ALGEBRAIC TOOLS IN QUANTUM COMPUTING ALGORITHMS

(Universal Quantum Gates)

Lucía Absalom Bautista

ALGEBRAIC TOOLS IN QUANTUM COMPUTING ALGORITHMS

(Universal Quantum Gates)

Lucía Absalom Bautista

Memoria presentada como parte de los requisitos para la obtención del título de Grado en Matemáticas por la Universidad de Sevilla.

Tutorizada por

José M. Tornero Sánchez

Índice general

Abstract / Resumen **1**

1. Introduction **3**

2. Basic concepts of quantum computing **5**

2.1. Why is algebra important for quantum computing 5

2.2. A first word on unitary operators 6

2.3. Qubits 8

2.4. Single qubit gates 11

2.5. Tensor product 20

2.6. Quantum entanglement 25

2.7. Multiple qubit gates 26

3. Universal quantum gates **31**

3.1. First steps 32

3.2. Exact quantum gates 33

3.2.1. 2–level unitary matrices 34

3.2.2. 2–level unitary matrices can be used to factorize any unitary matrix 35

3.2.3. $\mathbf{C_{NOT}}$ gates and single qubit gates are universal 39

3.3. A finite universal gate set 46

3.3.1. Approximating unitary operators 46

3.3.2. Finding a finite universal set for 1-qubit operations 48

3.4. A note about complexity 52

Abstract / Resumen

Abstract

Universal quantum gates play a crucial role in quantum computing as they form the building blocks for quantum circuits. Unlike classical circuits which are made up of classical gates such as AND, OR, and NOT gates, quantum circuits are composed of quantum gates that operate on qubits instead of classical bits. These gates can be used to perform various operations on qubits such as rotations, phase shifts, and entanglement, allowing for the execution of quantum algorithms.

In this dissertation, we will explore the fundamental principles behind universal quantum gates and their significance in quantum computing. We will examine the various types of quantum gates including the single-qubit gates such as the Hadamard gate, Pauli gates, and phase gates, as well as the multi-qubit gates such as the $\mathbf{C_{NOT}}$ gate.

This work aims to provide an introductory overview of universal quantum gates and their applications in quantum computing, leaving aside the current state of the art in research and development of quantum gate technology. We hope that this dissertation will shed some light on the ongoing effort to build more powerful and reliable quantum computers that will pave the way for groundbreaking advances in a variety of fields, including cryptography, chemistry, and optimization.

Resumen

Las puertas cuánticas universales desempeñan un papel crucial en la computación cuántica, ya que forman los bloques de construcción de los circuitos cuánticos. A diferencia de los circuitos clásicos, formados por compuertas clásicas como AND, OR y NOT, los circuitos cuánticos se componen de compuertas cuánticas que operan con qubits en lugar de con bits clásicos. Estas puertas pueden utilizarse para realizar diversas operaciones sobre los qubits, como rotaciones, cambios de fase y entrelazamiento, lo que permite la ejecución de algoritmos cuánticos.

En esta memoria exploraremos los principios fundamentales de las puertas cuánticas universales y su importancia en la computación cuántica. Examinaremos los distintos tipos de puertas cuánticas, incluidas las puertas de un solo qubit, como la puerta de Hadamard, las puertas de Pauli y las puertas de fase, así como las puertas multiqubit, como la puerta $\mathbf{C_{NOT}}$.

El objetivo de este trabajo es proporcionar una visión introductoria de las puertas cuánticas universales y sus aplicaciones en la computación cuántica, sin entrar en el estado actual de la investigación y el desarrollo de la tecnología de puertas cuánticas. Esperamos que esta memoria arroje algo de luz sobre el esfuerzo en curso para construir ordenadores cuánticos más potentes y fiables que allanen el camino para avances revolucionarios en diversos campos, como la criptografía, la química y la optimización.

1 | Introduction

Quantum computing is a rapidly advancing field of research that has the potential to revolutionize computing as we know it. While traditional computing uses binary digits, or bits, to represent data as either 0 or 1, quantum computing uses quantum bits, or qubits, that can exist in multiple states simultaneously, allowing for vastly more complex calculations to be performed.

So, at least theoretically, quantum computers will be able to solve certain problems much faster than classical computers, including factoring large numbers and searching through large amounts of data. One could think of it as a library, with traditional computers searching for books one by one, while a quantum computer can search through all the books at once.

Quantum computing also has the potential to improve our understanding of fundamental physics, enable more secure communication through quantum cryptography, and accelerate the development of new materials and drugs through quantum simulations.

As a result, quantum computing is an area of growing interest for researchers, with major tech companies and governments investing significant resources into its development. However, there are still significant technical and practical challenges that need to be overcome before quantum computing can become a practical technology for widespread use.

Observation 1.1. As a general rule, when we cite or use linear algebra results without further details, they will be part of the syllabus of *Álgebra Lineal y Geometría I*. As for quantum computing definitions and results, two sources have been used the most: [7, 4]. Although there are no new results in this memoir, some of the results in these references have incomplete or imprecise proofs (in line with the rigour standard of the subject). We have consistently tried to overcome this, for most of the work.

Two exceptions to this rule are the Gottesman–Knill (section 3.1) and the Sovolay–Kitaev (section 3.4) theorems which are prominent results in this field, but their proofs fall well outside the scope of this memoir.

2 Basic concepts of quantum computing

This chapter introduces the main actors at play: qubits and (quantum) gates. The target is giving a rigorous mathematical structure to the different aspects of quantum computing. Most of the definitions come from [8, 6].

2.1 Why is algebra important for quantum computing

Algebra is essential for quantum computing because it provides the mathematical framework necessary to understand and manipulate the behaviour of quantum systems. Quantum computing relies heavily on linear algebra, which is the branch of algebra that deals with vector spaces and linear transformations.

In particular, the following algebraic concepts are central for quantum computing:

1. **Complex numbers**: Quantum mechanics is built upon complex numbers. In quantum computing, complex numbers are used to represent the states of qubits, which can be in a superposition (which are, essentially, certain linear combinations) of multiple states simultaneously.

2. **Vectors and matrices**: In quantum mechanics, the state of a system is represented by a vector in a complex vector space. These vectors can be manipulated using matrices, which represent quantum gates and operations on qubits.

3. **Inner product and unitary transformations**: The inner product is a way of measuring the similarity between two vectors in a vector space. In quantum computing, the inner product is used to calculate the probability of measuring a particular state of a qubit. Unitary transformations,

which are used to represent quantum gates, are linear transformations that preserve the inner product.

4. **Tensor products**: Quantum systems can be composed of multiple qubits, and the state of the entire system is represented by a tensor product of the states of each qubit. Tensor products are also able to represent entanglement, which is a phenomenon where the states of two or more qubits become correlated.

2.2 A first word on unitary operators

In classical computing, we can represent the state of a system as a binary number which can be modified using logical gates. Similarly, in probabilistic computing states are depicted as classical probability distributions and the operations that we perform on the system are represented as classical stochastic matrices. However, in quantum computing, the state of a system is represented by a superposition of quantum states (which is nothing more than a linear combination of vectors), and the operations that we perform on the system are represented by unitary operators.

A unitary operator is a linear transformation that preserves the inner product between vectors, and it does therefore preserves the norms of vectors. This property will be vital because unitary vectors will define a discrete probability distribution which will be a most interesting object related to this set–up.

Furthermore, unitary operators are reversible, meaning that they can be undone by applying their inverse. This is crucial feature, because quantum circuits are composed of a sequence of unitary operators that must be applied in a specific order to perform a desired computation. So we will be able to *undo* the computation and recover the initial state, as each unitary operator must have an inverse that can be applied in the reverse order.

Overall, unitary operators are essential in quantum computing because they enable the manipulation of quantum states and the implementation of quantum algorithms. They ensure that the probabilities associated with each quantum state are consistent during the computation, and they allow the computation to be reversed if necessary.

Definition 2.1. *Let H be a finite-dimensional Hilbert space. A unitary operator is a linear operator $U : H \to H$ which preserves the inner product of the Hilbert space, H. In other words, for all vectors x and y in H we have: $\langle Ux, Uy\rangle_H = \langle x, y\rangle_H$.*

Example 2.1. On finite-dimensional complex Hilbert spaces we can represent unitary operators by unitary matrices. In the case of $\mathbb{R}^n$ these matrices are called orthogonal matrices.

As we know from linear algebra, equivalently we have:

Definition 2.2. *A unitary operator is a linear operator $H \to H$ on a finite-dimensional (real or complex) Hilbert space H given by a mtrix U that satisfies $U^*U = UU^* = I$, where U^* is the conjugate transpose of U, and I is the identity matrix.*

Example 2.2. An example of unitary operator are rotations in $\mathbb{R}^2$ or $\mathbb{R}^3$. It is easy to see that these do not change the length of a vector or the angle between two vectors.

Example 2.3. Let us consider a unitary matrix

$$U = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \qquad \text{with } a, b, c, d \text{ in } \mathbb{C}.$$

We know from basic linear algebra that this matrix is unitary if and only if $U^{-1} = U^*$ or, expressed in another way:

$$UU^* = U^*U = I$$

This implies in particular

$$1 = \det(U) \cdot \det(U^*) = \det(U) \cdot \overline{\det(U)} \implies |\det(U)| = 1,$$

that is, $\det(U) = e^{i\theta}$, for some $\theta \in [0, 2\pi)$. Imposing this condition and equating U^* and U^{-1} term to term we find that U has to be:

$$U = \begin{bmatrix} a & b \\ -e^{i\theta}\overline{b} & e^{i\theta}\overline{a} \end{bmatrix}$$

We also have the condition $|a|^2 + |b|^2 = 1$ (from $UU^* = I$). This means

$$a = e^{i\alpha}\cos\xi, \qquad b = e^{i\beta}\sin\xi,$$

for some $\alpha, \beta, \xi \in [0, 2\pi)$ and we can express our matrix as:

$$U = \begin{bmatrix} e^{i\alpha}\cos\xi & e^{i\beta}\sin\xi \\ -e^{i(\theta-\beta)}\sin\xi & e^{i(\theta-\alpha)}\cos\xi \end{bmatrix}$$

2.3 Qubits

A qubit (short for quantum bit) is the basic unit of quantum information in quantum computing. It is similar to the classical bit used in classical computing, but unlike a classical bit, which can only take on a value of either 0 or 1, a qubit can be in a superposition of both 0 and 1 at the same time.

A qubit can be represented by a two-dimensional vector in a complex vector space, known as a quantum state. Let us introduce these concepts formally.

We will work in the Hilbert space $\mathbb{C}^2$, with the usual scalar product. In the quantum computing context it is customary to write the vector in Dirac notation:

$$u \longleftrightarrow |u\rangle, \qquad v^* \longleftrightarrow \langle v|.$$

In particular, the usual scalar product of u and v becomes $\langle v|u\rangle$.

Definition 2.3. *The vectors:*

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \text{ and } |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

are called the basis states of a quantum bit.

In linear algebra $\{|0\rangle, |1\rangle\}$ is called the usual / standard basis, but in the context of quantum computing they might be also called the computational basis.

In addition to these two states, qubits can be in a state of quantum superposition (again, this can be considered just argot for unitary linear combination), combining these two states in what is called a pure qubit state.

Definition 2.4. *A (pure) qubit state $|\psi\rangle$ is a unit vector in $\mathbb{C}^2$. When written as a linear combination of the basis*

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

the coefficients $\alpha, \beta \in \mathbb{C}$ are called the amplitudes of the state. Note that they must satisfy $|\alpha|^2 + |\beta|^2 = 1$.

Essentially $|\alpha|^2$ and $|\beta|^2$ determine a Bernoulli probability distribution interpreted as

$$\begin{aligned} P\big(|0\rangle\big) &= P\big(\text{observing the qubit and getting the state } |0\rangle\big) = |\alpha|^2, \\ P\big(|1\rangle\big) &= P\big(\text{observing the qubit and getting the state } |1\rangle\big) = |\beta|^2, \end{aligned}$$

where we should note $|\alpha|^2 = \overline{\alpha} \cdot \alpha = \langle \psi|0\rangle \cdot \langle 0|\psi\rangle$ (and similarly for β).

This distribution is the real object we are concerned about, as the qubits, at the end of all manipulations must be *measured* (that is, observed) and this forces them to collapse to a basis state, according to the probability distribution above.

The most interesting quantum algorithms take advantage of the qubit ability to exist in multiple states simultaneously, thus performing certain computations much faster than classical computers. However, qubits are also very delicate and susceptible to decoherence, which is the tendency to change of a quantum state due to environmental noise. This is one of the major challenges in building practical quantum computers.

The Bloch sphere

We already know that a pure qubit state can be expressed as a linear combination of the basis states $|0\rangle$ and $|1\rangle$, where the coefficients in front of the basis states determine the probability amplitudes of measuring the qubit in either state. These coefficients are complex numbers that can be written in polar form, where the modulus represents the probability amplitude and the argument represents the so-called *relative phase* between the states.

Observation 2.1. It could be the case that different qubits, say $|\psi\rangle$ and $|\xi\rangle$ might in fact give rise to the same probability distribution. For example, we can immediately deduce that this is the case if we take a $\theta \in [0, 2\pi)$ and define

$$|\psi\rangle = e^{i\theta}|\xi\rangle.$$

This phenomenon is usually called *change of phase*. As the probability distribution associated to both states is the same, which is all we actually care about, we will generally consider these qubit states as the same. This phenomenon will also arise later when we describe the equivalence of quantum gates.

