UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

FIB

HARROW-HASSIDIM-LLOYD ALGORITHM: STUDY AND IMPLEMENTATION USING QIBO

ALEXANDRE ALEMANY ORFILA

Thesis supervisor: JORDI DELGADO PIN (Department of Computer Science)

Thesis co-supervisor: ALBA CERVERA LIERTA

Degree: Bachelor Degree in Informatics Engineering (Computing)

Thesis report

Facultat d'Informàtica de Barcelona (FIB)

Universitat Politècnica de Catalunya (UPC) - BarcelonaTech

26/06/2023

Abstract

In this thesis report we present the study of the Harrow-Hassidim-Lloyd quantum algorithm. This study compiles the theoretical study of the algorithm from scratch including the algorithms used as subroutine in the HHL like the Quantum Fourier Transform and the Quantum Phase Estimation and their implementation using Qibo, an open source python framework for programming quantum computers.

An introduction to quantum computing and an explanation about the state of the art are given in order to be able to follow this report. The main idea is to study and understand the relevance of quantum computing through an specific case: the HHL algorithm. This algorithm is aimed to solve systems of linear equations.

Key words: quantum computing, quantum algorithms, quantum Fourier transform, quantum phase estimation, Harrow-Hassidim-Lloyd algorithm, QFT, QPE, HHL.

Acknowledgements

I would like to acknowledge to my supervisors, Alba and Jordi, to support this project about quantum computing. Not only with their knowledge about the topic but motivating me and encouraging me to get into this field accepting to supervise this project.

Then I would like to thank to my friends from university that helped me during these years and shared with me great moments on and off campus.

Finally, I should thank to my family, I could not be at university without their support.

Contents

List of Tables

List of Figures

Code cells

1 Context of the project

The general goal of a bachelor final project (Spanish acronym: TFG) is to prove the achievement of the competences acquired during the bachelor of informatics engineering in the Barcelona School of Informatics (FIB) at the Universitat Politècnica de Catalunya (UPC). Precisely, this project is part of the computing specialization. The computer science specialization is aimed to study the foundations of computer science and makes the student capable of designing complex informatics systems keeping in mind critical points of efficiency, reliability and security.

This project will be conducted in the branch of algorithmia and complexity theory working on the theoretical computer science field and its relation with quantum computing. The purpose is to achieve an extensive study of a quantum algorithm to solve linear equations systems including its correctness and complexity in time. Furthermore, the quantum algorithm will be implemented in an open source python framework called Qibo [12] and it will be run in the first Spanish quantum computer based in the Barcelona Supercomputer Center, if possible.

This project has the support of the director PhD. Jordi Delgado Pin, member of the Computer Science Department (CS) at the UPC [9] and of the co-director PhD. Alba Cervera Lierta, senior researcher in QUANTIC, the quantum computing research group at the BSC [10] and coordinator of the Spanish quantum computing project, Quantum Spain [11].

2 Introduction of the project

2.1 Introduction

In 1936 Alan Turing presented what became the foundation for theories about computing and computers, the Turing machine [1]. This concept defined an abstract computational machine that helped to investigate the extent and limitations of what can be computed. The Turing Machine is the starting point to question what is an algorithm, what is computation or what is an efficient computer. That work, among a lot of other contributions to the field, is what made Alan Turing to be called the father of computer science.

Later in 1945, John Von Neumann proposed what we know as the Von Neumann architecture. This computer architecture was a revolutionary idea for that time and the base for the modern computer we know nowadays. The idea was that both programs and data are stored in memory [2]. And this work became John Von Neumann as one of the most important characters of the computer architecture field.

But if we take a look at the beginnings of the past century, physicists started working on quantum physics. This revolution was led by Max Planck and the reason this was an inflection point was to start considering the light as a particle, a photon [3]. Quantum physics enabled inventions like the transistor, a fundamental piece for a computer. But also was the beginning for quantum mechanics.

Although devices that were invented thanks to the knowledge about quantum mechanics are used daily and found in computers like the transistor, quantum properties are not exploited in the classical computation. For example, the simulation of quantum systems is exponential problem.

So, what if we would like to simulate computationally nature which has a quantum behavior? That question was stated by the physicist Richard Feynman in his work "Simulating Physics with Computers" [4] published in 1982 and starting what a lot of people think is the beginning of quantum computing. In that work he postulated that to simulate quantum systems, it would be needed to build quantum computers. This is indeed a computer capable of working with quantum properties and states. This was the gate to an emerging field of study that presented us a new computation model that using quantum mechanics would let us, not only to simulate, but to solve a small set of problems of optimization, of group theory, number theory... in a more efficient way getting results in a reasonable amount of time.

This Bachelor Final Project will research the power of quantum computing studying an specific algorithm that solves the problem of solving systems of linear equations. This algorithm is the HHL quantum algorithm [5] getting an exponential speedup against the classical method (in optimal conditions). For that we will need to understand the basics of some of the most used quantum algorithms: the Quantum Fourier Transform and the Quantum Phase estimation. The idea of the project is to study this algorithm from scratch.

2.2 Problem statement

The goal of this project is the study of the application of quantum computing in order to efficiently solve a given system of linear equation using the Harrow-Hassidim-Lloyd quantum algorithm (HHL algorithm [41]). Throughout this thesis we will see that this problem has been well studied using classical computing and with linear complexity. However the HHL algorithm promises an exponential speedup (in optimal conditions) against the classical method. Thus, the project will focus on the study of the HHL algorithm from scratch for the proper understanding of the technique. To do so, this project will go through the fundamentals of quantum computing, the quantum algorithms needed for the HHL algorithm such as the QFT and QPE and then the analysis and implementation of the quantum algorithm.

As a result, we produce a detailed description of one of the most known quantum algorithms and its implementation using a real open source python quantum computing framework, Qibo [12]. With this we will contribute in an open source project. Furthermore, the code in this project will be able to be ran in the quantum computer from the Barcelona Supercomputing Center.

3 State-of-Art of Quantum computing

Quantum computing is a new computing paradigm. This new model of computation has been briefly explained in the introduction of this project. In this section we will get deeper in the definition and the actual situation of quantum computing.

Although quantum computing is not a new field, it is getting every year a bigger relevance as times goes on. This is because the quantum computing research is been expanding outside the laboratories and universities and many companies are investing on it due to the potential applications. There is a big effort from public and private institutions trying to build and scale (in number of qubits) real quantum computers (or fault-tolerant quantum computers) as well as developing software that enables to run quantum algorithms on them. However, this is still a novel field of study with a long way to go.

The point of this efforts and hard work is that quantum computing can tackle problems that are very hard and useful and at the same time these problems are unfeasible or intractable on any classical computer. This is because these computers take advantage of quantum properties to manipulate and transform data. This is why they do not work using bits as we know, but using qubits, quantum bits. This fact lets the computer to behave in a total different way than classical computers. Therefore, new "paths" to get a solution for a set of problems have been found and there is research to find new "paths" applicable to more problems that could give an efficient solution. These paths are the quantum algorithms, like the ones we are going to study in this project.

3.1 Quantum hardware

Talking about quantum hardware, means talking about the quantum computer itself: how is it build. In this section we are going to see the situation of quantum computers nowadays. There are different types of quantum hardware and different types of quantum computers, in the following subsections we will discuss about these classifications.

3.1.1 Types of quantum hardware

Quantum computers can be classified by their type of architecture i.e. the type of qubits the quantum computer uses to process the information. There are different trendy technologies that are currently applied in the industry.

- Ion traps qubits: Trapped ion quantum computers utilize the electronic states of charged atoms known as ions to implement qubits. These ions are confined and suspended above a microfabricated trap using electromagnetic fields. Trapped-ion systems employ lasers to apply quantum gates, manipulating the electronic state of the ion. Unlike synthetic qubits, trapped ion qubits rely on naturally occurring atoms [22].

- Photon qubits: A quantum photonic processor is a device designed to manipulate light for computational purposes. Photonic quantum computers employ

quantum light sources that emit squeezed-light pulses, with qubits that are equivalent to modes of a continuous operator, such as position or momentum [22].

- Superconducting qubits: Superconducting quantum computers use superconducting electronic circuits that operate at very low temperatures. Superconductivity is a property observed in certain materials where electrical resistance becomes zero and magnetic flux fields are expelled below a critical temperature. Superconducting qubits, built with these circuits, enable the storage and processing of information in a quantum computer. This technology is considered one of the most promising approaches to quantum computing. Ongoing research aims to improve the performance and scalability of superconducting quantum computers[22].

3.2 Quantum software

In the current time, it may seem that the industry is focused on quantum hardware, the truth is that there is a whole sector in quantum computing called quantum software. Quantum software is gaining importance every year. In fact, in terms of quantum technology, quantum software lead the 36% of investment against the 46% of investment in quantum hardware [23].

Quantum software is key in quantum computing. Quantum software is the tool that will manage and program the quantum computers. This means the full stack from the user that writes code to the execution of the code in the quantum computer. The application of the theoretical quantum algorithms that have been designed and that will be designed have to be adapted into software solutions. Not only this, but the implementation of requirements engineering, architecture and design, development, testing, debugging, and maintenance phases [24].

Nowadays, we find different options to program quantum computers. These programming languages and frameworks allow a low-level programming i.e. designing quantum circuits that will be then sent to the quantum computer or simulator[24]. Nevertheless, the design of the circuit does not need to be taking in mind the distribution of qubits of the actual device that will be used to run the circuit. This does not mean that we do not need software development for hardware issues. For example, it is needed in order to map the circuit that the user sends to the quantum computer according to the topology of the device. Thus, there are different topics and problems to tackle from the software perspective. We are far from a high level programming as we are used to in classical computing.

3.2.1 Quantum programming languages and frameworks

There are several companies and research laboratories developing their quantum programming languages. In this section, the most known will be introduced:

- Qiskit is an open source python framework developed by IBM. IBM also has it is a platform where the user can design circuits or can program them using

Qiskit. This framework allow the access to different quantum computers as well as to simulators.

- Cirq is an open source python framework developed by Google. Cirq has access to devices and simulators to test and run the code of the user too.
- Pennylane is an open source python framework developed by Xanadu. This framework is mainly addressed for quantum machine learning although it can be used as the other frameworks.

3.2.1.1 Qibo [12] Finally Qibo is also an open source python framework. This framework is not owned by any company and has the goal to enable developers to delegate all complicated aspects of hardware or platform implementation to the library so they can focus on the problem and quantum algorithms at hand. This will be the framework that will use the first Spanish quantum computer from the Quantum Spain project [11] and the one that will be used in this project.

3.3 Current Era of quantum computing

3.3.1 Noisy Intermediate-Scale Quantum Era

As said before, quantum computing is still a novel field. This means that in the current time it is still studied how to build a fault-tolerant quantum computer i.e. a quantum computer with quantum error correction with less noisy qubits, where noise refers to the multiple factors that can affect the accuracy of the calculations a quantum computer performs. However, we do have quantum computers although they are not perfect. This is called the Noisy Intermediate-Scale Quantum Era (NISQ).

In the NISQ era, a well behaved quantum computer is a goal to achieve, but also to get the maximum from the experimentation of the quantum computers we have today. In the meantime, different companies and researchers opts for the architecture and study the performance and the applications of their NISQ quantum computers. These computers are noisy and they produce errors on their calculations. Thus, algorithms like the HHL algorithm can not be executed nowadays because the current error rates and noise obtained inevitably destroy any precision in the results [25]. This is why currently NISQ algorithms are being studied and studied so current quantum computers can be useful and worth. NISQ algorithms are designed to be implementable on NISQ devices in the near term, i.e. the next few years. This also implies that NISQ programmers aim to use as many qubits as it is physically implementable [25].

3.3.2 Fault-tolerant quantum computing Era

In the long term, we should view the NISQ era as a step towards full fault tolerance and the development of more powerful quantum devices. We do not expect NISQ devices to be fully capable of realising the power of quantum computers [25].

The fault-tolerant quantum computing Era will begin when we have quantum computers that have quantum error correction and can perform properly. However,

as said in the previous section, we are far from this point. Furthermore, it is important to study and investigate about quantum computing through the NISQ era since it will give knowledge for the fault-tolerant quantum computing Era, not only about quantum computing but about quantum algorithms and quantum mechanics itself. Nowadays, there are not many algorithms known[25] [26]. It is important to study and develop NISQ algorithms to see how quantum computers can be useful today and study fault-tolerant algorithms to be able to experiment with them once fault-tolerant quantum computers are available.

3.4 Quantum industry and investment

Quantum computing is an emerging field of study as said before. But not only in the academia but also in industry. Around the world many countries and company are investing a lot of resources and money in order to be part of this new "business". If we take a look at the Figure 15 from QURECA[27] we can see an overall investment of 36 billions of dollars and it is projected to reach 42.4 billion of dollars by 2027.

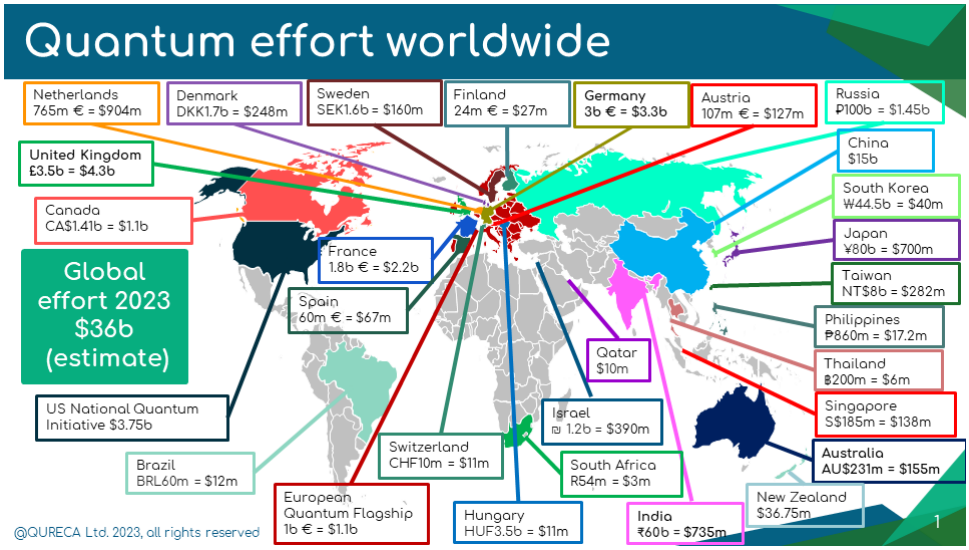


Figure 1: World map of investment in quantum technologies. Image obtained from QURECA[27]

Lets take a look to the main global powers: The United States of America, China and the European Union. In these cases the investment is 3.75 billions of dollars, 15 billions of dollars and 1.1 billions of dollars respectively . This 2023 the United Kingdom and Germany announced on their own quantum strategies that imply an investment of 4.3 billions 3.3 billions of dollars respectively. Spain also announced this year that a quantum strategy is being planned, although 67 millions of dollars are already being invested in the Quantum Spain project. This money comes from the Spanish ministry of economic affairs and digitalization. In particular, from the general secretary of state of digitalization and artificial intelligence. It is founded by NextGeneration European funds.

When talking about private investment, there was an estimated investment of 2.35 billions of dollars into quantum technology start-ups [28]. Slightly more than 2021. McKinsey & Company did an analysis that shows that the four industries

likely to see the earliest economic impact from quantum computing—automotive, chemicals, financial services, and life sciences—stand to potentially gain up to $1.3 trillion in value by 2035 [28].

In the meantime, we have the big tech companies investing in research and development of quantum computing. IBM released the general purpose quantum computer with more qubits at the moment with 433 qubits and it is planned to launch a quantum computer with more than 1000 qubits this 2023 [30]. Google announced to have achieved the quantum supremacy in 2019 [31] and it is team in Google Quantum AI is focused on the development of quantum computer and it is applications in AI. In fact, we can see that in 2019 the private investment increases to a really large degree in figure 4

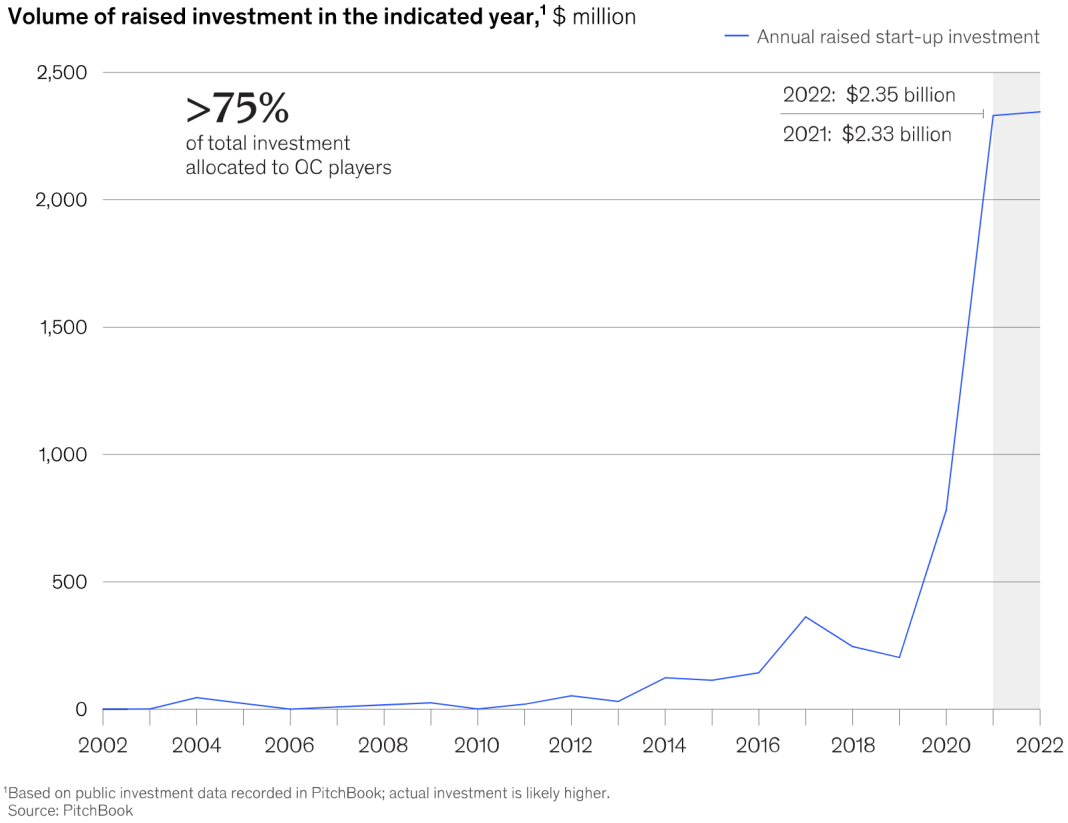


Figure 2: Graph of private investment in quantum technologies from [28]

Summarizing, quantum computing is becoming a strong field of study that grows quickly. This is why the effort of different disciplines, not only physics, but computer science and engineering are needed in order to keep developing this field. There are big challenges to address in quantum computing and there may be more as we keep doing research and learning about it.

