



Escola Superior de Tecnologia
i Ciències Experimentals · ESTCE

Introduction to Quantum Computing for Graduate Students in Chemistry

BACHELOR'S THESIS REPORT

AUTHOR: ESTER BOCHONS RODILLA

SUPERVISOR: JUAN IGNACIO CLIMENTE PLASENCIA

Index

Aims.....	2
1. Introduction	3
2. Classical information and classical computing	6
2.1. Information storage and encoding	6
2.2. Information manipulation	8
3. Quantum mechanics	11
4. Quantum computing	17
4.1. The qubit.....	17
4.2. Quantum computation protocol.....	19
5. Computational project	25
5.1. Programming project 1: Simulate measurement of a 3-qubit register.....	25
5.2. Programming project 2: First full quantum computations.....	27
5.3. Programming project 3: Implement Grover’s quantum search	30
6. Conclusions	34
Annex. Full computation project code	34
References	44

Aims

- 1- To provide an overview of the concepts of classical computation and quantum mechanics on which quantum computing is based.
- 2- Introduce the basic ideas about quantum computing and expose its main differences with classical computing.
- 3- To program different examples of quantum computation with 3-qubit systems, as well as Grover's search algorithm.

1. Introduction

Quantum computation and quantum information are defined as the study of information processing tasks that can be accomplished using quantum mechanical systems [1]. Their birth cannot be understood as the evolution of just one field, but as the merging of several branches of science, such as computer science, quantum mechanics, information theory and cryptography.

It uses phenomena such as superposition, which establishes that if a physical system may be in one of many configurations—arrangements of particles or fields—then the most general state is a combination of all of these possibilities [2]; or entanglement, that occurs when a pair or group of particles interact in such way that the quantum state of each particle cannot be described independently of the state of the others. [3]

Although quantum computers are mostly theoretical constructs at the present time, it has been proved that quantum computing will eventually outperform classical computing in many purposes [4]. A good example of this is quantum cryptography, which has the potential to encrypt data for longer periods of time than classical cryptography [5]. Quantum simulation is also one of the most relevant potential applications, since simulations of quantum systems that are carried out numerically on classical computers are subject to the exponential growth of required resources as the size of the quantum system increases. And it has been claimed that quantum computers will be able to mimic these systems efficiently in the polynomial scale [6].

As a result, interest and investment in the field of quantum computing has increased dramatically in recent years. The market of quantum computing is projected to reach \$64.980,0 million by 2030 from just \$507,1 million in 2019. In the public sector, China has remained at the forefront of technological advances, launching the first quantum satellite in 2016. In the U.S., the Trump administration authorized in 2018 \$1.200,0 million to be spent on the quantum science over the next five years, while in 2020 India set a budget of \$1.120,0 million for the same period. Europe, on the other hand, has a total initiative of €1.000,0 million providing funding for the next ten years. [7]

The history of computation possibly begins with the appearance of the abacus in 500-300 BC, but its main development did not come until the 19th and 20th centuries, with Boolean algebra (1854), the theory of computation (Turing, 1936) and information theory (Shannon and Weaver, 1940s). It was in that 40s decade that John Eckert and John Mauchly developed the ENIAC (Electronic Numerical Integrator and Calculator), the first fully electronic computer, in 1946.

Parallel to the development of computation, in the early 20th century, the world of physics took a plot twist. The inadequacy of classical physics to the microscopic domain became increasingly evident due to various empirical facts, such as blackbody radiation and the photoelectric effect. Thus was born the theory of quantum mechanics, which differs from classical physics in that energy, momentum, angular momentum, and other quantities of a bound system are restricted to discrete values(quantization), objects have characteristics of both particles and waves (wave-particle duality), and there are limits to how accurately the value of a physical quantity can be predicted prior to its measurement, given a complete set of initial conditions (Heisenberg's uncertainty principle, 1927). [2]

Computing and the quantum world did not come together until well into the 20th century. In the 1970s, Paul Benioff began to research the theoretical feasibility of quantum computing. His research culminated in a paper, published in 1980, that described a quantum mechanical model of Turing Machines [8]. Shortly thereafter, Richard Feynman and Yuri Manin suggested that quantum computing has the potential to simulate things that classical computing cannot. [9] [10]

The origin of this idea can be found in the *decision problem*, posed by David Hilbert and Wilhelm Ackerman in 1928:

“Given a statement of a first-order logic¹ defined by a finite set of axioms, is there an algorithm that can decide whether the statement is true or false?” [11]

In 1936, Alonzo Church and Alan Turing both independently demonstrated that the existence of such an algorithm was impossible [12] [13]. The concept of the Turing machine, an abstract notion of what we know as a programmable computer, appeared for the first time. And further work led to the Church-Turing thesis.

“Any algorithmic process can be simulated efficiently using a probabilistic Turing machine.” [14]

In 1994, Peter Shor demonstrated that the problem of finding the prime factors of an integer could be solved efficiently on a quantum computer – a problem that takes classical computers an exponentially long time to solve for large numbers [15]. His algorithm launched a huge interest in the field of quantum computing. And the Church-Turing quantum thesis was formulated:

“A quantum Turing machine can efficiently simulate any realistic model of computation.” [16]

Decoherence (loss of entanglement due to interactions of a quantum system with external factors) is a major obstacle for the development of robust quantum computers. However, the physical realization of quantum computing became significantly more tenable in 1995, when Quantum Error Correction (QEC) emerged from several groups around the world. This theory makes it possible to protect quantum information from errors due to decoherence and other quantum noise [17]. But the first correcting code did not appear until 1997, when Alexei Kitaev proposed the surface code, a topological quantum error correcting code that is currently considered the most promising platform for realizing a scalable, fault-tolerant quantum computer. [18]

Many companies have ventured into the quantum computing race since Paul Benioff began his research. But among these companies, IBM and Google are the ones that have achieved the best results. IBM launched in 2017 the first industry initiative to build commercially available universal quantum computing systems [19]. The project, named as IBM Quantum, is pioneer in providing quantum computing service, and it has led to

¹ First order logic is a formal system used in mathematics, philosophy, linguistics and computer science. It is a language with quantifiers that reach only individual variables, and with predicates and functions whose arguments are only constants or individual variables. That is, instead of propositions like "Mars is a planet", we would have "there exists x such that x is Mars and x is a planet", where "there exists" is a quantifier. [35]

the design and construction in 2019 of the first integrated quantum computing system for commercial use [20]. That same year, Google AI, in collaboration with NASA, claimed they had realized a quantum computation that would be unfeasible on any classical computer [21]

Although the history of quantum computing is still short and there are still many discoveries to be made, there are already several quantum algorithms of great relevance. One of the best known, apart from Shor's, is Grover's algorithm, invented by Lov K. Grover in 1996 [22]. It is described as a database search algorithm which requires fewer function evaluations than a normal search, thus substantially reducing the search time. Inverting a function can be related to searching in a sequence, if we consider that this function produces the value of y as the position occupied by the value x in this sequence. Thus, if we have the function $y=f(x)$, which can be evaluated in a quantum computer, Grover's algorithm allows us to calculate the value of x given the value of y as input.

In this context of increasing awareness of the potential of quantum computation for the society, we have devised this Bachelor's Thesis Report as an attempt to provide the reader with a basic overview of the fundamentals of quantum computing from the point of view of an undergraduate student in Chemistry.

To this end, this work starts by introducing ideas from classical information and classical computing such as bit, byte, number encoding and the like. Logic gates will be described, with a review of the operators on which most current computational algorithms rely. In the next chapter, we will provide the backgrounds of quantum mechanics, emphasizing the phenomena of superposition and entanglement. And we will also study why the wave function collapses into a single (non-entangled) state when measuring, being that normally in quantum mechanics it evolves deterministically according to the Schrödinger equation as a linear superposition of different states. We call this the measurement problem. Based on these ideas, we will be able to introduce the basics in quantum information and quantum computing and to expose its fundamental differences with classical computing. For this purpose, we will explain the qubit and the difficulties of its physical implementation, review the quantum computing protocol and briefly describe the usual quantum gates. In a last stage, we will provide a few examples of routines and computational codes aimed for quantum computers, programmed with Mathematica, which altogether constitute a complete, student-made implementation of Grover's algorithm. The latter exercise is probably the most challenging and self-instructing part of this work.

2. Classical information and classical computing

2.1. Information storage and encoding

This chapter introduces the basic concepts required to understand how classical information is stored and manipulated, as well as its physical implementation in today's computers. With the information contained in this chapter we intend to provide the reader with some background on classical computing concepts that will be revisited in chapter 4 once we introduce quantum computing. In this way, hopefully, we will be able to capture the conceptual breakthrough that quantum computation entails.

The smallest unit of classical information is the bit (**binary digit**). It is a binary variable, generally represented as 0 or 1, where the number indicates one of two possible values or states, such as true or false, open or closed, north or south, and so on. We can store one bit in any electronic device or any other physical system that exists in either of two possible distinct states. These two states can be, for example, two positions of an electrical switch, two different directions of magnetisation or polarisation, or two voltage levels allowed by a circuit. Generally, in today's digital equipment (PCs, smartphones, game consoles, etc.) bits are implemented by using transistors. [23]

A transistor is an electronic device that regulates the flow of current or voltage in a circuit, acting as a switch and/or amplifier for electrical or electronic signals. As shown in figure 2.1, it has three terminals: base, collector and emitter. If the input signal is 1 (current through the base), the output signal is 1 (current between the collector and emitter). If the input signal is 0 (no current through the base), the output signal is also 0.

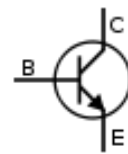


Figure 2. 1.
Transistor symbol

Bits are extremely useful because any discrete value such as numbers, words and images can be encoded using sequences of bits. With a single bit, there are only two (2^1) possible patterns to store information in, *0* or *1*, which is of limited usefulness. However, every bit we add double the possibilities. For example, with two bits, we have four (2^2) possible patterns: *00*, *01*, *10* and *11*. With three bits, we have eight (2^3) possible patterns: *000*, *001*, *010*, *011*, *100*, *101*, *110* and *111*. In general, *n* bit yields 2^n different patterns for storing information, so we find that the number of messages (*m*) that can be delivered by *n* bits is:

$$m = 2^n \quad (2.1)$$

Of course, human beings do not handle information as sequences of bits, so in order to be able to interact and exchange further information between computers and humans, there are rules that have been developed that associate sequences of ones and zeros with letters and decimal numbering. The basic unit of information used in classical computing is the byte, which consists of a group of 8 bits with a storage capacity of up to $m=256$ values, enough to store any of the most usual characters or alphabetic letters.

[ASCII](#), for example, is an encoding convention that represents each typed character by a number. The code uses 8-bit sequences, so the numbers range from *0* to *255* and each is stored in one byte to represent the upper and lower case letters of the English alphabet, plus punctuation marks, digits *0* to *9* and some control information. Thus, if we insert a message such as “Hello” in a computer that uses this code, it would be stored as follows:

Character	H	e	l	l	o
Code	72	101	108	108	111
Byte	01001000	01100101	01101100	01101100	01101111

Table 2. 1. Example of correspondence between alphabetic letters and ASCII

However, the number of patterns available in the ASCII code is insufficient to represent the alphabet of many Asian and some Eastern European languages, so [Unicode](#) had to be designed. This encoding typically stores each character in 2 bytes, so it uses 16-bit sequences, with which 65536 different patterns can be composed. Enough to be able to write texts in languages such as Chinese, Japanese and Hebrew.

A byte works well for characters, but for computational purposes we are very much interested in number manipulation too. Integer numbers can be easily encoded with bits by writing them in binary (rather than decimal) base. As an example, the table below shows the correspondence for numbers from 0 to 7, for which we need 3 bits, which offer 8 (2^3) different patterns to encode each digit.

000	0	100	4
001	1	101	5
010	2	110	6
011	3	111	7

Table 2. 2. example of correspondence between binary and decimal numbers

In general, to encode integers in binary notation, each position is associated with a weight, just as in the decimal system (ones, tens, hundreds, etc.). In the case of binary notation, the most right-hand digit is associated with 2^0 , the next position to the left with 2^1 , the next with 2^2 , and so on up to 2^n , where n is the number of bits. To obtain the corresponding value, as in base ten, we multiply the value of each digit by the weight associated with its position and then sum the results. The following tables show an example, comparing decimal and binary notation (with 8 bits).

Decimal pattern

2	3	3	Weight	Result	Sum
			x 1	3	233
			x 10	30	
			x 100	200	

Binary pattern

1	1	1	0	1	0	0	1	Weight	Result	Sum
								x 1 (2^0)	1	233
								x 2 (2^1)	0	
								x 4 (2^2)	0	
								x 8 (2^3)	8	
								x 16 (2^4)	0	
								x 32 (2^5)	32	
								x 64 (2^6)	64	
								x 128 (2^7)	128	






As we have seen previously, with 8 bits we can only store 256 different numbers, so to achieve a larger range, integers are usually stored in 8 bytes, that can store numbers between -9223372036854775808 and 9223372036854775807.

2.2. Information manipulation

With all of the above, we can now understand the basis of classical information, but how are the individual bits manipulated in a computer?

We can understand bit manipulation as handling of *true/false* values, if we imagine that bit *0* represents the *false* value and bit *1* represents the *true* value. To manipulate these values, functions belonging to Boolean algebra are used. Given one or more binary arguments (*true/false* or *0/1*), these functions produce a single binary output, applying a logical operation such as conjunction (and), disjunction (or) or negation (not). The electronic device that generates the output of a Boolean operation when given the input values of that operation is called a logic gate. In today's computers, they are often implemented with small electronic circuits in which the digits *0* and *1* represent voltage levels. [23]

To understand how they work, we provide a review of the eight most commonly used logic gates, along with their symbol and their logic table or truth table.