If we allow ourselves to change the phase of a given qubit freely, then we can choose the first coordinate to be a non–negative real number and write the state of a pure qubit as

$$|\psi\rangle = \begin{bmatrix} \cos(\theta/2) \\ e^{i\varphi}\sin(\theta/2) \end{bmatrix} \qquad \text{where } \theta \in [0, \pi),\ \varphi \in [0, 2\pi).$$

The complex numbers $\cos(\theta/2)$ and $e^{i\varphi}\sin(\theta/2)$ are the coefficients associated with the basis states $|0\rangle$ and $|1\rangle$, respectively. Here, θ is the polar angle that determines the relative weight between the $|0\rangle$

and $|1\rangle$ states, and φ is the azimuthal angle that specifies the phase difference between them. The $1/2$ factor is introduced to arrange the interval as spherical coordinates.

The Bloch sphere is a unit sphere where the north pole represents the state $|0\rangle$ and the south pole represents the state $|1\rangle$ and, as, we will see, any other state of the qubit can also be represented. Precisely, to determine the Cartesian coordinates of the point on the Bloch sphere, we use the following formulas:

$$\begin{cases} x &= \sin\theta\cos\varphi \\ y &= \sin\theta\sin\varphi \\ z &= \cos\theta \end{cases}$$

Therefore, to visualize a qubit on the Bloch sphere, we need to know the amplitudes of the qubit (that is, the coordinates with respect to $\{|0\rangle, |1\rangle\}$) and then find φ and θ from them. Then we can map the probability amplitudes into coordinates on the sphere.

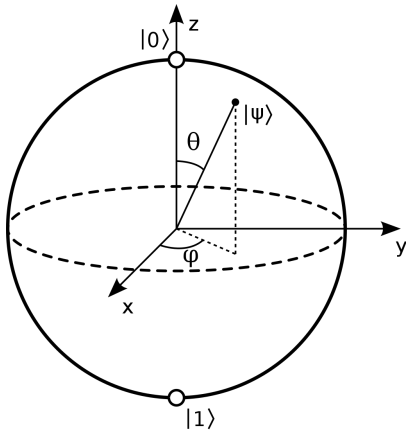


Figura 2.1: The Bloch sphere

The Bloch sphere allows us to intuitively understand various properties of the qubit. For example, if the point is located near the north pole, it indicates a high probability of measuring the qubit in the state $|0\rangle$, whereas if it is near the south pole, it indicates a high probability of measuring the qubit in the state $|1\rangle$.

We will now introduce a couple of states that, apart for being widely used, are very easy to visualize in the Bloch sphere:

The so called *+ state* is defined by

$$|+\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle = \frac{1}{\sqrt{2}}\begin{bmatrix}1\\1\end{bmatrix},$$

and its representation in the Bloch sphere is the point $(1, 0, 0)$, located along the equator, in the positive part of the x-axis.

On the other side , the *- state*, given by

$$|-\rangle = \frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle = \frac{1}{\sqrt{2}}\begin{bmatrix}1\\-1\end{bmatrix},$$

is represented by the point $(-1, 0, 0)$, also located along the equator but at an opposite point from the representation of the *+ state*.

The set $\{|+\rangle, |-\rangle\}$ is called the *Hadamard basis* and is, in fact, an orthonormal basis of $\mathbb{C}^2$.

In the following section, we will explore how we can alter a qubit. We will see that the Bloch sphere is a valuable tool for understanding and visualizing these single qubit operations, such as rotations and manipulations of the qubit state.

2.4 Single qubit gates

As we are working with unitary vectors in $\mathbb{C}^2$, if we want to preserve both the algebraic structure (essentially linear combinations) and the unitary character of our main objects we are bound to use unitary matrices, as we know from basic linear algebra.

Definition 2.5. *A 1-qubit quantum gate is a unitary matrix in $\mathcal{M}_2(\mathbb{C})$.*

We will usually refer to *gates* when they are operating on a quantum state, whereas we will use *matrices* for the purely mathematical object. They can be loosely assumed to be equivalent, though, and both will usually be written in boldface, as in $\mathbf{A}, \mathbf{B}, \mathbf{C}\ldots$

For the rest of the section we will introduce some important gates which will play a significant role in the sequel.

Definition 2.6. *Two unitary matrices, $\mathbf{A}$ and $\mathbf{B}$ are considered to be equivalent when $\mathbf{A} = z\mathbf{B}$, where z is a complex number of module 1.*

Unitary equivalent matrices represent gates that perform the same quantum operation up to a change of phase. This means that although the gate may have a different matrix representation, its effect on the quantum states is identical, and which basically means that they affect the probability distribution associated with the qubits in the same way.

Definition 2.7. *The Pauli gates (noted as* $(\mathbf{X}, \mathbf{Y}, \mathbf{Z})$ *or* $(\sigma_1, \sigma_2, \sigma_3)$*) are the three matrices acting on a single qubit which correspond, respectively, to a rotation around the x, y and z axes of the Bloch sphere by* π *radians (also called axial symmetry).*

$$\mathbf{X} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \qquad \mathbf{Y} = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}, \qquad \mathbf{Z} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}.$$

Proposition 2.1. The set of the Pauli matrices, together with $\mathbf{I}_2$, the 2×2 identity matrix, form a basis of the $\mathcal{M}_2(\mathbb{C})$ space.

Proof. It is an easy exercise in linear algebra, as $\{\mathbf{I}_2, \mathbf{X}, \mathbf{Y}, \mathbf{Z}\}$ are linearly independent (meaning no matrix in the set can be expressed as a linear combination of the others), and $\dim \mathcal{M}_2(\mathbb{C}) = 4$.

Example 2.4. We already said that, mathematically, the Pauli gate can be represented as:

$$\mathbf{X} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}.$$

Let us consider an example where we apply the Pauli $\mathbf{X}$ gate to an initial qubit state $|\psi\rangle = 0.6|0\rangle + 0.8|1\rangle$.

Applying the Pauli $\mathbf{X}$ gate to this initial state, we just multiply the state vector by the Pauli $\mathbf{X}$ matrix:

$$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} 0.6 \\ 0.8 \end{bmatrix} = \begin{bmatrix} 0.8 \\ 0.6 \end{bmatrix}.$$

Thus, the resulting state after applying the Pauli $\mathbf{X}$ gate is $\mathbf{X}|\psi\rangle = 0.8|0\rangle + 0.6|1\rangle$.

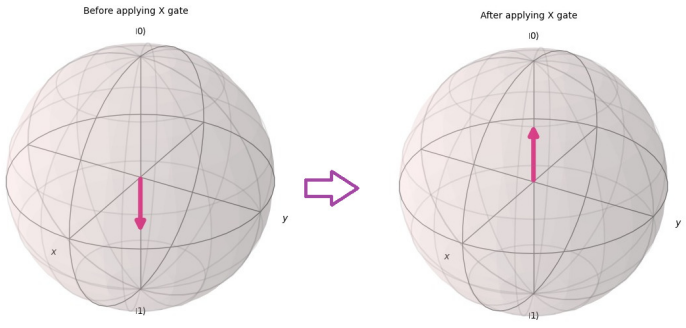


Figura 2.2: **X** rotation

This shows how the Pauli **X** gate flips the amplitudes of the $|0\rangle$ and $|1\rangle$ basis states. The resulting state is a superposition of $|0\rangle$ and $|1\rangle$, but with their amplitudes swapped, equivalent to the classical **NOT** gate.

Example 2.5. In this example we want to show graphically how the **Y** gate works by taking the state $|0\rangle$ and applying the gate to it.

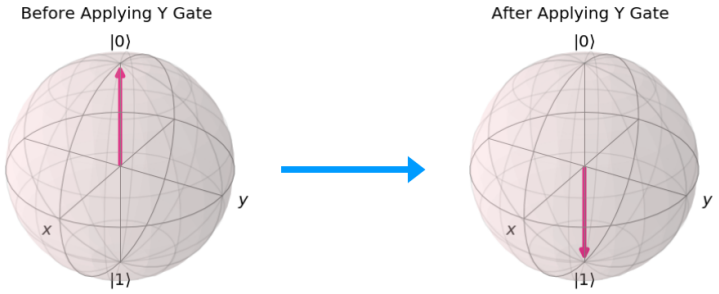


Figura 2.3: **Y** rotation

As we know

$$\mathbf{Y}|0\rangle = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ i \end{bmatrix}.$$

Mind that this vector is equivalent (up to a change of phase) to $|1\rangle$.

Proposition 2.2. The Pauli matrices verify the anticommutation relations:

$$\sigma_i\sigma_j = -\sigma_j\sigma_i, \quad \text{with } i \neq j; \qquad \sigma_i^2 = \mathbf{I}.$$

The proof is direct. Written in gate terms the anticommutations go as follows:

$$\mathbf{XY} = -\mathbf{YX}, \qquad \mathbf{XZ} = -\mathbf{ZX} \qquad \mathbf{YZ} = -\mathbf{ZY}.$$

Observation 2.2. From these relations many others can be derived. Let us show, for instance, that $\mathbf{XYX} = -\mathbf{Y}$. We know that $\mathbf{XY} = -\mathbf{YX}$ and $\mathbf{X}^2 = \mathbf{I}$. Then,

$$\mathbf{XYX} = -\mathbf{YXX} = -\mathbf{YI} = -\mathbf{Y}$$

Definition 2.8. *The Hadamard gate, usually noted as* $\mathbf{H}$*, maps the basis states in the following way:*

$$|0\rangle \mapsto \frac{1}{\sqrt{2}}\big(|0\rangle + |1\rangle\big) = |+\rangle, \qquad |1\rangle \mapsto \frac{1}{\sqrt{2}}\big(|0\rangle - |1\rangle\big) = |-\rangle.$$

It has, therefore, the following matrix representation:

$$\mathbf{H} = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

The Hadamard gate can also be expressed as a $\pi/2$–rotation around the y-axis, followed by a π–rotation around the x-axis. So, $\mathbf{H} = \mathbf{XY}^{1/2}$.

One useful property of the Hadamard gate is that it is self-inverse, meaning $\mathbf{H} = \mathbf{H}^{-1}$, and so

$$\mathbf{H}\left(\frac{|0\rangle + |1\rangle}{\sqrt{2}}\right) = |0\rangle, \qquad \mathbf{H}\left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right) = |1\rangle.$$

Example 2.6. The Hadamard gate in action:

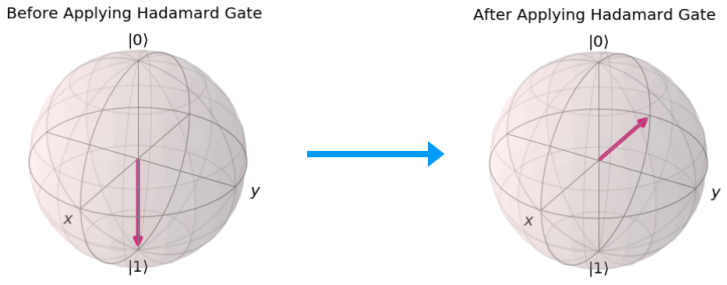


Figura 2.4: $\mathbf{H}$ rotation

Observation 2.3. We have the following three useful identities:

$$\mathbf{HXH} = \mathbf{Z}; \quad \mathbf{HYH} = -\mathbf{Y}; \quad \mathbf{HZH} = \mathbf{X}.$$

These expressions can be obtained by direct computation:

$$\mathbf{HXH} = \frac{1}{2}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}\begin{bmatrix}0 & 1\\ 1 & 0\end{bmatrix}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix} = \frac{1}{2}\begin{bmatrix}2 & 0\\ 0 & -2\end{bmatrix} = \begin{bmatrix}1 & 0\\ 0 & -1\end{bmatrix} = \mathbf{Z}. \tag{2.1}$$

$$\mathbf{HYH} = \frac{1}{2}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}\begin{bmatrix}0 & -i\\ i & 0\end{bmatrix}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix} = \frac{1}{2}\begin{bmatrix}0 & 2i\\ -2i & 0\end{bmatrix} = -\begin{bmatrix}0 & -i\\ i & 0\end{bmatrix} = -\mathbf{Y}. \tag{2.2}$$

$$\mathbf{HZH} = \frac{1}{2}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}\begin{bmatrix}1 & 0\\ 0 & -1\end{bmatrix}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix} = \frac{1}{2}\begin{bmatrix}0 & 2\\ 2 & 0\end{bmatrix} = -\begin{bmatrix}0 & 1\\ 1 & 0\end{bmatrix} = \mathbf{X}. \tag{2.3}$$

The above can also be calculated in a different way. We should also explain how, as it will be useful in further computations. Let us bear in mind that, from direct calculation,

$$\mathbf{H} = \frac{1}{\sqrt{2}}(\mathbf{X} + \mathbf{Z}).$$

Thus,

$$\mathbf{HXH} = \frac{1}{2}(\mathbf{X} + \mathbf{Z})\mathbf{X}(\mathbf{X} + \mathbf{Z}) = \frac{1}{2}(\mathbf{I} + \mathbf{ZX})(\mathbf{X} + \mathbf{Z}) = \frac{1}{2}(\mathbf{X} + \mathbf{Z} + \mathbf{Z} + \mathbf{XZX}) = \mathbf{Z}.$$

$$\mathbf{HYH} = \frac{1}{2}(\mathbf{X} + \mathbf{Z})\mathbf{Y}(\mathbf{X} + \mathbf{Z}) = \frac{1}{2}(\mathbf{XY} + \mathbf{ZY})(\mathbf{X} + \mathbf{Z}) = (\mathbf{XYX} + \mathbf{ZXY} + \mathbf{ZYX} + \mathbf{ZYZ}) = -\mathbf{Y}.$$

Once we know that $\mathbf{HXH} = \mathbf{Z}$ it is easy to see that

$$\mathbf{HZH} = \mathbf{H} \cdot (\mathbf{HXH}) \cdot \mathbf{H} = \mathbf{X},$$

as $\mathbf{H}^2 = \mathbf{I}$.