4 Key Concepts and Introduction to Quantum Computing

In order to better understand this thesis and to give some previous background to the reader, some basic key concepts will be introduced in this section.

4.1 Quantum Computing

Quantum computing is a new paradigm of computation that uses the quantum properties in order to compute the data. One of the most significant difference with classical computing is that quantum computing uses qubits (quantum bits) as the smallest unit of information. A qubit is a quantum state. This section and the next one will be explained following "Ref. [6]".

A quantum state is a mathematical entity that provides a probability distribution for the outcomes of each possible measurement on a system. A qubit has analog states for the classical 0 and 1, these are a $|0\rangle$ and $|1\rangle$. A single qubit will be represented with this two states as references, the computational basis states.

There are some concepts to understand the qubits and the difference between them and the classical bits:

- Superposition: As we said, a qubit is a quantum state, which is a probability distribution. It is said that qubits can be 0, 1 or both at the same time, but this is wrong. A qubit can have a 100% of probabilities to be measured and get 0 or 1. But you can have any other probability distribution to get one state or the other (if these two probabilities when added are 100%). This is called superposition of two states. In fact, a qubit state can be represented as follows:

  $$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$$

  Where α and β are complex numbers, these complex numbers are called amplitude, and $|\alpha|^2$ and $|\beta|^2$ are the probabilities of getting the 0 and 1 respectively. Naturally, in order to get a 100% of probability, $|\alpha|^2 + |\beta|^2 = 1$. So when $|psi\rangle = |0\rangle$ means that $|\psi\rangle = 1|0\rangle + 0|1\rangle$. So the probability to read this quantum state is 100% $|0\rangle$. Same for $|\psi\rangle = |1\rangle$

- Collapse: When a qubit is measured, the measurement will be either 0 or 1. Even if the qubit it is in superposition, it will collapse to one state or other. This is a postulate of quantum mechanics and the main reason why quantum computing is probabilistic. This is why experiments have to be executed more than once to obtain a distribution of probabilities.

- Entanglement: Two qubits can be entangled, which means that they are correlated. Two qubits are entangled if it is not possible to represent them with two separated and differentiated wave functions i.e. the entangled state cannot be written as a tensor product such as $|\psi\rangle \neq |\psi_A\rangle \otimes |\psi_B\rangle$

It is also interesting to introduce the Bloch Sphere 3. Since $|\alpha|^2 + |\beta|^2 = 1$, we can rewrite the previous qubit expression as:

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

Where θ and φ are real numbers that define a point on the unit three-dimensional sphere, the Bloch sphere, giving a geometric representation of a single qubit. It is important to notice that the Bloch sphere can not represent more than one qubit.

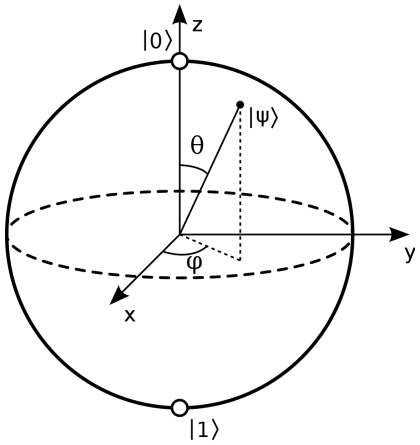


Figure 3: Representation of the Bloch sphere from [29]

4.2 Quantum gates

The quantum gates are the operators or instructions used in quantum computing in order to to manipulate and transform the qubits. As well as in classical computing there are logical gates, quantum gates can be used in quantum computing for this purpose. Quantum circuits are designed in order to use quantum gates. This quantum circuits are represented as wires (lines) which represent the qubits and the boxes which represent the gates. When a wire goes through a box, the operator is applied to the qubit.

This gates or operators are unitary operations that can be represented as unitary matrix. An operator U (or a matrix) is unitary when $U^*U = I$. Where I is the identity and U^* is the transposed conjugated U. this unitarity constraint is the only constraint on quantum gates. In quantum computing the processes have to be unitary.

A quantum operator has to be hermitian too. This means that the conjugated and transposed matrix of the operator has to be the same as the original matrix. So we want to satisfy that $M^* = M$. As we can see, the matrix F is hermitian.

Some basic gates and the ones needed in this project will be introduced in this section:

- X gate: The X gate is the analog get for the classical NOT gate i.e. given an $|0\rangle$, using the X gate on it, the obtained output would be $|1\rangle$. This means that

the probability of getting a 0 is now the probability of getting a 1 and vice versa. What this gate is really doing is a rotation on the X-axis of 180^{o}. This is the circuit representation:

$$|0\rangle \text{ —}\boxed{X}\text{— } |1\rangle$$

$$|1\rangle \text{ —}\boxed{X}\text{— } |0\rangle$$

The unitary matrix that represents the X gate: $\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$

- Hadamard gate: This gate creates a superposition in a given qubit where the probabilities of the $|0\rangle$ and $|1\rangle$ are 50% in both cases. In the following figure, the states of applying this gates are given. This probability, as seen before, is given by the amplitude. The resulting states of the Hadamard gate can be rewritten as $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ and $\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$ respectively. The amplitude of both states is $\frac{1}{\sqrt{2}}$ which means that the probability of getting one of both states is $|\frac{1}{\sqrt{2}}|^2 = \frac{1}{2}$. However, they are different states and this can effect the workflow of the state. This is the circuit representation:

$$|0\rangle \text{ —}\boxed{H}\text{— } \tfrac{|0\rangle+|1\rangle}{\sqrt{2}}$$

$$|1\rangle \text{ —}\boxed{H}\text{— } \tfrac{|0\rangle-|1\rangle}{\sqrt{2}}$$

The unitary matrix that represents the H gate: $\frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$

- U1 gate: Or first general Unitary gate is a gate that transforms the qubit applying a rotation as follows.

$$|x\rangle \text{ —}\boxed{U1}\text{— } e^{i\theta}\,|x\rangle$$

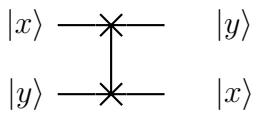
The unitary matrix that represents the U1 gate: $\begin{pmatrix} 1 & 0 \\ 0 & e^{i\theta} \end{pmatrix}$

- RY gate: The Ry gate is a single-qubit rotation around the Y-axis. specifically applies a rotation depending on θ. For example, being $|0\rangle$ the input state and $\theta = \frac{\pi}{2}$:

$$|0\rangle \text{ —}\boxed{RY}\text{— } \tfrac{1}{\sqrt{2}}\,|0\rangle + \tfrac{1}{\sqrt{2}}\,|1\rangle$$

The unitary matrix that represents the RY gate: $\begin{pmatrix} \cos\frac{\theta}{2} & -\sin\frac{\theta}{2} \\ \sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{pmatrix}$

- SWAP gate: This 2-qubit gate is aimed to swap the two input qubits. This means that having $|ab\rangle$, after applying this gate, the output will be $|ba\rangle$. This is its circuit representation:

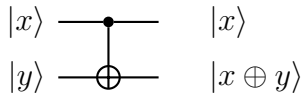


The unitary matrix that represents the SWAP gate: $\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$

- C-NOT gate: This gate is a 2-qubit gate that given two qubits, one will work as the control and the other will work as the target. The exact operation is: when the control qubit is $|1\rangle$, an X gate will be applied to the target qubit. Nothing happens if the control qubit is $|0\rangle$. Thus, we can define the operation:

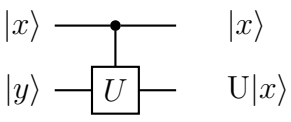
$$|xy\rangle \longmapsto |x(x \oplus y)\rangle$$

where x is the control qubit, y is the target qubit and $\oplus$ is the binary addition. The circuit would look like:



The unitary matrix that represents the U1 gate: $\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$

- C-U gate: Finally, any gate can be controlled, like the C-NOT gate. This means that having a U gate, if needed, it can be applied conditionally having a control qubit that applies U to the target qubit when the control qubit is $|1\rangle$. The general circuit representation is:



The unitary matrix that represents the U1 gate: $\begin{pmatrix} I & 0 \\ 0 & U \end{pmatrix}$. Where I is the unitary matrix.

The measurement of a qubit is also important. The measurement is a classical gate that collapses the qubit giving a classical state based on the computational basis (a qubit will always return 0 or 1). It is not possible to keep the quantum

properties after measure the qubit. Thus, in general, quantum circuits will not have mid-measurements.

Finally, in order to implement these circuits, different programming languages or python frameworks exist in order to design and send them to a quantum computer. The programming of the circuits (or algorithms) are not quantum. Software is and will be classical. Once the circuit is implemented, it will be sent to the quantum hardware and then the output will be obtained again as a classical state.

4.3 Mathematical basics

Quantum computing can be expressed as linear algebra operations. Quantum states can be represented as vectors and it has been shown that operators can be represented as unitary matrix. This means that a qubit being $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ can be expressed as $\begin{pmatrix} a \\ b \end{pmatrix}$. Therefore, the computational basis states ($|0\rangle$ and $|1\rangle$) can be expressed as $\begin{pmatrix} 1 \\ 0 \end{pmatrix}$ and $\begin{pmatrix} 0 \\ 1 \end{pmatrix}$ respectively.

When we have a 2 qubit system, we can express them with the tensor product. Thus, when having $|xy\rangle$, it can also be expressed as $|x\rangle \otimes |y\rangle$. Having

$$|x\rangle = \begin{pmatrix} a \\ b \end{pmatrix} \; |y\rangle = \begin{pmatrix} c \\ d \end{pmatrix}$$

Then, $|x\rangle \otimes |y\rangle = \begin{pmatrix} ac \\ ad \\ bc \\ bd \end{pmatrix}$.

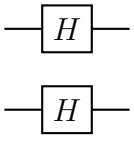
When two qubits can be expressed using the tensor product, this means that they are separated states i.e. not entangled.

The tensor product can be used with operators too. When two operators $U_1 = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$ and $U_2 = \begin{pmatrix} x & y \\ z & t \end{pmatrix}$ are expressed in a tensor product, we have that:

$$U_1 \otimes U_2 = \begin{pmatrix} a \cdot U_2 & b \cdot U_2 \\ c \cdot U_2 & d \cdot U_2 \end{pmatrix}$$

.

For example, if two Hadamard gates are applied to two qubits like in the following circuit:



Mathematically can be expressed with the tensor product:

$$H \otimes H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \otimes \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix}$$

Now we have the basics to start working on quantum computing. This section also justifies that the simulation of quantum systems is exponential because of the use of vectors and matrices. Therefore, a quantum computer, that can deal with quantum systems efficiently, is perfect for problems that involve quantum systems.

5 Quantum Fourier Transform

The Quantum Fourier Transform (QFT) is one of the most famous quantum algorithms. The QFT is the core of a lot of important quantum algorithms as the Quantum Phase Estimation (QPE), algorithm that will be studied in this project.

However, in order to understand this quantum algorithm, it is important to start for its classical version, the Fourier transform. We will see what is this transform, why it is used and how it is implemented.

5.1 Classical Fourier Transform

The Fourier Transform was created by Jean-Baptiste Joseph Fourier in the 1821, when he discovered that any function, whether continuous or discontinuous, can be expanded into a series of sines [33]. Once this transform is applied to a function, this returns a new function in a form that describes the frequencies of the original function. In simpler words, it lets you to transform a time function into a frequency function.

This makes it a useful tool for the fields of physics and telecommunications because it lets to analyze signals, for example for audio processing, image processing...

Nevertheless the functions that are wanted to transform have to meet certain conditions. These are:

- The function has to be totally integrable.
- In every given finite interval, there must be a finite number of maximums and minimums.
- In every given finite interval, there must be a finite number of discontinuities.

5.1.1 Formula of the Fourier Transform

The goal of the Fourier transform, as said before, is to decompose a waveform, which is a function of time, into the frequencies that make it up. In order to achieve this, the formula that is applied is following one:

$$F(k) = \int_{-\infty}^{\infty} f(x) e^{-i2\pi kx} dx \tag{1}$$

Where:

- $F(k)$ is the transformed function
- $f(x)$ is the function to transform.
- k is the frequency.
- 2π represents a full round of a circumference.

In order to explain what the Fourier transform does in a simple way, lets imagine a function $f(x)$. What the Fourier transform does is to find the frequencies that compose $f(x)$. To do so, the function is transformed in a 2-dimension plane where $f(x)$ is wrapped following a center point and a given frequency k [34].

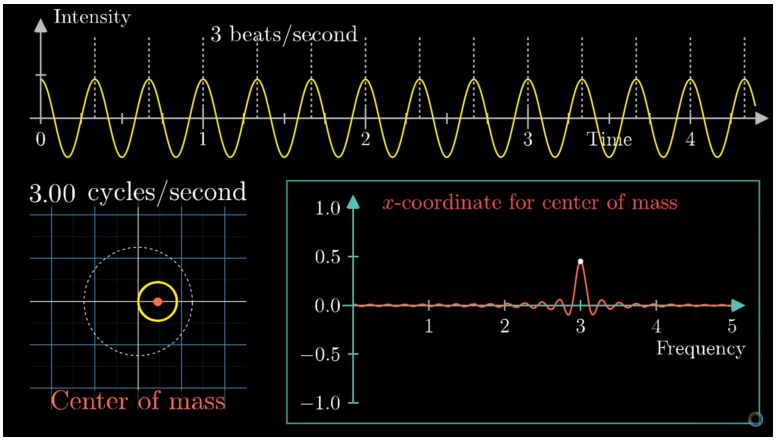


Figure 4: Visual help for understand how a function is wrapped. Image from[34]

As we can see in the image, the function from above, is wrapped in the "center of mass" graphic.

Once $f(x)$ is mapped in a 2-dimension plane, in fact, the average of the points distributed on the plane is calculated and represents the center of mass of the wrapped function. The resulting function $F(k)$ represents this center of mass depending on the frequency of wrapping of the function $f(x)$. This $F(k)$ will have a peak in each point where k equals the frequencies that compose $f(x)$.

5.2 Discrete Fourier Transform

The Discrete Fourier Transform is the Fourier transform applied to a set of points of the function to be transformed. This transform is very important in order to be able to computationally apply the Fourier Transform very accurately. Since a computer can not treat all the points of a function the Discrete version of the Fourier Transform is used in practice. Nevertheless, the DFT has more applications like the multiplication of polynomials. The Quantum Fourier Transform will be explained from he DFT.

The DFT defines a function $f : \mathbb{R}^n \longmapsto \mathbb{C}^n$. This means that given a set of n real numbers, returns n complex numbers[32]. These are the n real numbers transformed to its fourier basis.

5.2.1 Formula of the Discrete Fourier Transform

Given a set of points X_N the following formula represents the Discrete Fourier Transform [32]:

$$X_k = \sum_{n=0}^{N-1} x_n e^{-i2\pi kn/N} \tag{2}$$

Where:

- X_k is the transformed set of points.
- x_n Is the point to be transformed.
- n is the current iteration of the formula.
- N is the total number of iterations and the length of X_N.
- $k : 0 <= k < N - 1$.

Since this is a discrete version of the Fourier transform, it is not possible to use the time or frequency. In this version, the time is substituted by the n and the frequency by k/N.