<i>GATE</i>	<i>SYMBOL</i>	<i>TRUTH TABLE</i>																		
<i>Buffer: acts as an identity function, generating no voltage or a voltage that is the same as the input.</i>		<table border="1"> <thead> <tr> <th colspan="2">INPUT</th> <th>OUTPUT</th> </tr> <tr> <th>A</th> <th></th> <th>Q</th> </tr> </thead> <tbody> <tr> <td>0</td> <td></td> <td>0</td> </tr> <tr> <td>1</td> <td></td> <td>1</td> </tr> </tbody> </table>	INPUT		OUTPUT	A		Q	0		0	1		1						
INPUT		OUTPUT																		
A		Q																		
0		0																		
1		1																		
<i>NOT: Its output is the opposite of the input.</i>		<table border="1"> <thead> <tr> <th colspan="2">INPUT</th> <th>OUTPUT</th> </tr> <tr> <th>A</th> <th></th> <th>Q</th> </tr> </thead> <tbody> <tr> <td>0</td> <td></td> <td>1</td> </tr> <tr> <td>1</td> <td></td> <td>0</td> </tr> </tbody> </table>	INPUT		OUTPUT	A		Q	0		1	1		0						
INPUT		OUTPUT																		
A		Q																		
0		1																		
1		0																		
<i>AND: Both input values must be true to obtain a true output.</i>		<table border="1"> <thead> <tr> <th colspan="2">INPUT</th> <th>OUTPUT</th> </tr> <tr> <th>A</th> <th>B</th> <th>Q</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>0</td> </tr> <tr> <td>0</td> <td>1</td> <td>0</td> </tr> <tr> <td>1</td> <td>0</td> <td>0</td> </tr> <tr> <td>1</td> <td>1</td> <td>1</td> </tr> </tbody> </table>	INPUT		OUTPUT	A	B	Q	0	0	0	0	1	0	1	0	0	1	1	1
INPUT		OUTPUT																		
A	B	Q																		
0	0	0																		
0	1	0																		
1	0	0																		
1	1	1																		
<i>OR: If one of the input values is true, we will get a true output.</i>		<table border="1"> <thead> <tr> <th colspan="2">INPUT</th> <th>OUTPUT</th> </tr> <tr> <th>A</th> <th>B</th> <th>Q</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>0</td> </tr> <tr> <td>0</td> <td>1</td> <td>1</td> </tr> <tr> <td>1</td> <td>0</td> <td>1</td> </tr> <tr> <td>1</td> <td>1</td> <td>1</td> </tr> </tbody> </table>	INPUT		OUTPUT	A	B	Q	0	0	0	0	1	1	1	0	1	1	1	1
INPUT		OUTPUT																		
A	B	Q																		
0	0	0																		
0	1	1																		
1	0	1																		
1	1	1																		
<i>NAND: if one or both of the values are false, it generates a true output.</i>		<table border="1"> <thead> <tr> <th colspan="2">INPUT</th> <th>OUTPUT</th> </tr> <tr> <th>A</th> <th>B</th> <th>Q</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>1</td> </tr> <tr> <td>0</td> <td>1</td> <td>1</td> </tr> <tr> <td>1</td> <td>0</td> <td>1</td> </tr> <tr> <td>1</td> <td>1</td> <td>0</td> </tr> </tbody> </table>	INPUT		OUTPUT	A	B	Q	0	0	1	0	1	1	1	0	1	1	1	0
INPUT		OUTPUT																		
A	B	Q																		
0	0	1																		
0	1	1																		
1	0	1																		
1	1	0																		

NOR: if one or both of the values are true, it generates a false output.



INPUT		OUTPUT
A	B	Q
0	0	1
0	1	0
1	0	0
1	1	0

XOR: only generates a true output if the values of the inputs are different. That is, equal inputs generate a false output.



INPUT		OUTPUT
A	B	Q
0	0	0
0	1	1
1	0	1
1	1	0

XNOR: opposite to XOR, only generates a true output if the inputs are the same.



INPUT		OUTPUT
A	B	Q
0	0	1
0	1	0
1	0	0
1	1	1

Table 2. 3. List of logic gates

A good example to understand how logic gates work is their use for integer addition. Now that we know how to encode integer numbers with bits, their sum in binary notation will not be so different from the sum in base ten.

$$\begin{array}{r}
 0 \ 1 \ 0 \ 0 \ 1 \ 1 \ 1 \ 0 \\
 + 0 \ 1 \ 0 \ 0 \ 1 \ 0 \ 1 \ 1 \\
 \hline
 \text{Carry} \ 1 \ 0 \ 0 \ 1 \ 1 \ 1 \ 0 \\
 \hline
 1 \ 0 \ 0 \ 1 \ 1 \ 0 \ 0 \ 1
 \end{array}$$

We add the bits in the same position, starting from the right, and if the base value is reached, we restart the addition from zero and carry one to the next bit. Therefore we will need, for each pair of bits, an output for the sum and an output for the carried value. One can check from the truth tables of the logic gates, that the XOR gate allows us to obtain the result of the addition, while the AND gate determines whether or not we carry one to the next position. The circuit formed by these two gates is called a half-adder.

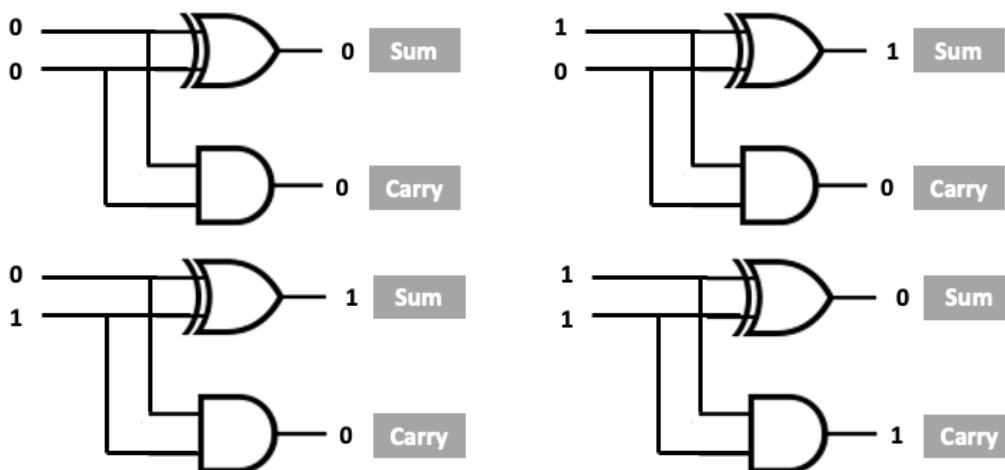


Figure 2. 2. Different inputs and outputs for a half-adder circuit

With an additional half-adder, we can add the carry output in the following position. The next carry output is generated with an OR gate, using the outputs of the AND gates of both half-adders as input. All these together form a full-adder circuit.

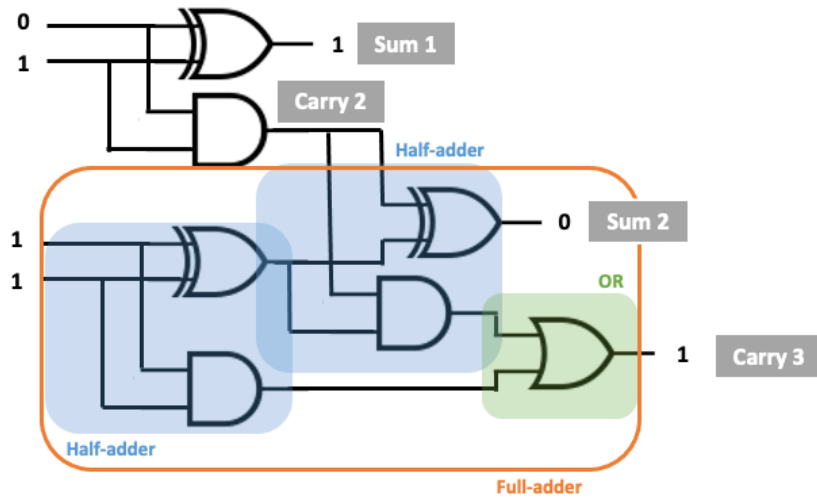


Figure 2. 3. Example of full-adder circuit

By connecting a half-adder circuit with 7 full-adder circuits, we can add 8-bit binary numbers, see Fig.2.3. This is precisely the kind of computational circuit that enables your ordinary calculator or computer to carry out simple additions. [23]

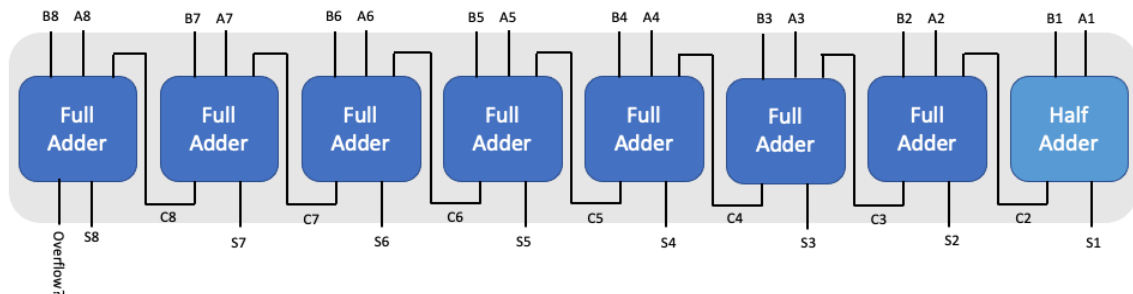


Figure 2. 4. Basic circuit for addition

3. Quantum mechanics

The aim of this chapter is to illustrate the reader with some of the key concepts and phenomena on which quantum computing is based. We will introduce the concepts of superposition and entanglement, discuss the probabilistic nature of quantum physics and also the so-called measurement problem. Rather than providing a self-contained introduction to quantum mechanics, which the reader can find in several textbooks (e.g. Refs. [24], [25]), we will address these concepts by drawing analogies of classical physics in the quantum limit and simple examples.

Some of these ideas may be difficult to understand, as they are suited to the microscopic domain and are not applicable to our experience in everyday life. That is why we will start this chapter by discussing the Dirac polariser experiment [2]. This is a good example to introduce some quantum concepts because photons, being bosonic particles, can accumulate to provide macroscopic effects we can observe directly.

The polarisation of light is defined as the angle formed by the electric field vector of electromagnetic radiation and the axis perpendicular to its displacement (Fig 3.1). Generally, conventional light sources, such as the sun, emit unpolarized light, whose electric field oscillates equally in all directions. We can filter it using a polarizer, which is a grating that only allows light that oscillates in the plane parallel to the filter gratings to pass through. Light transmitted on the other side is considered polarized light, since its electric field oscillates in only one direction.

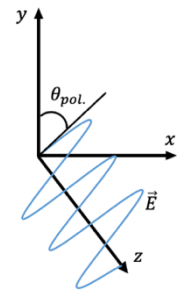


Figure 3.1. Polarisation angle

Imagine shining a beam of unpolarised light through a vertical polariser. The intensity of light observed on the other side will be lower, since only the fraction of light polarised parallel to the filter will pass through it. If we then add a horizontally polarised filter, no light would reach the other side at all, as all of it is vertically polarized.

If the light beam is not able to pass through two filters, it would be logical to think that the addition of a third filter cannot change this fact. But, surprisingly, if between the two aforementioned filters we place a polariser with an angle of 45° , we will see that part of the light is able to pass through all three filters (Fig. 3.2).

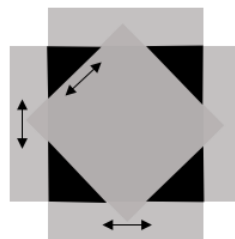


Figure 3.2. Polariser at 45° between two parallel polarisers

In terms of classical physics (electromagnetism), this behavior can be described by the classic principle of superposition and by the vector character of the incident radiation. The vector decomposition of the electric field of the incident radiation is as follows:

$$\vec{E}_\theta = E_0 \vec{i} + E_{90} \vec{j} \quad (3.1)$$

Where E_0 is the x-axis component of the electric field and E_{90} is the y-axis component.

The intensity is proportional to the emergent amplitude, and the squares of the components at 0 and 90 are related to each other:

$$|E_0|^2 = \cos^2\theta |E_\theta|^2 ; |E_{90}|^2 = \sin^2\theta |E_\theta|^2 \quad (3.2)$$

$$|E_0|^2 + |E_{90}|^2 = |E_\theta|^2 (\cos^2\theta + \sin^2\theta) = |E_\theta|^2 \quad (3.3)$$

The fraction of light passing through the first polariser is only the vertical component ($\vec{E}_\theta' = E_\theta \sin\theta$). As there is no component with projection on the x-axis, no light reaches the other side if a horizontal polarizer is added ($\vec{E}_\theta'' = 0$).

Using this classical electromagnetism approach we can estimate what proportion of the radiation passes through each polarizer. But if only vertically polarized light pass through the first filter, how is it possible that it can also pass through two other filters with different polarization? We need to approach the problem from the point of view of quantum physics. To this end we keep in mind that, as revealed by the photoelectric effect [24], light is made of collection of photons, each with its own frequency and polarization.

The photon is the indivisible unit of light and, therefore, it is not possible for only a fraction of it to pass through the filter. There are only two possible outcomes: to pass or not to pass. We can deduce that there will be a certain **probability** that a photon will be polarised in one direction or the other. Thus, when a beam of light passes through a polarizer, some photons pass and others do not, and therefore the intensity of the emerging light is lower.

The quantum state of a photon with any polarisation can be written as:

$$|\psi_v\rangle = c_x |x\rangle + c_y |y\rangle \quad (3.4)$$

In our case $|x\rangle$ would be the state fully polarized along the x-axis and $|y\rangle$ would be the state polarized along the y-axis. So the photon state is a **superposition** of two orthogonal states ($\langle x|y\rangle = 0$). Necessarily, the probability that the photon is polarised in any direction must be 100%. Therefore, the wave function will be normalized,

$$\langle \psi_v | \psi_v \rangle = c_x^2 + c_y^2 = 1 \quad (3.5)$$

The square of the coefficients c_x and c_y are related to the probability that the photon is polarised in one direction or the other.

It is important to note that the **measurement changes the state of the system**. In our case the measuring device is the polarizer. The photons that manage to pass through the first filter must be polarized along the y-axis, so their state has changed to:

$$|\psi_{v2}\rangle = |y\rangle + 0|x\rangle \quad (3.6)$$

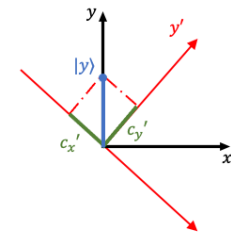


Figure 3. 3. polarization axes after passing through the filter at 45° (x', y')

Since now $c_x^2 = 0$, these photons will not be able to pass through a second horizontal polarizer. But if we add in between a polarizer which is not orthogonal (e.g. 45°), the projection of the photon polarized along $|y\rangle$ over $|y'\rangle$ is finite (Fig.3.3). We could rewrite the wave function with the new axes:

$$|\psi_{v2}\rangle = |y\rangle = c_x'|x'\rangle + c_y'|y'\rangle \quad (3.7)$$

Where both c_x' and c_y' are finite. Now the photons can pass through the second polarizer with probability $c_y'^2$ and collapse to $|\psi_{v3}\rangle = |y'\rangle$. (Fig. 3.4 (b))

As we can see, $|y'\rangle$ is no longer orthogonal to $|x\rangle$, or to $|y''\rangle$ if we rewrite the axes for the horizontal polarizer (Fig. 3.4 (c)). Consequently, photons will be able to pass through the third filter with a finite probability.

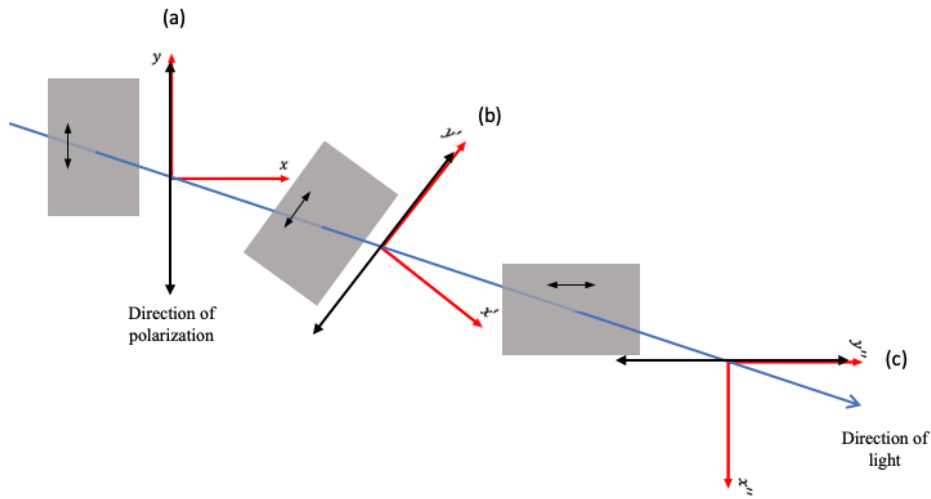


Figure 3. 4. Polarization direction and new axes for the three filters

So far we have managed to introduce the concept of superposition and some other basic ideas of quantum physics. Thus, let's leave aside Dirac's experiment and move on to another important phenomenon: entanglement.