Definition 2.9. *The phase gate* $\mathbf{S}$ *and the* $\pi/8$*–gate* $\mathbf{T}$ *are defined by*

$$\mathbf{S} = \begin{bmatrix}1 & 0\\ 0 & i\end{bmatrix}, \qquad \mathbf{T} = \begin{bmatrix}1 & 0\\ 0 & e^{\frac{i\pi}{4}}\end{bmatrix}.$$

The **T**-gate (as it is also known) introduces a phase shift of $\pi/4$ to the $|1\rangle$ state:

$$\mathbf{T}|1\rangle = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\frac{\pi}{4}} \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ e^{i\frac{\pi}{4}} \end{bmatrix} = e^{i\frac{\pi}{4}}|1\rangle$$

When applied to a qubit on the $|0\rangle$, the **T**−gate has no effect on it:

$$\mathbf{T}|0\rangle = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\frac{\pi}{4}} \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} = |0\rangle$$

Its alternative name (*the* $\pi/8$*−gate*) is due to the fact that **T** is equivalent to

$$\begin{bmatrix} e^{\frac{-i\pi}{8}} & 0 \\ 0 & e^{\frac{i\pi}{8}} \end{bmatrix}.$$

On the other hand, gate **S** applies a phase shift of $\pi/2$ to the qubit state $|1\rangle$, while leaving the qubit state $|0\rangle$ unchanged. This is similar to the **T**−gate, when applied to the $|0\rangle$ state, **S** has no effect:

$$\mathbf{S}|0\rangle = \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} = |0\rangle.$$

Finally, it is interesting to note that the **S**−gate is related to the **T**−gate, as $\mathbf{S} = \mathbf{T}^2$.

Example 2.7. In this example, the **T**−gate is applied on a qubit in the state $|+\rangle$. We can calculate the result after applying the gate by simply doing the following:

$$\mathbf{T}|+\rangle = \mathbf{T} \begin{bmatrix} 1/\sqrt{2} \\ 1/\sqrt{2} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{bmatrix} \begin{bmatrix} 1/\sqrt{2} \\ 1/\sqrt{2} \end{bmatrix} = \begin{bmatrix} 1/\sqrt{2} \\ (1+i)/2 \end{bmatrix}$$

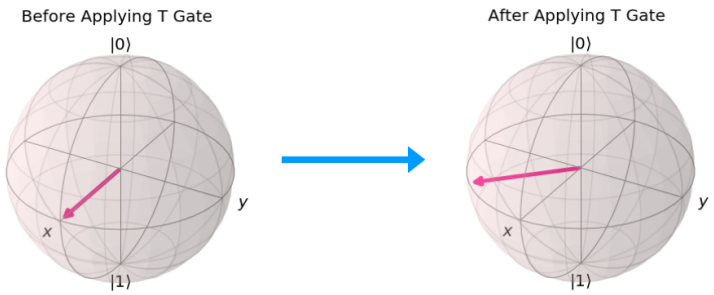


Figura 2.5: **T** gate

Example 2.8. In this example, the **S** gate involves a qubit state $|+\rangle$ as was the case,

$$\mathbf{S}|+\rangle = \mathbf{S}\begin{bmatrix}1/\sqrt{2}\\1/\sqrt{2}\end{bmatrix} = \begin{bmatrix}1 & 0\\0 & i\end{bmatrix}\begin{bmatrix}1/\sqrt{2}\\1/\sqrt{2}\end{bmatrix} = \begin{bmatrix}1/\sqrt{2}\\i/\sqrt{2}\end{bmatrix}$$

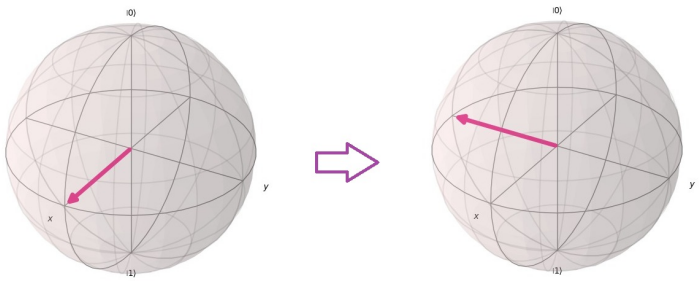


Figura 2.6: **S** gate

Definition 2.10. *The rotation operators appear as a result of exponentiating the Pauli matrices. They are the following:*

$$\mathbf{R}_i(\theta) \equiv e^{-i\theta\sigma_i/2} = \cos\left(\frac{\theta}{2}\right)\mathbf{I}_2 - i\sin\left(\frac{\theta}{2}\right)\sigma_i, \qquad \textit{for } i = x, y, z.$$

Explicitely,

$$\begin{aligned}
\mathbf{R}_x(\theta) \equiv e^{-i\theta\mathbf{X}/2} &= \cos\left(\frac{\theta}{2}\right)\mathbf{I}_2 - i\sin\left(\frac{\theta}{2}\right)\mathbf{X} = \begin{bmatrix}\cos(\theta/2) & -i\sin(\theta/2)\\ -i\sin(\theta/2) & \cos(\theta/2)\end{bmatrix}.\\
\mathbf{R}_y(\theta) \equiv e^{-i\theta\mathbf{Y}/2} &= \cos\left(\frac{\theta}{2}\right)\mathbf{I}_2 - i\sin\left(\frac{\theta}{2}\right)\mathbf{Y} = \begin{bmatrix}\cos(\theta/2) & -\sin(\theta/2)\\ \sin(\theta/2) & \cos(\theta/2)\end{bmatrix}.\\
\mathbf{R}_z(\theta) \equiv e^{-i\theta\mathbf{Z}/2} &= \cos\left(\frac{\theta}{2}\right)\mathbf{I}_2 - i\sin\left(\frac{\theta}{2}\right)\mathbf{Z} = \begin{bmatrix}e^{-i\theta/2} & 0\\ 0 & e^{i\theta/2}\end{bmatrix}.
\end{aligned}$$

Note that when we perform them we are basically performing rotations around the x, y, z axes of the Bloch sphere. These rotation gate operators will be important because they help us to decompose any 2×2 unitary matrices.

If we want to perform a general rotation of a qubit about an axis $n = (n_x, n_y, n_z)$ we can use the following proposition:

Proposition 2.3. A rotation of an angle θ about a general axis $n = (n_x, n_y, n_z)$ (normalized to 1) of the Bloch sphere can be expressed as:

$$\mathbf{R}_n(\theta) \equiv e^{-i\theta(n_x\mathbf{X}+n_y\mathbf{Y}+n_z\mathbf{Z})/2} = \cos\left(\frac{\theta}{2}\right)\mathbf{I} - i\sin\left(\frac{\theta}{2}\right)\left(n_x\mathbf{X} + n_y\mathbf{Y} + n_z\mathbf{Z}\right),$$

where $\left(n_x\mathbf{X} + n_y\mathbf{Y} + n_z\mathbf{Z}\right)^2 = \mathbf{I}$.

Proof. It follows from the definition by direct calculation. See [7] for the details. ∎

Proposition 2.4. Any unitary 2×2 matrix $\mathbf{U}$ can be written in the form below:

$$\mathbf{U} = e^{i\delta}\,\mathbf{R}_z(\gamma)\mathbf{R}_y(\beta)\mathbf{R}_z(\alpha),$$

where $\alpha, \beta, \gamma, \delta \in [0, 2\pi)$.

Proof. As $\mathbf{U}$ is a unitary matrix we already know we can write it as

$$\mathbf{U} = \begin{bmatrix} e^{i\omega}\cos\xi & e^{i\phi}\sin\xi \\ -e^{i(\theta-\phi)}\sin\xi & e^{i(\theta-\omega)}\cos\xi \end{bmatrix}.$$

Let us now compute $e^{i\delta}\,\mathbf{R}_z(\gamma)\mathbf{R}_y(\beta)\mathbf{R}_z(\alpha)$. Remember that

$$\mathbf{R}_z(\gamma) = \begin{bmatrix} e^{-i\gamma/2} & 0 \\ 0 & e^{i\gamma/2} \end{bmatrix}, \qquad \mathbf{R}_y(\beta) = \begin{bmatrix} \cos(\beta/2) & -\sin(\beta/2) \\ \sin(\beta/2) & \cos(\beta/2) \end{bmatrix}.$$

So we have

$$\mathbf{R}_y(\beta)\cdot\mathbf{R}_z(\alpha) = \begin{bmatrix} \cos(\beta/2) & -\sin(\beta/2) \\ \sin(\beta/2) & \cos(\beta/2) \end{bmatrix}\cdot\begin{bmatrix} e^{-i\alpha/2} & 0 \\ 0 & e^{i\alpha/2} \end{bmatrix} = \begin{bmatrix} e^{-i\alpha/2}\cos\beta/2 & -e^{i\alpha/2}\sin\beta/2 \\ e^{-i\alpha/2}\sin\beta/2 & e^{i\alpha/2}\cos\beta/2 \end{bmatrix},$$

and then,

$$\begin{aligned} e^{i\delta}\,\mathbf{R}_z(\gamma)\mathbf{R}_y(\beta)\mathbf{R}_z(\alpha) &= e^{i\delta}\begin{bmatrix} e^{-i\gamma/2} & 0 \\ 0 & e^{i\gamma/2} \end{bmatrix}\cdot\begin{bmatrix} e^{-i\alpha/2}\cos\beta/2 & -e^{i\alpha/2}\sin\beta/2 \\ e^{-i\alpha/2}\sin\beta/2 & e^{i\alpha/2}\cos\beta/2 \end{bmatrix} \\ &= \begin{bmatrix} e^{i(2\delta-\alpha-\gamma)/2}\cos\beta/2 & -e^{i(2\delta+\alpha-\gamma)/2}\sin\beta/2 \\ e^{i(2\delta-\alpha+\gamma)/2}\sin\beta/2 & e^{i(2\delta+\alpha+\gamma)/2}\cos\beta/2 \end{bmatrix}. \end{aligned}$$

So, if we say

$$\alpha = \phi - \omega, \qquad \beta = -2\xi, \qquad \gamma = \theta - \phi - \omega, \qquad \delta = \frac{1}{2}\omega$$

we get the desired description for $\mathbf{U}$. ∎

There are many other ways to represent a unitary matrix $\mathbf{U}$ in terms of a product of rotation matrices, as there is nothing particularly special about the y and z–axis and we can actually generalise regarding the above statement with the following proposition whose proof can be found in [7]:

Proposition 2.5. Let m and n be non-parallel real unit vectors of $\mathbb{R}^3$. For any quantum gate $\mathbf{U}$ acting on a single qubit there exist real numbers α, β, γ and δ so that

$$\mathbf{U} = e^{i\alpha}\mathbf{R}_n(\beta)\mathbf{R}_m(\gamma)\mathbf{R}_n(\delta).$$

Observation 2.4. We can express the $\mathbf{T}$–gate as

$$\mathbf{T} = e^{i\pi/8}\mathbf{R}_z\left(\frac{\pi}{4}\right).$$

It is an immediate result arising from the definition of $\mathbf{T}$.

Observation 2.5. The following expression will become useful in some later demonstrations:

$$\mathbf{R}_x\left(\frac{\pi}{4}\right) = e^{-i\pi/8}\mathbf{HTH}.$$

We know that $\mathbf{T} = \mathbf{R}_z(\pi/4)$ up to a global phase, $e^{-i\pi/8}$. We can therefore say that:

$$\begin{aligned}\mathbf{HTH} &= e^{-i\pi/8}\mathbf{H}\mathbf{R}_x(\pi/4)\mathbf{H} &&= e^{-i\pi/8}\mathbf{H}\big(\cos(\pi/8)\mathbf{I} - i\sin(\pi/8)\mathbf{Z}\big)\mathbf{H}\\ &= e^{-i\pi/8}\big(\cos(\pi/8)\mathbf{I} - i\sin(\pi/8)\mathbf{X}\big) &&= e^{-i\pi/8}\mathbf{R}_x(\pi/4),\end{aligned}$$

where we have used $\mathbf{HZH} = \mathbf{X}$ and $\mathbf{HIH} = \mathbf{H}^2 = \mathbf{I}$.