5.2.2 Discrete Fourier Transform operator

In the previous section, the formula of the DFT was introduced. However, the DFT can be expressed as a matrix operator. This allows its application in linear algebra. Thus, given a vector $x = [x_0, x_1, ..., x_{N-1}]^T$ that represents the set of points to be transformed, and a matrix F, which is the $N \times N$ DFT matrix, the vector x can be transformed using the following F:

$$F_{jk} = \frac{1}{\sqrt{N}} e^{-i2\pi kj/N}, \quad j, k = 0, 1, \ldots, N-1. \tag{3}$$

This formula can be simplified using $\omega = e^{-i2\pi/N}$:

$$F_{jk} = \frac{1}{\sqrt{N}} \omega^{jk}, \quad j, k = 0, 1, \ldots, N-1. \tag{4}$$

The matrix will look like the following:

$$F = \frac{1}{\sqrt{N}} \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega & \omega^2 & \cdots & \omega^{N-1} \\ 1 & \omega^2 & \omega^4 & \cdots & \omega^{2(N-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{N-1} & \omega^{2(N-1)} & \cdots & \omega^{(N-1)(N-1)} \end{pmatrix} \tag{5}$$

Now the matrix is defined. In order to transform the vector x it is only needed to multiply the matrix and the vector using the form $\mathbf{y} = F\mathbf{x}$ so the output vector will be the transformed vector.

$$\mathbf{y} = F\mathbf{x} = \frac{1}{\sqrt{N}} \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega & \omega^2 & \cdots & \omega^{N-1} \\ 1 & \omega^2 & \omega^4 & \cdots & \omega^{2(N-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{N-1} & \omega^{2(N-1)} & \cdots & \omega^{(N-1)(N-1)} \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_{N-1} \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \\ y_{N-1} \end{pmatrix} \tag{6}$$

Essentially, the matrix F is the quantum operator for the Quantum Fourier Transform. In the following section the formulas will be adapted for transforming the Qubits.

5.3 From DFT to QFT

Now we understand the Fourier Transform and its discrete form (DFT), it is time to adapt it for quantum computing. First important thing is to understand the difference between the DFT and the QFT.

The DFT is transforming a vector of real numbers into another vector of complex numbers. Computationally these are two different vectors using different storage space. However the QFT is mapping a set of qubits states into another set of qubits states, although the qubits are not changed. So we are working with quantum states. The QFT is only modifying the amplitudes of the qubits but they do not have an specific storage space and their mapping will not need more space [32].

5.3.1 Formula of the Quantum Fourier Transform

The DFT and the QFT are practically the same mathematically speaking but very different conceptually. Once this difference is understood, the path from DFT to QFT becomes simpler. The first step is to understand that $N = 2^n$ because the QFT will be mapping qubits which have 2^n combinations given n qubits. Now, lets take a look at the formula of the Quantum Fourier Transform which is the following [32] [35]:

$$|k\rangle \mapsto F_N|k\rangle = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} \omega_N^{jk} |j\rangle, \quad j, k = 0, 1, \ldots, N-1 \tag{7}$$

Where:

- $N = 2^n$.
- $\omega = e^{-i2\pi/2^n}$.
- $F_N|k\rangle$ is the transformed set of points.
- $|j\rangle$ is the point to be transformed.
- j is the current iteration of the formula.
- N is the total number of iterations.

This formula is analog to the DFT formula but using Dirac notation since we are expressing quantum states. And the quantum operator is exactly the same as the DFT operator.

5.3.2 Quantum operator for the Quantum Fourier Transform

We already know that the operator of the QFT is the same as the DFT which is:

$$F = \frac{1}{\sqrt{2^n}} \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega & \omega^2 & \cdots & \omega^{2^n-1} \\ 1 & \omega^2 & \omega^4 & \cdots & \omega^{2(2^n-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{2^n-1} & \omega^{2(2^n-1)} & \cdots & \omega^{(2^n-1)(2^n-1)} \end{pmatrix} \tag{8}$$

This operator already transforms our quantum state to a quantum state in the Fourier basis. However, quantum operators have some conditions. In this section, these conditions will be checked in order to know if the DFT operator can be used as a quantum operator to define the QFT operator.

The matrix has to be unitary as well. This can be proved showing that any two columns of the matrix are orthogonal [32]:

$$\sum_{j=0}^{N-1} \frac{1}{\sqrt{N}} (\omega_N^{jk})^* \frac{1}{\sqrt{N}} \omega_N^{jk'} = \frac{1}{N} \sum_{j=0}^{N-1} \omega_N^{j(k'-k)} = \begin{cases} 1 & \text{if } k = k' \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

In this section we proved that the operator of the DFT can be used as the quantum operator for the QFT because it accomplishes the different conditions for being a quantum operator. Next step is to define a circuit that is equivalent to the QFT operator.

5.4 Efficient quantum circuit for the QFT

The quantum operator for the Quantum Fourier Transform (QFT) is defined in he previous section. So it is possible to implement a quantum circuit that performs the QFT. In this section, a quantum circuit will be presented to implement the QFT in a quantum computer using quantum gates [32] [6].

In order to do so, the formula of the QFT will be mapped to be implemented for n qubits. Lets remember the QFT formula:

$$|k\rangle \mapsto F_N|k\rangle = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} \omega_N^{jk} |j\rangle \tag{10}$$

For this reasoning, it is useful to substitute $\omega = e^{-i2\pi/2^n}$. Thus,

$$F_N|k\rangle = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} \omega_N^{jk} |j\rangle = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} e^{-i2\pi jk/2^n} |j\rangle \tag{11}$$

The quantum state $F_N|k\rangle$ is a tensor product of n qubits. There is no entanglement. So it is possible to rewrite $F_N|k\rangle$ as follows. Let $|k\rangle = |k_1 ... k_n\rangle$, being k_1 the most significant qubit. Then, the integer $j = j_1 ... j_n$, Notice that $j/2^n = \sum_{l=1}^{n} j_l 2^{-l}$. Now it is possible to rewrite the QFT formula as follows:

$$F_N|k\rangle = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} e^{-i2\pi jk/2^n}|j\rangle$$

$$= \frac{1}{\sqrt{2^n}} \sum_{j\in\{0,1\}^n} e^{i2\pi \sum_{l=1}^{n} j_l 2^{-l} k}|j_1...j_n\rangle$$

$$= \frac{1}{\sqrt{2_n}} \sum_{j\in\{0,1\}^n} \prod_{l=1}^{n} e^{i2\pi j_l k/2^l}|j_1...j_n\rangle$$

$$= \bigotimes_{l=1}^{n} \frac{1}{\sqrt{2}}(|0\rangle + e^{i2\pi k/2^l}|1\rangle \tag{12}$$

Now the formula has been written in a way that is much easier to understand for its implementation in a circuit. First we need to introduce some notation. Being $k_1k_2k_3...$ a string of bits, we can represent decimal numbers as $0.k_1k_2k_3...$ where $k_i = \frac{1}{2^i}$, $k_1 = \frac{1}{2}$, $k_2 = \frac{1}{4}$, $k_3 = \frac{1}{8}...$ and $0.k_1k_2k_3... = k_1 + k_2 + k_3....$

Once this notation is understood, lets see some examples:

- for $N = 2$:

$$F_2|k_1\rangle = \bigotimes_{l=1}^{1} \frac{1}{\sqrt{2}}(|0\rangle + e^{i2\pi k/2^l}|1\rangle) = \frac{1}{\sqrt{2}}(|0\rangle + e^{i\pi k}|1\rangle) = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \tag{13}$$

  The resulting expression is the Hadamard gate. So applying the QFT to one single qubit is the same as applying the Hadamard gate.

- for $N = 4$

$$F_4|k_1k_2\rangle = \bigotimes_{l=1}^{2} \frac{1}{\sqrt{2}}(|0\rangle + e^{i2\pi k/2^l}|1\rangle) = \frac{1}{\sqrt{2}}(|0\rangle + e^{i2\pi 0.k_2}|1\rangle) \otimes \frac{1}{\sqrt{2}}(|0\rangle + e^{i2\pi 0.k_1k_2}|1\rangle)$$

$$= \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes \frac{1}{\sqrt{2}}(|0\rangle + e^{i2\pi 0.k_1k_2}|1\rangle) \tag{14}$$

  To implement this expression, a new gate is introduced. It is needed a gate to rotate the qubit $0.k_1k_2$ times a circumference the qubit in the Z-axis.

The needed gate for the previous implementation is the $R_s = \begin{pmatrix} 1 & 0 \\ 0 & e^{i2\pi/2^s} \end{pmatrix}$. This gate does a rotation to the qubit in the Z-axis and it let us to implement efficiently the QFT circuit. If we take a look to this gate, we see that it is analog to the U1 gate introduced before where $\theta = \frac{2\pi}{2^s}$.

lets take the last example again:

- for $N = 4$:
  The first qubit will have a Hadamard to get the state of the expression. The second qubit will have a Hadamard as well because, as the previous qubit, $e^{i2\pi 0.k_2} = (-1)^{k2}$ (which substituted in the expression gives the Hadamard expression). Then, conditioned by k_3, the R_2 gate will be applied to k_2. This is

because, applying the Hadamard, the qubit k_2 has the state $\frac{1}{\sqrt{2}}(|0\rangle+e^{i2\pi 0.k_2}|1\rangle)$. Then applying the Control R_2 conditioned by k_3, multiplies $|1\rangle$ with the phase $e^{i2\pi 0.0k_3}$ producing the final desired state.

The final step is to swap the qubits so we have the same order as in the formula. The resulting circuit is:

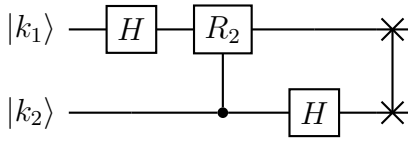


Figure 5: 2-qubits QFT circuit. Original creation

- Finally, lets see $N = 8$ The formula to follow would be:

$$F_9|k_1k_2k_3\rangle = \frac{1}{\sqrt{2}}(|0\rangle+e^{i2\pi 0.k_3}|1\rangle)\otimes\frac{1}{\sqrt{2}}(|0\rangle+e^{i2\pi 0.k_2k_3}|1\rangle)\otimes\frac{1}{\sqrt{2}}(|0\rangle+e^{i2\pi 0.k_1k_2k_3}|1\rangle) \tag{15}$$

As we can see, the pattern keeps the same, the qubits k_1 and k_2 keep having the structure as before. The change is that adding a new qubit like now, we keep applying a Hadamard to the new qubit and then apply the rotations conditioned by the rest of the qubits. In this case We should apply a Hadamard to the k_3 and then a conditioned rotation by k_2 and a conditioned rotation by k_1. Finally, we have to correct the order of the qubits adding a swap between the k_3 and k_1. The final circuit would be:

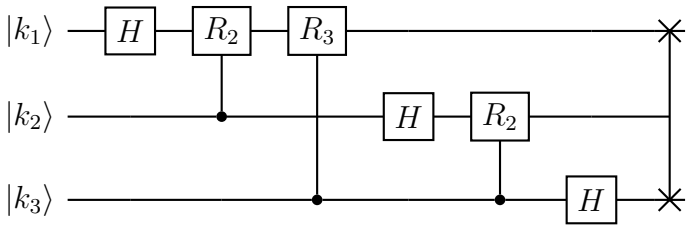


Figure 6: 3-qubits QFT circuit. Original creation

5.5 Complexity of the Quantum Fourier Transform

Now, the circuit implementation of the QFT is introduced and explained, it is complexity can be discussed. The time complexity of quantum algorithms is determined by the number of 1-qubit and 2-qubit gates needed to implement the circuit that represents the quantum algorithm. This means, that having a look at our previous section, the complexity if the Quantum Fourier Transform can be figured out.

In order to implement the QFT for n qubits, we've seen that the most significant qubit has n gates. Then, each qubit needs one less gate. Finally at most every qubit

needs an extra gate for swapping. Knowing this, the complexity of the QFT can be stated. This is $\mathcal{O}(n^2)$ because each qubit will use at most n gates, so $n \times n = n^2$. Thus, the time complexity of the QFT is $\mathcal{O}(n^2)$.

5.6 Implementation using Qibo and experimentation

In order to finally close the Quantum Fourier Transform part, an implementation using Qibo will be given implementing the general circuit as this project states.

```
def myQFT(nqubits):

    #Declare an empty circuit of n qubits
    circuit = Circuit(nqubits)

    for q1 in range(nqubits):

        #We apply a Haddamard gate to each qubit
        circuit.add(gates.H(q1))

        for q2 in range(q1 + 1, nqubits):
            #We define theta. theta is needed to apply the
   following gate.
            theta = math.pi / 2 ** (q2 - q1)

            #We apply the controlled rotations using CU1 gate
            circuit.add(gates.CU1(q2, q1, theta))

    #We apply swaps to every pair of qubits
    for i in range(nqubits // 2):
        circuit.add(gates.SWAP(i, nqubits - i - 1))

    return circuit
```

Code cell 1: Implementation of the QFT circuit using Qibo

The previous function returns a circuit of n qubits, being n an integer as input. The output will be the Quantum Fourier Transform. The QFT is already implemented in Qibo but it has been decided to implement it in order to understand better Qibo and the QFT. This implementation can be found in the repository of this project in Github [36].

In order to test this code and experiment with the code, the simulator that Qibo provides will be used. In this case, we will run some circuits of 1, 2 and 3 qubits and check if the final state of the circuit is the expected one that encodes the input in its Fourier basis. Notice that measuring the qubits is not useful since qubits would collapse to a computational basis state.

In order to test it, we will run a program in order to try different numbers of qubits n and giving different possible input states. We will output the probabilities of the state and the actual final state of the circuit. In order to check the probabilities, they should always be the same for every possible collapse state. The state will be checked following the formula of the transformation.

- For $n = 1$ we have two options as input. When the input is the state $|0\rangle$, the output of the execution is:

```
(0.70711+0j)|0> + (0.70711+0j)|1>
tf.Tensor([0.5 0.5], shape=(2,), dtype=float64)
```

Code cell 2: Bash output for $n = 1$ qubits and input $|0\rangle$

And when the input state is $|1\rangle$:

```
(0.70711+0j)|0> + (-0.70711+0j)|1>
tf.Tensor([0.5 0.5], shape=(2,), dtype=float64)
```

Code cell 3: Bash output for $n = 1$ qubits and input $|1\rangle$

In both cases we get the proper state, since we know that the QFT applied only to one qubit is the same as applying the H gate.

- For $n = 2$ we have four input options: $|00\rangle$, $|01\rangle$, $|10\rangle$, $|11\rangle$:

When the input state is $|00\rangle$, the output we get is:

```
(0.5+0j)|00> + (0.5+0j)|01> + (0.5+0j)|10> + (0.5+0j)|11>
tf.Tensor([0.25 0.25 0.25 0.25], shape=(4,), dtype=float64)
```

Code cell 4: Bash output for $n = 2$ qubits and input $|00\rangle$

When the input state is $|01\rangle$, the output we get is:

```
(0.5+0j)|00> + (-0.5+0j)|01> + (0.5+0j)|10> + (-0.5+0j)|11>
tf.Tensor([0.25 0.25 0.25 0.25], shape=(4,), dtype=float64)
```

Code cell 5: Bash output for $n = 2$ qubits and input $|01\rangle$

When the input state is $|10\rangle$, the output we get is:

```
(0.5+0j)|00> + 0.5j|01> + (-0.5+0j)|10> + (-0-0.5j)|11>
tf.Tensor([0.25 0.25 0.25 0.25], shape=(4,), dtype=float64)
```

Code cell 6: Bash output for $n = 2$ qubits and input $|10\rangle$

When the input state is $|11\rangle$, the output we get is:

```
(0.5+0j)|00> + (-0-0.5j)|01> + (-0.5+0j)|10> + 0.5j|11>
tf.Tensor([0.25 0.25 0.25 0.25], shape=(4,), dtype=float64)
```

Code cell 7: Bash output for $n = 2$ qubits and input $|11\rangle$

In all cases the final state is correct if we calculate manually the formulas stated before.

- Finally we will try our function with $n = 3$. We will not try every case but our input states will be $|000\rangle$, $|001\rangle$, $|010\rangle$, $|101\rangle$ and $|111\rangle$.

When the input state is $|001\rangle$, the output we get is:

```
(0.35355+0j)|000> + (-0.35355+0j)|001> + (0.35355+0j)|010> +
    (-0.35355+0j)|011> + (0.35355+0j)|100> + (-0.35355+0j)|101>
     + (0.35355+0j)|110> + (-0.35355+0j)|111>
tf.Tensor([0.25 0.25 0.25 0.25], shape=(4,), dtype=float64)
```

Code cell 8: Bash output for $n = 3$ qubits and input $|001\rangle$

When the input state is $|010\rangle$, the output we get is:

```
(0.35355+0j)|000> + 0.35355j|001> + (-0.35355+0j)|010> +
   (-0-0.35355j)|011> + (0.35355+0j)|100> + 0.35355j|101> +
   (-0.35355+0j)|110> + (-0-0.35355j)|111>
tf.Tensor([0.25 0.25 0.25 0.25], shape=(4,), dtype=float64)
```

Code cell 9: Bash output for $n = 3$ qubits and input $|010\rangle$

When the input state is $|101\rangle$, the output we get is:

```
(0.35355+0j)|000> + (-0.25-0.25j)|001> + 0.35355j|010> +
   (0.25-0.25j)|011> + (-0.35355+0j)|100> + (0.25+0.25j)|101>
   + (-0-0.35355j)|110> + (-0.25+0.25j)|111>
tf.Tensor([0.25 0.25 0.25 0.25], shape=(4,), dtype=float64)
```

Code cell 10: Bash output for $n = 3$ qubits and input $|101\rangle$

When the input state is $|111\rangle$, the output we get is:

```
(0.35355+0j)|000> + (0.25-0.25j)|001> + (-0-0.35355j)|010> +
   (-0.25-0.25j)|011> + (-0.35355+0j)|100> + (-0.25+0.25j)
   |101> + 0.35355j|110> + (0.25+0.25j)|111>
tf.Tensor([0.25 0.25 0.25 0.25], shape=(4,), dtype=float64)
```

Code cell 11: Bash output for $n = 3$ qubits and input $|111\rangle$

Once the states have been checked, the output of our function is correct. We can conclude that that the experimentation using our QFT function has been satisfactory. It is important to notice that these experiments have been executed using a simulator. If we could use real hardware, we could gate different and wrong results because of the noise of the qubits.