Mathematically, entanglement is defined as the state of a composite system that cannot be expressed as a direct product of its individual states. In this state, one constituent cannot be fully described without considering the others.

Consider a system consisting of two indistinguishable particles. It is a fundamental property of quantum physics that they must be symmetric or antisymmetric with respect to the exchange of particles. If the particles are indistinguishable, the Hamiltonian must remain unaffected by exchanging their coordinates:

$$\hat{P}_{kl}\hat{H} = \hat{H}\hat{P}_{kl} ; \quad [\hat{P}_{kl}, \hat{H}] = 0 \quad (3.8)$$

That is, the Hamiltonian (\hat{H}) and the permutation operator (\hat{P}_{kl}) commute. This in turn implies that they share a complete set of eigenfunctions. Therefore, when applying the

operator on a wave function, we obtain the same function, multiplied by its eigenvalue, λ .

$$\hat{P}_{kl} \psi(k, l) = \psi(l, k) = \lambda \psi(k, l) \quad (3.9)$$

Since \hat{P}_{kl} is a hermitic operator, λ must be a real number. If we apply the operator twice:

$$\hat{P}_{kl} \hat{P}_{kl} \psi(k, l) = \hat{P}_{kl} \lambda \psi(l, k) = \lambda^2 \psi(k, l) \quad (3.10)$$

But if we permute twice the coordinates of a wave function, it remains unchanged, then $\lambda^2 = 1$. From this we can deduce that $\lambda = \pm 1$. Particles with $\lambda = +1$ are called bosons (e.g., photons) and we say that they are symmetric with respect to the permutation. On the other hand, particles with $\lambda = -1$ are called fermions (e.g., electrons) and these are antisymmetric with respect to the permutation.

Let us imagine now that we have a system formed by two particles that do not interact (perhaps because they are far apart in space). The Hamiltonian of the system could be written as

$$\hat{H} = \sum_i \hat{h}_i \quad (3.11)$$

where \hat{h}_i is the Hamiltonian acting on the i -th single particle. We could think that, since there is separation of variables, the wave function could be written as the product of the independent particles.

$$\psi(\tau_1, \tau_2) = \phi_a(\tau_1) \phi_b(\tau_2) \quad (3.12)$$

where ϕ_a would be the orbital of particle 1 and ϕ_b the orbital of particle 2.

However, the wave function (3.12) does not have the required symmetry properties because it does not fulfill Eq. (3.9). As we have seen above, for indistinguishable particles it must be symmetric or antisymmetric with respect to the permutation. If our system consists of two bosons, the required wave function would be:

$$\psi = N[\phi_a(\tau_1) \phi_b(\tau_2) + \phi_a(\tau_2) \phi_b(\tau_1)] \quad (3.13)$$

While in case of electrons, which are fermions, the function

$$\psi = N[\phi_a(\tau_1) \phi_b(\tau_2) - \phi_a(\tau_2) \phi_b(\tau_1)] = \begin{bmatrix} \phi_a(\tau_1) & \phi_a(\tau_2) \\ \phi_b(\tau_1) & \phi_b(\tau_2) \end{bmatrix} \quad (3.14)$$

is the function to be considered (a Slater determinant). If the composite system is in this state, it is impossible to attribute to either electron 1 or electron 2 a definite pure state. We say then that it is in an **entangled state**.

Incidentally, we note from Eq. (3.14) that if $\phi_a = \phi_b$, the wave function vanishes. This is a manifestation of the Pauli principle, which states that there cannot be in a system two identical fermions occupying the same spinorbital.

One last fundamental aspect of quantum mechanics, which we will rely on in future chapters when discussing quantum computing, is that of time-dependent perturbations in a quantum system.

The evolution of the state of a system in time is given by the time-dependent Schrodinger equation [26]:

$$-\frac{\hbar}{i} \frac{\partial \psi}{\partial t} = \hat{H}(q, t) \psi \quad (3.15)$$

where \hbar is the reduced Planck constant, ψ the system's eigenfunction and $\hat{H}(q, t)$ is the Hamiltonian operator with q and t representing space and time coordinates, respectively.

When studying Eq.(3.15) for chemical purposes, one is usually concerned with equilibrium situations, where the system does not change with time. Then, the Hamiltonian can be simply written as $\hat{H}(q)$, and it is reasonable to make a separation of variables and write ψ_i as a product of a function of space coordinates and a function of time:

$$\psi_i(q, t) = f_i(q) \cdot \chi_i(t) = f_i(q) \cdot e^{-Eit/\hbar} \quad (3.16)$$

$$\hat{H}(q) \psi_i(q, t) = \hat{H}(q) f_i(q) \cdot \chi_i(t) = \chi_i(t) \hat{H}(q) f_i(q) \quad (3.17)$$

This corresponds to a stationary state, since the probability distribution is independent of time, $\langle \psi_i | \psi_i \rangle = \langle f_i | f_i \rangle$.

In the case that we would need to manipulate the state of a quantum system, the Hamiltonian becomes time-dependent, and the solutions are no longer stationary states, but combinations of stationary states.

$$\psi_i(q, t) = \sum_i f_i(q) \cdot e^{-Eit/\hbar} \quad (3.18)$$

To illustrate how this changes the behavior of quantum systems,, we will use one of the simplest examples: the particle in a 1D box [26].

$$\hat{H} \psi = \frac{\hat{p}}{2m} \psi = -\frac{\hbar^2}{2m} \frac{d^2}{dx^2} \psi = E \psi \quad (3.19)$$

If we apply a time-independent Hamiltonian as in (3.17), the eigenfunctions obtained are:

$$\psi_n(x) = \sqrt{\frac{2}{L}} \sin \frac{n\pi x}{L} \quad (3.20)$$

and the corresponding probability density is:

$$|\psi(x)|^2 = \frac{2}{L} \sin^2 \frac{n\pi x}{L} \quad (3.21)$$

Fig. (3.5) shows the functions and probability densities of the states corresponding to the two lowest suffices needed to study the time-dependent example we want.

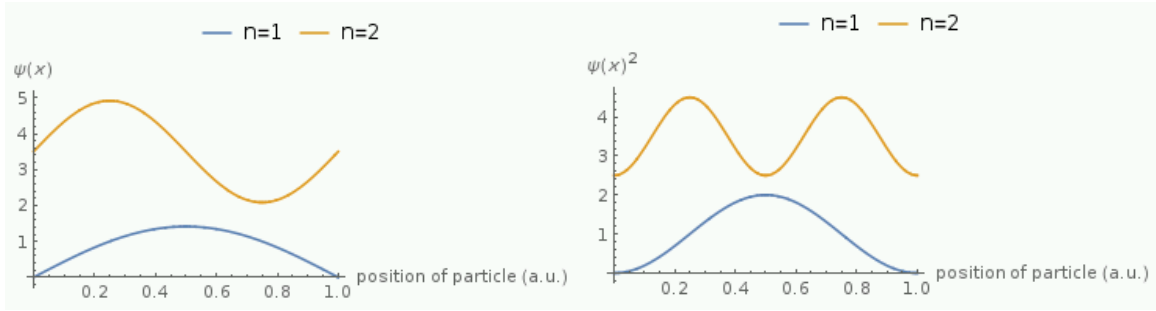


Figure 3. 5. Plots for $\psi(x)$ and $|\psi(x)|^2$ using Mathematica

As we can see, in any of both stationary states, there is a finite probability of finding the particle on either side of the box at any time. This is not the case for non-stationary states. Let suppose as an example that we induce a spectroscopic transition between the fundamental state and the first excited state of the particle in the box.

A good approximation to this non-stationary state would be a 50% combination of the first two stationary states

$$\psi(x, t) = \frac{1}{\sqrt{L}} \left(\sin \pi \frac{x}{L} \cdot e^{-iE_1 t/\hbar} + \sin 2\pi \frac{x}{L} \cdot e^{-iE_2 t/\hbar} \right). \quad (3.22)$$

$$|\psi(x, t)|^2 = \frac{1}{\sqrt{L}} \left(\sin^2 \frac{\pi x}{L} + \sin^2 \frac{2\pi x}{L} + 2 \sin \frac{\pi x}{L} \sin \frac{2\pi x}{L} \cos \omega t \right) \quad (3.23)$$

If we compare (3.23) with (3.21) we see that, unlike in stationary states, the probability density in non-stationary states depends on time. Plotting the evolution of $|\psi(x, t)|^2$ with time, we can see that the location of the particle changes and it bounces from wall to wall. Therefore, it is almost impossible to find the particle on either side of the box at certain times.

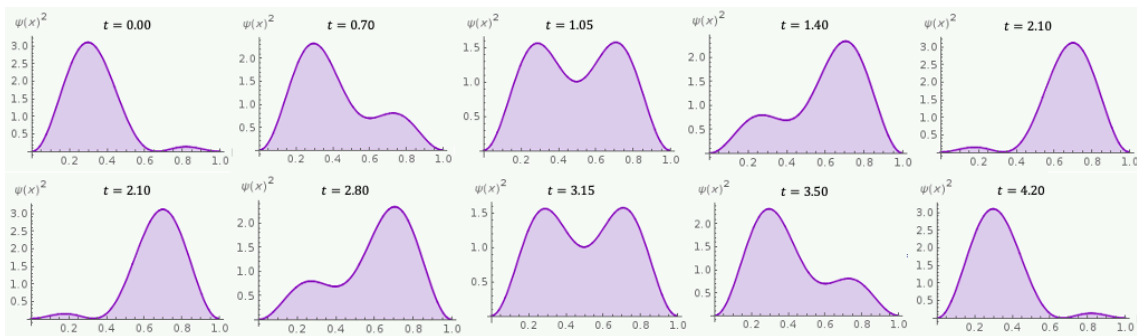


Figure 3. 6. Plots for $|\psi(x)|^2$ with evolution of time using Mathematica

4. Quantum computing

4.1. The qubit

We have already seen in Chapter 2 that conventional bits can take only the values 0 or 1, thus they can be implemented by any system with two possible states (e.g. two voltage levels in a circuit). This has proved to be an extremely efficient way of coding information.

Over the last 50 years, the information technology industry has been exponentially developing following Moore's law, which states that the number of transistors in a microprocessor double about every two years. The advances achieved by engineers in the lithography of silicon have allowed the exponential miniaturisation of electronics, but, as transistors reach the nanoscopic scale, classical physics no longer applies and we must stick to the laws of quantum mechanics.

As we saw in Chapter 3, quantum systems differ from classical ones in that the states can occur as a superposition of two orthogonal states (Eq.3.4). This brings about the idea that it may be possible to implement bits based on such kind of states. We call them quantum bits (or qubits). Hence, a qubit could be in state $|0\rangle$, state $|1\rangle$, or a combination of both:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \quad (4.1)$$

Where α and β are complex numbers. Because the amount of information that a system of qubits can represent increases exponentially, superposition allows quantum algorithms to process information in a fraction of the time it would take for even the fastest classical computer systems to solve certain problems.

However, although a qubit may exist in a superposition of states, we know that if we measure it we will not find it like that. It will collapse in state $|0\rangle$ with probability $|\alpha|^2$, or in state $|1\rangle$ with probability $|\beta|^2$.

Quantum states behave mathematically in an analogous way to physical vectors. This is why we can also describe a qubit by a column vector:

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \quad ; \quad |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \quad ; \quad |\psi\rangle = \begin{pmatrix} \alpha \\ \beta \end{pmatrix} \quad (4.2)$$

The state of a qubit, $|\psi\rangle$, is a vector in a two-dimensional complex vector space, whose standard basis, $\{|0\rangle, |1\rangle\}$, consists of two distinguishable orthonormal states.

Since $|\alpha|^2 + |\beta|^2 = 1$, we can also rewrite Eq. 4.1 as:

$$|\psi\rangle = e^{i\gamma} \left(\cos \frac{\theta}{2} |0\rangle + e^{i\varphi} \sin \frac{\theta}{2} |1\rangle \right) \quad (4.3)$$

Where γ , θ and φ are real numbers. And since the global phase factor, $e^{i\gamma}$, has no observable effect [27], then we can write:

$$|\psi\rangle = \cos\frac{\theta}{2}|0\rangle + e^{i\varphi}\sin\frac{\theta}{2}|1\rangle \quad (4.4)$$

This state vector describes a point on the surface of a 3D sphere known as the Bloch sphere, which can be observed in Fig. 4.1. The points on this surface can also be expressed in Cartesian coordinates as:

$$(x, y, z) = (\sin\theta\cos\varphi, \sin\theta\sin\varphi, \cos\theta) \quad (4.5)$$

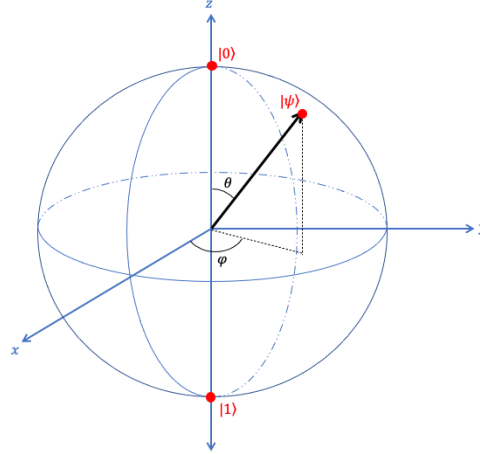


Figure 4. 1. State of a qubit on the Bloch sphere

We have already learned three ways to represent the state of a single qubit: vectors in ket notation, column vector or points on the Bloch sphere surface. Now we can move on to composite systems.

In the last chapter we saw that some quantum systems can be written as the direct product of their independent particles. Therefore, if we have a qubit in state $|\psi_1\rangle$ and a second qubit in state $|\psi_2\rangle$, the state of the combined system would be:

$$|\psi\rangle = |\psi_1\rangle \otimes |\psi_2\rangle \quad (4.6)$$

A system of two qubits then is a vector that can be written in the following ways:

$$|\psi\rangle = \alpha_{00}|00\rangle + \alpha_{01}|01\rangle + \alpha_{10}|10\rangle + \alpha_{11}|11\rangle \quad (4.7)$$

$$|\psi\rangle = \begin{pmatrix} \alpha_{00} \\ \alpha_{01} \\ \alpha_{10} \\ \alpha_{11} \end{pmatrix}$$

Where $|ij\rangle$ represents the basis states of the system and the coefficients α_{ij} are the corresponding amplitudes.

As in the case of a single qubit, the measurement result, $x = (00,01,10,11)$, is obtained with probability $|\alpha_x|^2$. Furthermore, the state of the 2-qubit system after the measurement will be $|x\rangle$ and the normalisation condition must be satisfied, so we know that $|\alpha_{00}|^2 + |\alpha_{01}|^2 + |\alpha_{10}|^2 + |\alpha_{11}|^2 = 1$.