Lema 2.1. It is also possible to express the Hadamard gate with $\mathbf{R}_z$ and $\mathbf{R}_x$ rotations. In particular,

$$\mathbf{H} = e^{i\frac{\pi}{2}}\,\mathbf{R}_z\left(\frac{\pi}{2}\right)\mathbf{R}_x\left(\frac{\pi}{2}\right)\mathbf{R}_z\left(\frac{\pi}{2}\right).$$

Proof. We start with the product of:

$$
\begin{aligned}
\mathbf{R}_z\left(\frac{\pi}{2}\right)\mathbf{R}_x\left(\frac{\pi}{2}\right)\mathbf{R}_z\left(\frac{\pi}{2}\right) &= \mathbf{R}_z\left(\frac{\pi}{2}\right)\left(\frac{\sqrt{2}}{2}\mathbf{I} - i\frac{\sqrt{2}}{2}\mathbf{X}\right)\left(\frac{\sqrt{2}}{2}\mathbf{I} - i\frac{\sqrt{2}}{2}\mathbf{Z}\right) \\
&= \mathbf{R}_z\left(\frac{\pi}{2}\right)\left(\frac{1}{2}\mathbf{I}^2 - \frac{i}{2}\mathbf{Z} - \frac{i}{2}\mathbf{X} - \frac{1}{2}\mathbf{XZ}\right) \\
&= \left(\frac{\sqrt{2}}{2}\mathbf{I} - i\frac{\sqrt{2}}{2}\mathbf{Z}\right)\left(\frac{1}{2}\mathbf{I}^2 - \frac{i}{2}\mathbf{Z} - \frac{i}{2}\mathbf{X} - \frac{1}{2}\mathbf{XZ}\right) \\
&= \frac{\sqrt{2}}{4}\mathbf{I} - \frac{\sqrt{2}}{4}i\mathbf{Z} - \frac{\sqrt{2}}{4}i\mathbf{X} - \frac{\sqrt{2}}{4}\mathbf{XZ} - \frac{\sqrt{2}}{4}i\mathbf{Z} \\
&\qquad - \frac{\sqrt{2}}{4}\mathbf{Z}^2 - \frac{\sqrt{2}}{4}\mathbf{ZX} + \frac{\sqrt{2}}{4}i\mathbf{ZXZ} \\
&= -\frac{\sqrt{2}}{2}i\mathbf{Z} - \frac{\sqrt{2}}{4}i\mathbf{X} + \frac{\sqrt{2}}{4}i\mathbf{ZXZ} \\
&= -\frac{\sqrt{2}}{2}i\mathbf{Z} - \frac{\sqrt{4}}{4}i\mathbf{X} - \frac{\sqrt{2}}{4}i\mathbf{X} \\
&= -\frac{\sqrt{2}}{2}i(\mathbf{Z} + \mathbf{X}) = -i\mathbf{H}.
\end{aligned}
$$

2.5 Tensor product

In quantum computing, the tensor product is used to combine quantum states of multiple quantum systems, such as qubits, into a joint state that represents the combined system.

The importance of the tensor product in quantum computing arises from the fact that quantum mechanics is fundamentally different from classical mechanics as a theory. In classical mechanics, the state of a system can be described by a set of classical variables, such as position and momentum, which can be combined by using simple addition or multiplication. However, in quantum mechanics, the state

of a system is described by a complex vector in a higher-dimensional Hilbert space by using the tensor product, which has some interesting nuances, as we will see shortly.

Definition 2.11. *Let V and W be complex vector spaces of dimensions n and m, respectively. The tensor product of V and W, denoted by $V \otimes W$, is an (nm)-dimensional vector space whose elements are linear combinations of the symbols $v \otimes w$ that satisfy the subsequent properties:*

- $\alpha(v \otimes w) = (\alpha v) \otimes w = v \otimes (\alpha w)$,
- $(v_1 + v_2) \otimes w = (v_1 \otimes w) + (v_2 \otimes w)$,
- $v \otimes (w_1 + w_2) = (v \otimes w_1) + (v \otimes w_2)$,

where $\alpha \in \mathbb{C}, v, v_1, v_2 \in V$ and $w, w_1, w_2 \in W$.

Note that the elements of the form $v \otimes w$ generate $V \otimes W$ but *not all elements* of $V \otimes W$ can be written as $v \otimes w$. On the other hand, if

$$\mathcal{B}_V = \{v_1, \dots, v_n\}, \qquad \mathcal{B}_W = \{w_1, \dots, w_m\},$$

are bases of V and W (respectively), it is easy to see that, in fact, the set

$$\{v_i \otimes w_j \mid i = 1, \dots, n;\ j = 1, \dots, m\}$$

is a basis of $V \otimes W$.

Definition 2.12. *Let A and B be linear operators defined on V and W respectively, then the linear operator $A \otimes B$ operating on $V \otimes W$ is defined by*

$$(A \otimes B)(v \otimes w) = Av \otimes Bw,$$

with $v \in V$ and $w \in W$.

If A and B are $n \times n$ and $m \times m$ matrices respectively, which correspond to the matrix representations of the linear operators A and B with respect to the canonical base, then the linear operator $C = A \otimes B$ (called the tensor product, or the Kronecker product of A and B) can be built in the following way:

$$c_{i,j} = a_{q_1,q_2} b_{r_1,r_2}$$

where the indices are numbered from 0 and we have the Euclidean divisions

$$i = n \cdot q_1 + r_1, \qquad j = m \cdot q_2 + r_2.$$

A more illustrative (and easy) way of producing $C = A \otimes B$ is the following:

$$C = A \otimes B = \begin{bmatrix} a_{11}B & \cdots & a_{1n}B \\ \vdots & & \vdots \\ a_{n1}B & \cdots & a_{nn}B \end{bmatrix} = \begin{bmatrix} a_{11}b_{11} & \cdots & a_{11}b_{1m} & & a_{1n}b_{11} & \cdots & a_{1n}b_{1m} \\ \vdots & & \vdots & \cdots & \vdots & & \vdots \\ a_{11}b_{n1} & \cdots & a_{11}b_{mm} & & a_{1n}b_{m1} & \cdots & a_{1n}b_{mm} \\ & \vdots & & & & \vdots & \\ a_{n1}b_{11} & \cdots & a_{n1}b_{1m} & & a_{nn}b_{11} & \cdots & a_{nn}b_{1m} \\ \vdots & & \vdots & \cdots & \vdots & & \vdots \\ a_{n1}b_{m1} & \cdots & a_{n1}b_{mm} & & a_{nn}b_{m1} & \cdots & a_{nn}b_{mm} \end{bmatrix}$$

In quantum computing, the tensor product arises when we want to manipulate multi-qubit systems. Let us say we have an n system with individual quantum states:

$$|\psi_0\rangle,\ |\psi_1\rangle,\ |\psi_2\rangle, \ldots,\ |\psi_{n-1}\rangle.$$

The way to combine these systems as a whole quantum state would be:

$$|\phi\rangle_n = |\psi_0\rangle \otimes |\psi_1\rangle \otimes |\psi_2\rangle \otimes \cdots \otimes |\psi_{n-1}\rangle := |\psi_0\psi_1\psi_2 \cdots \psi_{n-1}\rangle.$$

What we are doing by considering tensor products (as opposed to a direct product) is somehow capturing every possible combination of these systems.

Example 2.9. If we consider the two basic qubit states $|0\rangle$ and $|1\rangle$ we get:

$$|\psi\rangle_2 = |0\rangle \otimes |1\rangle = |01\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \otimes \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}.$$

And it actually makes sense that this system is described with a 4-dimensional basis because if we want to show all the possible combinations of both states we will have a basis formed by $|00\rangle$, $|01\rangle$, $|10\rangle$ and $|11\rangle$.

Definition 2.13. *An n–qubit is a unit element of* $\left(\mathbb{C}^2\right)^{\otimes n}$.

That is, an n–qubit is a linear combination of modulus 1 in the following vectors

$$\left\{ |x_{n-1}\rangle \otimes \cdots \otimes |x_1\rangle \otimes |x_0\rangle := |x_{n-1} \cdots x_1 x_0\rangle \;\mid\; x_i \in \{0,1\} \text{ for } i = 0, \dots, n-1 \right\},$$

which we will also call computational basis, as in the $n = 1$ case. Moreover, if we consider $j \in \mathbb{Z}$ to be the integer with binary expression $(j)_2 = x_{n-1} \dots x_1 x_0$, this is usually written as

$$|x_{n-1} \cdots x_1 x_0\rangle = |j\rangle_n,$$

where the n subscript is compulsory to avoiding ambiguity. The computational basis is hence written as

$$\left\{ |0\rangle_n, |1\rangle_n, \dots, |2^n - 1\rangle_n \right\},$$

Furthermore, if we write an n–qubit $|\psi\rangle_n$ as a superposition of the basis states we get:

$$|\psi\rangle_n = \sum_{j=0}^{2^n-1} \alpha_j |j\rangle_n,$$

with the amplitudes $\alpha_j \in \mathbb{C}$ verifying

$$\sum_{j=0}^{2^n-1} |\alpha_j|^2 = 1.$$

As in the 1–qubit case, this should be interpreted as a probability distribution. The probability of obtaining $|j\rangle_n$ after measuring $|\psi\rangle_n$ is, in fact, given by $|\alpha_j|^2$.

What if we wanted to apply gates to our multiqubit system? Let us consider our above system and imagine we only want to apply an $\mathbf{X}$ gate to the first qubit. We can do this by using the definition of the $\otimes$ linear operator and model in our example as follows:

$$\left(\mathbf{X}|\psi_1\rangle\right) \otimes \left(\mathbf{I}|\psi_2\rangle\right) = (\mathbf{X} \otimes \mathbf{I}) \left(|\psi_1\rangle \otimes |\psi_2\rangle\right)$$

where

$$\mathbf{X} \otimes \mathbf{I} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 \cdot \mathbf{I} & 1 \cdot \mathbf{I} \\ 1 \cdot \mathbf{I} & 0 \cdot \mathbf{I} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

as the identity $\mathbf{I}$ simply models the *do nothing result* we want on the second qubit.

Observation 2.6. If we have two separate qubits:

$$|a\rangle = \begin{bmatrix} a_0 \\ a_1 \end{bmatrix} \qquad |b\rangle = \begin{bmatrix} b_0 \\ b_1 \end{bmatrix}$$

We can describe their combined state by using the Kronecker product:

$$|ba\rangle = |b\rangle \otimes |a\rangle = \begin{bmatrix} b_0 \cdot \begin{bmatrix} a_0 \\ a_1 \end{bmatrix} \\ b_1 \cdot \begin{bmatrix} a_0 \\ a_1 \end{bmatrix} \end{bmatrix} = \begin{bmatrix} b_0 a_0 \\ b_0 a_1 \\ b_1 a_0 \\ b_1 a_1 \end{bmatrix}$$

As we can see in this simple example, the 2–qubits arising as tensor products of two 1–qubits are a very special case: they always verify the equation $x_0 x_3 = x_1 x_2$. It is clear that not all the elements in our space have to verify this.

In particular, this shows that there are *more* 2*–qubit states* than those arising from the product of two 1–qubit states. We will give specific examples in the sequel.

Definition 2.14. *We can say that a state (or an n–qubit, more generally) is separable if it can be written as the tensor product of* 1*–qubit states.*

The elements of the computational basis are clearly separable states. But there are also examples of this phenomenon which are not easily detected at plain sight.

Example 2.10. Let

$$|\psi\rangle = \begin{bmatrix} -4/5 \\ 3/5 \end{bmatrix}, \qquad |\phi\rangle = \begin{bmatrix} 1/2 \\ \sqrt{3}/2 \end{bmatrix}.$$

If we compute the tensor product of both states we get a separable state:

$$|\psi\rangle|\phi\rangle = |\psi\rangle \otimes |\phi\rangle = |\psi\phi\rangle = \begin{bmatrix} -4/5 \\ 3/5 \end{bmatrix} \otimes \begin{bmatrix} 1/2 \\ \sqrt{3}/2 \end{bmatrix} = \begin{bmatrix} -2/5 \\ -2\sqrt{3}/5 \\ 3/10 \\ 3\sqrt{3}/10 \end{bmatrix},$$

and then

$$|\psi\phi\rangle = -\frac{2}{5}|00\rangle - \frac{2\sqrt{3}}{5}|01\rangle + \frac{3}{10}|10\rangle + \frac{3\sqrt{3}}{10}|11\rangle.$$

Our next section will refer to those states that are not separable, that is, those that do not come from the tensor product of two elements. We call these entangled states.

2.6 Quantum entanglement

Quantum entanglement is a phenomenon in which the state of a composite quantum system cannot be expressed as a product of the individual states of its subsystems. In other words, the quantum state of the composite system is a highly correlated joint state that cannot be factorized into separate states of the individual subsystems.

Definition 2.15. *An n–qubit general state $|\psi\rangle_n$ is called mixed or entangled if it is not separable, that is, if no 1–qubit states $|\psi_1\rangle, \ldots, |\psi_n\rangle$ exist so that*

$$|\psi\rangle_n = |\psi_1\rangle \otimes \cdots \otimes |\psi_n\rangle.$$

Definition 2.16. *The Bell states are specific quantum states of two qubits that represent the simplest examples of quantum entanglement. They can be represented as follows*

$$|\beta(x,y)\rangle = \frac{|0y\rangle + (-1)^x|1\overline{y}\rangle}{\sqrt{2}},$$

where $x, y \in \{0,1\}$ and $\overline{y}$ is the negation of y.

Therefore, the four Bell states can be explicetely described as:

$$|\Phi^+\rangle = \frac{1}{\sqrt{2}}\big(|00\rangle + |11\rangle\big), \qquad |\Phi^-\rangle = \frac{1}{\sqrt{2}}\big(|00\rangle - |11\rangle\big),$$

$$|\Psi^+\rangle = \frac{1}{\sqrt{2}}\big(|01\rangle + |10\rangle\big), \qquad |\Psi^-\rangle = \frac{1}{\sqrt{2}}\big(|01\rangle - |10\rangle\big).$$

Example 2.11. Let us consider a Bell state, say

$$|\Phi^+\rangle = \frac{|00\rangle + |11\rangle}{\sqrt{2}},$$

and let us prove that there are no single qubit states $|a\rangle$ and $|b\rangle$ so that $|\Phi^+\rangle = |a\rangle|b\rangle$.