6 Quantum Phase Estimation

The quantum phase Estimation is the next step towards the Harrow-Hassidim-Lloyd algorithm. This algorithm which uses the Quantum Fourier Transform is a very important algorithm that is part of the core of many quantum algorithms, not only the one that studies this thesis, but for example the Shor's algorithm [37] as well. In this section, the Quantum Phase Estimation (QPE) algorithm will be introduced and studied. An implementation using Qibo will be suggested.

6.1 Problem statement

Let U be a unitary transformation on n qubits and $|\psi\rangle$ is an eigenvector of U with eigenvalue $e^{2\pi i\phi}$ where $0 \leq \phi \leq 1$. Consider that U or $|\psi\rangle$ or $e^{2\pi i\phi}$ are not explicitly known but instead control-U^{2^j} operations can be performed where $j \in \mathbb{R} : 0 \leq j < m$. Also, assume that a single preparation of the state $|\psi\rangle$ is given. The problem is to obtain an m-bit estimator of ϕ. This m-bit string will estimate $2^m\phi$. Being a $= 2^m\phi$ we can represent ϕ using the notation $\phi = 0.j_1j_2j_3... = \frac{a}{2^m}$.

6.2 Quantum Phase Estimation Algorithm

The quantum phase estimation (QPE) algorithm solves the problem stated before with a good estimation of $|\psi\rangle$. The QPE algorithm is based on the phase kickback. The phase kickback is basically a method where the eigenvalue of a gate or transform applied to a qubit is moved to a different qubit using controlled gates [38]. Then, phase kickback is used in order to write the phase of U in the Fourier basis. The control qubits will control the control-U^{2^j} gates [40].

Consider that the phase ϕ can be represented with m bits of precision. Then our algorithm would precise of two registers: the register of qubits that will encode the phase ϕ and the register that will represent the single preparation of the state $|\psi\rangle$ [39]. So the setup of our algorithm would look like:

$$|\psi_0\rangle = |0\rangle^{\otimes m} |\psi\rangle \tag{16}$$

Next step is to apply a Hadamard gate to the first register of qubits so all the qubits are in superposition. The quantum state would change as follows:

$$|\psi_0\rangle \mapsto |\psi_1\rangle = H^{\otimes m} |0\rangle^{\otimes m} |\psi\rangle = \frac{1}{2^{\frac{m}{2}}}(|0\rangle + |1\rangle)^{\otimes m} |\psi\rangle \tag{17}$$

The next step is to apply the Control-U^{2^j} where $j \in \mathbb{R} : 0 \leq j \leq m-1$. Each qubit $j+1$ of the first register will be the control qubit for a Control-U^{2^j} that will be applied to $|\psi\rangle$. Since U is a unitary operator with eigenvector $|\psi\rangle$ such that $U|\psi\rangle = e^{2\pi i\phi}|\psi\rangle$, this means:

$$U^{2^j}|\psi\rangle = U^{2^j-1}U|\psi\rangle = U^{2^j-1}e^{2\pi i\phi}|\psi\rangle = e^{2\pi i 2^j\phi}|\psi\rangle \tag{18}$$

Then, applying the Control-U^{2^j} to $|\psi_1\rangle$ we get this state:

$$|\psi_1\rangle \mapsto |\psi_2\rangle = \frac{1}{2^{\frac{m}{2}}}\left(|0\rangle + e^{2\pi i\phi 2^{m-1}}|1\rangle\right) \otimes \cdots \otimes \left(|0\rangle + e^{2\pi i\phi 2^0}|1\rangle\right) \otimes |\psi\rangle$$

$$= \frac{1}{2^{\frac{m}{2}}} \sum_{k=0}^{2^m-1} e^{2\pi i\phi k}|k\rangle \otimes |\psi\rangle \tag{19}$$

Looking at the new state $|\psi_2\rangle$, we can notice that this is a similar expression of the Quantum Fourier Transform. In this case, $\phi = 0.a_1...a_m$. So, in order to obtain ϕ, the inverse QFT can be applied to the first register of qubits. This will produce the state $|a_1...a_m\rangle$ exactly, which will give us the value of ϕ.

The inverse QFT is very similar to the QFT. So after applying it to $|\psi_2\rangle$, the new state would look like:

$$|\psi_2\rangle \longmapsto QFT^{-1}\,|\psi_3\rangle = \frac{1}{2^m} \sum_{x=0}^{2^m-1} \sum_{k=0}^{2^m-1} e^{-\frac{2\pi ik}{2^m}(x-2^m\phi)}\,|x\rangle \otimes |\psi\rangle \tag{20}$$

Finally, it is only needed to measure the first register of m qubits. After applying the IQFT, the measured state will be:

$$|\psi_3\rangle = |2^m\phi\rangle \otimes |\psi\rangle \tag{21}$$

Since we only want to measure the first register. We would get $|2^m\phi\rangle$.

6.2.1 Probability of measuring ϕ properly

As the name suggests, the quantum phase estimation gives an accurate approximation of ϕ. In case that ϕ is a fraction of a power of two (an integer), the probability of obtaining the right ϕ is 1. However, in general ϕ does not accomplish that condition (it may not even be a rational number). In this section, we will study the probability of obtaining the best m-bit approximation of ϕ.

To do so, let $\frac{a}{2^m} = 0.a_1...a_m$ be the be best m-bit estimate of ϕ. Then, let $\phi = \frac{a}{2^m} + \delta$ where $0 < |\delta| \leq \frac{1}{2^{m+1}}$. Lets start from the application of the IQFT:

$$\frac{1}{2^m} \sum_{x=0}^{2^m-1} \sum_{k=0}^{2^m-1} e^{\frac{-2\pi ixk}{2^m}} e^{2\pi i\phi k}\,|x\rangle = \frac{1}{2^m} \sum_{x=0}^{2^m-1} \sum_{k=0}^{2^m-1} e^{\frac{-2\pi ixk}{2^m}} e^{2\pi i(\frac{a}{2^m}+\delta)k}\,|x\rangle$$

$$= \frac{1}{2^m} \sum_{x=0}^{2^m-1} \sum_{k=0}^{2^m-1} e^{\frac{-2\pi i(a-x)k}{2^m}} e^{2\pi i\delta k}\,|x\rangle \tag{22}$$

Since the coefficient of $|a_0...a_m\rangle$ is a geometric series:

$$\frac{1}{2^m} \sum_{y=0}^{2^m-1} (e^{2\pi i\delta})^y = \frac{1}{2^m} \left(\frac{1-(e^{2\pi i\delta})^{2^m}}{1-e^{2\pi i\delta}} \right) \tag{23}$$

Knowing that $|\delta| \leq \frac{1}{2^{m+1}}$, $2\pi\delta 2^m \leq \pi$. Thus, $|1 - e^2\pi i\delta 2^m| \geq \frac{2\pi\delta 2^m}{\pi/2} = 4\delta 2^m$. Also, $|1 - e^{2\pi i\delta}| \leq 2\pi\delta$. Using all this we can state that the probability of observing $a_1...a_m$ when measuring is:

$$\left| \frac{1}{2^m} \left(\frac{1-(e^{2\pi i\delta})^{2^m}}{1-e^{2\pi i\delta}} \right) \right|^2 \geq \left(\frac{1}{2^m} \left(\frac{4\delta 2^m}{2\pi\delta} \right) \right)^2 = \frac{4}{\pi^2} \tag{24}$$

6.3 Circuit for the Quantum Phase Estimation

In the previous section, the Quantum Phase Estimation algorithm has been step by step explained justifying it formally. In this section the previous steps of the algorithm will be translated to a quantum circuit. It will be explain with an example of 4 qubits in the first register and the ψ eigenvector represented with one wire.

Firstly, the circuit will start mapping the $|\psi_0\rangle$ and $|\psi_1\rangle$ states. As we can see, the transform produced is done by the application of Hadamard gates to all the qubits of the m-bits register.

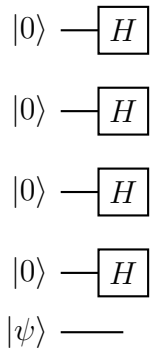


Figure 7: Application of Hadamard gates to estimation qubits in the QPE

Next step, is applying the Control-U^{2^j}. This gates take as the control qubit each of the qubits of the first register and they are applied to $|\psi\rangle$. So the circuit will be:

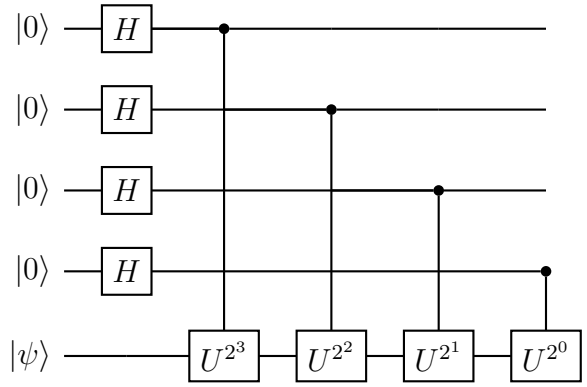


Figure 8: Application of Unitary gates U in the QPE

Finally, we only need to apply the inverse quantum Fourier Transform and measure the m-qubits register. This will give us the desired output of $|2^m\phi\rangle$. Thus, the final circuit of $m = 4$ would look like:

6.4 Complexity of the Quantum Phase Estimation

The time complexity of the Quantum Phase Estimation circuit is determined by the number of gates i.e. m control-U^{2^j} gates and $\mathcal{O}(m^2)$. This means that in general,

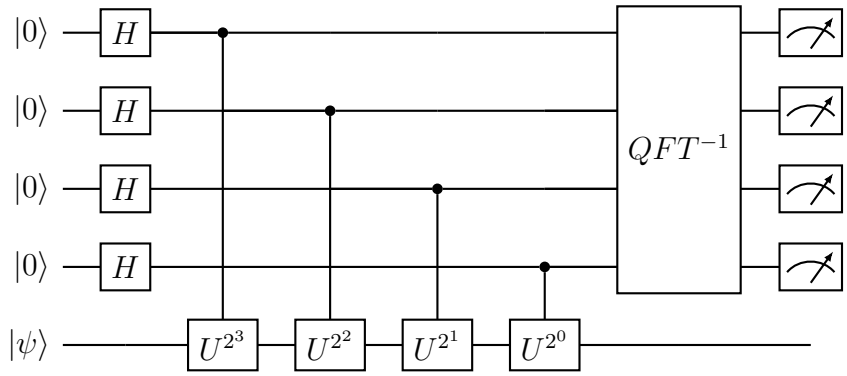


Figure 9: Application of the IQFT in the QPE

the cost of this algorithm is $\mathcal{O}(m + m^2) = \mathcal{O}(m^2)$.

Nevertheless, the complexity of the QPE can be improved. The success probability of this algorithm can be amplified to $1 - \epsilon$ for any $\epsilon > 0$ by inflating m to $m' = m + \mathcal{O}(\log(1/\epsilon))$, and rounding off the resulting m'-bit string to its most significant m bits [39].

6.5 Implementation using Qibo and experimentation

In this section, the implementation of the Quantum Phase Estimation is given using Qibo. A python function that returns a circuit will be defined. The previous Quantum Fourier Transform Function will be used in order to apply the inverse QFT to the first register of m qubits.

```
def QPE(mqubits, circuit, matrix = None, operator = None):
    qpe = Circuit(mqubits+circuit.nqubits)

    #The H gates are added to the firsts mqubits
    for q in range(0,mqubits):
        if q < mqubits: qpe.add(gates.H(q))

    #The circuit that gives the state that will be the input of the
    QPE is attached to the whole QPE circuit
    qpe.add(circuit.on_qubits(*range(mqubits,mqubits+circuit.
   nqubits)))

    #If a gate U is provided, they are set
    if (matrix is None):
      reps = 2**(mqubits-1)
      for q in range(mqubits):
          for i in range(reps):
              qpe.add(operator(*range(mqubits, circuit.nqubits+
   mqubits)).controlled_by(q))
          reps //= 2

    #If a matrix is given instead, the matrix is used as a gate and
    we add the enough gates
```

```
    else:
      reps = 2**(mqubits-1)
      for q in range(mqubits):
          for i in range(reps):
              qpe.add(gates.Unitary(matrix,*range(mqubits, circuit.
nqubits+mqubits)).controlled_by(q))
          reps //= 2


    #Apply the IQFT
    qft = QFT(mqubits)
    iqft = qft.invert()
    qpe.add(iqft.on_qubits(*range(0, mqubits)))

    return qpe
```

Code cell 12: Implementation of the QPE circuit using Qibo

This code can be found in the GitHub repository of this project[36].

In order to test the proposed function, we will try two scenarios:

- We will use the input state $|1\rangle$, and we will try to get the phase of the T-gate using three qubits of estimation. The phase of this gate is $\frac{1}{8}$ The output for this experiment is:

```
Counter({'001': 1000})
```

Code cell 13: Bash output for first test

As we can see we got the binary string 001. This is 1 in decimal numbers and 0.125 or $\frac{1}{8}$ in the notation introduced in the problem statement. We got the same output all the times that the experiment was executed.

- In the following scenario, we will give the same input and we will use the S-gate with three qubits of estimation. The phase of the S-gate is $\frac{1}{4}$. The output for this experiment is:

```
Counter({'010': 1000})
```

Code cell 14: Bash output for second test

As in the previous example, we got the correct value. 010 is 4 in decimal notation and 0.25 or $\frac{1}{4}$ in the notation introduced in the problem statement. We got the same output all the times that the experiment was executed.

It makes no sense to plot these results because we clearly see that we get the same value after one thousand executions. This is because we are using simulators and the phase of the chosen gates can be precisely estimated with three qubits. If the phase was not able to be encoded into an integer or we could use real hardware, the final results will give a different distribution giving different measured values.

7 Harrow-Hassidim-Lloyd algorithm

In this thesis, the Quantum Fourier Transform and the Quantum Phase Estimation has been studied and implemented. At this point, the background needed to start studying the main algorithm of this project is achieved. Thus, in this section the Harrow-Hassidim-Lloyd algorithm will be introduced. In order to do so, we will explain the steps that the algorithm follows, its correctness and complexity. We will also see an improved version of the algorithm.

7.1 Problem Statement

Given a $N \times N$ matrix A and a vector $\vec{b}$, we want to find a vector $\vec{x}$ such that $A\vec{x} = \vec{b}$. This problem is known as the Linear System Problem (LSP) and it has many applications. This problem can be tackled from the quantum point of view using the HHL algorithm. This algorithm is addressed to solve the Quantum Linear System Problem (QLSP). This is, given a matrix A and a $|b\rangle$, find an n-qubit state $|\tilde{x}\rangle$ such that the probabilistic distributions of $|x\rangle$ and $|\tilde{x}\rangle$ are similar (specifically, the difference should be lower than ϵ) and $Ax = b$, where A can have either real or complex entries.

7.1.1 Limitations of the HHL Algorithm

However, the HHL algorithm has some assumptions [32]:

- the state $|b\rangle$ must be loaded to the circuit efficiently i.e. a circuit that can compute $|b\rangle$ efficiently is needed in order to preserve the efficiency.
- A is a s-sparse hermitian matrix. We can adapt a matrix $C = \begin{pmatrix} 0 & A \\ A^* & 0 \end{pmatrix}$ (where A^* is the transposed conjugated matrix of A) to be hermitian if A it is not. But for simplicity we will assume that A is hermitian.
- A is invertible. This can be expressed also defining that the ratio between its largest and smallest singular value is at most some κ. The stronger assumption that κ (condition number) is small, intuitively says that A is invertible in a stable or robust way, so that small errors don't lead to massive errors in the solution vector x.

7.2 HHL Algorithm

The problem to solve is defined and the assumptions of the algorithm are stated. Now we can start studying how the algorithms works and performs in order to give as a solution for the QLSP.

7.2.1 Preparations for the algorithm

In order to start explaining the steps that the HHL algorithm follow, some preparations are needed to make our data suitable for it. Lets remind that we want to find $|x\rangle$ such that $Ax = b$ [32]. In the following list $\lambda_i \in \lambda$ are the eigenvalues of the matrix A and $|u_i\rangle$ are the eigenvectors.