However, this is not the case for all two-qubit systems. If each qubit forms a closed system, i.e. they are prepared independently and kept isolated, their state can be written in product form. But if we allow the qubits to interact, it may not be possible to represent their state by a tensor product. Then we say that they are **entangled**.

The last question to be addressed about the qubit is its physical implementation in a computer. We need a system with two possible states, as in the case of the bit, but in this case the system must satisfy the peculiarities of quantum mechanics that we have already seen throughout this paper.

In a classical computer, as we saw in chapter 2, each bit is stored in a physical system as a small capacitor in which different values of the charge represent 0 and 1. But this does not represent a good qubit, as it cannot remain in a superposition state, or in an entangled state.

A good example of a qubit would be a photon, as we saw in the previous chapter, or a particle with spin-1/2, such as an electron, where only the spin can change. The quantum state of this particle would be:

$$|\psi\rangle = \alpha|\uparrow\rangle + \beta|\downarrow\rangle \quad (4.8)$$

where $|\uparrow\rangle$ and $|\downarrow\rangle$ are the states with $S_z = \pm \frac{\hbar}{2}$. Being S_z the projection of the electron spin momentum over the z-axis. From now on, when talking about computation, we will refer to N of these 2-state systems as an N-qubit register, considering them together as a quantum system.

The biggest problem is not really finding systems that we can use as qubits but building a computer with them. Such computers are fragile and difficult to control. Any influence from the outside world would collapse the computation and the qubit would no longer represent many states at once. [28]

Currently one of the most favored qubit designs is based on semiconductor circuits. Other examples of physical systems with which to implement a qubit are both nuclear spin and atomic spin [29]. Recently, much attention has also been paid to the use of quantum dots, which are semiconductor nanoparticles with optical and electronic properties characteristic of quantum mechanics [30]. Physicists and materials scientists still have to work to isolate the qubits as much as possible, as well as to keep them in their quantum states long enough to perform calculations.

4.2. Quantum computation protocol

The quantum computing protocol is divided into three main steps: initialisation, manipulation and measurement.

Initialisation simply consists of establishing a defined initial state for our quantum system. Most often the N-qubit register is initialised in one of its basis states. This means that one of the amplitudes $\alpha_{ijk\dots}$ will be equal to 1, while the rest will be equal to 0. For a 3-qubit register, which state can be written as

$$\begin{aligned}
|\psi\rangle = & \alpha_{000}|000\rangle + \alpha_{001}|001\rangle + \alpha_{010}|010\rangle + \alpha_{011}|011\rangle + \alpha_{100}|100\rangle \\
& + \alpha_{101}|101\rangle + \alpha_{110}|110\rangle + \alpha_{111}|111\rangle \quad (4.9)
\end{aligned}$$

One of these initial states could be, for example:

$$|\psi\rangle = |011\rangle = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad (4.10)$$

Analogous to classical computing, in quantum computers information is **manipulated** using gates. But in this case the gates are unitary operations, i.e. they preserve the norm of the state. These operations must not collapse the state of the system as in measurement.

Like qubits, quantum operators can be represented by matrices. A quantum gate with n inputs and outputs can be represented by a matrix of degree 2^n . In the case of single qubit gates, $2^1 = 2$. Of these, the simplest is the quantum NOT gate (sometimes called the Pauli X gate).

$$U_{NOT} = X \equiv \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \quad (4.11)$$

It changes the state of the qubit from $|0\rangle$ to $|1\rangle$ and vice versa. In terms of the Bloch sphere, this can be visualized as a rotation through an angle π about the x axis.

$$\begin{aligned}
U_{NOT}|0\rangle &= \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} = |1\rangle \\
U_{NOT}|1\rangle &= \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} = |0\rangle
\end{aligned}$$

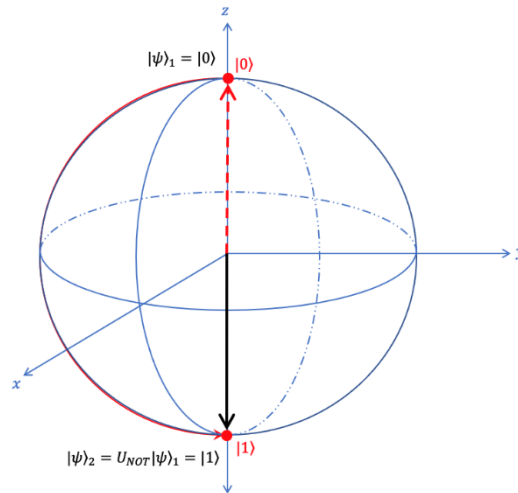


Figure 4. 2. Bloch sphere representation of the quantum NOT gate acting on a qubit

Another family of relevant single-qubit gates are phase shift gates, which change or alter the relative phase of the amplitudes α and β , leaving the probabilities $|\alpha|^2$ and $|\beta|^2$ unchanged.

$$P = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\varphi} \end{pmatrix} \quad (4.12)$$

Where φ is the phase shift. This is equivalent to drawing a horizontal circle on the Bloch sphere of φ radians. We could also compare it to cutting the sphere at the equator, along the $x - y$ axes, so that the $|1\rangle$ hemisphere rotates, while the $|0\rangle$ hemisphere remains static. In general, the action of the phase shift on a qubit is:

$$P|\psi\rangle = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\varphi} \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} \alpha \\ e^{i\varphi}\beta \end{pmatrix}$$

When $\varphi = \pi$, $e^{i\pi} = \cos \pi + i \sin \pi = -1$, and we obtain the Pauli Z gate.

$$P(\pi) = Z \equiv \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \quad (4.13)$$

It generates a half-turn in the Bloch sphere about the z axis, as shown in Fig. 4.3.

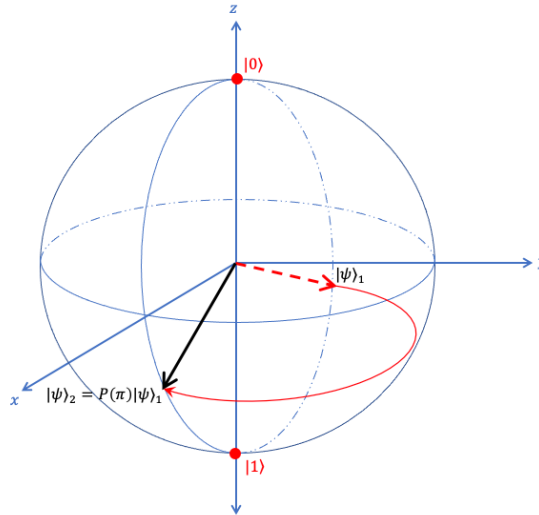


Figure 4. 3. Bloch sphere representation of the phase shift π gate acting on a qubit

If we let $\varphi = \pi/2$, then we have that $e^{i\pi/2} = \cos(\pi/2) + i \sin(\pi/2) = i$. The gate we obtain is called the S gate, which has the matrix representation

$$P(\pi/2) = S = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix} \quad (4.14)$$

It is like a quarter turn anti-clockwise about the z axis (Fig. 4.4).

And finally, we obtain the gate T if $\varphi = \pi/4$, whose matrix representation is as follows:

$$P(\pi/4) = T = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{pmatrix} = e^{i\pi/8} \begin{pmatrix} e^{-i\pi/8} & 0 \\ 0 & e^{i\pi/8} \end{pmatrix} \quad (4.15)$$

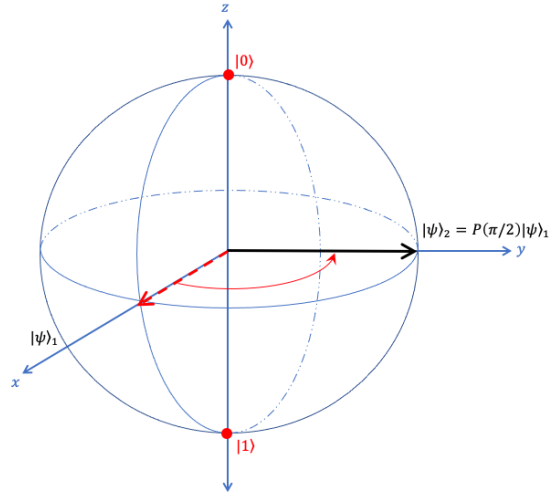


Figure 4. 4. Bloch sphere representation of the phase shift $\pi/2$ gate acting on a qubit

In terms of the Bloch sphere, we can visualize this as an eight turn clockwise about the z axis. (Fig.4.5)

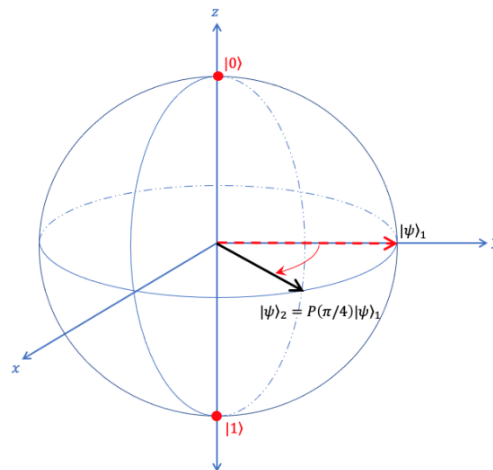


Figure 4. 5. Bloch sphere representation of the phase shift $\pi/4$ gate acting on a qubit

The Hadamard gate is also one of the most relevant single qubit gates, which we will use later in this paper.

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \quad (4.16)$$

Visualized on a Bloch sphere, the Hadamard gate interchanges the x and the z axes, and inverts the y axis (Fig. 4.6 and 4.7)

By applying this gate on a qubit in one of the basis states, we obtain a superposition of both:

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

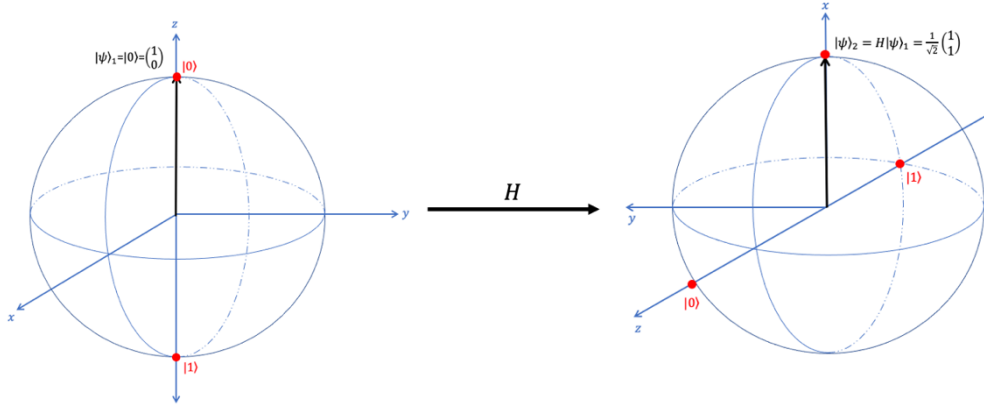


Figure 4. 6. Bloch sphere representation of the Hadamard gate acting on a qubit in one of the basis states

The Hadamard gate can also take a superposition of both basis states and put them back together into a single state:

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

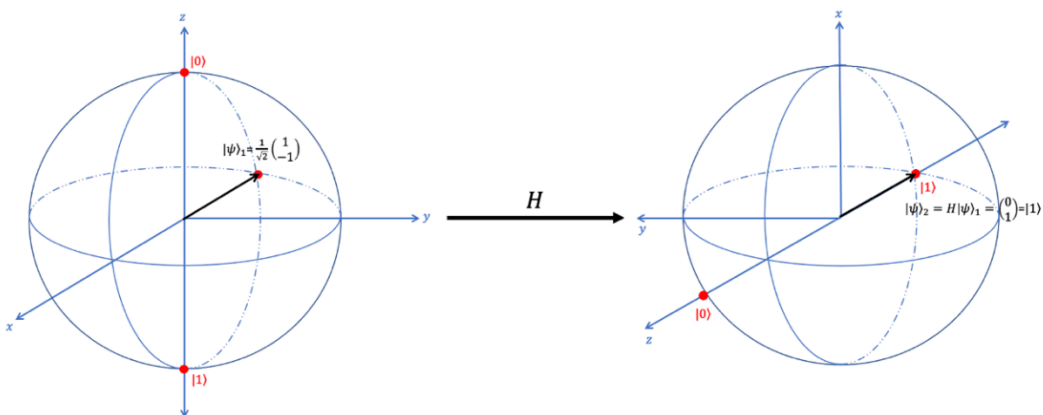


Figure 4. 7. Bloch sphere representation of the Hadamard gate acting on a qubit in a superposition state

The last gate we will discuss in this chapter is the CNOT (Controlled NOT) gate. It has two input qubits, the first qubit is known as the control qubit and the second one as the target qubit. Its matrix representation is

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

As one can easily check from matrix products, if the control qubit is in state 1, then the target qubit changes from one basis state to another. If the control qubit is set to 0, the target qubit is left unchanged. The general set of operations is:

$$CNOT |00\rangle = |00\rangle$$

$$CNOT |01\rangle = |01\rangle$$

$$CNOT |10\rangle = |11\rangle$$

$$CNOT |11\rangle = |10\rangle$$

The remaining step to finalize the computation would be **measurement**. As for the initialisation, we often seek to find the register in one of the 2 basis states. To do this we measure each of the qubits individually.

We already know from Chapter 3 that the probability of measuring each basis state is given by the square of its amplitude. Moreover, if the result of the measurement is, for example, $|110\rangle$, $|\psi\rangle$ collapses to the basis state $|110\rangle$ and any subsequent measurement will always give $|110\rangle$.

Finally, we will discuss the quantum circuit model, an example of which is shown in Fig. 4.8. It is read from left to right, starting from the initial state of the N-qubit register. The qubits are carried along "wires", which in this case only mean time flow. On the right side we can see the quantum circuit symbol for the measurement.

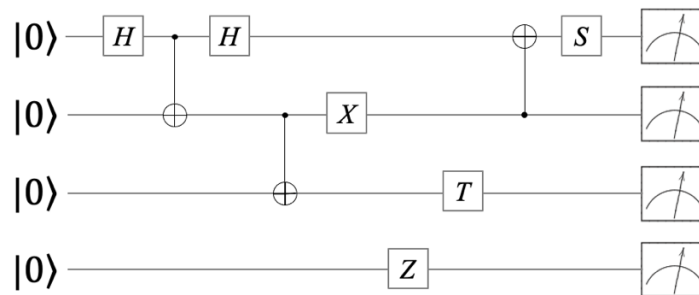


Figure 4. 8 Example of quantum circuit

In the central area of the circuit we can see several quantum gates. These are generally represented by a box with the symbol of each gate in it. A quantum gate acting on n qubits will receive the input qubits through n wires and another n wires will carry the output qubits away from the gate.

Fig. 4.9 illustrates another interesting convention of quantum circuits. Being U any unitary matrix acting on n qubits, we can define a controlled- U gate, which is a natural extension of the controlled-NOT gate. This gate, like the CNOT shown in Fig. 4.10, has a control qubit, indicated by a black dot, and n target qubits, indicated by the wires on the boxed U , or the symbol \oplus in the case of CNOT.