Suppose, for the sake of contradiction, that $|\Phi^+\rangle = |a\rangle|b\rangle$ for some single qubit states $|a\rangle$ and $|b\rangle$. Then, we have that

$$|a\rangle = \alpha|0\rangle + \beta|1\rangle, \quad |b\rangle = \gamma|0\rangle + \delta|1\rangle; \qquad \text{for some } \alpha, \beta, \gamma, \delta \in \mathbb{C},$$

so that $|\alpha|^2 + |\beta|^2 = 1$ and $|\gamma|^2 + |\delta|^2 = 1$. Therefore,

$$|a\rangle|b\rangle = \big(\alpha|0\rangle + \beta|1\rangle\big)\,\big(\gamma|0\rangle + \delta|1\rangle\big) = \alpha\gamma|00\rangle + \alpha\delta|01\rangle + \beta\gamma|10\rangle + \beta\delta|11\rangle,$$

where we have used the linearity of the tensor product (though we supress the $\otimes$ symbols in the above expression). Hence,

$$|\psi\rangle = \frac{|00\rangle + |11\rangle}{\sqrt{2}} = \alpha\gamma|00\rangle + \alpha\delta|01\rangle + \beta\gamma|10\rangle + \beta\delta|11\rangle,$$

which forces $\alpha\delta = 0$ and $\beta\gamma = 0$. However, then we have the situation that at least one of $\alpha\gamma$ or $\beta\delta$ is also zero, and we thus reach a contradiction.

Observation 2.7. The four Bell states for an orthonormal basis for the two qubit state space.

Proposition 2.6. A quantum computer that does not use entangled qubits has the same computational power as a classical one.

Although the detailed proof of this result goes beyond the scope of this dissertation we can give a sketch of the main argument. If there is no entanglement, then an n–qubit state is separable into tensor products of n 1–qubit states. Thus, a classical computer simply has to retain the $2n$ complex amplitudes in its memory to simulate the quantum computer, which can be done efficiently (in polynomial time, to be precise). So, whatever the quantum computer solves, the classical computer can also solve it with the same complexity (roughly) if the qubits are in a pure state.

2.7 Multiple qubit gates

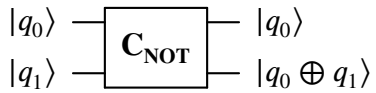
As we are now working now with unit vectors in $\mathbb{C}^{(2^n)} = \otimes_n \mathbb{C}^2$, the definitions match the respective ones for the case of $n = 1$, as one should expect.

Definition 2.17. *An n-qubit quantum gate is a unitary matrix in* $\mathcal{M}_{2^n}(\mathbb{C})$.

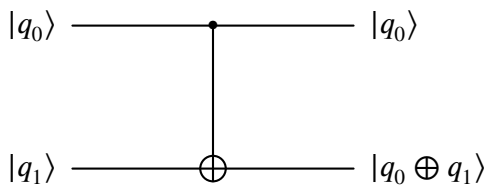
Of course one can construct n–qubit quantum gates by using tensor products of 1–qubit quantum gates but, in the same way as with n–qubits, there are many gates which can not be written like that. Let us begin our first approach to the multiple qubit gates through one of these cases: (probably) the most important example (we will see why in the next chapter).

Definition 2.18. *The* $\mathbf{C_{NOT}}$ *gate is a conditional* 4×4 *quantum gate, that is, it acts on* 2*–qubit states, performing an* $\mathbf{X}$ *gate on the second qubit (target), if the state of the first qubit (control) is* $|1\rangle$.

The gate is drawn on a circuit like this, with $|q_0\rangle$ as the control and $|q_1\rangle$ as the target:



or, more commonly,



When our 1–qubits are not a proper superposition of $|0\rangle$ and $|1\rangle$ (thus behaving as classical bits), this gate is very simple and intuitive to understand. We can use the classical truth table (mind that, in quantum as in classical computation, the *first* qubit is the one of the right):

Input	Output
00	00
01	11
10	10
11	01

And acting on our 4-dimensional vector, it is represented by the matrix:

$$\mathbf{C_{NOT}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix},$$

If it acts on the first qubit, using the second one as a control then the matrix representation is

$$\mathbf{C_{NOT}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix},$$

which is probably the most common one in the literature.

Example 2.12. Let us apply the first instance of $\mathbf{C_{NOT}}$ to an element outside the computational basis. Consider the state $|0\rangle \otimes |+\rangle = |0+\rangle$,

$$|0+\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |01\rangle),$$

and let us apply a $\mathbf{C_{NOT}}$ gate:

$$\mathbf{C_{NOT}}|0+\rangle = \frac{1}{\sqrt{2}}\left(|00\rangle + |11\rangle\right).$$

This state is an old acquaintance of ours (the first Bell state) and is, interestingly enough, entangled, although it is the image of a pure state.

Observation 2.8. Abusing notation, we will use the notation $\mathbf{C_{NOT}}$ for *any* gate acting on an n–qubit state (for $n \geq 2$) in an similar way to the examples above. That is, any gate which swaps *one* qubit (the **NOT** operation) if and only if certain requirements on the states of the other ones (the control qubits) are met.

Observation 2.9. There is nothing particularly relevant about the **NOT** operation (or the **X** gate, which is the same). Given a 1–qubit gate **U** we can, in fact, define a controlled–**U** gate in a certain bit, which will mean that the gate operates (as **U**) over this particular bit if and only if the control bits verify certain conditions.

Observation 2.10 (Permutation notation). From the Dirac notation it is straightforward that a gate performing the permutation of the basis states $|a\rangle_n$ and $|b\rangle_n$ can be written as:

$$|a\rangle_n\,{}_n\langle b| + |b\rangle_n\,{}_n\langle a| + \sum_{\substack{i=0 \\ i\neq a,b}}^{2^n-1} |i\rangle_n\,{}_n\langle i|.$$

This way of writing the gates is called *permutation notation* in quantum computing and it is widely used because it is quite compact, and it is most used for swapping elements of the computational basis or, more generally, for matrices similar to $\mathbf{I}_n$.

Clearly, it is nothing more than writing gates as a set of pairs of the form

$$\big(|\text{image}\rangle\ \langle\text{basis element}|\big),$$

where the images might be a linear combination of the computational basis, of course.

For example, the second $\mathbf{C_{NOT}}$ gate above can be represented as:

$$\mathbf{C_{NOT}} = |00\rangle\langle 00| + |01\rangle\langle 01| + |10\rangle\langle 11| + |11\rangle\langle 10|.$$

3 | Universal quantum gates

Any quantum algorithms that are of general interest are complex sequences of unitary transformations that act on n qubits. The question of a universal set of gates arises on its own: Is it possible to decompose these complex operations into much simpler ones? Is it possible that this set is made of *elementary* gates, whatever that might mean?

In this section, using the tools acquired in the previous sections, we will explore how to find universal gate sets: i.e., roughly speaking, sets of gates which allow us to create any possible quantum circuit. In order to do so, we have to take into account several drawbacks and limitations that will increase the complexity of the process. On the one hand one should have to deal with one of the biggest handicaps of quantum computing: errors that occur in a quantum system, which affect the accuracy of the calculations, so that instead of perfectly recreating a given unitary operator, we will get a good enough approximations of it. These issues are beyond the scope of our study and concern big sections of quantum computing related to error correction and fault tolerance that help us to achieve more accurate results.

We might on the other hand we might be able to find methods that approximate unitary matrices. But some further problems might appear. Maybe it will turn out that we need to use a huge amount of gates, deeming the method unpractical. Maybe, because of the amount of operations we need to make, the complexity of the algorithm will make our approach unfeasible. Bear in mind that there are uncountably many unitary transformations, but with our set of gates we can only build a countably infinite number of quantum circuits, meaning that we must compromise in the approximation (arbitrary precision, ideally).

So our task is to find a set of elementary gates that approximate any unitary operator and that, at the same time, are efficient. In the following section we will give an intuitive idea of the conditions that a set of quantum gates must fulfil in order to be universal.

The next sections draw on the arguments provided by [7, 4].

3.1 First steps

Definition 3.1. *A set of gates is said to be universal if for any integer $n \geq 1$, any n-qubit unitary operator can be approximated to arbitrary precision by a quantum circuit using only gates from that set.*

A natural question that arises from this definition is what does a set of quantum gates need to have in order to be universal. We will start discussing four ways where a gate set fails to be universal. Of course, a potential gate set must be able to reach all the states in which a qubit can be found in order to be universal. Hence:

A) We must be able to create a superposition.

Example 3.1. We have studied the $\{\mathbf{C_{NOT}}\}$ gate and we have seen that it maps computational basis states like $|10\rangle$, to other computational basis states, like $|11\rangle$, so it cannot create essentially superpositon states but only permute the amplitudes that already exist. A gate set consisting only of a $\mathbf{C_{NOT}}$ gate can never be universal.

B) We must be able to create entanglement.

Example 3.2. Let us take a gate now that is able to create superposition states like the Hadamard gate. Can we achieve entanglement with it? Well, because it only acts on one qubit, the answer is no. Furthermore, as we mentioned earlier, one can see that because it acts only on one qubit it will be rather easy to replicate on a classical computer. A set of gates made up of just 1−qubit gates can never be universal.

C) We must be able to reproduce gates with complex entries.

Example 3.3. This statement is pretty logical, unitary transformations exist with complex entries, so if we want to be able to approximate them we need to have a matrix with complex entries in our set. The set of gates consisting of $\{\mathbf{C_{NOT}}, \mathbf{H}\}$ meets conditions A) and B) but its elements only consist of real entries so it cannot generate matrices with complex values.

For our last condition we will introduce a definition of a gate set widely used in quantum computing: the Clifford gate set.

Definition 3.2. *We will call the Clifford gate set* $\mathcal{C} = \{\mathbf{H}, \mathbf{C_{NOT}}, \mathbf{S}\}$.

Why have we specified this set? One very important result in quantum computing is the Gottesman-Knill theorem, which tells us the following [2]:

Theorem 3.1 (Gottesman-Knill). *A quantum circuit using only the following elements can be simulated efficiently on a classical computer:*

- *Preparation of qubits in computational basis states,*
- *Clifford gates, and*
- *Measurements in the computational basis.*

Hence,

D) We must be able to operate outside the Clifford group gate set.

Because of the Gottesman-Knill theorem, if a set of gates is contained in the Clifford group, any algorithm that we perform with them can be efficiently simulated on a classical computer and these algorithms will not be enough to realize exponential quantum speed–ups and therefore this set of gates cannot be universal.

It is therefore natural to look for sets of gates to achieve universality. And a number of interesting questions might also appear in this context, like trying to find a set which is minimal (in the cardinal sense) or which is the most efficient (insofar as it achieves the shortest gate factorization).

In the following sections we will tackle the problem with different approaches and we will try to provide some understanding of the conditions these sets of gates need to meet as well as some examples of universal gate sets.

3.2 Exact quantum gates

In this section we will forget about some efficiency issues and we will allow our algorithms to use uncountable gate sets. This way we will build any unitary transformation we want in an exact way.

First of all we will study how to decompose any unitary transformation into the product of a particular type of gate that will be introduced next.

3.2.1 2–level unitary matrices

Definition 3.3 (2–level unitary matrices). *A 2–level matrix is a matrix which act non–trivially only on two or fewer vector components.*

In other words, if we fix our standard computational basis $|0\rangle_n, |1\rangle_n, |2\rangle_n, \rangle|2^n-1\rangle_n$, we can say that $\mathbf{A}$ is a 2–level matrix if it acts nontrivially only in the two-dimensional subspace spanned by two basis elements $|i\rangle_n$ and $|j\rangle_n$; i.e., $\mathbf{A}$ decomposes (as an operator) as a direct sum:

$$\mathbf{A} = \mathbf{A}^{(2)} \oplus \mathbf{I}^{(2^n-2)},$$

where $\mathbf{A}^{(2)}$ is a 2×2 operator acting on the span of $|i\rangle_n$ and $|j\rangle_n$, and $\mathbf{I}^{(2^n-2)}$ is the identity operator acting on the complementary (2^n-2)–dimensional subspace given by $\langle |i\rangle_n, |j\rangle_n \rangle^{\perp}$.

Example 3.4. Consider a 3×3 matrix (acting on a 3–dimensional vector space)

$$\mathbf{A} = \begin{bmatrix} a & 0 & b \\ 0 & 1 & 0 \\ c & 0 & d \end{bmatrix}$$

If we apply this matrix to an arbitrary vector (x, y, z),

$$\begin{bmatrix} a & 0 & b \\ 0 & 1 & 0 \\ c & 0 & d \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} ax + bz \\ y \\ cx + dz \end{bmatrix},$$

is therefore $\mathbf{A}$ a 2–level matrix, as only the first and third component were altered.

Example 3.5. From the classical Gaussian algorithm we are already familiar with the so–called elementary matrices, all of which are 2–level matrices:

- $\mathbf{E}_{ij}$ matrices, which consist of identity matrices with the i–th and j–th columns swapped.
- $\mathbf{E}_i(\alpha)$ matrices, which consist of identity matrices with the i–th column multiplied by $\alpha \in \mathbb{C}$.
- $\mathbf{E}_{ij}(\alpha)$ matrices, which consist of identity matrices where the j–th column multiplied by $\alpha \in \mathbb{C}$ has been added to the i–th column.