- In order to load $|b\rangle$, we can state that $|b\rangle = \sum_i b_i |u_i\rangle$, where b_i is the i^{th} element of $\vec{b}$.
- In the same vein, we can load a matrix A in a quantum state because since A is hermitian, it can be expressed as $A = \sum_i \lambda_i |u_i\rangle \langle u_i|$.
- Since A is invertible and hermitian, A^{-1} can be expressed as $A^{-1} = \sum_i \frac{1}{\lambda_i} |u_i\rangle \langle u_i|$
- Regarding κ, for simplicity lets assume that the smallest singular value a_s of A is $a_s \geq 1/\kappa$ and the largest value a_l is $a_l \leq \kappa$. The smaller the condition number κ is, the better the algorithm will perform.It is assumed we know κ.
- Finally $|x\rangle$ has to be expressed using $|b\rangle$ and A^{-1} because if $Ax = b$, then $x = A^{-1}b$. Thus $|x\rangle$ can be expressed as $|x\rangle = \sum_i \frac{b_i}{\lambda_i} |u_i\rangle$

7.2.2 Workflow of the algorithm

At this point, all the background and previous steps needed for the Harrow-Hassidim-Lloyd algorithm are known and the steps that this algorithms follow can be explained following the original paper [41].

For this algorithm we'll need three qubit registers: the first will load the $|b\rangle$ state, the second will store the eigenvalues of the matrix A and the third register will be a single ancilla qubit. Thus, our input quantum state will be:

$$|\psi_0\rangle = |0\rangle_{n_b} |0\rangle_{n_l} |0\rangle \longmapsto |\psi_1\rangle = |b\rangle_{n_b} |0\rangle_{n_l} |0\rangle = \sum_i b_i |u_i\rangle_{n_b} |0\rangle_{n_l} |0\rangle \tag{25}$$

Now that our state has $|b\rangle$ loaded, Quantum phase estimation will be applied on our two registers. In order to apply QPE, as seen previously in this project, a unitary operator from which we will extract its phase it is needed. This operator U will be $U = e^{iAt}$ being A the matrix. The resulting state after applying QPE will be:

$$|\psi_1\rangle \longmapsto |\psi_2\rangle = \sum_i b_i |u_i\rangle_{n_b} |\lambda_i\rangle_{n_l} |0\rangle \tag{26}$$

Where λ_i is the i-th eigenvalue of the matrix A and $|\lambda_i\rangle_{n_l}$ is the n_l-bit representation of the respective eigenvalue. Using the QPE, we've been able to get the eigenvalues of the matrix and map them in our second register.

Then, a rotation will be applied to the ancilla qubit. This rotation will be conditioned on $|\lambda_i\rangle_{n_l}$. The resulting state after he rotation will be:

$$|\psi_2\rangle \longmapsto |\psi_3\rangle = \sum_i b_i |u_i\rangle_{n_b} |\lambda_i\rangle_{n_l} \left(\sqrt{1 - \frac{C^2}{\lambda_i^2}} |0\rangle + \frac{C}{\lambda_i} |1\rangle \right) \tag{27}$$

Actually this is not only one rotation but one for each eigenvalue λ_i. They can be applied by a Y rotation matrix where $\theta = cos^{-1} \frac{C}{\lambda_i}$. Also note that C is just a scaling factor used to make sure that all of our quantum states are normalized. $\mathcal{O}(1/\kappa)$.

Once we do this rotation, it is time to uncompute the eigenvalues register using the inverse of Quantum Phase Estimation. This will return the register to $|0\rangle$ and it is done in order to unentangle the qubits of the register that stores $|b\rangle$ and the register with the eigenvalues. This give us the following quantum state:

$$|\psi_3\rangle \longmapsto |\psi_4\rangle = \sum_i b_i |u_i\rangle_{n_b} |0\rangle_{n_l} \left(\sqrt{1 - \frac{C^2}{\lambda_i^2}} |0\rangle + \frac{C}{\lambda_i} |1\rangle \right) \tag{28}$$

After applying the inverse QPE, the tricky part comes. The extra qubit will be measured and if the measure equals 0, the circuit has to run again. On the other hand if the measure equals 1, we got the state that will give us $|x\rangle$. This is why this algorithm will have to be executed several times in order to get the probabilistic distrubution afterwards. Since we want to keep the amplitude of $|1\rangle$, we apply amplitude amplification. $\mathcal{O}(\kappa)$ will be enough repetitions and the obtained state will be:

$$|\psi_4\rangle \longmapsto |\psi_5\rangle = \left(\sqrt{\frac{1}{\sum_i C^2 |b_i|^2 / |\lambda_i|^2}} \right) \sum_i C \frac{b_i}{\lambda_i} |u_i\rangle_{n_b} |0\rangle_{n_l} \tag{29}$$

If we remember the expected $|x\rangle$ we are looking for, this is $|x\rangle \approx \sum_i \frac{b_i}{\lambda_i} |u_i\rangle$. Now if we take a look at the state we have now, we can see that they are quite similar. Our current state is component-wise proportional to the solution vector $|x\rangle$.

Finally the state that it is obtained at the end of the circuit, when the ancilla qubit collapses to $|1\rangle$, will have encoded the normalized solution of the system. This means that the probabilities of the state $|x\rangle$ will be what encodes the solution of the system of linear equations to solve.

7.3 Complexity of the algorithm

This algorithm has an exponential speedup compared to the classical alternatives. The strength of this algorithm is that it is not needed to write down all of A, $\vec{b}$ or $\vec{x}$ and it works using only $\mathcal{O} \log N$)-qubit registers. Although taking in mind the error rate ϵ and the s that determines the sparsity of A and its condition number κ, the complexity of this algorithm is $\mathcal{O}(\log(N) s^2 \kappa^2 / \epsilon)$ [41].

lets go deeper on the explanation of the complexity.

- The first step we have is loading our data, As said previously, we assume that we have an efficient unitary operator that let us to do this in $\mathcal{O}(\log N)$.
- The second step is to apply the QPE. The complexity of the QPE it has been studied earlier in this project. This step will cost $\mathcal{O}(\log(N))$ to the algorithm.
- Finally, the conditional rotation is applied to the ancilla qubit and has to be measured several times in order to get its collapse to $|1\rangle$. This step will require at most $\mathcal{O}(\kappa)$.
- Due to the conditions of the matrix A regarding κ and the s factor that determines the sparse of A, the claimed run-time proportional to $\mathcal{O}(\kappa^2 s^2)$ will be achieved [5].

All this together, give us the time complexity of $\mathcal{O}(\log(N)s^2\kappa^2/\epsilon)$.

7.3.1 Justification of the complexity

The aim of this section is to explain the complexity of the HHL algorithm but ignoring the sparsity, the conditional number and the error rate.

Starting from the QFT, in this project it is seen that the complexity is $\mathcal{O}(n^2)$ given n qubits.

Then the complexity of the QPE is marked by the QFT but accepting some error, the complexity can be reduced to $m' = m + \mathcal{O}(\log(1/\epsilon))$. In terms of n qubits, the cost of the QPE can be $\mathcal{O}(n + \log(1/\epsilon)) = \mathcal{O}(n)$ i.e. in linear time regarding the number of qubits.

Finally, knowing that N is the size of $\vec{b}$ and $\vec{x}$, assuming that $|b\rangle$ can be loaded in $\mathcal{O}(\log(N))$ steps and that the rotation can be implemented in $\mathcal{O}(l)$ being l the number of qubits in the register that encodes the eigenvalues of the matrix, the complexity of the QPE in the HHL has to be justified: since it is known that is linear in terms of the number of qubits and that $N = 2^n$:

$$\begin{aligned} N &= 2^n \\ \log(N) &= \log(2^n) \\ \log(N) &= n\log(2) \\ \mathcal{O}(\log(N)) &= \mathcal{O}(n) \end{aligned} \tag{30}$$

Thus, it can be stated that the HHL algorithm has a complexity of $\mathcal{O}(\log(N))$ if the parameters we are ignoring are optimal.

7.4 Circuit of the HHL algorithm

At this point of the project, we know how the algorithm works and it is complexity. So now a quantum circuit can be designed in order to implement the HHL algorithm. This circuit will be designed using the circuits previously seen in this project.

In order to build the circuit, the steps that the algorithm follows will be revisited and adapted to a circuit. Starting by defining the registers and the loading of $|b\rangle$. The following is the circuit that represents the state $|\psi_1\rangle$:

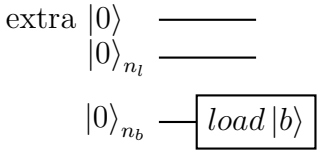


Figure 10: $|b\rangle$ is loaded in the HHL circuit

Then, Quantum Phase Estimation is applied to both registers. The following is the circuit that represents the state $|\psi_2\rangle$:

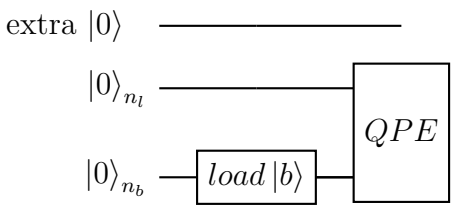


Figure 11: Application of QPE in HHL circuit

Once we have the eigenvalues of A mapped in the second register, we apply the conditional rotation to the ancilla qubit. The following is the circuit that represents the state $|\psi_3\rangle$:

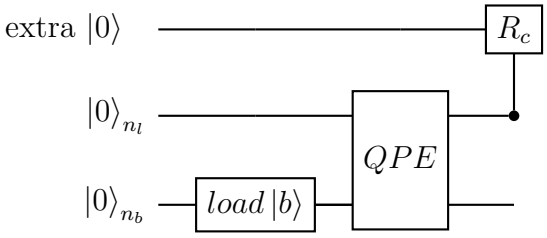


Figure 12: Application of the conditional rotation in the HHL circuit

Now, the inverse of the quantum phase estimation has to be applied to unentangle both registers. The following is the circuit that represents the state $|\psi_4\rangle$:

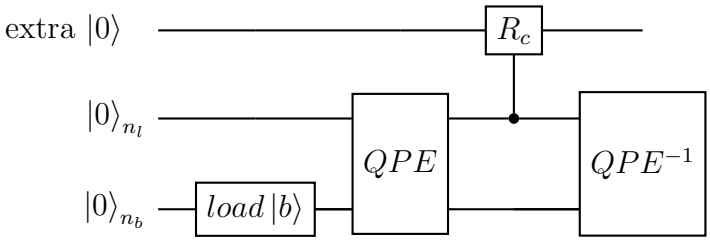


Figure 13: Application of inverse QPE in HHL circuit

Finally, a measurement of the ancilla qubit is needed to know if the output will be the desired one. The following is the circuit that represents the state $|\psi_5\rangle$:

7.4.1 Detailed circuit

A more detailed circuit can be shown. In this case will be the whole circuit at once but being able to see through the QPE operator.

As we can see, in this representation of the circuit, all the elements of our problem are set. In this way, we can see in a graphic composition how all the elements are used and mapped in the workflow of the algorithm and in the circuit.

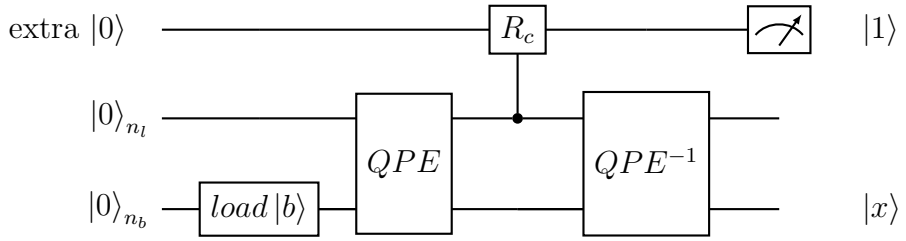


Figure 14: Final HHL circuit

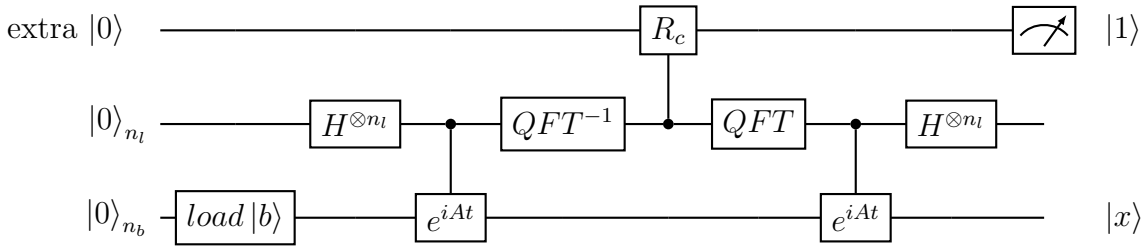


Figure 15: Final detailed HHL circuit

7.5 Implementation of the HHL algorithm

Currently there is not an efficient general circuit implementation for this quantum algorithm. Due to the time and dimensions of this project, it can not be studied a way to implement efficiently the HHL algorithm. Specifically, the assumption of finding a circuit that loads $|b\rangle$ efficiently and the implementation of the conditional rotation are still studied. There are different papers that give possible implementations but they are still not trivial and not easy to implement generically in the algorithm.

This is why in this section it will be shown and explained the implementation of an example for 4 qubits. This example will be followed through the following paper [42].

In this example, we will solve a system of linear equations of size N = 2. The Matrix A and the vector b will be:

$$A = \begin{pmatrix} 1 & -\frac{1}{3} \\ -\frac{1}{3} & 1 \end{pmatrix}$$

$$\vec{b} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

We want to find the vector $\vec{x}$ such $A\vec{x} = \vec{b}$. The solution is:

$$\vec{x} = \begin{pmatrix} \frac{3}{8} \\ \frac{9}{8} \end{pmatrix}$$

Whereby the ratio of $|x_0|^2$ and $|x_1|^2$ is $1:9$. The goal is to design a quantum circuit using Qibo[12] in order to get a quantum state x which probabilities are similar to the ratio of the elements of $\vec{X}$ using the HHL algorithm.

First thing we have to do is to implement the U operator that will implement the hamiltonian that represents A. The following function will give us this operator.

```
def transform_to_hamiltonian(matrix, t):
    # Step 1: Get the eigenvalues and eigenvectors of the matrix
    eigenvalues, eigenvectors = np.linalg.eigh(matrix)

    # Step 2: Construct the diagonal matrix
    E = np.diag(eigenvalues)

    # Step 3: Construct the transformation matrix
    V = eigenvectors

    # Step 4: Exponential of A
    expE = np.exp(1j * E * t)

    for i in range(len(expE)):
      for j in range(len(expE)):
        if D[i][j] == 0: expE[i][j] = 0

    # Step 5: Construct the Hamiltonian operator
    H = V @ expE @ np.conjugate(V).T
    return  H
```

In the first step is to get the eigenvectors and eigenvalues of A. In order to do so, we use the library Numpy. This will store the eigenvalues $\lambda_0 = \frac{2}{3}$ and $\lambda_1 = \frac{4}{3}$ and the eigenvectors $\vec{u_0} = \begin{pmatrix} \frac{-1}{\sqrt{2}} \\ \frac{-1}{\sqrt{2}} \end{pmatrix}$ and $\vec{u_1} = \begin{pmatrix} \frac{-1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix}$.

The following steps are aimed to build the hamiltonian using the information we already got. In order to get the desired operator, it is needed a suitable t (remember that the operator will be $U = e^{iAt}$). Since we will have two qubits dedicated to store the encoded eigenvalues, it is desired to choose a t that let us encode this eigenvalues such that $\tilde{\lambda_0} = 01$ and $\tilde{\lambda_0} = 10$. Thus, the chosen t will be using the following formula: $\tilde{\lambda_j} = M\lambda_j t/2\pi$ where $M = 2^m$ (m being the number of qubits that encode the eigenvalues, in this case $m = 2$). Solving the equation we get that $t = \frac{3\pi}{4}$.

The step 2 will store the diagonal matrix of the eigenvalues E, where the eigenvalues of A will be in the diagonal of a matrix and the rest of elements will be 0. The third step is to create a matrix V with our two eigenvectors and then the fourth step will return a diagonal matrix $expE$ where the diagonal is $e^i\lambda_j t$. In our case, matrix will be $\exp E = \begin{pmatrix} i & 0 \\ 0 & -1 \end{pmatrix}$.

To obtain U in the original basis, we apply the last step multiplying V by $expE$ by V^*. After following these steps, we got the operator U that will be used in the Quantum Phase Estimation. This operator will be $E = \frac{1}{2}\begin{pmatrix} -1+i & 1+i \\ 1+i & -1+i \end{pmatrix}$.

Having all this information, now we can implement this example of HHL algo-

rithm. The following code is the implementation.

```
def HHL(b_circuit, A, l, C):

    #Step 1: Get the hamiltonian operator U using the previous
   function.
    U = transform_to_hamiltonian(A,(3*np.pi/4))

    #Step 2: Apply QPE
    qpe = QPE(l, b_circuit, U)

    hhl = Circuit(qpe.nqubits+1)
    hhl.add(qpe.on_qubits(*range(1,hhl.nqubits)))

    #Step 3: Apply conditional rotation
    theta = [math.pi, math.pi/3]
    for q in range(l):
        hhl.add(gates.CRY(q+1,0,theta[q]))

    c = Circuit(1)

    #Step 3: Apply Inverse QPE
    iqpe = (QPE(l, c, U)).invert()
    hhl.add(iqpe.on_qubits(*range(1,hhl.nqubits)))

    return hhl
```

The most important part to look at in this function is the third step which applies the conditional rotation to an extra qubit using the qubit register that encodes the eigenvectors. As seen in the section where the correctness of the algorithm is explained, the angle of rotation θ rotation is $2arcsin(\frac{C}{\lambda})$. In this case, $C = 1$ and λ will be our $\tilde{\lambda}$. Thus $\theta = 2arcsin(\frac{1}{\tilde{\lambda}_j})$. This is $\theta_0 = \pi$ and $\theta_0 = \frac{\pi}{3}$.