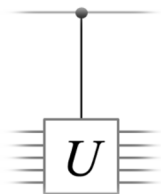


Figure 4. 9 Controlled-U

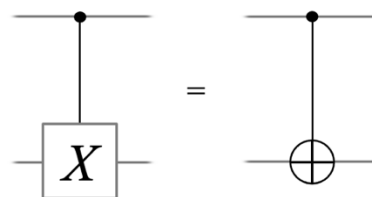


Figure 4. 10 Two different representations for CNOT

5. Computational project

In this final chapter, after reviewing all the necessary concepts throughout the work, we will simulate the operation of a quantum computer, implementing some realistic examples of quantum algorithms. For this purpose, we will take three programming projects proposed for undergraduate students in a scholar paper [31]. The program used to carry out the project was Wolfram Mathematica, since it has all the necessary functionalities for the work to be done and has been used several times throughout the degree, so its use is familiar.

Because, of course, we do not have access to a quantum computer, the various computations have been implemented on a classical computer. This is not able to take advantage of quantum programming to save computational time, but it is entirely feasible for obtaining results that we can evaluate.

The only limitation of using a classical computer is the number of qubits that can be used, which is a maximum of about 27. The problem is that the number of states grows as 2^N and for 27 qubits, we have 134217728 states. Each of the states also has a complex coefficient. Therefore, there are two numbers of 16 bytes each:

$$2^{16} \text{ bytes} * 2^{27} \text{ states} = 4294967296 \text{ bytes} \approx 4 \text{ Gbytes}$$

This is the total RAM, or at most half of it, that a normal laptop might have.

5.1. Programming project 1: Simulate measurement of a 3-qubit register

First, we implement the initialization and measurement of our 3-qubit system. In eq.4.9 we have already seen how to define the state of 3 qubits. We must set the number of qubits and create a vector state with eight positions, in which we will store the coefficients corresponding to each basis state.

$$\text{basis} = \{\{0,0,0\}, \{0,0,1\}, \{0,1,0\}, \{0,1,1\}, \{1,0,0\}, \{1,0,1\}, \{1,1,0\}, \{1,1,1\}\}$$

After creating a state vector full of zeros, we must insert in it the desired values for the coefficients of the wave function and make sure that it is normalized. For example, we could set as initial state an equal superposition of states $|000\rangle$ and $|111\rangle$, see eq.5.1.

$$|\psi\rangle = \frac{1}{\sqrt{2}} (|000\rangle + |111\rangle) = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \quad (5.1)$$

```
In[292]= nqubit = 3;
```

```
In[293]=  $\psi$  = Table[0, 2 ^ nqubit]
```

```

In[304]=  $\psi[[1]] = 1.0;$ 
In[305]=  $\psi[[8]] = 1.0;$ 
In[306]=  $\psi$ 
Out[306]= {1., 0, 0, 0, 0, 0, 0, 1.}

In[307]= (* Normalize *)
In[308]= norm =  $\sum_{i=1}^{2^{\text{nqubit}}} \text{Conjugate}[\psi[[i]]] * \psi[[i]]$ 
Out[308]= 2.

In[309]=  $\psi = \frac{1}{\text{Sqrt}[\text{norm}]} * \psi$ 
Out[309]= {0.707107, 0., 0., 0., 0., 0., 0., 0.707107}

```

According to what we have studied in previous chapters, if we make several measurements, the result will be $|000\rangle$ 50% of the time and $|111\rangle$ the other 50% of the time, but the result of each single measurement will be random.

Therefore, to program the measurement in Mathematica, we first choose a random number r between 0 and 1. This respects the randomness of the measurement and coincides with the sum of the quadratic coefficients. In our case there is a 50% probability of finding the system in each of the two possible states, so if r is between 0 and 0.5, the result of the measurement will be $|000\rangle$. If, on the other hand, r is greater than 0.5, the result of the measurement will be $|111\rangle$.

$$|\psi\rangle = \frac{1}{\sqrt{8}} \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \quad (5.2)$$

In order to bring it to the general case, we will use a slightly more complicated example, the superposition of the eight basis states shown in 5.2. In this case the probability of finding the system in each of the states is 12.5%. If r is between 0 and 0.125, the result of the measurement is $|000\rangle$. If it is between 0.125 and 0.25, the result is $|001\rangle$. Between 0.25 and 0.375 is $|010\rangle$, between 0.375 and 0.5 is $|100\rangle$, ... To program this, in a first step we define $q = |\alpha_{000}|^2$, so that if $q > r$, the result of the measurement is $|000\rangle$. If not, we redefine $q = q + |\alpha_{001}|^2$ and compare again. If $q > r$ the result is $|001\rangle$ and, if not, we repeat the same step for the following states until $q > r$ is satisfied and we find the result. This can be programmed in several consecutive steps or using a loop. The code used in this work is as follows:

```

In[402]= r = RandomReal[]
Out[402]= 0.779249

In[403]= q = 0;
Do[q = q +  $\psi$ cat2[[n]];
Print["q=", q, " ", "r=", r];

```

```

If[q > r, Print["measurement=", base[[n]];
Break[]], {n, 2^nqubit}]
q=0.125    r=0.779249
q=0.25    r=0.779249
q=0.375    r=0.779249
q=0.5    r=0.779249
q=0.625    r=0.779249
q=0.75    r=0.779249
q=0.875    r=0.779249
measurement={1, 1, 0}

```

Using another loop, we can repeat the measurement as many times as we wish and thus study the probability of obtaining each result. In Figure 5.1 we can see represented the results of repeating the measurement simultaneously for both states. And below is the code used to perform it.

```

In[1105]:= Ntrials = 100 000;
In[1115]:= measurementcat = Table[0, 2^nqubit];
In[1116]:= Do[r = RandomReal[];
q = 0;
Do[q = q + ψcat2[[n]];
If[q > r, result = n;
Break[]], {n, 2^nqubit}];
measurementcat[[result]] = measurementcat[[result]] + 1, {i, Ntrials}]

```

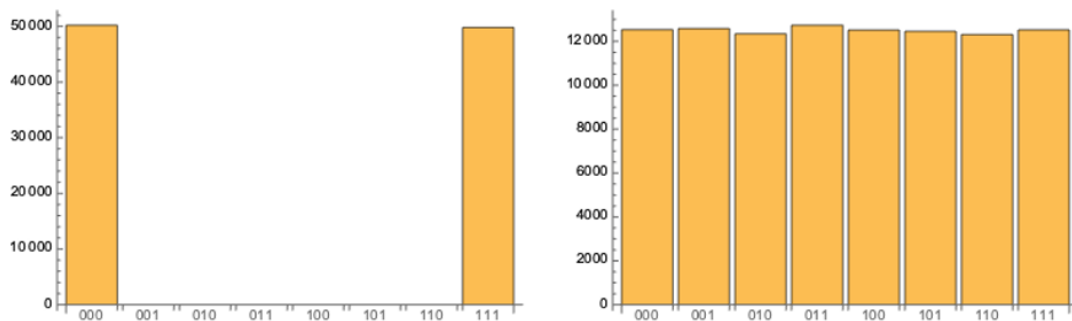


Figure 5. 1. Results of the measurement in programming project 1. On the x-axis we can see the label of each state and on the y-axis the number of times it has been measured. On the left the state defined by 5.1 and on the right the state defined by 5.2.

5.2. Programming project 2: First full quantum computations

For the second project, four different calculations included in Fig. 5.2. were programmed. These are not particularly complicated examples and will allow us to illustrate the effect of the most frequent single-qubit gates. Previously the necessary logic gates must be built, which in the case of 3 qubits are represented by $2^3 \times 2^3$ matrices. The matrices corresponding to the Hadamard gate acting on each of the 3 qubits ($\hat{H}^{(1)}$, $\hat{H}^{(2)}$ and $\hat{H}^{(3)}$) are shown below, as well as the phase shift gate ($\hat{R}_\theta^{(3)}$) acting on the third qubit. $\hat{H}^{(1)}$, $\hat{H}^{(2)}$ and $\hat{H}^{(3)}$ are constructed with the tensor product of the Hadamard gate seen in 4.16

and two identity matrices, with the H gate being multiplied at the position of the qubit on which it operates.

$$\hat{H}^{(1)} = \hat{H} \otimes \hat{I} \otimes \hat{I} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 \end{bmatrix} \quad (5.3)$$

$$\hat{H}^{(2)} = \hat{I} \otimes \hat{H} \otimes \hat{I} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 \end{bmatrix} \quad (5.4)$$

$$\hat{H}^{(3)} = \hat{I} \otimes \hat{I} \otimes \hat{H} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \end{bmatrix} \quad (5.5)$$

$$\hat{P}_\theta^{(3)} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & e^{i\theta} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & e^{i\theta} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & e^{i\theta} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & e^{i\theta} \end{bmatrix} \quad (5.6)$$

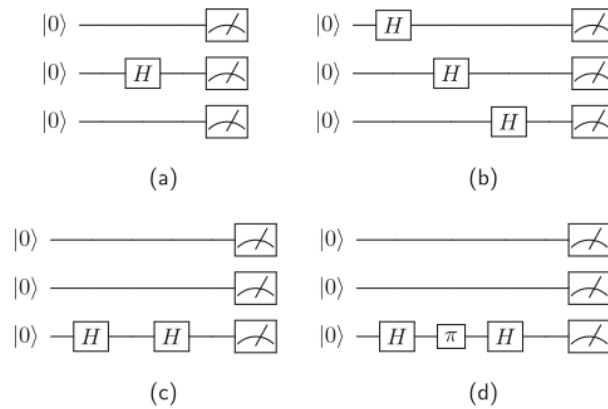


Figure 5.2. Calculations to be solved for programming project 2. The initial state of the qubit register is read from top to bottom. As we saw in chapter 4, H represents the Hadamard gate and π represents the phase shift gate with $\theta = \pi$.

Once $|\psi\rangle$ is initialized to state $|000\rangle$, the gates are applied by multiplying $|\psi\rangle$ by the left. They must be applied in the same order as shown in the circuit diagram. The gates in Fig.5.2(b), for example, are carried out by the following operation

$$|\psi\rangle \leftarrow \hat{H}^{(3)}\hat{H}^{(2)}\hat{H}^{(1)}|\psi\rangle \quad (5.7)$$

```
In[1139]:=  $\psi_{out} = H3.H2.H1.\psi_{in}$ 
```

As in the previous project, the measurement after each of these calculations was repeated in order to observe the trend of the results. The representation of these results can be seen in Figure 5.3.

In Fig.5.2(a) a Hadamard gate is applied to qubit 2, so that it is in an equal superposition of states $|0\rangle$ and $|1\rangle$. This is confirmed by the measurement results observed in Fig.5.3.(a), which vary randomly between states $|000\rangle$ and $|010\rangle$.

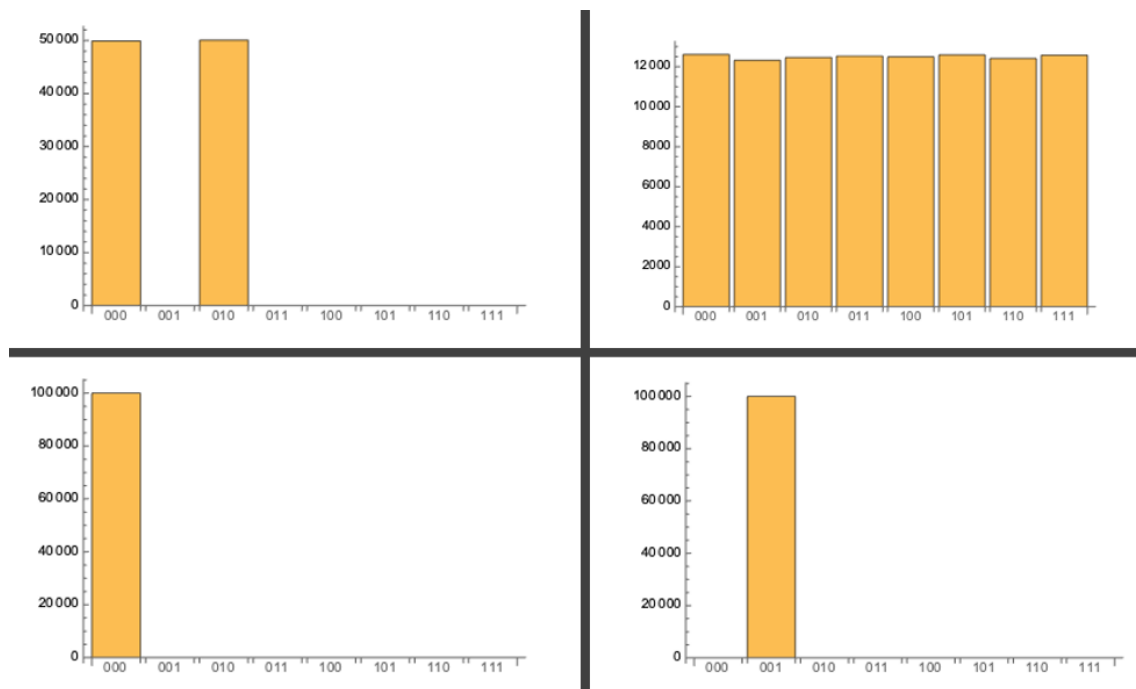


Figure 5. 3. Results of the calculations in programming project 2

In the calculation of Fig.5.2(b), Hadamard gates are applied to each of the qubits. This will leave each of the three in an equal superposition of states $|0\rangle$ and $|1\rangle$ and, therefore, the result varies randomly among the eight possible basis states, as we can see in Fig.5.3(b).

To simulate the circuit of Fig.5.2(c), two successive Hadamard gates are applied to the same qubit, so it will remain unchanged and the measurement result should always be $|000\rangle$. This can also be checked in Fig.5.3(c).

In Fig.5.2(d) a first Hadamard gate is applied to qubit 3, which leaves it in an equal superposition of states $|0\rangle$ and $|1\rangle$. Next, the phase shift phase, with $\theta = \pi$, generates a half-turn on the Bloch sphere about the z-axis. By applying another Hadamard gate on

the resulting state, qubit 3 is left in state $|1\rangle$. We can see these operations represented on the Bloch sphere in Fig.5.4. As a result of the computation, the measurement should always give $|001\rangle$, which can be seen in Fig.5.3(d).