Note that $\mathbf{E}_{ij}$ matrices are always unitary, while $\mathbf{E}_i(\alpha)$ matrices are unitary if and only if $|\alpha| = 1$. $\mathbf{E}_{ij}(\alpha)$ type matrices, on the other hand, are never unitary (except for the trivial case of $\alpha = 0$).

3.2.2 2–level unitary matrices can be used to factorize any unitary matrix

Our goal in this section is to decompose a unitary matrix, $\mathbf{U}$, which acts on a d–dimensional Hilbert space, into the product of 2–level unitary matrices.

Let us suppose $\mathbf{U}$ is:

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{d1} & a_{d2} & \cdots & a_{dd} \end{bmatrix}$$

In order to achieve our purpose we will slightly modify the Gaussian elimination. Remember that this process essentially consist in two phases. The first one takes our matrix to row–echelon form by iterating the following process:

Step 0) Move every null row to the lower end of the matrix. This step uses $d - 1$ matrices at most $\mathbf{E}_{ij}$ and it only has to be performed once.

Step 1) Locate a pivot (i.e., a non–zero element) in the leftmost available column and move it upwards as far as possible. If there are none, move it to the next column. This step uses one $\mathbf{E}_{ij}$ matrix at most.

Step 2) Cancel out all elements below the pivot. This step uses $d - i$ matrices $\mathbf{E}_{ij}(\alpha)$ at most (with $i < j$) if we are in the i–th row.

Step 3) Consider the submatrix comprising the rows below and the columns at the right of the pivot and go back to Step 1.

The second phase gives us the reduced row–echelon form, using matrices $\mathbf{E}_i(\alpha)$ and $\mathbf{E}_{ij}(\alpha)$, but we are not interested in that, as we can use the properties of the unitary matrices to avoid this part.

So, from the remarks above, in order to achieve a row–echelon form using unitary matrices, the main obstruction is the need for $\mathbf{E}_{ij}(\alpha)$ matrices, which allow us to cancel terms below a given pivot, because they are not unitary. However, this can be corrected.

Imagine we have a 2×2 matrix

$$\mathbf{A} = \begin{bmatrix} \alpha & * \\ \beta & * \end{bmatrix},$$

with $\alpha, \beta \in \mathbb{C}^*$. We can then consider the following matrix:

$$\mathbf{U}_{12}(\alpha, \beta) = \begin{bmatrix} \dfrac{\overline{\alpha}}{\sqrt{|\alpha|^2 + |\beta|^2}} & \dfrac{\overline{\beta}}{\sqrt{|\alpha|^2 + |\beta|^2}} \\ \dfrac{-\beta}{\sqrt{|\alpha|^2 + |\beta|^2}} & \dfrac{\alpha}{\sqrt{|\alpha|^2 + |\beta|^2}} \end{bmatrix} = \frac{1}{\sqrt{|\alpha|^2 + |\beta|^2}} \begin{bmatrix} \overline{\alpha} & \overline{\beta} \\ -\beta & \alpha \end{bmatrix},$$

which can be easily checked to ensure it is unitary. We therefore have:

$$\mathbf{U}_{12}(\alpha, \beta)\mathbf{A} = \begin{bmatrix} \gamma & * \\ 0 & * \end{bmatrix}$$

where $\gamma = \sqrt{|\alpha|^2 + |\beta|^2} \neq 0$.

So we can adapt the Gaussian algorithm in order to use only unitary transformations. The process is exactly the same except that, instead of using $\mathbf{E}_{ij}(\alpha)$ matrices we have to use $\mathbf{U}_{ij}(\alpha, \beta)$ matrices, which are identity matrices $\mathbf{I}_d$ except for the following elements:

$$(i,i) := \frac{\overline{\alpha}}{\sqrt{|\alpha|^2 + |\beta|^2}}; \quad (i,j) := \frac{\overline{\beta}}{\sqrt{|\alpha|^2 + |\beta|^2}}; \quad (j,i) := \frac{-\beta}{\sqrt{|\alpha|^2 + |\beta|^2}}; \quad (j,j) := \frac{\alpha}{\sqrt{|\alpha|^2 + |\beta|^2}}.$$

These matrices, as the original $\mathbf{E}_{ij}(\alpha)$, cancel out the element in the j–th row lying below the pivot (which is in the i–th row) and depend on both the pivot and the element to be cancelled. These new matrices are also unitary.

Observation 3.1. A major difference, besides the unitary character of $\mathbf{U}_{ij}(\alpha, \beta)$ and $\mathbf{E}_{ij}(\alpha)$ is that the latter only modify the j–th row, while the former modify both the i–th and the j–th row. And it does have to be this way because they do in fact achieve more than predicted.

The following result is a very well–known lemma frequently used with Schur's Lemma, to characterise normal matrices as those which are diagonalizable using orthonormal bases (as in *Álgebra Lineal y Geometría I*, for instance).

Proposition 3.1. A normal matrix which is upper triangular must in fact be diagonal.

Therefore, as unitary matrices are normal, if we start with a unitary $\mathbf{U}$ and apply the modified version of the Gaussian algorithm we will get the following expression:

$$\mathbf{U}_m \cdots \mathbf{U}_1 \cdot \mathbf{U} = \begin{bmatrix} \gamma_1 & & & \\ & \gamma_2 & & \\ & & \ddots & \\ & & & \gamma_d \end{bmatrix} = \mathbf{E}_1(\gamma_1)\mathbf{E}_2(\gamma_2)\cdots\mathbf{E}_d(\gamma_d),$$

where the matrices $\mathbf{U}_i$ are unitary, either of the $\mathbf{E}_{ij}$–type or of the $\mathbf{U}_{ij}(\alpha,\beta)$–type. And, as the resultant diagonal matrix must also be unitary, $|\gamma_i| = 1$, for $i = 1, \ldots, d$. That is, the matrices $\mathbf{E}_i(\gamma_i)$ are as well unitary.

From this expression it is clear that

$$\mathbf{U} = \mathbf{U}_1^* \cdots \mathbf{U}_m^* \cdot \mathbf{E}_1(\gamma_1) \cdots \mathbf{E}_d(\gamma_d),$$

which is a factorization in 2–level unitary matrices.

Example 3.6. Let us try to decompose the following matrix into the product of 2–level unitary matrices:

$$\mathbf{U} = \frac{1}{2}\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & i & -1 & -i \\ 1 & -1 & 1 & -1 \\ 1 & -i & -1 & i \end{bmatrix}.$$

We must first compute the matrix noted above as $\mathbf{U}_{12}(1,1)$,

$$\mathbf{U}_1 = \mathbf{U}_{12}(1,1) = \begin{bmatrix} \sqrt{2}/2 & \sqrt{2}/2 & 0 & 0 \\ -\sqrt{2}/2 & \sqrt{2}/2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

and

$$\mathbf{U}_1 \cdot \mathbf{U} = \frac{1}{2}\begin{bmatrix} \sqrt{2} & \sqrt{2}/2 + \sqrt{2}i/2 & 0 & \sqrt{2}/2 - \sqrt{2}i/2 \\ 0 & -\sqrt{2}/2 + \sqrt{2}i/2 & -\sqrt{2} & -\sqrt{2}/2 - \sqrt{2}i/2 \\ 1 & -1 & 1 & -1 \\ 1 & -i & -1 & i \end{bmatrix},$$

where, as we remarked above, both the first and the second row have been altered. So in order to make another 0 in the $(3,1)$ position we need the matrix $\mathbf{U}_{13}\left(\sqrt{2},1\right)$:

$$\mathbf{U}_2 = \mathbf{U}_{13}\left(\sqrt{2},1\right) = \begin{bmatrix} \sqrt{2/3} & 0 & 1/\sqrt{3} & 0 \\ 0 & 1 & 0 & 0 \\ -1/\sqrt{3} & 0 & \sqrt{2/3} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

We now compute this and get:

$$\mathbf{U}_2\mathbf{U}_1\mathbf{U} = \frac{1}{2}\begin{bmatrix} \sqrt{3} & \sqrt{3}i/3 & \sqrt{3}/3 & -\sqrt{3}i/3 \\ 0 & -\sqrt{2}/2 + \sqrt{2}i/2 & -\sqrt{2} & -\sqrt{2}/2 - \sqrt{2}i/2 \\ 0 & -\sqrt{6}/2 - \sqrt{6}i/6 & \sqrt{6}/3 & -\sqrt{6}/2 + \sqrt{6}i/6 \\ 1 & -i & -1 & i \end{bmatrix}.$$

As one might notice, the computations quickly become cumbersome (at least, to be done by hand).

Observation 3.2. From the description of the procedure it is not complicated to bound the number of matrices we need. Step 0 is in fact not necessary as $\mathbf{U}$ is unitary. So, for the i–th column (where $i = 1, \dots, d$) we need $(d - i + 1)$ 2–level matrices at most, as explained in the step description. That is, the full diagonalization takes, at most,

$$d + (d-1) + \cdots + 1 = \frac{1}{2}d(d+1) \text{ 2–level unitary matrices.}$$

Once we have a diagonal matrix we might need d matrices $\mathbf{E}_i(\cdot)$ (we maight take two at a time, but there is no substantial difference), so an upper bound for the number of factors in this decomposition is $d(d+3)/2$.

As for lower bounds in the number of factors, we will mention a result which needs some technical adjustments beyond outside the scope of this work. So we are not giving a full proof, but rather an extensive road map. A fully detailed account can be found in [9].

Lema 3.1. There exists a $d \times d$ unitary matrix $\mathbf{U}$ which cannot be decomposed as a product of less than $(d-1)$ unitary 2–level matrices.

Proof. (Outline) Suppose $\mathbf{U}$ is a $d \times d$ unitary matrix which can be decomposed using less than $(d-1)$ unitary 2–level matrices.

Every time we multiply using one of these 2–level matrices we arrive at a new matrix. These new matrices can be seen as vertices of a graph and the 2–level matrices as the connections or the edges connecting these nodes of the graph. Our hypothesis is that we are actually using less than $(d-1)$ edges in our decomposition.

If we had a graph with d vertices and there were no connection matrices or edges then we would have d different components in the graph. If our graph had k components and we added a new edge, then the least number of components that we could have is $(k-1)$. That is, we can add an edge that connects two different components (having $(k-1)$ components) or an edge that links vertices of the same components (resulting in k components).

Taking these two statements into account, it is easy to see that if a graph has n nodes and m edges, we thus have at least $n-m$ components so if in our case we have d vertices and $(d-2)$ edges we will have at least 2 components.

It is not complicated to show that, if we want to have only $(d-2)$ links, then the graph is not connected because it has at least two different components. That is, we can get D_1 and D_2 subsets of our graph that are not connected by any edges. So until we rearrange this we can write $\mathbf{U}$ as $\mathbf{U}_1 \oplus \mathbf{U}_2$ where $\mathbf{U}_1$ and $\mathbf{U}_2$ are two unitary matrices that can be identified with the two graphs, D_1 and D_2 stated above.

If $\mathbf{U}$ can be decomposed into this form it means that $\mathbf{U}$ has to be block-diagonal in the computational basis and clearly not all matrices can be written like this. One example of this is the matrix of the Quantum Fourier Transform, which is the matrix in the above Example 3.6.

3.2.3 $\mathbf{C_{NOT}}$ gates and single qubit gates are universal

We have just shown that an arbitrary unitary matrix acting on a d–dimensional Hilbert space may be written as a product of 2–level unitary matrices. Now we will show that single qubit and $\mathbf{C_{NOT}}$ gates can be used together to implement an arbitrary 2–level unitary operation on the state space of n qubits. By combining these results we see that single qubit and $\mathbf{C_{NOT}}$ gates can be used to implement an arbitrary unitary operation on n qubits, and are therefore a universal gate set for quantum computation.

Theorem 3.2. *The set of single qubit gates and the* $\mathbf{C_{NOT}}$ *gates are universal.*

We have already proved that we could write any unitary transformation as the product of 2–level matrices, so, in particular, it suffices for us to assume that the matrix we want to implement is 2–level.

Now let us assume we have a gate given by 2–level unitary matrix $\mathbf{U}$ on a n–qubit quantum computer and $\tilde{\mathbf{U}}$ is the non-trivial 2×2 submatrix of $\mathbf{U}$. Let us write $|s\rangle_n$ and $|t\rangle_n$ as the two qubits from the computational basis where our matrix acts in a non-trivial way and take their binary expansions as $s_1 \dots s_n$ and $t_1 \dots t_n$.

Example (Theorem 3.2)

In order to illustrate this process we will simultaneously tackle the following example:

$$\mathbf{U} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & a & 0 & 0 & 0 & 0 & c & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & b & 0 & 0 & 0 & 0 & d & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \qquad \tilde{\mathbf{U}} = \begin{bmatrix} a & c \\ b & d \end{bmatrix}.$$

This is a 2–level matrix, acting non-trivially only on states $|s\rangle = |001\rangle_3$ and $|t\rangle = |110\rangle_3$.

Our main objective is to bring together the two qubits where $\mathbf{U}$ acts in a non-trivial way, so we can apply the submatrix $\tilde{\mathbf{U}}$. In order to do this we will introduce the Gray codes.

Definition 3.4. *Given two binary numbers s and t, the Gray code connecting s and t is a sequence of binary numbers, starting with s and concluding with t, such that any adjacent members on the list differ by exactly one bit.*

Example 3.7. For instance, with $s = 100101$ and $t = 010111$ we have the Gray code

$$\begin{matrix} 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 \end{matrix}$$

Let us denote $g_1, \dots, g_m$ as the elements of a Gray code that connect s and t, where $s = g_1$ and $t = g_m$. Because we are using an n–qubit quantum computer, s and t can only differ at most in n locations so we can assume $m \leq n + 1$.