Applying QPE before and after this rotation using the operator U obtained previously, the construction of the HHL circuit is done.

The next code is a main that calls the HHL function using the matrix and vector defined at the beginning of the section.

```

def main():
    #A is the matrix from the equation Ax=b
    A = np.array([[1, (-1/3)],
                  [(-1/3), 1]])

    #b is a circuit that encodes the vector b
    b = Circuit(1)
    b.add(gates.X(0))

    #HHL function returns the HHL Circuit
    hhl = HHL(b, A, 2, 1)

    #Measurements are added to the circuit
    hhl.add(gates.M(0))
    hhl.add(gates.M(3))

    #The circuit is simulated
    simulated_final_state = hhl(nshots=1000)
```

```
20
21      #Print the cicuit
22      print(hhl.draw())
23
24      #Print the simulated final state of the circuit
25      print(simulated_final_state)
26
27      #Print the probabilites of the esimulated final state of the
        circuit
28      probabilities = simulated_final_state.probabilities
29      print(probabilities(qubits=[3,0]))
```

This code will be available in Github [36].

Now we can test this example and see the results. The output of this code is:

```
1 (0.43301+0j)|0000> + (0.43301+0j)|0001> + (-0.25-0j)|1000> +
     (0.75-0j)|1001>
2 tf.Tensor([0.1875 0.0625 0.1875 0.5625])
```

Code cell 15: Bash output after executing the HHL example

The probabilities that we got are the probabilities after reading the extra qubit (the one where we apply the conditional rotation) and the qubit that give us the result of the system. In this case, we can plot the results:

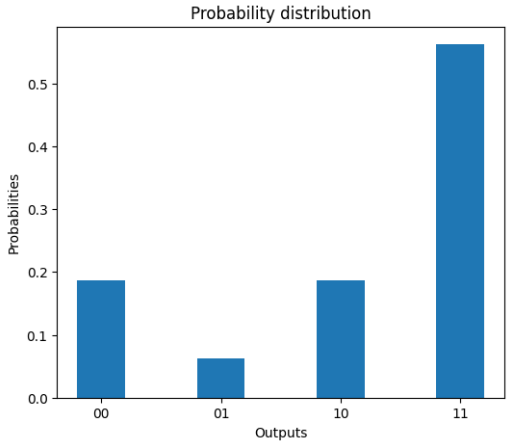


Figure 16: Bar plot of the probability distribution of the measurements of the HHL example circuit. Original creation

We have to check the cases where the extra qubit collapses to 1. These are the 01 and 11 bars. In the first case the probability is 0.0625 and in the second case is 0.5625. The ratio of these two probabilities is $0.0625 : 0.5625 = 1 : 9$. The ratio we got is the same ratio of $|x_0|^2$ and $|x_1|^2$ meaning that we got the correct final state and therefore, the solution to the system of linear equations. It is important to remember that this is just an example. The implementation of this circuit is very specific.

As said in the different experiments of this thesis, the results we got are good because we are using the simulator that Qibo provides. We could not test the implementations in real hardware, which means that we could not study the performance

of these algorithms with noise. However, with simulators we can have an accurate test of running the algorithms without noise which is also interesting and useful.

8 Conclusions of the Project

8.1 Conclusions

In this project, quantum computing has been studied and understood through the review of an specific algorithm: the Harrow-Hassidim-Lloyd algorithm. This algorithm is an example of the power and utility of quantum computing showing an exponential speedup against classical alternatives and justifies the efforts done by different researchers in this field. Quantum computing merges quantum mechanics and computer science giving the option to tackle different problems more efficiently than classical computers.

The HHL algorithm is a novel algorithm that solves the Quantum Linear Solver Problem. Although, it has different assumptions to keep in mind and it is very theoretical (we can not find an specific agreed implementation that gives the theoretical time complexity), this algorithm is a gate to new applications and fields of study regarding quantum computing, for example for quantum machine learning. Specifically, the use of the HHL algorithm could be used in the implementation of the quantum support vector machines (QSVM) [45]

Throughout this report, we have detailed the study of the whole algorithm from the single gates and core algorithms used to the actual HHL algorithm including the implementation of the algorithms needed and the mathematical justifications in order to understand the workflow and performance of the HHL algorithm.

This thesis compile the different tasks done:

- Understanding of quantum circuits and quantum computing notation as well as the basic mathematical procedures used in both.
- Review and understanding of different papers (original papers and enhancement proposals) about the quantum algorithms explained in this project.
- review of the mathematical procedures required to justify the correctness.
- Study of time complexity of the quantum algorithms.
- The general implementation for the Quantum Fourier Transform and the Quantum Phase Estimation using Qibo.
- The implementation of an example of the HHL circuit using Qibo explaining it step by step.
- The study of the results of the circuits using the simulator backend that provides Qibo.

This project almost satisfies all the objectives set at the beginning. They were the full study of this algorithm theoretically and its implementation including the implementation of the procedures needed for it (the Quantum Fourier Transform and Quantum Phase estimation). Although it could not be possible to do a general implementation of the HHL circuit, a given example is implemented and explained.

Meanwhile the QPE implementation will be adapted in order to create a class available in Qibo and contribute to this open source framework.

The general implementation of the HHL algorithm was tried but it was not possible to find any reference giving information about how to general implement the different parts of this circuit. However two papers were found [43] [44] proposing some solid implementations. Unfortunately, we could not study them due time and lack of knowledge and experience in quantum computing and mechanics.

It was not possible to run the algorithms on the first Spanish quantum computer installed in the Barcelona Supercomputing Center because the installation was delayed. This caused that we did not have enough time left to wait until the computer was ready to use. However, the proposed code in this report will be available to use it afterwards the deadline of this project if the BSC is interested.

Finally I would like to exalt the role of the computer scientist in quantum computing. This field has been seen as a physics topic during during its study until today the computer science got more relevance in the field. This is because both classical and quantum software, experts in complexity and information theory and theoretical computer science are needed in order to make this field grow. For that reason, this computer science bachelor's final project (TFG) is studying a quantum algorithm. Because computer scientists can play an important role in this field together with physicists, mathematicians and different engineers. After all, quantum computing is a new computing paradigm. Quantum computing is computing.

8.2 Future Work

In order to keep working on this project, the main point would be to study how to implement efficiently the HHL algorithm. This means finding a general and efficient implementation of the conditioned rotation and without knowing the eigenvalues of the matrix A beforehand, a general and efficient implementation for loading the state $|b\rangle$ and for getting the operator that expresses the hamiltonian matrix of the matrix A.

Following the previous points, the study and testing of these implementations will be key in order to find more efficient methods and this methods will also be useful for other known and future algorithms. As well as the study of applications of the HHL algorithm as a subroutine for bigger algorithms, for example in quantum machine learning.

It is also relevant the study of the Variational Quantum Linear Solver (VQLS) [46]. VQLS is a NISQ algorithm that solves the QLSP problem. This algorithm is based on the HHL algorithms in order to be able to use NISQ quantum computers and get reliable solutions. Its study should be considered in future work in order to understand better the NISQ algorithms and to compare its performance with the HHL algorithm. In the same vein, it may be interesting to study hybrid quantum-classical algorithms in order to solve the QLSP problem.

9 References

[1] De Mol, Liesbeth. Turing Machines. The Stanford Encyclopedia of Philosophy, edited by Edward N. Zalta, Winter 2021, Metaphysics Research Lab, Stanford University, 2021. Stanford Encyclopedia of Philosophy, https://plato.stanford.edu/archives/win2021/entriesuring-machine/.

[2] John von Neumann: The Father of the Modern Computer. Devlin's Angle, https://www.maa.org/external_archive/devlin/devlin_12_03.html.

[3] Gribbin, John. Erwin Schrodinger and the quantum revolution. John Wiley Sons, Inc, 2012.

[4] Feynman, Richard P. Simulating Physics with Computers. International Journal of Theoretical Physics, vol. 21, n.º 6-7, june of 1982, pp. 467-88. DOI.org (Crossref), https://doi.org/10.1007/BF02650179.

[5] Quantum Algorithm for Linear Systems of Equations. Wikipedia, 2023. Wikipedia, shorturl.at/dtUWX.

[6] Nielsen, Michael A., y Isaac L. Chuang. Quantum computation and quantum information. 10th anniversary ed, Cambridge University Press, 2010.

[7] Equation. Wikipedia, 2023. Wikipedia, shorturl.at/JMSX6.

[8] System of Linear Equations. Wikipedia, 30 de enero de 2023. Wikipedia, shorturl.at/hmoIX.

[9] JDelgado's Website. https://www.cs.upc.edu/ jdelgado/.

[10] ALBA CERVERA LIERTA Position: SENIOR RESEARCH ENGINEER Barcelona Supercomputing Center-Centro Nacional de Supercomputación CASE-Quantic email: alba [dot] cervera [at] bsc [dot], et al. People. BSC-CNS, https://www.bsc.es/cervera-lierta-alba.

[11] D3veloperSCS_qu4ntum. quantumspain. Quantum Spain, https://quantumspain-project.es/.

[12] website of the QIBO Framework, https://qibo.science/

[13] Gaussian Elimination. Wikipedia. Wikipedia, shorturl.at/oAQ18.

[14] Doolittle Algorithm: LU Decomposition. GeeksforGeeks, 15th of October 2017, https://www.geeksforgeeks.org/doolittle-algorithm-lu-decomposition/.

[15] Liu, Xiaonan, et al. Survey on the Improvement and Application of HHL Algorithm. Journal of Physics: Conference Series, vol. 2333, n.º 1, August 2022, p. 012023. DOI.org (Crossref), https://doi.org/10.1088/1742-6596/2333/1/012023.

[16] Basermann, A., et al. Parallel Iterative Solvers for Sparse Linear Systems in Circuit Simulation. Future Generation Computer Systems, vol. 21, n.º 8, October 2005, pp. 1275-84. DOI.org (Crossref), https://doi.org/10.1016/j.future.2004.09.007.

[17] Baraniuk, Richard G. Compressive Sensing. IEEE SIGNAL PROCESSING MAGAZINE, vol. 118, July 2007.

[18] Overleaf, Online LaTeX Editor. https://www.overleaf.com

[19] TeamGantt: Online Gantt Chart Maker Software - Free Forever. https://www.teamgantt.com

[20] Informe Empleo Informática Marzo 2023 - tecnoempleo. https://www.tecnoempleo.com/informe-empleo-informatica.php.

[21] Precio de la tarifa de luz por horas HOY — Consulta ahora. https://tarifaluzhora.es/.

[22] What is Quantum Computing? - AWS. Amazon Web Services, Inc., https://aws.amazon.com/what-is/quantum-computing/?nc1=h_ls.

[23] The Quantum Insider. Quantum Technology Investment Update 2022 review. febrero de 2023, https://thequantuminsider.com/wp-content/uploads/2023/02/Quantum-Technology-Investor-Update_vFF.pdf.

[24] Ali, Shaukat, et al. When Software Engineering Meets Quantum Computing. Communications of the ACM, vol. 65, n.o 4, April 2022, pp. 84-88. DOI.org (Crossref), https://doi.org/10.1145/3512340.

[25] Lau, Jonathan Wei Zhong, et al. NISQ Computing: Where Are We and Where Do We Go? AAPPS Bulletin, vol. 32, n.o 1, September 2022, p. 27. Springer Link, https://doi.org/10.1007/s43673-022-00058-z.

[26] Stephen Jordan. Quantum Algorithm Zoo, https://quantumalgorithmzoo.org/.

[27] Maninder. Quantum Initiatives Worldwide - Update 2023. Qureca, 15^{th} of May 2023, https://qureca.com/quantum-initiatives-worldwide-update-2023/.

[28] Record investments in quantum technology — McKinsey. 24^{th} of Aprif 2023. https://www.mckinsey.com/capabilities/mckinsey-digital/our-insights/quantum-technology-sees-record-investments-progress-on-talent-gap

[29] Bloch Sphere. Wikipedia, 14 de junio de 2023. Wikipedia, https://en.wikipedia.org/w/index.php?title=Bloch_sphereoldid=1160187164.

[30] IBM Unveils 400 Qubit-Plus Quantum Processor and Next-Generation IBM Quantum System Two. IBM Newsroom, https://newsroom.ibm.com/2022-11-09-IBM-Unveils-400-Qubit-Plus-Quantum-Processor-and-Next-Generation-IBM-Quantum-System-Two.

[31] Gibney, Elizabeth. Hello Quantum World! Google Publishes Landmark Quantum Supremacy Claim. Nature, vol. 574, n.o 7779, octubre de 2019, pp. 461-62. www.nature.com, https://doi.org/10.1038/d41586-019-03213-z.

[32] De Wolf, Ronald. Quantum Computing: Lecture Notes. 16^{th} of January 2023, https://homepages.cwi.nl/ rdewolf/qcnotes.pdf.

[33] Fourier Transform. Wikipedia, 9 de mayo de 2023. Wikipedia, https://en.wikipedia.org/w/index.php?title=Fourier_transformoldid=1153987231.

[34] But what is the Fourier Transform? A visual introduction. www.youtube.com, https://www.youtube.com/watch?v=spUNpyF58BY.

[35] Quantum Fourier Transform. https://learn.qiskit.org.

[36] Alemany Orfila, Alexandre. Thesis Repository - HHL algorithm. Github, https://github.com/alexao8/TFG-HHL_algorithm.

[37] Shor, Peter W. Polynomial-Time Algorithms for Prime Factorization and Discrete logarithms on a Quantum Computer. SIAM Journal on Computing, vol. 26, n.o 5, October 1997, pp. 1484-509. arXiv.org, https://doi.org/10.1137/S0097539795293172.

[38] Phase Kickback. https://learn.qiskit.org

[39] Cleve, Richard, et al. Quantum Algorithms Revisited. Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences, vol. 454, n.o 1969, January 1998, pp. 339-54. arXiv.org, https://doi.org/10.1098/rspa.1998.0164.

[40] Quantum Phase Estimation. https://learn.qiskit.org

[41] Harrow, Aram W., et al. Quantum algorithm for solving linear systems of equations. Physical Review Letters, vol. 103, n.o 15, octubre de 2009, p. 150502. arXiv.org, https://doi.org/10.1103/PhysRevLett.103.150502.

[42] Morrell Jr, Hector Jose, et al. Step-by-Step HHL Algorithm Walkthrough to Enhance the Understanding of Critical Quantum Computing Concepts. arXiv, 24 de marzo de 2023. arXiv.org, http://arxiv.org/abs/2108.09004.

[43] Vazquez, Almudena Carrera, et al. Enhancing the Quantum Linear Systems Algorithm using Richardson Extrapolation. ACM Transactions on Quantum Computing, vol. 3, n.o 1, marzo de 2022, pp. 1-37. arXiv.org, https://doi.org/10.1145/3490631.

[44] Yan, Shilu, et al. Module for arbitrary controlled rotation in gate-based quantum algorithms. arXiv, 16 de julio de 2021. arXiv.org, http://arxiv.org/abs/2107.08168.

[45] Zhang, Yao, y Qiang Ni. Recent Advances in Quantum Machine Learning. https://eprints.lancs.ac.uk/id/eprint/154554/1/QML_survey.pdf.

[46] Bravo-Prieto, Carlos, et al. Variational Quantum Linear Solver. arXiv, 2^{nd} of June. arXiv.org, http://arxiv.org/abs/1909.05820.

Appendices

Appendix A

Project Management: Introduction and Scope

1 Stakeholders

The main stakeholder of this project is the student, in charge of its research, study, development and implementation that will have the roll of main researcher. The director and co-director will give assistance to the project by follow-up biweekly meetings and giving academical, material and moral support.

Since this is a research project that is will not present a final product ready to use, to identify the the final beneficiary can be difficult. Nevertheless, this thesis can be helpful for anyone that wants to learn about quantum computing without an strong background as well as for other researchers or companies that think that the use of the presented algorithm can be useful for their research or product. Solving a linear equations system is not naive since this can be used for speedup machine learning methods such as neural networks.

Finally, the society in general will be able to get profit of it since the applications of the HHL algorithm can indirectly arrive to the devices and applications we use. Even though, as said before, the main beneficiary of this project are researchers and companies that think this method can give a better performance to their work.

2 Justification

Quantum computing is an emerging field for research and business environments. Regarding this, the main reason for undertaking this project is the willing to learn in quantum computing and being able to vindicate the role of the computer scientist in this field. Besides this, the reasons for this thesis are:

- This thesis can help to computer science students to introduce themselves in quantum computing.
- HHL algorithm is one of the first quantum algorithms that shows an exponential speedup and has a clear application. This study can help to better understand the workflow of the algorithm from scratch.
- Qibo is an open-source framework for programming quantum computers [12]. Specifically this will be the framework that will use the first Spanish quantum computer, so to learn about this framework and document this project using it will be useful for future research.
- The implementation of the algorithm in QIBO will be able to be a contribution to the open-source framework that does not have the HHL algorithm implemented as a function.