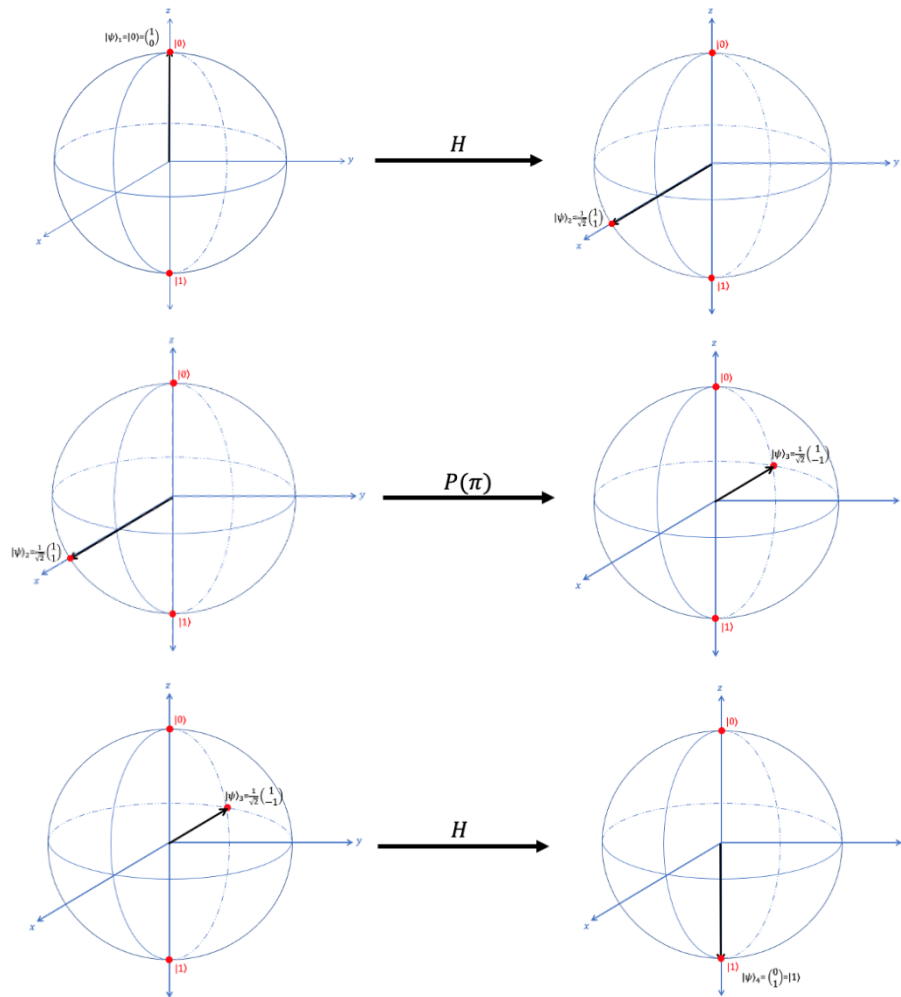


Figure 5. 4. Computation of Fig.5.2(d) represented on the Bloch sphere

5.3. Programming project 3: Implement Grover's quantum search

Imagine I ask you to think of a number from 1 to 8, which are numbers that we can encode with 3 qubits. Classically, if I wanted to guess that number, I could be lucky enough to get it right the first time or unlucky enough to go through all of them before getting to the right one. This means that the average number of tries needed to get to the correct answer is 4.

Generally speaking, if we wanted to perform a search in a database with N entries, we would need to check an average of $N/2$ entries to find the desired one. In our case of eight numbers this does not seem like much, but in databases with thousands or millions of entries, the search time increases considerably.

Using Grover's search algorithm, whose circuit diagram can be seen in Fig.5.5, the average number of entries in the database that we must consult is reduced to $(\pi/4)\sqrt{D}$. If we are talking about a database with one million entries, this reduces the search from 500,000 to 785 attempts on average, which represents a clear example of quantum supremacy.

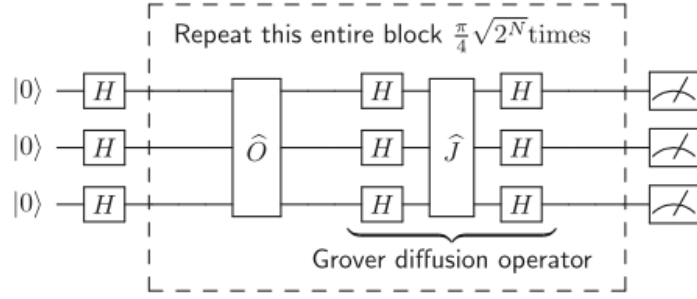


Figure 5. 5. Quantum circuit diagram for Grover's algorithm

To perform the Grover search, the database must be put in quantum oracle form. The quantum oracle (\hat{O}) is like the identity matrix, except that it has -1 as the diagonal element for the correct answer. If, emulating the above case, I try to guess a number from 1 to 8 but with a quantum search, obviously I should not know the correct answer beforehand, so the oracle really must be like a black box. To each element of the diagonal would correspond a number from 1 to 8 and you should put -1 in the correct position without telling me which one it is. In 5.8 the correct answer is number 7.

$$\hat{O} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad (5.8)$$

In addition to Hadamard gates, a special operator \hat{J} will be needed, whose matrix is always as follows:

$$\hat{J} = \begin{bmatrix} -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad (5.9)$$

In summary, first the 3-qubit register is initialized to state $|000\rangle$ and three H gates are applied to it to create a superposition of the eight basis states. Then the oracle changes the input state of the correct answer to -1. The Grover diffusion operator, shown in Fig.5.5, is used to amplify the amplitude of the correct answer. If we imagine it on Bloch's sphere, each repetition of the dotted block, which we will call the iteration step, rotates the state of the system to bring it a little closer to the correct answer.

Since the number of iterations must be close to $(\pi/4)\sqrt{D}$, and in our case $D=8$, we will repeat the step twice.

$$\left(\frac{\pi}{4}\right)\sqrt{8} = 2.22 \approx 2$$

The measurement results are shown in Fig.5.6. We can see that the result is $|110\rangle$ around or above 90% of the time in all cases. The state $|110\rangle$ corresponds to position 7, which is the number we were looking for. Performing 7 measurements (Fig.5.6(a)) gives only one wrong result, which means that if you performed only one measurement you could get it right at most at the second attempt. The code used for the computation was as follows:

```
In[654]:=  $\psi_{out} = (H3.H2.H1.J.H3.H2.H1.Oracle) . (H3.H2.H1.J.H3.H2.H1.Oracle) . H3.H2.H1.\psi_{in}$ 
```

```
Out[654]=  $\left\{ -\frac{1}{8\sqrt{2}}, -\frac{1}{8\sqrt{2}}, -\frac{1}{8\sqrt{2}}, -\frac{1}{8\sqrt{2}}, -\frac{1}{8\sqrt{2}}, -\frac{1}{8\sqrt{2}}, \frac{11}{8\sqrt{2}}, -\frac{1}{8\sqrt{2}} \right\}$ 
```

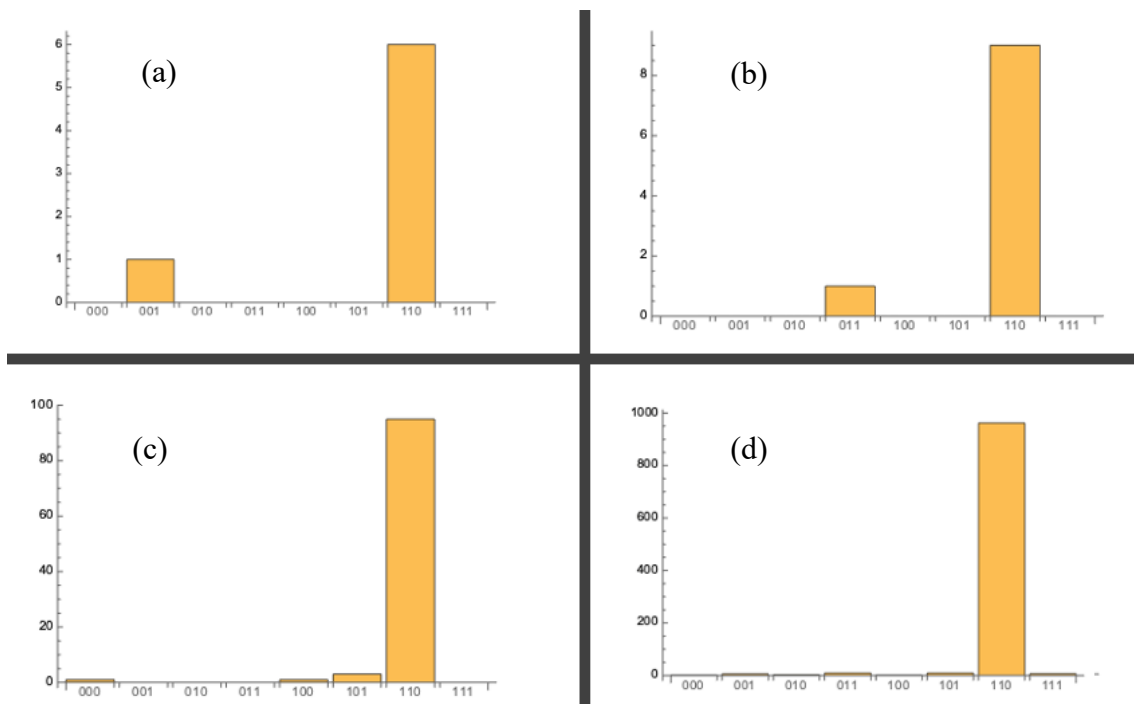


Figure 5. 6. Results of Grover's search with correct answer $|110\rangle$ increasing the number of repetitions of the measurement: (a) 7 times. (b) 10 times. (c) 100 times. (d) 1000 times

In Fig.5.7 we can also see the results of two computations in which the correct answer was changed to $|010\rangle$ and the number of iterations also varied. As a result, the percentage of times the answer is correct decreases.

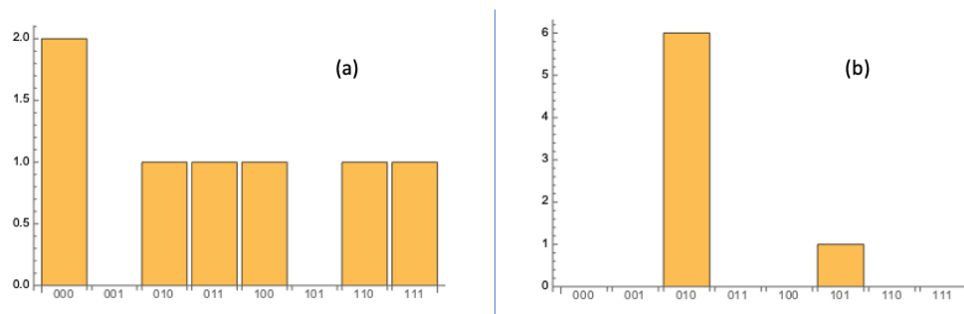


Figure 5. 7. Results of Grover's search with correct answer $|010\rangle$. (a) With 3 iterations. (b) Only one iteration step.

With the previous examples we have been able to see the power of Grover's algorithm, which is able to arrive at the correct answer in very few runs. On the other hand, the weak point of the algorithm is also demonstrated, since its answer is probabilistic, like quantum mechanics, and not deterministic, as in classical physics.

6. Conclusions

The information contained in this work, especially the examples put into practice in chapter 5, has allowed us to illustrate the power of quantum computations. Moreover, it gives us an idea of the classical resources needed to simulate them.

Although we have not been able to perform these computations on a quantum computer, the results obtained have shown us a small proof of quantum supremacy. This may lead us to think that a quantum computer would be exponentially more powerful than a classical one with the same number of bits. However, both quantum computers and quantum algorithms are currently at an early age, so the results they offer are not much better than those of a classical computer in most situations. We may not have to wait long for this to change, judging by the speed at which advances are being made in both quantum physics and material science.

It is important to note that, although the results have not yet surpassed classical computation for the most part, the development of quantum computing has moved us from seeing quantum systems as phenomena that must be explained as they are found in nature, to seeing them as systems that we can design and create. This has led to the development of ingenious algorithms to simulate, in a classical way, quantum systems that were originally considered hard to simulate. The ability to perform these simulations is expected to be a significant advance in the field of chemistry. [32] [33] [34]

While the project may seem at first sight only loosely related to the contents of the Grade in Chemistry, the knowledge of the fundamentals of quantum mechanics and computation, together with self-instructing work through surveying of specialized literature, have eventually culminated in this report, which provides a self-contained explanation to the basics of the quantum computation for graduates in Chemistry, thus paving the way for future students who feel attracted by this fascinating and emerging field

Annex. Full computation project code

```

In[552]= ClearAll["Global`*"]
In[553]= (* 1.Initialization and measurement of a 3-qubit register *)
In[554]= (*Set the number of qubits and create a vector for it's state*)
In[555]= nqubit = 3;
In[556]=  $\psi$  = Table[0, 2 ^ nqubit]
Out[556]= {0, 0, 0, 0, 0, 0, 0, 0}

In[557]= (*Create a vector with all the basis states*)
In[558]= B[nqubit_] := Tuples[{0, 1}, nqubit]
In[559]= base = B[nqubit]
Out[559]= {{0, 0, 0}, {0, 0, 1}, {0, 1, 0}, {0, 1, 1}, {1, 0, 0}, {1, 0, 1}, {1, 1, 0}, {1, 1, 1}}

In[560]= (*Give values for some positions of the state vector*)
In[561]=  $\psi$ [[1]] = 1.0;
In[562]=  $\psi$ [[8]] = 1.0;
In[563]=  $\psi$ 
Out[563]= {1., 0, 0, 0, 0, 0, 0, 1.}

In[564]= (* Normalize *)
In[565]= norm =  $\sum_{i=1}^{2^{nqubit}}$  Conjugate[ $\psi$ [[i]]] *  $\psi$ [[i]]
Out[565]= 2.

In[566]=  $\psi$  =  $\frac{1}{\text{Sqrt}[\text{norm}]}$  *  $\psi$ 
Out[566]= {0.707107, 0., 0., 0., 0., 0., 0., 0.707107}

In[567]= (*Create a vector with probabilities*)
In[568]=  $\psi$ 2 = Conjugate[ $\psi$ ] *  $\psi$ 
Out[568]= {0.5, 0., 0., 0., 0., 0., 0., 0.5}

In[569]= (* Performing a single measurement *)
In[570]= r = RandomReal[]
Out[570]= 0.598784

In[571]= q = 0;
Do[q = q +  $\psi$ 2[[n]];
Print["q=", q, " ", "r=", r];
If[q > r, Print["measurement=", base[[n]]];
Break[]], {n, 2 ^ nqubit}

q=0.5 r=0.598784
q=0.5 r=0.598784
q=0.5 r=0.598784
q=0.5 r=0.598784
q=0.5 r=0.598784
q=0.5 r=0.598784
q=0.5 r=0.598784
q=1. r=0.598784
measurement={1, 1, 1}

