Quantum Computing

Team 191 :
Linda He
Rostane Feknous
Alexandre Morlat
Thibaut Lamoureux
Arthur Cluet

November 2022

Contents

1 General introduction

1.1 Preamble

Modern digital computers store the necessary data in magnetic memory in the form of 0-1 bits; they operate on this data via logic gates (circuits that combine the logic signals presented to their inputs as voltages).

The fact that the data is digitized has profound implications:
- Although considerable, the amount of data a computer can store and its processing speed are limited.
- The inevitable errors inherent in any physical device are relatively easy to control.

These computers have a basic component of computing, the microprocessor, whose complexity is measured by the number of transistors - an electronic component in most electronic circuits (logic circuits, amplifier, voltage stabilizer, signal modulation, etc.) in both low and high voltage - that it contains. In short, the more transistors a microprocessor contains, the more complex operations it will be able to perform, and/or process large numbers. Therefore, the performance of computer hardware, intrinsically linked to the number of transistors in the processor, follows a trend - Moore's empirical law - which states that the computing power of our computers doubles every 18 months. This is due to the fact that the engraving fineness of the transistors is reduced by half over this period. The problem is that this "law" will not be able to apply forever as we are inexorably approaching the size of atoms. This poses a problem because when we manipulate such small components, certain quantum phenomena such as the tunnel effect (electrons "pass through walls" instead of being stopped) appear and because of these, transistors no longer function as usual transistors.

Quantum computers, on the other hand, are designed to go beyond the classical and usual functioning (mechanical and numerical) and thus use quantum mechanical phenomena to solve computational problems that classical computers cannot solve efficiently (or at all). To build a functional quantum computer, one must overcome enormous difficulties and obtain conditions such that quantum effects become dominant or, at least, perceptible. Indeed, quantum states are extremely fragile, the very fact of observing them disturbs them. In order to manipulate qubits, it is generally necessary to use cold atoms, trapped and cooled by laser (at a temperature close to -273 °C, the absolute zero), which requires a cumbersome hardware. In short, such a computer must operate at low temperatures close to absolute zero.

1.1.1 How does a quantum computer work?

A quantum computer is the equivalent of a classical computer, except that its calculations are performed at the atomic scale. It is based on the laws of quantum physics, which is concerned with the behavior of matter and light at the microscopic level.

At this scale, strange and totally counter-intuitive phenomena occur:
-The principle of quantum superposition: An object can be in several states until it is measured.
-The principle of quantum entanglement: Two objects can influence each other even when separated by a great distance.

Thus, while the classical computer works with "bits", information stored in a binary way (with 0's and 1's), a quantum computer works with "qubits", made up of superpositions of states between 0 and 1. In a simplified way, instead of exploring each of the paths one by one, a quantum computer is able to explore all the paths at the same time. And thus to calculate much faster. Thus, for some computational tasks, quantum computing offers exponential speedups.

Let's go back to four fundamental notions mentioned:

1.1.2 What is wave packet reduction?

A simple way to understand this phenomenon is to consider the following analogy: We will imagine that a bit is materialized by a coin. When this coin is on the tails side, we can say that it codes the value 1, and when it is on the heads side, it codes the value 0.

In the macroscopic world, when a coin is placed on its edge, the slightest disturbance (jolt, sneeze...) makes it fall on the heads or tails side, with generally a 50% probability for each side. At the quantum scale, it is even more sensitive. So much so that it is in fact impossible to "see" the coin on its edge because the simple light disturbs the coin and makes it fall immediately. For the coin to be on edge, it must be placed in a perfectly sealed black box, with no interaction with you or the environment you are in. Under these strict conditions, the coin will spontaneously evolve on the edge, even on "tilted" edges. On the other hand, as soon as you open the box, the coin will instantly fall on heads or tails, with probabilities of 50/50... or even non-symmetrical probabilities depending on the inclination on which it was just before opening. This phenomenon is called wave packet reduction, also known as wave function collapse. Thus, in order for the quantum coin to be on the slice, one must neither look at it nor interact with it. Even if the coin can be on edge, it is impossible to "see" it in such a state; it can only be seen lying on its heads or tails side, after it has collapsed.