3 Equations and systems of equations

In mathematics, an equation [7] is a formula that expresses the equality of two expressions. This expressions can have variables. To solve an equation means to find the value of the values in order to satisfy the equality of the equation. For example:

$$2x - 5 = x + 3$$

Where the value of x is 8. Moreover, this equation is a linear equation because it can be rewrote in the form of $a_1x_1 + a_2x_2 + ... + a_nx_n + b = 0$. Where x_i is the i^{th} variable of n variables, a_i is the i^{th} constant of n constants and b is another constant.

A system of equations is a set of equations with the same number of variables [8]. Being a system means that all equations have to be considered collectively, not individually i.e. the value of the variables have to satisfy all equations. Specifically, this project focuses on systems of linear equations. For example:

$$\begin{cases} 3x + 5y + z = 0 \\ 7x - 2y + 4z = 0 \\ -6x + 3y + 2z = 0 \end{cases}$$

The solution for this system is $(x, y, z) = (0, 0, 0)$.

Now that a system of equations is defined, for this project is useful to know that we can represent a system of linear equations using matrices and vectors. In this case the equation would look like $A\vec{x} = \vec{b}$ being A a matrix, $\vec{x}$ a vector with

the variables and $\vec{b}$ the solution of the system. So taking the previous example, we would have:

$$\begin{bmatrix} 3 & 5 & 1 \\ 7 & -2 & 4 \\ -6 & 3 & 2 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{A.1}$$

4 Classical algorithms for solving systems of linear equations

The main goal of this project is to study the application of quantum computing to solve systems of linear equations. However, classical options have been already developed. In this section two main classical algorithms are introduced: Gaussian elimination and LU decomposition.

4.1 Gaussian elimination

Gaussian elimination is an known algorithm for solving systems of linear equations [13]. This algorithm consists of a sequence of operations performed on the corresponding matrix of coefficients. It is also used to compute the rank of a matrix, the determinant of a square matrix, and the inverse of an invertible matrix.

To perform row reduction on a matrix, one uses a sequence of elementary row operations to modify the matrix until the lower left-hand corner of the matrix is filled with zeros, as much as possible. There are three types of elementary row operations: Swapping two rows, multiplying a row by a nonzero number and adding a multiple of one row to another row. Using these operations, a matrix can always be transformed into an upper triangular matrix, and in fact one that is in row echelon form. Once all of the leading coefficients (the leftmost nonzero entry in each row) are 1, and every column containing a leading coefficient has zeros elsewhere, the matrix is said to be in reduced row echelon form. This final form is unique; in other words, it is independent of the sequence of row operations used.

4.2 LU decomposition

In numerical analysis and linear algebra, lower–upper (LU) decomposition or factorization factors a matrix as the product of a lower triangular matrix and an upper triangular matrix [14]. Computers usually solve square systems of linear equations using the LU decomposition, and it is also a key step when inverting a matrix, or computing the determinant of a matrix. LU decomposition can be viewed as the matrix form of Gaussian elimination.

4.3 Applications

Solvers for systems of linear equations are widely used in different fields. On of these is machine learning. This kind of algorithms are found in different regression and classification models in machine learning [15]. Due to the size of the datapoints to treat can be significant, the study of the implementation of the HHL algorithm into

new machine learning algorithms can drive into an exponential speedup for training models.

This algorithms can be applied to circuit simulations too [16]. The study of the simulation of large circuits has the objective to achieve an improvement of the time execution. To do so, it is tried to build distributed and parallel algorithms.

Another field is the signal compressing. Linear algebra is used for signal compressing [17]. This problem is important for, for example, storing or transmitting photos, videos or audios. Other applications in this field can be imaging systems like medical scanners or radars and high-speed analog-to-digital converters.

However, this project will not only talk about the final applications of the HHL algorithm but will try to show the approach and advantage of using quantum computing and the different subroutines needed for the principal algorithm. Quantum computing will be useful for cryptography, the health sector, financial issues, simulation... And a lot of the algorithms needed for these applications use the subroutines that will be presented in this project like the quantum Fourier transform.

5 Scope

Once the problem is specified, since the available time and resources are limited, it is necessary and important to define the scope of this project as well as the requirements that can be needed during this time and the obstacles and risks that can be found.

5.1 Main Objective

The main objective of this project is the study of the Harrow-Hassidim-Lloyd algorithm. This include the correctness, complexity of the algorithm, the comparison with the classical alternatives and the implementation using QIBO.

5.1.1 Sub-objectives

Building upon the main objective, there are some sub-objectives that are the following ones:

1. Learning and delving deeper into the field of quantum computing which is an emerging field that can reach a huge impact in society.
2. Learning about quantum information theory and quantum complexity.
3. Contributing to an open-source project like QIBO, which involves implementing the HHL algorithm and creating a class in the framework. This class will allow users to run the algorithm without needing to program it themselves.

5.2 Requirements

To ensure the proper development of the project, it is proper to define the following requirements:

5.2.1 Functional requirements

The Harrow-Hassidim-Lloyd algorithm uses different subroutines. This subroutines will be studied from scratch. It is primordial to understand the subroutine (what it does, its complexity and correctness) to start working on the next subroutine that uses the previous one.

Besides that, in order to contribute to QIBO, it is needed to implement a functional class that defines a generic HHL algorithm capable to define a quantum circuit for any number of qubits.

5.2.2 Non-functional requirements

This project should work as a guide to understand the Harrow-Hassidim-Lloyd algorithm and quantum computing. To achieve that, this thesis must proceed in incremental steps from 0 and from the simplest concepts to the development of a complex quantum algorithm. This would make this project to be an starting point for researchers and students.

5.3 Obstacles

When starting a project, it is important to think and keep in mind possible obstacles that can appear during the development of this project. The considered obstacles are the following:

- The main obstacle is that it is needed a familiarization with the topic. Quantum mechanics and mathematics background are essential. This will cause that the firsts weeks will be aimed to get the enough background to keep the track of the project.
- Quantum computing is an emerging technology with a lot of expectation and scepticism at the same time. This project should show the real power of quantum computing and the approach that can be obtained and used by computer scientists.
- At the same time, there are not full tolerant quantum computer i.e. the actual systems have errors. This can affect to the results of the execution of the algorithm.

5.4 Risks

As well as the obstacles, there are some risks that may have to be faced:

- It is possible that the Spanish quantum computer is not installed on time. This will cause the use of simulators or other systems, for example, from IBM.
- It is possible that the process to develop and understand one of the subroutines needed for the final algorithm takes longer than expected.
- To make this project too complex omitting steps or not being clear enough. This would cause that the project would not be able to be used as a guide for interested people that ones to get introduced in quantum computing.

- If somebody else (a company or an individual) is working on the implementation of the HHL algorithm for the QIBO framework.

6 Methodology and rigour

In order to achieve the goal of this project, it is important to define a strategy. We think that the best methodology to address the problem is through a modular approach. The quantum algorithm can be divided by modules that will be studied and implemented. Once the module is fully understood, the problem will be scaled to the next module.

Besides that, the director, co-director and student will meet every two weeks. The student will update the status of the project and will address questions to the supervisors. The director and co-director will give feedback of the work done during the two previous weeks. The supervisors will have access to the memory and code of this project at anytime using a shared overleaf document and a GitHub repository.

6.0.1 Tools and resources

In order to keep working properly in this project, a computer with access to internet and some software installed will be needed. The software needed is python, for programming; Visual Studio Code, as the IDE that will be used for programming; Jupyter Notebook, to create and run the algorithm step by step; QIBO, the open source framework for programming quantum computers that will be used in this project, and LaTeXand overleaf to write the thesis and let the co-directors to review the text.

In order to test the algorithm, the quantum computer of the Barcelona Supercomputer Center could be used if it is installed before the deadline of this project arrives. If not, simulators of quantum computers provided by QIBO will be used.

Appendix B

Project Management: Project planning

Previously, we defined the objectives of this project. In order to achieve them, a time planning will be defined specifying the different types of tasks that are needed to complete this project.

It is important to state that this is a research project. This means that this planning may suffer changes depending on results, obstacles... and at the same time, it is important to keep in mind that this is a research project. This means that the planning can be change if new tasks or goals are defined in order to achieve the main objective. To go back some steps it may also be needed if once a goal is achieved it is considered that it can improve or it is not clear enough (e.g. mathematically)

This project started the 20^{th} of February and the end date will depend on the date of the presentation of the thesis to the tribunal. This can be in between the 26^{th} of June and the 31^{st} of June. The total estimated time of dedication is about 540 hours. It will be distributed in 4 hours per day on average.

0.1 Tasks description

In this section, the tasks to be done will be defined. As we said, they can suffer changes during the development of the project. However, it is a good practice to define the tasks previously and arrange the changes if needed afterwards during the development of the project.

This project has three main parts: the contextualization and familiarization of quantum computing (I), the study of the state of the art of the proposed quantum algorithm and its implementation (Q) and the study of the classical alternatives and their implementation (C). At the same time, during the execution of these, the documentation of the project (D), the management of the project (MP) and there will be biweekly meetings with the co-directors of the projects (M). Finally the conclusions will be elaborated and the defense of the thesis will be presented (F).

0.2 I - Initial Steps and Familiarization of Quantum Computing

0.2.1 I1 - Understanding the Qubits and quantum states

This task is about doing research and getting start in the quantum computing fundamentals. This includes the study of quantum states, its relation with the qubits and the way to operate them using quantum gates. This task will cost around 20 hours.

0.2.2 I2 - Familiarization of quantum algorithms

This task is the following step of the previous one. The goal is to understand how quantum algorithms are designed and how do they perform on a quantum computer. Some simple quantum algorithms, like the Deutsch-Jozsa algorithm, will be seen in order to understand how do they work. This task will cost around 16 hours.

0.2.3 I3 - Familiarization of quantum complexity

This task is about understanding and learning how to interpret the complexity of the quantum algorithms. It will be important to do some research about circuit complexity for classical algorithms. These two parts will help for understanding how the complexity is measured in quantum algorithms, the steps that a quantum algorithm needs to perform its purpose and the speedup of the quantum algorithms against the classical ones. This task will cost around 28 hours.

0.3 Q - Study of the Quantum Algorithm and its Implementation

0.3.1 Q1 - Study and development of Quantum Fourier Transform

This task implies the starting point of the understanding and development of the HHL algorithm. In this task it will be needed to study the what the Quantum Fourier Transform (QFT) does, what is the difference between the QFT and the classical Fourier Transform, understand it is complexity and correctness and to implement it in a quantum circuit using Qibo. This task will cost around 32 hours.

0.3.2 Q2 - Study and development of Quantum phase Estimation

This task implies to understand the Quantum Phase Estimation (QPE) that uses the QFT. In this task it will be needed to study the what the QPE does and why is it needed, understand it is complexity and correctness and to implement it in a quantum circuit using Qibo. This task will cost around 32 hours.

0.3.3 Q3 - Study of the HHL algorithm

This task implies using the previous implementation to develop the HHL algorithm. This is understanding how this algorithm works, it is complexity and correctness. It will be important to study special cases of this algorithm and possible different implementations. This task will cost around 48 hours.

0.3.4 Q4 - Implementation of the HHL algorithm

The goal of this task is the implementation of the quantum algorithm in Qibo. First of all, the implementation will be for specific cases of 2, 3 and 4 qubits. Then, a generic implementation will be developed. Finally, once we achieve it, it will be adapted to work as a new class for the open source framework. This task will cost 20 hours.

0.3.5 Q5 - Test of the implementation

In this task, we will test the implementations done previously. This task will occur interspersed with Q4. This task will cost 20 hours.

0.4 C - Study of the Classical Alternatives and its Implementation

0.4.1 C1 - Research of classical algorithms

This task cover the research of classical algorithms that can solve systems of linear equations. There will be two types of algorithms: the algorithms that will be implemented and the research of papers that present new algorithms that speedup the actual ones (this may use techniques like parallelization that we can not implement in this project). This task will cost around 12 hours.

0.4.2 C2 - Study and implementation

The goal of this task is to study the correctness and complexity of the classical algorithms and it is implementation in python since this will be the programming language that uses Qibo. This task will cost around 32 hours.

0.4.3 C3 - Test of the implementations

In this task, we will test the implementations done previously. This task will occur interspersed with C2. This task will cost 20 hours.

0.4.4 C4 - Comparison with the quantum alternative

This task consists in comparing the the performance between the quantum algorithm and the classical alternatives. The goal of this project is to justify and conclude that quantum computing can get a big time speedup against classical options for some problems. This task will cost around 12 hours.

0.5 D - Documentation of the project

The goal of this task is the documentation of the project writing a document explaining all the process of this thesis. This task will cost around 92 hours.

0.6 MP - management of the project

0.6.1 MP1 - Contextualization and introduction

This task consists in the research and documentation of the introduction and contextualization of this project. This is part of the GEP part of the thesis. This task will cost around 20 hours.

0.6.2 MP2 - Time planning

This task consists in the research and documentation of the the time planning and management of this project. This is part of the GEP part of the thesis. This task will cost around 20 hours.

0.6.3 MP3 - Sustainability and costs

This task consists in the research and documentation of the sustainability and costs of this project. This is part of the GEP part of the thesis. This task will cost around 20 hours.

0.7 M - Meetings

During the project, one hour meetings with the co-directors will happen biweekly. This task will cost around 12 hours.

0.8 F - Final conclusions and defense

0.8.1 F1 - Conclusions

This task consists in the development of the final ideas and conclusion of the project. This task will cost around 12 hours.

0.8.2 F2 - Defense of the thesis

This task will cover the preparation of the defense of the thesis including the design of the slides and the presentation. This task will cost around 12 hours.

0.9 Project Resources

0.9.1 Human Resources

Four roles are found in this project: head of the project, researcher, developer and tester. However, since this is an individual student project, this four roles will be assumed by the student. The roles are introduced here:

- Head of the project (H): In charge of the management and planning of the project, scheduling and leading the meetings and documenting the project.
- Researcher (R): In charge of the research including the study of the algorithms.
- Developer (D): In charge of the implementations of the algorithms.
- Tester (T): In charge of the testing of the implementations, executing them and reporting the errors.

0.9.2 Material Resources

This project requires few but important material resources:

- Hardware:
  - Laptop
  - Quantum computer
- Software :
  - IDE - Visual Studio Code
  - Overleaf (LaTeXeditor) [18]
  - Version control system. (GitHub)
- Workstation

0.10 Summary of tasks

The table 1 join and summerize all the tasks indicating the time estimation in hours, the temporal dependencies among the tasks (where A < B means that A is needed to start B) and the needed project resources.

0.11 Gantt chart

In the Annex A, the Gantt chart can be found. This chart shows the distribution of the tasks. This chart has been designed using TeamGantt [19].

0.12 Risk Management

In the previous submission, the risks of the project are stated. We can see in the Table 2 that these are mainly the possibility of not having the first Spanish quantum computer on time, the possibility of needing more time for the study of the algorithms, the clarity and simplicity of the thesis and the possibility of other people developing this algorithm for Qibo.

In order to mitigate this risks, it is proposed:

- If the Spanish quantum computer is not installed by the end of May, it will be needed to use simulators of quantum computers. This simulators can be provided by Qibo. If it is needed to test on real devices, it will be needed to translate the code to a new framework like Qiskit and use the IBM quantum computers.
- When planning the time needed for every task, the time assigned to the study and development of the algorithms is intended to be enough time. However, the total time that is estimated for this project is a bit lower than the total time we can spend doing the project, so it is possible to enlarge the time needed up to 60 more hours.

Code	Descriptive name	Time estimation	Temporal dependences	HHRR	MMRR
I1	Qubits and quantum states	20h	-	R	C
I2	Quantum algorithms	16h	I1 < I2	R	C
I3	Quantum complexity	28h	I2 < I3	R	C
Q1	Quantum Fourier Transform	32h	I3 < Q1	R, D	C, IDE, G
Q2	Quantum phase Estimation	32h	Q1 < Q2	R, D	C, IDE, G
Q3	Study of HHL algorithm.	48h	Q2 < Q3	R	C
Q4	Implementation	20h	Q3 < Q4	D	C, IDE, G
Q5	Testing HHL	20h	Q4 < Q5	DS, T	QC, G
C1	Research	12h	-	R	C
C2	Study and Implementation	32h	C1 < C2	R, D	C, IDE, G
C3	Testing classical algorithms	20h	C2 < C3	T	C, IDE, G
C4	Comparison	12h	C3,Q5 < C4	R	C
D	Documentation	92h	-	H	L, C
MP1	Contextualization	20h	-	H	L, C
MP2	Time Planning	20h	MP1 < MP2	H	L, C
MP3	Sustainability and costs	20h	MP2 < MP3	H	L, C
M	Meetings	12h	-	H	C
F1	Conclusions	12h	C4 < F1	H	L, C
F2	Defense	12h	F1 < F2	H	C
P	Project	480h	-	-	-

Table B.1: Summary of the tasks of the project. original creation. Human Resources (HHRR): H - Head of the project, R - Researcher, D - Developer, T - Tester. Material Resources (MMRR): C - laptop computer, QC - Quantum Computer, IDE - Visual Studio Code, L - LaTeX

Risk	**Degree**	**Probability**
Not having access to quantum computer	High	Medium
Need for more time for quantum algorithm	High	High
Not achieving simplicity	High	Medium
Other team working on the same topic	Low	Medium

Table B.2: Classification of risks depending on degree of impact and the probability to happen. Original creation

- It is key to write a simple and very well detailed thesis. The way to mitigate this will be trying to specify all the steps taken during the project in the thesis. During the meetings, the co-directors will also give their opinion to know if changes are needed and things should be simpler.