Example (Theorem 3.2)

As our matrix acted non-trivially only on the states $|001\rangle$ and $|110\rangle$ we can write our Gray code connecting these states as:

$$\begin{array}{lll} 0 & 0 & 1 \to g_1 \\ 0 & 0 & 0 \to g_2 \\ 0 & 1 & 0 \to g_3 \\ 1 & 1 & 0 \to g_4 \end{array}$$

In this example we have a maximal length Gray code (for $n = 3$).

As a sum up we will begin by describing the implementation of the factoring algorithm on three steps as follows:

Step 1) We want to swap states $|g_1\rangle$ with $|g_2\rangle$, $|g_2\rangle$ with $|g_3\rangle$ and so on until we reach $|g_{m-1}\rangle$, following our Gray code.
For every switch, from $|g_i\rangle$ to $|g_{i+1}\rangle$ let us say that both states differ in the j–th qubit, so we perform a controlled bit flip on this qubit which is conditional on the other qubits being identical to those in both $|g_i\rangle$ and $|g_{i+1}\rangle$. This sequence that can be represented as follows:

$$|g_1\rangle \mapsto |g_2\rangle \mapsto \cdots \mapsto |g_{m-2}\rangle \mapsto |g_{m-1}\rangle,$$

and the effect of all the swaps in the sequence is given by:

$$|g_1\rangle \mapsto |g_{m-1}\rangle, \qquad |g_2\rangle \mapsto |g_1\rangle, \qquad |g_3\rangle \mapsto |g_2\rangle, \quad \ldots, \quad |g_{m-1}\rangle \mapsto |g_{m-2}\rangle$$

But bear in mind that, as we fixed $(n-1)$ control bits, each step swaps only the two relevant elements of the computational basis, all the other states are not involved and subsequently are left unchanged.

Step 2) Now we apply a controlled-$\tilde{U}$ operation on the qubit where $|g_{m-1}\rangle$ and $|g_m\rangle$ differ, being conditional on all the other qubits having the same values as $|g_m\rangle$ and $|g_{m-1}\rangle$.

Step 3) The third and last step is to revert all the operations from Step 1.

It is easy to check that these steps do precisely the same as the gate $\mathbf{U}$ given. Let us inspect more closely each step, applying them to our example.

Step 1

Since adjacent elements in the Gray code differ by only one bit, as we have seen, the swaps of Step 1 (and Step 3, for that matter) can be described by $\mathbf{C_{NOT}}$ operations: flipping a particular bit only if all of the other bits are as given.

As these gates only swap two elements of the computational basis they can be realized as $\mathbf{E}_{ij}$ matrices as well, where $|i\rangle_n$ and $|j\rangle_n$ are the elements of the computational basis swapped by the $\mathbf{C_{NOT}}$ gate.

Example (Theorem 3.2)

In our example, if we have to begin the algorithm doing a double swap:

$$|001\rangle = |g_1\rangle \;\mapsto\; |000\rangle = |g_2\rangle \;\mapsto\; |010\rangle = |g_3\rangle.$$

If want to represent the permutation from state $|001\rangle$ to state $|000\rangle$, as they are the first two vectors of the computational basis, we can do it with the matrix:

$$\mathbf{E}_{12} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

Mind that, as explained above, this is indeed a $\mathbf{C_{NOT}}$ gate acting on 3 qubits, if the 2 last qubits are 0 then the first qubit is turned to the opposite state.

Now, if we want to go from state $|000\rangle$ to state $|010\rangle$ we have to use a different $\mathbf{C_{NOT}}$ gate, which is also a permutation. In particular we need a gate acting on two qubits: it will change the second qubit if the first and third qubit are 0. As it interchanges the first and third elements of the

Example (Theorem 3.2) (cont)

computational basis, the matrix we need is

$$\mathbf{E}_{13} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

Step 2

Now we must apply the controlled–$\tilde{\mathbf{U}}$ gate acting on the qubit, which is the different one between $|g_{m-1}\rangle$ and $|g_m\rangle$. That is, we have to perform

$$\hat{\mathbf{U}} = \tilde{\mathbf{U}} \otimes \mathbf{I}$$

where $\tilde{\mathbf{U}}$ acts on $\{|g_{m-1}\rangle, |g_m\rangle\}$ and $\mathbf{I}$ on its orthogonal space (with dimension $2^n - 2$).

Example (Theorem 3.2)

In our working example we have

$$|g_{m-1}\rangle = |g_3\rangle = |010\rangle; \qquad |g_m\rangle = |g_4\rangle = |110\rangle$$

So our controlled–$\tilde{\mathbf{U}}$ gate acts in the following way:

$$\begin{aligned} |010\rangle &\rightarrow \big(a|0\rangle + b|1\rangle\big)|10\rangle = a|010\rangle + b|110\rangle \\ |110\rangle &\rightarrow \big(c|0\rangle + d|1\rangle\big)|10\rangle = c|010\rangle + d|110\rangle \end{aligned}$$

Example (Theorem 3.2) (cont)

while the rest of the elements of the computational basis does not vary. That is,

$$\hat{\mathbf{U}} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & a & 0 & 0 & 0 & c & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & b & 0 & 0 & 0 & d & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Step 3

As noted, this step only consist of undoing the $\mathbf{C_{NOT}}$ gates of Step 1 in reverse order.

Example (Theorem 3.2)

In our working example, this last step will involve the $\mathbf{C_{NOT}}$ gates given by $\mathbf{E}_{13}$ and $\mathbf{E}_{12}$ (in this order).

Example 3.8. Inspired by the comment section in the post [10] we have also seen how we can implement this whole procedure in our previous example, this time using the permutation notation.

Step 1: The unitary operator can be described in the following way:

$$\mathbf{S} = \underbrace{|010\rangle\langle 001| + |001\rangle\langle 000| + |000\rangle\langle 010|}_{\text{permutations of Gray codes}} +$$

$$+ \underbrace{|011\rangle\langle 011| + |100\rangle\langle 100| + |101\rangle\langle 101| + |110\rangle\langle 110| + |111\rangle\langle 111|}_{\text{trivial action}}$$

The matrix expression is

$$\mathbf{S} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

Step 2: As we saw above, the single-qubit operation on the first qubit acts on these states in the following way:

$$\begin{aligned} |010\rangle &\rightarrow \big(a|0\rangle + b|1\rangle\big)|10\rangle = a|010\rangle + b|110\rangle \\ |110\rangle &\rightarrow \big(c|0\rangle + d|1\rangle\big)|10\rangle = c|010\rangle + d|110\rangle \end{aligned}$$

None of the other states are affected so we can write the full action of this single qubit gate as:

$$\hat{\mathbf{U}} = \underbrace{(a|010\rangle + b|110\rangle)\langle 010| + (c|010\rangle + d|110\rangle)\langle 110|}_{\text{Action of single qubit gate}} +$$

$$\underbrace{+|000\rangle\langle 000| + |001\rangle\langle 001| + |011\rangle\langle 011| + |100\rangle\langle 100| + |101\rangle\langle 101| + |111\rangle\langle 111|}_{\text{trivial action}}$$

And, as noted, the matrix is given by

$$\hat{\mathbf{U}} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & a & 0 & 0 & 0 & c & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & b & 0 & 0 & 0 & d & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Step 3: Now is the part where we need to undo all the transformations we did in Step 1 and this basically

means taking $\mathbf{S}^*$, as $\mathbf{S}$ is a unitary matrix:

$$\mathbf{S}^* = \underbrace{|001\rangle\langle 010| + |000\rangle\langle 001| + |010\rangle\langle 000|}_{\text{undoing permutations of Gray codes}} +$$

$$+ \underbrace{|010\rangle\langle 010| + |100\rangle\langle 100| + |101\rangle\langle 101| + |110\rangle\langle 110| + |111\rangle\langle 111|}_{\text{trivial action}}$$

$$\mathbf{S}^* = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Now one can check (with a little time and patience) $\mathbf{S}^*\hat{\mathbf{U}}\mathbf{S} = \mathbf{U}$, as expected.

3.3 A finite universal gate set

3.3.1 Approximating unitary operators

In this section we will address the issue of quantum gate universality with a different approach. Instead of implementing the unitary operators exactly, we will seek to obtain an approximation of them, but still we need to measure how good that approximation is. To do this we will give some idea about errors between unitary operators in order to see that for any tolerance we may require, we can find a set of gates that approximate to any unitary operator.

In order to be able to talk about distances between operators we will use the common norm,

$$\left|\left||\psi\rangle\right|\right| = \sqrt{\langle\psi|\psi\rangle}$$

Suppose we approximate a given unitary transformation $\mathbf{U}$ by some other unitary transformation $\mathbf{V}$. The error in the approximation is defined as

$$E(\mathbf{U}, \mathbf{V}) = \max_{|\psi\rangle} \Big|\Big|(\mathbf{U} - \mathbf{V})|\psi\rangle\Big|\Big|,$$

where the maximum is above all normalized quantum states $|\psi\rangle$ in the state space. When we say that we *approximate the unitary operator*, we mean of course that we are trying to make this error smaller than a certain tolerance ($\varepsilon > 0$).

Proposition 3.2. For unitary transformations $\mathbf{U}_1$, $\mathbf{U}_2$, $\mathbf{V}_1$ and $\mathbf{V}_2$ we have

$$E(\mathbf{U}_2\mathbf{U}_1, \mathbf{V}_2\mathbf{V}_1) \leq E(\mathbf{U}_2, \mathbf{V}_2) + E(\mathbf{U}_1, \mathbf{V}_1).$$

Proof. From the definition, there must be a state $|\psi\rangle$ so that

$$\begin{aligned} E(\mathbf{U}_2\mathbf{U}_1, \mathbf{V}_2\mathbf{V}_1) &= \Big|\Big|(\mathbf{U}_2\mathbf{U}_1 - \mathbf{V}_2\mathbf{V}_1)|\psi\rangle\Big|\Big| \\ &= \Big|\Big|(\mathbf{U}_2\mathbf{U}_1 - \mathbf{V}_2\mathbf{U}_1)|\psi\rangle + (\mathbf{V}_2\mathbf{U}_1 - \mathbf{V}_2\mathbf{V}_1)|\psi\rangle\Big|\Big|, \end{aligned}$$

and using the triangular inequality,

$$\begin{aligned} E(\mathbf{U}_2\mathbf{U}_1, \mathbf{V}_2\mathbf{V}_1) &\leq \Big|\Big|(\mathbf{U}_2 - \mathbf{V}_2)\mathbf{U}_1|\psi\rangle\Big|\Big| + \Big|\Big|\mathbf{V}_2(\mathbf{U}_1 - \mathbf{V}_1)|\psi\rangle\Big|\Big| \\ &\leq E(\mathbf{U}_2, \mathbf{V}_2) + E(\mathbf{U}_1, \mathbf{V}_1). \end{aligned}$$

Corolary 3.1. For unitary transformations $\mathbf{U}_1, \ldots, \mathbf{U}_m$ and $\mathbf{V}_1, \ldots, \mathbf{V}_m$ we have

$$E(\mathbf{U}_m\mathbf{U}_{m-1}\cdots\mathbf{U}_1, \mathbf{V}_m\mathbf{V}_{m-1}\cdots\mathbf{V}_1) \leq \sum_{j=1}^{m} E(\mathbf{U}_j, \mathbf{V}_j).$$

Proof. This follows from the previous proposition by induction.

Instead of having to measure the $\varepsilon > 0$ tolerance mentioned above for the whole problem approximation, this result allows us to divide our problem into several instances and measure the total tolerance as the sum of the partial errors. This will be useful in the sequel.

3.3.2 Finding a finite universal set for 1-qubit operations

In this section we are going to prove that in order to find a universal quantum set we just need to focus on finding a set that generates some rotations of an irrational multiple of π. This statement is based on the fact that any unitary transformation in a quantum system can be represented by only this type of rotation.

We will first introduce a small technical proposition in order to help us provide the main result of the section.