```

```

In[572]:= (* Changing initialization state *)
In[573]:=  $\psi$ cat = Table[0, 2 ^ nqubit];
In[574]:=  $\psi$ cat[[1]] = 1.0;
            $\psi$ cat[[2]] = 1.0;
            $\psi$ cat[[3]] = 1.0;
            $\psi$ cat[[4]] = 1.0;
            $\psi$ cat[[5]] = 1.0;
            $\psi$ cat[[6]] = 1.0;
            $\psi$ cat[[7]] = 1.0;
            $\psi$ cat[[8]] = 1.0;

In[575]:=  $\psi$ cat
Out[575]= {1., 1., 1., 1., 1., 1., 1., 1.}

In[576]:= norm =  $\sum_{i=1}^{2^n \text{nqubit}}$  Conjugate[ $\psi$ cat[[i]]] *  $\psi$ cat[[i]]
Out[576]= 8.

In[577]:=  $\psi$ cat =  $\frac{1}{\text{Sqrt}[\text{norm}]}$  *  $\psi$ cat
Out[577]= {0.353553, 0.353553, 0.353553, 0.353553, 0.353553, 0.353553, 0.353553, 0.353553}

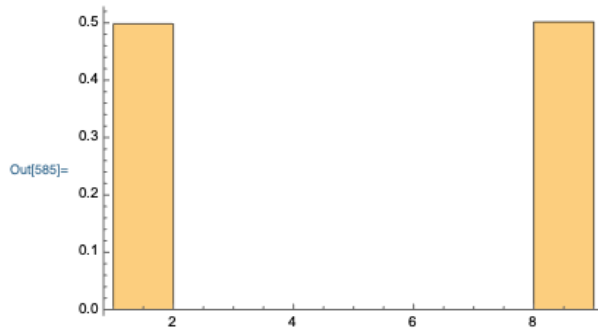
In[578]:=  $\psi$ cat2 = Conjugate[ $\psi$ cat] *  $\psi$ cat
Out[578]= {0.125, 0.125, 0.125, 0.125, 0.125, 0.125, 0.125, 0.125}
In[579]:= r = RandomReal[]
Out[579]= 0.92434

In[580]:= q = 0;
           Do[q = q +  $\psi$ cat2[[n]];
             Print["q=", q, " ", "r=", r];
             If[q > r, Print["measurement=", base[[n]]];
               Break[]], {n, 2 ^ nqubit}

           q=0.125    r=0.92434
           q=0.25    r=0.92434
           q=0.375    r=0.92434
           q=0.5     r=0.92434
           q=0.625    r=0.92434
           q=0.75    r=0.92434
           q=0.875    r=0.92434
           q=1.     r=0.92434
           measurement={1, 1, 1}

In[581]:= (* Performing Ntrials measurements recorded in a table of dimension Ntrials *)
In[582]:= Ntrials = 100 000;
In[583]:= measurement = Table[0, Ntrials];
In[584]:= Do[r = RandomReal[];
           q = 0;
           Do[q = q +  $\psi$ 2[[n]];
             If[q > r, result = n;
               Break[]], {n, 2 ^ nqubit}];
           measurement[[i]] = result, {i, Ntrials}
In[585]:= Histogram[measurement, {1}, Probability]

```



In[586]= (* Performing Ntrial measurements by adding (+1) to the position of the measured state in an nqubit dimension table *)

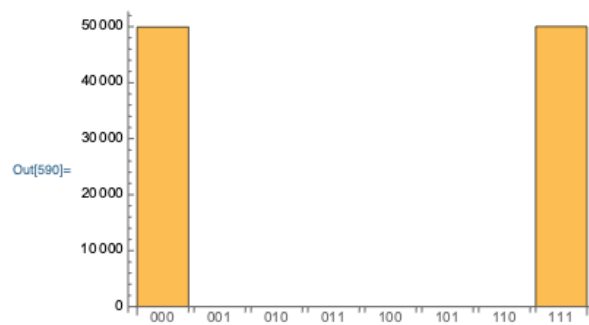
In[587]= measurement2 = Table[0, 2 ^ nqubit];

```
In[588]= Do[r = RandomReal[];
  q = 0;
  Do[q = q + ψ2[[n]];
    If[q > r, result = n;
      Break[]], {n, 2 ^ nqubit}];
  measurement2[[result]] = measurement2[[result]] + 1, {i, Ntrials}]
```

In[589]= measurement2

Out[589]= {49 947, 0, 0, 0, 0, 0, 0, 50 053}

```
In[590]= BarChart[measurement2,
  ChartLabels -> {"000", "001", "010", "011", "100", "101", "110", "111"}]
```



In[591]= (* Cat state measurement *)

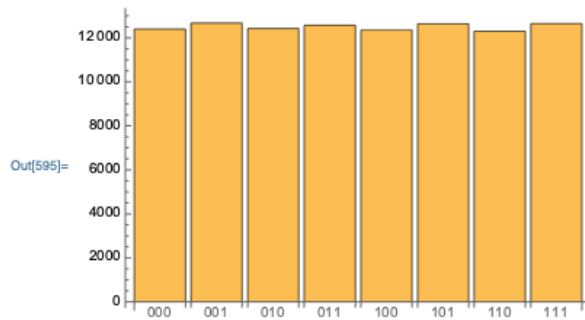
In[592]= measurementcat = Table[0, 2 ^ nqubit];

```
In[593]= Do[r = RandomReal[];
  q = 0;
  Do[q = q + ψcat2[[n]];
    If[q > r, result = n;
      Break[]], {n, 2 ^ nqubit}];
  measurementcat[[result]] = measurementcat[[result]] + 1, {i, Ntrials}]
```

In[594]= measurementcat

Out[594]= {12 396, 12 675, 12 423, 12 572, 12 353, 12 641, 12 297, 12 643}

```
In[595]= BarChart[measurementcat,
  ChartLabels -> {"000", "001", "010", "011", "100", "101", "110", "111"}]
```



```

In[596]=
In[597]= (* 2.First full quantum computacions with a 3-qubit register *)
In[598]=  $\psi_{in} = \{1, 0, 0, 0, 0, 0, 0, 0\}$ ;
In[599]= (*Hadamard gate acting on qubit 1*)
In[600]=  $H1 = \frac{1}{\sqrt{2}} * \{\{1, 0, 0, 0, 1, 0, 0, 0\}, \{0, 1, 0, 0, 0, 1, 0, 0\}, \{0, 0, 1, 0, 0, 0, 1, 0\},$ 
 $\{0, 0, 0, 1, 0, 0, 0, 1\}, \{1, 0, 0, 0, -1, 0, 0, 0\}, \{0, 1, 0, 0, 0, -1, 0, 0\},$ 
 $\{0, 0, 1, 0, 0, 0, -1, 0\}, \{0, 0, 0, 1, 0, 0, 0, -1\}\}$ ;
In[601]= (*Hadamard gate acting on qubit 2*)
In[602]=  $H2 = \frac{1}{\sqrt{2}} * \{\{1, 0, 1, 0, 0, 0, 0, 0\}, \{0, 1, 0, 1, 0, 0, 0, 0\}, \{1, 0, -1, 0, 0, 0, 0, 0\},$ 
 $\{0, 1, 0, -1, 0, 0, 0, 0\}, \{0, 0, 0, 0, 1, 0, 1, 0\}, \{0, 0, 0, 0, 0, 1, 0, 1\},$ 
 $\{0, 0, 0, 0, 1, 0, -1, 0\}, \{0, 0, 0, 0, 0, 1, 0, -1\}\}$ ;
In[603]= (*Hadamard gate acting on qubit 3*)
In[604]=  $H3 = \frac{1}{\sqrt{2}} * \{\{1, 1, 0, 0, 0, 0, 0, 0\}, \{1, -1, 0, 0, 0, 0, 0, 0\}, \{0, 0, 1, 1, 0, 0, 0, 0\},$ 
 $\{0, 0, 1, -1, 0, 0, 0, 0\}, \{0, 0, 0, 0, 1, 1, 0, 0\}, \{0, 0, 0, 0, 1, -1, 0, 0\},$ 
 $\{0, 0, 0, 0, 0, 0, 1, 1\}, \{0, 0, 0, 0, 0, 0, 1, -1\}\}$ ;
In[605]= (*Phase shift gate acting on qubit 3 *)
In[606]=  $phaseshift3 = \{\{1, 0, 0, 0, 0, 0, 0, 0\}, \{0, \text{Exp}[i*\theta], 0, 0, 0, 0, 0, 0\},$ 
 $\{0, 0, 1, 0, 0, 0, 0, 0\}, \{0, 0, 0, \text{Exp}[i*\theta], 0, 0, 0, 0\}, \{0, 0, 0, 0, 1, 0, 0, 0\},$ 
 $\{0, 0, 0, 0, 0, \text{Exp}[i*\theta], 0, 0\}, \{0, 0, 0, 0, 0, 0, -1, 0\},$ 
 $\{0, 0, 0, 0, 0, 0, 0, \text{Exp}[i*\theta]\}\}$ ;
In[608]= (* Circuit Fig. 3(a) Hadamard gate acting on qubit 2 *)
In[609]=  $\psi_{out} = H2.\psi_{in}$ 
Out[609]=  $\{\frac{1}{\sqrt{2}}, 0, \frac{1}{\sqrt{2}}, 0, 0, 0, 0, 0\}$ 
In[610]=  $\psi_{out2} = \text{Conjugate}[\psi_{out}] * \psi_{out}$ 
Out[610]=  $\{\frac{1}{2}, 0, \frac{1}{2}, 0, 0, 0, 0, 0\}$ 
In[611]=  $\text{measurement3} = \text{Table}[0, 2^n \text{qubit}]$ 
Out[611]=  $\{0, 0, 0, 0, 0, 0, 0, 0\}$ 
In[612]=  $\text{Do}[r = \text{RandomReal}[];$ 
 $q = 0;$ 
 $\text{Do}[q = q + \psi_{out2}[[n]];$ 
 $\text{If}[q > r, \text{result} = n;$ 
 $\text{Break}[]], \{n, 2^n \text{qubit}\}];$ 
 $\text{measurement3}[[\text{result}]] = \text{measurement3}[[\text{result}]] + 1, \{i, \text{Ntrials}\}$ 

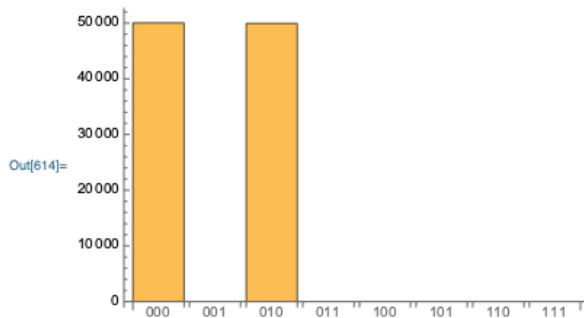
```

```
In[613]= measurement3
```

```
Out[613]= {50051, 0, 49949, 0, 0, 0, 0, 0}
```

```
In[614]= BarChart[measurement3,
```

```
ChartLabels -> {"000", "001", "010", "011", "100", "101", "110", "111"}]
```



```
In[615]= (* Circuit Fig. 3(b) Hadamard gates acting on each one of the qubits *)
```

```
In[616]=  $\psi_{\text{out}} = H3.H2.H1.\psi_{\text{in}}$ 
```

```
Out[616]=  $\left\{ \frac{1}{2\sqrt{2}}, \frac{1}{2\sqrt{2}}, \frac{1}{2\sqrt{2}}, \frac{1}{2\sqrt{2}}, \frac{1}{2\sqrt{2}}, \frac{1}{2\sqrt{2}}, \frac{1}{2\sqrt{2}}, \frac{1}{2\sqrt{2}} \right\}$ 
```

```
In[617]=  $\psi_{\text{out2}} = \text{Conjugate}[\psi_{\text{out}}] * \psi_{\text{out}}$ 
```

```
Out[617]=  $\left\{ \frac{1}{8}, \frac{1}{8}, \frac{1}{8}, \frac{1}{8}, \frac{1}{8}, \frac{1}{8}, \frac{1}{8}, \frac{1}{8} \right\}$ 
```

```
In[618]= measurement4 = Table[0, 2^nqubit]
```

```
Out[618]= {0, 0, 0, 0, 0, 0, 0, 0}
```

```
In[619]= Do[r = RandomReal[];
```

```
q = 0;
```

```
Do[q = q +  $\psi_{\text{out2}}[[n]$ ;
```

```
If[q > r, result = n;
```

```
Break[]], {n, 2^nqubit}];
```

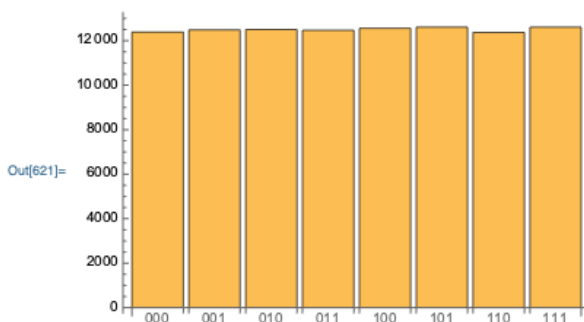
```
measurement4[result] = measurement4[result] + 1, {i, Ntrials}]
```

```
In[620]= measurement4
```

```
Out[620]= {12389, 12487, 12501, 12471, 12557, 12613, 12373, 12609}
```

```
In[621]= BarChart[measurement4,
```

```
ChartLabels -> {"000", "001", "010", "011", "100", "101", "110", "111"}]
```



```
In[622]= (* Circuit Fig. 3(c) *)
```

```
In[623]=  $\psi_{\text{out}} = H3.H3.\psi_{\text{in}}$ 
```

```
Out[623]= {1, 0, 0, 0, 0, 0, 0, 0}
```

```
In[624]=  $\psi_{\text{out2}} = \text{Conjugate}[\psi_{\text{out}}] * \psi_{\text{out}}$ 
```

```
Out[624]= {1, 0, 0, 0, 0, 0, 0, 0}
```



```
In[625]:= measurement5 = Table[0, 2^nqubit]
```

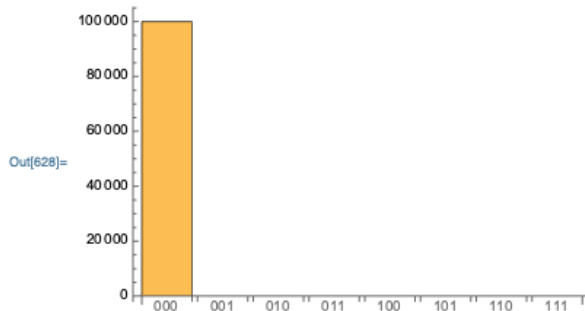
```
Out[625]= {0, 0, 0, 0, 0, 0, 0, 0}
```

```
In[626]:= Do[r = RandomReal[];  
q = 0;  
Do[q = q +  $\psi$ out2[[n]];  
If[q > r, result = n;  
Break[]], {n, 2^nqubit}];  
measurement5[result] = measurement5[result] + 1, {i, Ntrials}]
```

```
In[627]:= measurement5
```

```
Out[627]= {100 000, 0, 0, 0, 0, 0, 0, 0}
```

```
In[628]:= BarChart[measurement5,  
ChartLabels -> {"000", "001", "010", "011", "100", "101", "110", "111"}]
```



```
In[629]:= (* Circuit Fig. 3(d) *)
```

```
In[630]:=  $\theta = \pi$ ;
```

```
In[631]:=  $\psi$ out = H3.phaseshift3.H3. $\psi$ in
```

```
Out[631]= {0, 1, 0, 0, 0, 0, 0, 0}
```

```
In[632]:=  $\psi$ out2 = Conjugate[ $\psi$ out] *  $\psi$ out
```

```
Out[632]= {0, 1, 0, 0, 0, 0, 0, 0}
```

```
In[633]:= measurement6 = Table[0, 2^nqubit]
```