- It is difficult to mitigate a risk involving other teams working on the same goal. If this is the case, the time will not be affected but the goals will have to be redefined. This project could change the framework used to, for example, Pennylane. But it could also be useful to improve the work of the other team if they submit their contribution before this project ends. This project will also work as a guide, that can be considered as a contribution to the community even if it is not strictly new code but content.

This proposal will allow to not affect the time planning. Any change on the resources is needed for the proposal. Additionally, the time management during the project it is also very important to manage the time left. Thus, if during the project

development it is not possible to complete some task in the estimated time, to fix this:

- The dedicated time for the supposed task will be incremented to finish it in time.
- Replan the task giving it a lower amount of time (and changing the goal to a simpler one) or removing it if it does not present inconveniences.

0.13 Deviations in the time planning

During the competition of this project, the planning slightly changed. Specifically Q3, Q4 and Q5 had a delay of two weeks. The tasks from the study of classical algorithms and the comparison between classical and quantum algorithms were removed from the planning because we realized that comparing two different computational paradigms was not worth. Therefore, this project did not have any objective regarding classical computing. Thus, classical alternatives are mention in the appendix A but this thesis did not get deeper in it.

Appendix C

Project Management: Budget Management

In this section, the budget of the project will be discussed. The budget estimation will be studied and justified in order to achieve the right and complete development of the project. The necessary items related to the tasks and requirements, both human and material, specified in the Planning section are identified.

The goal of this section is to recognise and define the budget that will be needed for the project. Not only the resources needed but a contingency plan to mitigate possible obstacles or risks.

0.1 Budget

The different necessary items for this project are specified forth below.

0.2 Budget for Human Resources

In order to calculate the budget needed for human resources, the four roles defined previously in the time planning section will be considered. They are the head of the project, the researcher, the developer and the tester.

In the Table C.1, the annual salaries of each role are defined. This salaries have been designed from data of annual study from TecnoEmpleo portal [20] where the average salaries of different jobs can be found.

This salaries have to include the social security contribution. In order to do that the gross salary will be multiplied by 1.3.

Role	Gross Salary (€)	Social Security (€)	total salary (€)
Head of the project (H)	45,300	13,590	58,890
Researcher (R)	30,450	9,135	39,585
Developer (D)	34,150	10,245	44,395
Tester (T)	30,250	9,075	39,325

Table C.1: Annual salaries for the roles of the project. Original creation

The following table C.2 is the analog for the table 1. Instead of indicate the annual salaries, it indicates the salary per hours.

Role	Gross Salary (€)	Social Security (€)	total salary (€)
Head of the project (H)	21.78	6.53	28.31
Researcher (R)	14.64	4.40	19.03
Developer (D)	16.42	4.93	21.35
Tester (T)	14.54	4.36	18.90

Table C.2: salaries per hour for the roles of the project. Original creation

Now the salaries are known, the table C.3 shows the CPA. the CPA is the distribution of time per each role in each task and the total final cost regarding the the roles.

Code	Descriptive name	H(h)	R(h)	D(h)	T(h)	Cost (€)
I1	Qubits and quantum states	-	20	-	-	380.60
I2	Quantum algorithms	-	16	-	-	304.48
I3	Quantum complexity	-	28	-	-	540.40
Q1	Quantum Fourier Transform	-	24	8	-	634
Q2	Quantum phase Estimation	-	24	8	-	634
Q3	Study of HHL algorithm.	-	48	-	-	926.40
Q4	Implementation	-	-	20	-	427
Q5	Testing HHL	-	-	-	20	378
C1	Research	-	12	-	-	228.36
C2	Study and Implementation	-	24	8	-	634
C3	Testing classical algorithms	-	-	-	20	378
C4	Comparison	-	12	-	-	228.36
D	Documentation	92	-	-	-	2,604.52
MP1	Contextualization	20	-	-	-	566.20
MP2	Time Planning	20	-	-	-	566.20
MP3	Sustainability and costs	20	-	-	-	566.20
M	Meetings	12	-	-	-	339.72
F1	Conclusions	12	-	-	-	339.72
F2	Defense	12	-	-	-	339.72
P	Project	188	184	60	48	11,015.88

Table C.3: CPA table showing the dedicated total time per task for each role. original creation. H - Head of the project, R - Researcher, D - Developer, T - Tester.

0.3 Material costs

0.3.1 Hardware costs

This project requires for a minimum equipment for the research and the development of it. This equipment will be mainly composed by a laptop computer capable to run the required software. The laptop used is a Slimbook PRO X (2021) with Ubuntu as operative system. The main reason for this is that this is the personal laptop of the student in charge of this project and because it makes easier to move and travel with it.

The other key device needed is the quantum computer. It is needed in order to run and test the quantum algorithms developed in this project. The used device will be the first Spanish Quantum computer developed by Qilimanjaro Quantum Tech and hosted in the Barcelona Supercomputing Center.

In the table C.4, the amortization of these devices is shown.

Device	Price(€)	life expectancy (years)	Months in use	Amortization(€)
Laptop computer	1300	5	4	86.67
Quantum computer	8,100,000	5	4	540,000
Total	**8,101,300**	**-**	**-**	**540,086.67**

Table C.4: Costs of the hardware material resources. Original creation.

0.3.2 Software costs

This project is characterized for using open source software mainly. This means that the cost of the software are 0€. The open source software used is:

- Pyhton: open source programming language. Its libraries are open source too.
- Qibo: open source framework for programming quantum computers. This is the framework that uses the Spanish quantum computer.
- Overleaf: open source online Latex editor. It allows more than one writer and interactive work. The co-directors will be able to check the documentation using it.

The other software that is used it is not open source but it is free to use. These are:

- GitHub: free version control system for code. It is useful for uploading and hosting code and for its review.
- Visual Studio Code: this is an IDE that will be used for writing the code.

Thus, using only these software tools, the total cost of the software is 0€.

0.3.3 Electricity

In this section the electricity consumed will be studied. The average price is the electricity today is about 0.17203 €/kWh [21]. The Table C.5 shows how much will consume each device.

Device	Power(W)	Hours	Consumption(kWh)	cost(€)
Laptop Computer	30	480	14.4	2.48
Quantum Computer	20,000	25	500	172.03
Total	**-**	**-**	**-**	**174.51**

Table C.5: Electricity expenses for material resources. Original creation.

0.3.4 Internet

The contracted internet fee is 30€ per month. The dedicated time per day is about 4 days. This means a sixth part of the day a.k.a. a sixth part of each month. Thus, the price of the internet dedicated for internet is 5€ per month. The project is 5 complete months. Then, the total price of the internet for the project is 25€.

0.3.5 Workstation

The workstation is the apartment where the student lives. This apartment has a rent of 650€. The maximum time of dedication of this project per month is 120 hours in average. A month has 720 hours in total. This means that the price of the workstation that is dedicated to the realization of this project is 108.33€ per month. The total price will be 541.67€.

0.3.6 Office material

A part of the budget will be addressed to office material. Notebooks, pencils, pens, rubber, pen corrector... The minimum amount of office material will be used. So there will be 30€ planned for this section.

0.3.7 Total amount of material costs

Once calculated all the material costs of this project, the total amount is 1,621,360.98€. In table C.6, a summary of each item is shown.

Item	Cost(€)
Hardware	540,086.67
Electricity	174.51
Internet	25
Workstation	541.67
Office Material	30
Total	**540,857.85**

Table C.6: Total amount of material costs estimation. Original creation.

0.4 Total amount of costs

Adding the previous calculations to the calculated budget for human resources, The total amount of costs is gotten. In the table C.7, it is shown the total of both budgets and the final amount.

Element	Cost(€)
Human Resources budget	11,015.88
Material costs	540,857.85
Total	**551,873.73**

Table C.7: Estimation of the total cost of the project. Original creation.

0.5 Contingency plan

In order to financially cope with unexpected obstacles during the project, for instance, the possibility to need the dedication of more hours, the budget will have a contingency item. In general this item is a 10% of the total budget. However, this budget is mainly determined by the quantum computer. So the contingency item will have an amount of the 15% of the total budget without the cost of the quantum computer. This is 1,856.53€.

0.6 Incidental costs

It is important to take into account the additional cost that can bring unexpected events during the project. The following list expose possible incidents:

- Increment of hours of development: 20 extra hours will be added to develop the quantum algorithms. This is a possible situation that could be faced.
- Increment of hours of research: 30 extra hours will be added for doing research of quantum algorithms. There is a high possibility to face this situation in the future.
- Bugs in Qibo: This bugs should be fixed if they affect the project. 10 extra hours will be added to face this possible situation.
- Inexperience in Qibo: 10 extra hours will be added to learn and get experience in Qibo. The possibility of this situation is low.
- New laptop: If the laptop is damaged and a new laptop is needed. It will not need more hours but a new laptop will have to be bought.

In the table C.8 are shown this incidents.

Incident	**Cost(€)**	**Risk (%)**	**Total cost(€)**
Increment of hours of development	427	30%	128.10
Increment of hours of research	570.90	50%	285.45
Bugs in Qibo	213.50	10%	21.35
Inexperience in Qibo	213.50	15%	32.04
New laptop	1300	10%	130
Total			**695.94**

Table C.8: Budget dedicated to incidents. Original creation.

0.7 Final budget

Finally, all the items for the budget of the project are calculated. The final budget for this project is . The table C.9 itemizes the different parts of the budget.

Element	Cost(€)
Human Resources budget	11,015.88
Material costs	540,857.85
Contingency plan	1,781.06
Incidental costs	695.94
Total	**554,350.73**

Table C.9: Estimation of the final budget. Original creation.

0.8 Management control

If during the project, any contemplated incident appears, the incidental costs budget will be used. In case it is not enough or it happens something that is not contemplated in this budget, the contingency budget will be used. Besides that, the following guidelines will be applied to avoid alterations in the budget.

- The hours will be counted per each task.
- The approach of the project will be reduced if any task can not be fitted in the planning causing an increment in the expenses.
- Update the possible incidents during the process.

The meetings will be a tool for analyzing the time dedicated in the tasks and decide if any change is needed. The following indicators will be used for the calculation of the deviations:

$$Pricedeviation = (EC - RC) \cdot RCH$$

$$Consumptiondeviation = (ECH - RCH) \cdot EC$$

Where EC: estimation cost, RC: Real cost, RCH: Real consumption of hours i ECH: Estimation consumption of hours.

1 Sustainability

After doing the survey, keeping in mind that the university gave a very poor knowledge to the students about sustainability, I think that I was aware about my knowledge and the survey confirmed it. I know almost nothing and this will be a change to learn and know more about the sustainability techniques, procedures and indicators regarding my profession and project. I should remark that after viewing how long was the survey, it makes me be more sensible about the importance of this topic and how it affects our sector. In this section, we'll see how this project affects environmentally, economically and socially in terms of sustainability.

1.1 Economic Dimension

The economic part of this project is justified in this report. The items of the budget are justified and they are clearly specified. It is used the minimum required material, all the software used is free to use or even open source.

The most shocking item of the budget can be the quantum computer. It is important to understand that this is a research device, the quantum computer as a thing is still in development and its technology is very expensive. This computer will be used for research purpose in many other projects and will be useful for both developing and studying quantum software and developing better quantum computers as well as for understanding this new field. Obviously, a quantum computer is needed for executing a quantum algorithm. Besides this, The human resources costs could be lower if the personnel was a student researcher. However, with qualified personnel, the success of the project is ensured.

This solution does not improve economic issues directly in the short term. Classical options of the HHL algorithm can be executed in domestic classic computers. But in the long term, this solution will be used in different applications improving in time and using less resources than a supercomputer and getting better results (or just getting results). Thus, this is clearly an economic improvement but it is needed to wait some years to see this benefits working. Quantum computing research implies an estimated inversion of 33 billions of dollars in the world and 5.5 billions of dollars in Europe. So the price for this project must be justified with the reasons given.

1.2 Environment Dimension

This project is in general respectful with the environment. The material and software used is compared to the needs for an average person that has a computer at home. The key part is the quantum computer.

As said before, this is a simple problem but using an emergent technology. This technology is now competing against High Performance Computing (HPC). A supercomputer is a lot of CPU connected in parallel. A quantum computer has one quantum processor that is energetically more efficient and causes a smaller damage to the environment. Specifically, the Spanish quantum computer is made using

superconductors. This means that the energy consumption is determined by the refrigeration of the device. Manufacturing bigger quantum processor will not increase the energy consumption since the refrigeration system will be the same always. Moreover, using superconductors is very energetically efficient because it does not dissipate heat, so the refrigeration is also as much efficient as possible.

1.3 Social Dimension

This project has clear goals. The student in charge of this project will achieve enough knowledge for the good understanding of quantum computing from its fundamentals. This is very important because it gives a great opportunity to get deeper in a field of study that is emerging and gaining prominence. Quantum computing will be key for the future and this project will help the student in order to be part of this evolution.

In this project a quantum algorithm is presented. But the goal is to show the capabilities of quantum computing. The algorithm will be useful for solving problems that now are not solvable in a considerable amount of time. The society will not use directly the capabilities of quantum computing or the HHL algorithm. But this research will let the development of new drugs, simulations, new financial applications, better performance of machine learning, a more robust cryptography... Definitely, quantum computing is a must and a good thing for society.

Appendix D

Gantt Chart

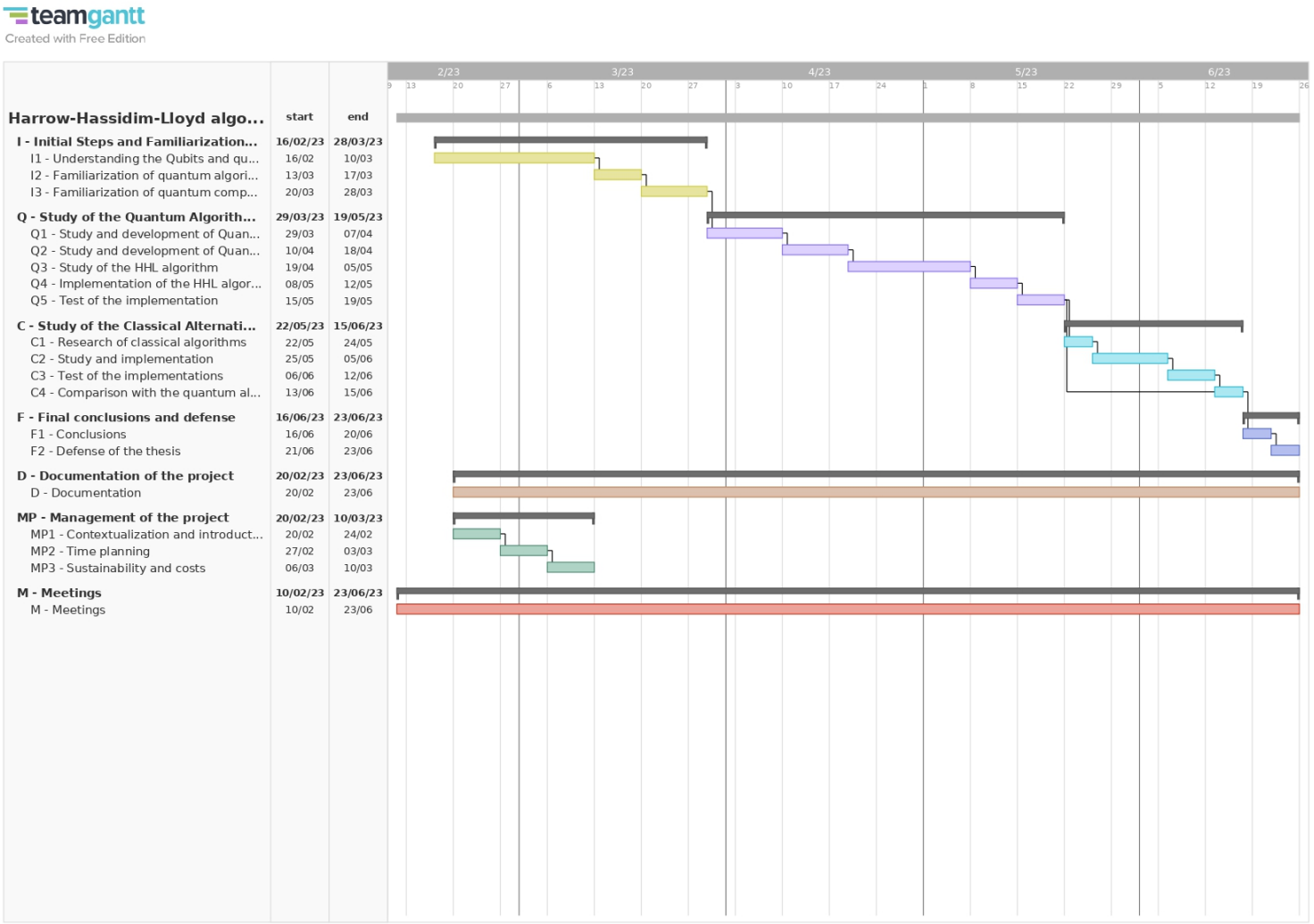


Figure D.1: Gantt chart. created with TeamGantt [19]. Original creation