Proposition 3.3. For arbitrary $\alpha, \beta \in [0, 2\pi)$, n a unitary vector,

$$E\left(\mathbf{R}_n(\alpha), \mathbf{R}_n(\alpha+\beta)\right) = \left|1 - e^{i\beta/2}\right|.$$

Proof. We need to compute, for all $|\psi\rangle$,

$$\begin{aligned}\left|\left|\left(\mathbf{R}_n(\alpha) - \mathbf{R}_n(\alpha+\beta)\right)|\psi\rangle\right|\right| &= \left|\left|\left(\mathbf{R}_n(\alpha) - \mathbf{R}_n(\alpha)\mathbf{R}_n(\beta)\right)|\psi\rangle\right|\right| \\ &= \left|\left|\mathbf{R}_n(\alpha)\cdot\left(\mathbf{I} - \mathbf{R}_n(\beta)\right)|\psi\rangle\right|\right| \\ &= \left|\left|\left(\mathbf{I} - \mathbf{R}_n(\beta)\right)|\psi\rangle\right|\right|,\end{aligned}$$

as $\mathbf{R}_n(\alpha)$ is unitary. If we then write, as in Proposition 2.3,

$$\mathbf{R}_n(\beta) = \cos\left(\frac{\beta}{2}\right)\mathbf{I} - i\sin\left(\frac{\beta}{2}\right)\left(n_x\mathbf{X} + n_y\mathbf{Y} + n_z\mathbf{Z}\right),$$

we can see that $\mathbf{I} - \mathbf{R}_n(\beta)$ can be written in matrix form as

$$\mathbf{M} = \begin{bmatrix} 1 - \cos(\beta/2) + in_z\sin(\beta/2) & -\sin(\beta/2)(n_y + in_x) \\ \sin(\beta/2)(n_y - in_x) & 1 - \cos(\beta/2) - in_z\sin(\beta/2) \end{bmatrix},$$

where $n = (n_x, n_y, n_z)$. This gives us, by an easy calculation,

$$\mathbf{M}^* \cdot \mathbf{M} = \left(2 - 2\cos\left(\frac{\beta}{2}\right)\right)\mathbf{I}$$

and therefore

$$\left|\left|\left(\mathbf{R}_n(\alpha) - \mathbf{R}_n(\alpha+\beta)\right)|\psi\rangle\right|\right|^2 = \left|\left|\left(\mathbf{I} - \mathbf{R}_n(\beta)\right)|\psi\rangle\right|\right|^2 = \left|\langle\psi|\mathbf{M}^*\cdot\mathbf{M}|\psi\rangle\right| = 2 - 2\cos\left(\frac{\beta}{2}\right),$$

which does not depend on $|\psi\rangle$, and the result follows from

$$\left|1 - e^{i\beta/2}\right| = \sqrt{2 - 2\cos\left(\frac{\beta}{2}\right)}.$$

Theorem 3.3. *A single qubit operation may be approximated to arbitrary accuracy using the Hadamard gate* $\mathbf{H}$ *and the* $\pi/8$ *gate* $\mathbf{T}$.

Proof. Let us say that $\varepsilon > 0$. Using Proposition 2.5, any quantum gate $\mathbf{U}$ action on a single qubit may be written (up to a phase change, which will not concern us for this matter) as

$$\mathbf{U} = \mathbf{R}_n(\alpha)\mathbf{R}_m(\beta)\mathbf{R}_n(\gamma),$$

where m and n are non–parallel vectors of $\mathbb{R}^3$ that can be chosen and $\alpha, \beta, \gamma \in [0, 2\pi)$ are then determined by $\mathbf{U}$. We will approximate each rotation factor separately and then apply Corollary 3.1.

Consider then a given angle $\alpha \in [0, 2\pi)$ and the gates $\mathbf{T}$ and $\mathbf{HTH}$. We are going to show that a successive product of these gates can be used to approximate $\mathbf{R}_n(\alpha)$, for a certain vector n, with an arbitrary accuracy. Note that

$$\begin{aligned}\mathbf{THTH} = \mathbf{R}_z\left(\frac{\pi}{4}\right)\mathbf{R}_x\left(\frac{\pi}{4}\right) &= \left(\cos\frac{\pi}{8}\mathbf{I} - i\sin\frac{\pi}{8}\mathbf{Z}\right)\left(\cos\frac{\pi}{8}\mathbf{I} - i\sin\frac{\pi}{8}\mathbf{X}\right)\\ &= \cos^2\frac{\pi}{8}\mathbf{I} - i\left(\cos\frac{\pi}{8}(\mathbf{X}+\mathbf{Z}) + \sin\frac{\pi}{8}\mathbf{Y}\right)\sin\frac{\pi}{8},\\ &= \cos^2\frac{\pi}{8}\mathbf{I} - i\sin\frac{\pi}{8}\left(\cos\frac{\pi}{8}\mathbf{X} + \sin\frac{\pi}{8}\mathbf{Y} + \cos\frac{\pi}{8}\mathbf{Z}\right),\end{aligned}$$

where we have used that $\mathbf{Y} = -i\mathbf{ZX}$.

If we look at the form of this expression, we find that it is quite similar to a rotation around the Bloch sphere (Proposition 2.3), where we can see that in this case it takes place around the axis

$$\overline{n} = \left(\cos\frac{\pi}{8}, \sin\frac{\pi}{8}, \cos\frac{\pi}{8}\right).$$

The difference lies in the fact that this vector $\overline{n}$ is not normalized. If we do normalize it so that it is written exactly as in Proposition 2.3, we get

$$\cos\left(\frac{\theta}{2}\right)\mathbf{I} - i\sin\left(\frac{\theta}{2}\right)\left(n_x\mathbf{X} + n_y\mathbf{Y} + n_z\mathbf{Z}\right),$$

where

$$\cos\left(\frac{\theta}{2}\right) = \cos^2\left(\frac{\pi}{8}\right), \qquad (n_x, n_y, n_z) \,||\, \left(\cos\frac{\pi}{8}, \sin\frac{\pi}{8}, \cos\frac{\pi}{8}\right), \qquad n_x^2 + n_y^2 + n_z^2 = 1.$$

and so we have a rotation around an axis collinear to $\overline{n}$ and through an angle θ, verifying

$$\theta = 2 \arccos\left(\cos^2\frac{\pi}{8}\right) \sim 1.09605682 \text{ rad}.$$

This angle θ is not a rational multiple of π but we will not prove this (the proof can be found in [1]). We will call this rotation (as usual) $\mathbf{R}_n(\theta)$.

Now we will show that by iterating $\mathbf{R}_n(\theta)$ we will be able to approximate any rotation $\mathbf{R}_n(\alpha)$ up to arbitrary accuracy. In order to see this let $\delta > 0$ be the desired accuracy, and N an integer larger than $2\pi/\delta$. Define θ_k the angle that we achieve after we apply k times the rotation so that $\theta_k \in [0, 2\pi)$ and

$$\theta_k = k\theta \mod 2\pi.$$

Due to irrationality, we will never hit 2π for any θ_j with $1 \leq j \leq N$, so there will be no repeats in the θ_j values. Using the pigeonhole principle, θ_j split the unit circle into N subintervals whose boundaries are the θ_j values. They would be most spread apart if they were equally distributed in the circle. Thus at least two elements θ_i, θ_j are at most $2\pi/N$ radians apart in absolute difference, at the two endpoints of the shortest interval. This then implies that there are different j and k in range $1, \ldots, N$ such that $|\theta_k - \theta_j| \leq 2\pi/N$. If we assume without loss of generality $k > j$ we then have

$$\left|\theta_{k-j}\right| < \frac{2\pi}{N} < \delta.$$

This, in turn, means that the sequence

$$\left\{\theta_{i(k-j)}\right\}_{i\in\mathbb{N}}$$

will approximate *any* given angle with accuracy better than δ. In particular there exists some $r \in \mathbb{N}$ such that $|\alpha - \theta_r| < \delta$. From Proposition 3.3 we get that

$$E\left(\mathbf{R}_n(\alpha), (\mathbf{THTH})^r\right) = E\left(\mathbf{R}_n(\alpha), \mathbf{R}_n(\theta)^r\right) = E\left(\mathbf{R}_n(\alpha), \mathbf{R}_n\left(\theta_r\right)\right) = \left|1 - e^{i|\alpha-\theta_r|/2}\right| < \left|1 - e^{i\delta/2}\right|.$$

So, taking an appropriate δ (precisely, one such that $\sqrt{2 - 2\cos(\delta/2)} < \varepsilon/3$), we get

$$E\left(\mathbf{R}_n(\alpha), (\mathbf{THTH})^r\right) < \frac{\varepsilon}{3}.$$

We need now a non–parallel axis m in order to fully factor $\mathbf{U}$. In order to do that, we can simply notice that, with the notations above, for every angle β we have,

$$\mathbf{H}\mathbf{R}_n(\beta)\mathbf{H} = \mathbf{R}_m(\beta),$$

with,

$$m = \left(\cos\frac{\pi}{8}, -\sin\frac{\pi}{8}, \cos\frac{\pi}{8}\right);$$

where we have used $\mathbf{HXH} = \mathbf{Z}$, $\mathbf{HZH} = \mathbf{X}$ and $\mathbf{HYH} = -\mathbf{Y}$ and $\mathbf{H}^2 = \mathbf{I}$.

So, using the same argument as above for $\mathbf{R}_n(\alpha)$, we can find some $s \in \mathbb{N}$ so that $|\beta - \theta_s| < \delta$ (same δ actually) and

$$E\left(\mathbf{R}_m(\beta), \mathbf{H}\cdot(\mathbf{THTH})^s\cdot\mathbf{H}\right) = E\left(\mathbf{H}\cdot\mathbf{R}_n(\beta)\cdot\mathbf{H}, \mathbf{H}\cdot(\mathbf{THTH})^s\cdot\mathbf{H}\right) = E\left(\mathbf{R}_n(\beta), (\mathbf{THTH})^s\right) < \frac{\varepsilon}{3}.$$

Of course the same goes for $\mathbf{R}_n(\gamma)$ and we get

$$E\left(\mathbf{R}_n(\gamma), (\mathbf{THTH})^t\right) < \frac{\varepsilon}{3},$$

for a certain $t \in \mathbb{N}$.

So as we factor $\mathbf{U}$ as

$$\mathbf{U} = \mathbf{R}_n(\alpha)\mathbf{R}_m(\beta)\mathbf{R}_n(\gamma),$$

we obtain have that there are $r, s, t \in \mathbb{N}$ such that

$$E\left(\mathbf{U}, (\mathbf{THTH})^r\cdot\mathbf{H}(\mathbf{THTH})^s\mathbf{H}\cdot(\mathbf{THTH})^t\right) < \varepsilon,$$

using Corollary 3.1. This finishes the proof. ∎

This proof not only helps us to prove the proposed theorem but also gives us a guideline for constructing a set of universal gates as we explained in the introduction to the section.

Finally if we want to implement any circuit containing m gates we just need to use the result given in the previous section by, using 1-qubit gates and $\mathbf{C}_{\mathbf{NOT}}$ gates, and where we replace the first ones with Hadamard and $\mathbf{T}$ gates. If the desired accuracy or the whole circuit is ε, we will have to approximate each 1-qubit gate to a tolerance of ε/m and, using Proposition 3.2, we obtain the result.

3.4 A note about complexity

The methods for approximating unitary transformations that we explained above appears to be pretty inefficient. Let us check the different stages of our decomposition for an n–qubit gate, which is realized as a $(2^n) \times (2^n)$ complex matrix:

1) Factoring our unitary matrix into a product of 2–level unitary gates needs $O\left((2^n)^2\right)$ matrices, as seen in section 3.2.2.
2) Every 2–level unitary matrix might need up to $2n$ $\mathbf{C_{NOT}}$ gates (for both uses of the Gray codes) and a 1–qubit gate, as shown in section 3.2.3.
3) The approximation a 1–qubit gate with $\mathbf{H}$ and $\mathbf{T}$ transformations with a tolerance $\varepsilon > 0$ needs $O(2^{1/\varepsilon})$ factors. This can be shown from the explicit computation of θ given in the proof of Theorem 3.3 [7].

Therefore a full approximation of a given unitary matrix in terms of the universal set $\{\mathbf{C_{NOT}}, \mathbf{H}, \mathbf{T}\}$ needs, at least theoretically, $O\left(4^n\left(2n + 2^{1/\varepsilon}\right)\right)$ gates.

This does not look practical at all. Fortunately, there is a powerful result called the Solovay–Kitaev theorem, that shows us that it is in fact possible to approximate any 1–qubit gate with an accuracy of ε with $O\left(\log^c(1/\varepsilon)\right)$ factors $\mathbf{H}$ and $\mathbf{T}$. The proof of this result goes far beyond the scope of this memoir, the first proof can be found in [5].

The c–constant appearing in the original statement can be shown to be $c \sim 3$ for any set of 1–qubit matrices which generate a dense subset in the special unitary group (that is, the unitary matrices with determinant 1). For some specific cases, c can achieve a value of 1 as shown in [3] although achieving this value seems rather difficult for the general case.

The Solovay–Kitaev implied a dramatic reduction in the expected number of gates a quantum computer needs in order to recreate a certain circuit, effectively establishing the feasibility of quantum computing as a valid and practical computation framework.

Bibliografía

[1] P.O. Boykin, T. Mor, M. Pulver, V. Roychowdhury and F. Vatan: *On universal and fault-tolerant quantum computing: a novel basis and a new constructive proof of universality for Shor's basis.* 40th Annual Symposium on Foundations of Computer Science (Cat. No.99CB37039) (1999) 486–494.

[2] D. Gottesman: *The Heisenberg representation of quantum computers.* International Conference on Group Theoretic Methods in Physics (1998), arXiv:quant-ph/9807006.

[3] A.W. Harrow, B. Recht, and I.L. Chuang: *Efficient Discrete Approximations of Quantum Gates.* Journal of Mathematical Physics **43** (2002) 4445–4451.

[4] P. Kaye, R. Laflamme, and M. Mosca: *An Introduction to Quantum Computing*, Oxford University Press (2006).

[5] A.Y. Kitaev: *Quantum computations: algorithms and error correction.* Russian Mathematical Surveys **52** (1997) 1191–1249.

[6] J. Lumbreras Zarapico: *Efficient unitary approximations in quantum computing: the Solovay-Kitaev Theorem.* Trabajo Fin de Grado, Universitat de Barcelona (2018).

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

[8] J. Ossorio-Castillo, and J.M. Tornero: *Quantum computing from a mathematical perspective: a description of the quantum circuit model* (2018), https://arxiv.org/abs/1810.08277.

[9] Mathematics Stack Exchange: *Prove that to fully connect n nodes we need at least $n-1$ pairwise links*, https://math.stackexchange.com/q/4027846.

[10] Mathematics Stack Exchange: *Decomposition of any* 2*–level matrix into single qubit and CNOT gates*, https://quantumcomputing.stackexchange.com/a/14314.