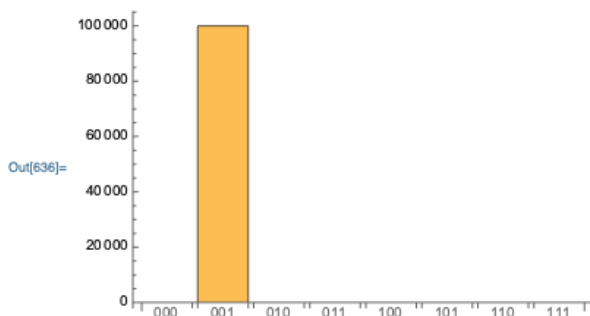
```
Out[633]= {0, 0, 0, 0, 0, 0, 0, 0}
```

```
In[634]:= Do[r = RandomReal[];  
q = 0;  
Do[q = q +  $\psi$ out2[[n]];  
If[q > r, result = n;  
Break[]], {n, 2^nqubit}];  
measurement6[result] = measurement6[result] + 1, {i, Ntrials}]
```

```
In[635]:= measurement6
```

```
Out[635]= {0, 100 000, 0, 0, 0, 0, 0, 0}
```

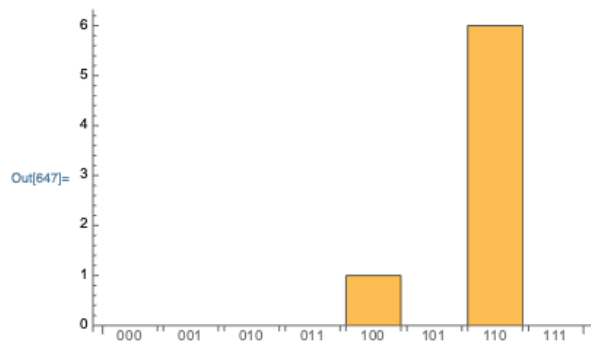
```
In[636]:= BarChart[measurement6,  
ChartLabels -> {"000", "001", "010", "011", "100", "101", "110", "111"}]
```



```

In[637]:= (* 3. Grover's quantum search algorithm *)
In[638]:= Ntrials = 7;
In[639]:= Oracle = {{1, 0, 0, 0, 0, 0, 0, 0}, {0, 1, 0, 0, 0, 0, 0, 0}, {0, 0, 1, 0, 0, 0, 0, 0},
  {0, 0, 0, 1, 0, 0, 0, 0}, {0, 0, 0, 0, 1, 0, 0, 0}, {0, 0, 0, 0, 0, 1, 0, 0},
  {0, 0, 0, 0, 0, 0, -1, 0}, {0, 0, 0, 0, 0, 0, 0, 1}};
In[640]:= J = {{-1, 0, 0, 0, 0, 0, 0, 0}, {0, 1, 0, 0, 0, 0, 0, 0}, {0, 0, 1, 0, 0, 0, 0, 0},
  {0, 0, 0, 1, 0, 0, 0, 0}, {0, 0, 0, 0, 1, 0, 0, 0}, {0, 0, 0, 0, 0, 1, 0, 0},
  {0, 0, 0, 0, 0, 0, 1, 0}, {0, 0, 0, 0, 0, 0, 0, 1}};
In[641]:=  $\frac{\pi}{4} \sqrt{2^{\text{nqubit}}} // \text{N}$ 
Out[641]:= 2.22144
In[642]:=  $\psi_{\text{out}} = (\text{H3.H2.H1.J.H3.H2.H1.Oracle}) . (\text{H3.H2.H1.J.H3.H2.H1.Oracle}) . \text{H3.H2.H1.}\psi_{\text{in}}$ 
Out[642]:=  $\left\{ -\frac{1}{8\sqrt{2}}, -\frac{1}{8\sqrt{2}}, -\frac{1}{8\sqrt{2}}, -\frac{1}{8\sqrt{2}}, -\frac{1}{8\sqrt{2}}, -\frac{1}{8\sqrt{2}}, \frac{11}{8\sqrt{2}}, -\frac{1}{8\sqrt{2}} \right\}$ 
In[643]:=  $\psi_{\text{out2}} = \text{Conjugate}[\psi_{\text{out}}] * \psi_{\text{out}}$ 
Out[643]:=  $\left\{ \frac{1}{128}, \frac{1}{128}, \frac{1}{128}, \frac{1}{128}, \frac{1}{128}, \frac{1}{128}, \frac{121}{128}, \frac{1}{128} \right\}$ 
In[644]:= measurement7 = Table[0, 2^nqubit]
Out[644]:= {0, 0, 0, 0, 0, 0, 0, 0}
In[645]:= Do[r = RandomReal[];
  q = 0;
  Do[q = q +  $\psi_{\text{out2}}[[n]]$ ;
  If[q > r, result = n;
  Break[], {n, 2^nqubit}];
  measurement7[[result]] = measurement7[[result]] + 1, {i, Ntrials}]
In[646]:= measurement7
Out[646]:= {0, 0, 0, 0, 1, 0, 6, 0}
In[647]:= BarChart[measurement7,
  ChartLabels -> {"000", "001", "010", "011", "100", "101", "110", "111"}]

```



```

Out[647]:=

```

```

In[648]:= (* Different oracle and different number of repetitions *)
In[649]:= Oracle = {{1, 0, 0, 0, 0, 0, 0, 0}, {0, 1, 0, 0, 0, 0, 0, 0}, {0, 0, -1, 0, 0, 0, 0, 0},
  {0, 0, 0, 1, 0, 0, 0, 0}, {0, 0, 0, 0, 1, 0, 0, 0}, {0, 0, 0, 0, 0, 1, 0, 0},
  {0, 0, 0, 0, 0, 0, 1, 0}, {0, 0, 0, 0, 0, 0, 0, 1}};
In[650]:=  $\psi_{\text{out}} = (\text{H3.H2.H1.J.H3.H2.H1.Oracle}) . (\text{H3.H2.H1.J.H3.H2.H1.Oracle}) .$ 
   $(\text{H3.H2.H1.J.H3.H2.H1.Oracle}) . \text{H3.H2.H1.}\psi_{\text{in}}$ 
Out[650]:=  $\left\{ \frac{7}{16\sqrt{2}}, \frac{7}{16\sqrt{2}}, -\frac{13}{16\sqrt{2}}, \frac{7}{16\sqrt{2}}, \frac{7}{16\sqrt{2}}, \frac{7}{16\sqrt{2}}, \frac{7}{16\sqrt{2}}, \frac{7}{16\sqrt{2}} \right\}$ 

```

```

In[651]=  $\psi_{out2} = \text{Conjugate}[\psi_{out}] * \psi_{out}$ 
Out[651]=  $\left\{ \frac{49}{512}, \frac{49}{512}, \frac{169}{512}, \frac{49}{512}, \frac{49}{512}, \frac{49}{512}, \frac{49}{512}, \frac{49}{512} \right\}$ 

In[652]= measurement8 = Table[0, 2^nqubit]
Out[652]= {0, 0, 0, 0, 0, 0, 0, 0}

In[653]= Do[r = RandomReal[];
q = 0;
Do[q = q +  $\psi_{out2}[[n]$ ;
If[q > r, result = n;
Break[]], {n, 2^nqubit}];
measurement8[result] = measurement8[result] + 1, {i, Ntrials}]

In[654]= measurement8
Out[654]= {2, 1, 0, 1, 1, 1, 0, 1}

In[655]= BarChart[measurement8,
ChartLabels -> {"000", "001", "010", "011", "100", "101", "110", "111"}]
Out[655]= 
In[656]=  $\psi_{out} = (\text{H3.H2.H1.J.H3.H2.H1.Oracle}).\text{H3.H2.H1}.\psi_{in}$ 
Out[656]=  $\left\{ -\frac{1}{4\sqrt{2}}, -\frac{1}{4\sqrt{2}}, -\frac{5}{4\sqrt{2}}, -\frac{1}{4\sqrt{2}}, -\frac{1}{4\sqrt{2}}, -\frac{1}{4\sqrt{2}}, -\frac{1}{4\sqrt{2}}, -\frac{1}{4\sqrt{2}} \right\}$ 

In[657]=  $\psi_{out2} = \text{Conjugate}[\psi_{out}] * \psi_{out}$ 
Out[657]=  $\left\{ \frac{1}{32}, \frac{1}{32}, \frac{25}{32}, \frac{1}{32}, \frac{1}{32}, \frac{1}{32}, \frac{1}{32}, \frac{1}{32} \right\}$ 

In[658]= measurement9 = Table[0, 2^nqubit]
Out[658]= {0, 0, 0, 0, 0, 0, 0, 0}

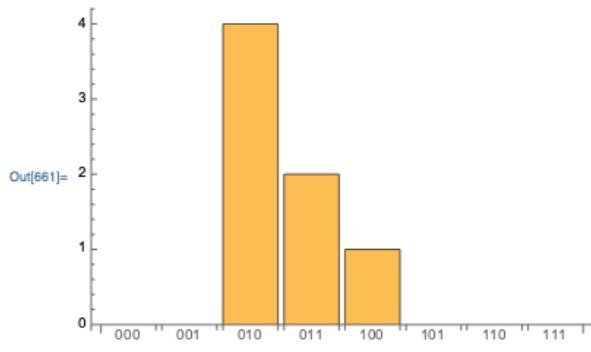
In[658]= measurement9 = Table[0, 2^nqubit]
Out[658]= {0, 0, 0, 0, 0, 0, 0, 0}

In[659]= Do[r = RandomReal[];
q = 0;
Do[q = q +  $\psi_{out2}[[n]$ ;
If[q > r, result = n;
Break[]], {n, 2^nqubit}];
measurement9[result] = measurement9[result] + 1, {i, Ntrials}]

In[660]= measurement9
Out[660]= {0, 0, 4, 2, 1, 0, 0, 0}

In[661]= BarChart[measurement9,
ChartLabels -> {"000", "001", "010", "011", "100", "101", "110", "111"}]

```



43

References

- [1] M. A. Nielsen and I. Chuang, Quantum computation and quantum information, 2002.
- [2] P. Dirac, The Principles of Quantum Mechanics (2^a ed.), Clarendon Press, 1947.
- [3] E. Schrödinger, "Discussion of probability relations between separated systems", *Mathematical Proceedings of the Cambridge Philosophical Society*, vol. 4, no. 31, pp. 555-563, 1935.
- [4] D. Deutsch and R. Jozsa, "Rapid solution of problems by quantum computation", *Proceedings of the Royal Society*, vol. A, no. 439, pp. 553-558, 1992.
- [5] D. Stebila, M. Mosca and N. Lütkenhaus, "The Case for Quantum Key Distribution", *Quantum Communication and Quantum Networking*, no. 36, pp. 283-296, 2010.
- [6] T. ohnson, S. Clark and D. Jaksch, "What is a quantum simulator", *EPJ Quantum Technology*, vol. 1, no. 10, 2014.
- [7] P. Bajpai, "Quantum computing: how to invest in it, and which companies are leading the way?", Nasdaq, 2020.
- [8] P. Benioff, "The Computer as a Physical System: A Microscopic Quantum Mechanical Hamiltonian Model of Computers as Represented by Turing Machines", *Journal of Statistical Physics*, vol. 5, no. 22, pp. 563-591, 1980.
- [9] R. Feynman, "Simulating Physics with Computers", *International Journal of Theoretical Physics*, vol. 6/7, no. 21, pp. 467-488, 1982.
- [10] Y. I. Manin, Vychislimoe i nevychislimoe [Computable and Noncomputable] (in Russian), Movska: Sovetskoe Radio, 1980.
- [11] D. Hilbert and W. Ackermann, Grundzüge der theoretischen Logik (Principles of Mathematical Logic), Berlín: Springer, 1928.
- [12] A. Church, "An unsolvable problem of elementary number theory", *American Journal of Mathematics*, no. 58, pp. 345-363, 1936.
- [13] A. Turing, "On Computable Numvers, with an Application to the Entscheidungsproblem", *Proceedings of the London Mathematical Society*, vol. 2, no. 42, pp. 230-265, 1936.
- [14] S. C. Kleene, Introduction to Metamathematics, North-Holland, 1952.
- [15] S. P. W., "Algorithms for quantum computation: discrete logarithms and factoring", in *Proceedings 35th Annual Symposium on Foundations of Computer Science*, 1994.
- [16] P. Kaye, R. Laflamme and M. Mosca, An introduction to quantum computing, Oxford University Press., 2007.
- [17] E. Knill, R. Laflamme and L. Viola, "Theory of Quantum Error Correction for General Noise", *Physical Review Letters*, vol. 11, no. 84, pp. 2525-2528, 2000.
- [18] A. S. Holevo, C. M. Caves, H. P. Yuen and L. Accardi, Quantum Communication, Computing, and Measurement, Springer US, 1997.
- [19] "IBM Building First Universal Quantum Computers for Business and Science", *IBM News Room*, 06 March 2017.
- [20] "IBM Unveil's World's First Integrated Quantum Computing System for Commercial Use", *IBM News Room*, 8 January 2019.

- [21] F. Tavares, "Google and NASA Achieve Quantum Supremacy", *NASA's Ames Research Center*, 23 October 2019.
- [22] L. K. Grover, "A fast quantum mechanical algorithm for database search", in *Proceedings of the 28th Annual ACM Symposium of the Theory of Computing*, 1996.
- [23] J. C. Amengual and V. R. Tomás, *Informática básica. Conceptos básicos de informática*, Castelló de la Plana: Publicacions de la Universitat Jaume I. Servei de Comunicació i Publicacions .
- [24] F. Schwabl, *Quantum Mechanics*, Springer-Verlag Berlin Heidelberg, 2007.
- [25] J. Planelles, *Noves notes de Química Quàntica*, Castelló de la Plana: Publicacions de la Universitat Jaume I. Servei de Comunicació i Publicacions, 2010.
- [26] J. Bertran, V. Branchadell, M. Moreno and M. Sodupe, *Química cuántica*, Madrid: Ed. Síntesis, S.A., 2002.
- [27] E. G. Rieffel and W. H. Polak, *Quantum Computing. A Gentle Introduction*, MIT Press, 2011.
- [28] E. Gibney, "Quantum computer quest", *Nature*, vol. 516, pp. 24-26, 2014.
- [29] Qubit, "Wikipedia", Available: https://en.wikipedia.org/wiki/Qubit#Physical_implementations. [Accessed July 2021].
- [30] S. E. Economou, J. I. Climente, A. Badolato, A. S. Bracker, D. Gammon and M. F. Doty, "Scalable qubit architecture based on holes in quantum dot molecules", *Physical Review B*, vol. 86, 2012.
- [31] D. Candela, "Undergraduate computational physics projects on quantum computing", *American Journal of Physics*, vol. 83, no. 8, pp. 688-702, 2015.
- [32] Y. Cao, J. Romero, J. P. Olson, M. Degroote, P. D. Johnson, M. Kieferová, I. D. Kivlichan, T. Menke, B. Peropadre, N. P. D. Sawaya, S. Sim, L. Veis and A. Aspuru-Guzik, "Quantum Chemistry in the Age of Quantum Computing", *Chemical Reviews*, vol. 119, no. 19, pp. 10856-10915, 2019.
- [33] E. A. Walker and S. Addamane Pallathadka, "How a Quantum Computer Could Solve a Microkinetic Model", *The Journal of Physical Chemistry Letters*, no. 12, pp. 592-597, 2021.
- [34] B. Bauer, S. Bravyi, M. Motta and G. Kin-Lic Chan, "Quantum Algorithms for Quantum Chemistry and Quantum Material Science", *Chemical Reviews*, vol. 120, no. 22, pp. 12685-12717, 2020.
- [35] D. J. P. E. Hodgson, *First Order Logic*, Philadelphia: Saint Joseph's University, 1995.