1.1.3 What is quantum superposition?

Imagine that you are exercising in your living room. You turn fully to the left, then fully to the right. Now you are trying to turn to the left and to the right at the same time. This is impossible to do unless you cut yourself in half! You obviously can't be in these two states simultaneously: facing left and facing right at the same time.

On the other hand, if you were a quantum particle, you could have a certain probability of facing the left side AND a certain probability of facing the right side thanks to a phenomenon known as superposition (or coherence).
Unlike classical particles, if two states A and B are valid quantum states of a quantum particle, any linear combination of the states is also a valid quantum state:

$$Qubit_{state} = \alpha A + \beta B.$$

This linear combination of quantum states A and B is called a superposition. Here, α and β are the probability amplitudes of A and B , respectively, so that :

$$|\alpha|^2 + |\beta|^2 = 1.$$

Only quantum systems like ions, electrons or superconducting circuits can exist in the superposition states that provide all the power of quantum computing. A quantum particle, like an electron, has its own "left or right side" property, called spin, which can be "up" or "down", so the quantum state of an electron is a superposition of "spin up" and "spin down".

In general, and to make the connection with classical binary computing, if a quantum system can be in two quantum states, these states are "state 0" and "state 1".

1.1.4 Qubits and probabilities

Classical computers store and process information in bits, which can have the state 1 or 0, but never both at the same time. The equivalent in quantum computing is the qubit. A qubit is a quantum system that can be in a superposition of two quantum states, 0 and 1. Each possible quantum state is associated with a probability amplitude (a weight in other words). Once you have measured a qubit, its state is reduced to state 0 or state 1 depending on the associated probability. Therefore, one of the possible states is obtained with a certain probability.

The probability of a qubit reducing in one direction or the other is determined by quantum interference. Quantum interference changes the state of a qubit to influence the probability of a certain outcome during measurement. This probabilistic state gives the full measure of the power of quantum

computing.

For example, with two bits in a conventional computer, each bit being able to store 1 or 0, you can store four possible values (00, 01, 10, and 11), but only one of these values at a time. With two qubits on top of each other, each qubit can be 1 or 0 or both, allowing you to represent the same four values simultaneously. With three qubits, you can represent eight values, with four qubits, 16 values, and so on.

1.1.5 What is entanglement?

One of the most interesting phenomena of quantum mechanics is the possibility that two or more quantum systems are entangled with each other. Entanglement is a quantum correlation between quantum systems. When qubits are entangled, they form a global system so that the quantum state of individual subsystems cannot be described independently. Two systems are entangled when the state of the global system cannot be written as a linear combination of the subsystems.

Intricate quantum systems can maintain this correlation even when they are separated by large distances. In other words, whatever operation or process you apply to one subsystem is correlated on the other subsystem as well. Because of the correlation between the intricate qubits, measuring the state of one qubit also provides information about the state of the other qubit. This particular property is very useful in the field of quantum computing.

Not all correlations between the measurements of two qubits mean that the two qubits are intricate. Classical bits can also be correlated. Two qubits are intricate when they have correlations that cannot be reproduced using classical bits. This difference between classical and quantum correlations is subtle, but it is essential to the speedup provided by quantum computers.

To simplify, let's take the analogy between bits and coins to illustrate this specificity of Qubits. Entanglement is a phenomenon that concerns not one but two (or more) quantum coins and that stipulates that, if your two coins have been created under very specific conditions, then their respective fates are linked. For example, if you glue two coins on their edges (like Siamese twins), they become linked. We can say here that our pieces are "intertwined". If one is on the tails side, the other is necessarily on the heads side. When you flip a coin with this set of two glued pieces, the two pieces will necessarily fall on opposite sides (one on tails, the other on heads). But where it becomes curious is that their osmosis will persist even if we move them away. Let's imagine that we finally detach the two pieces and lock the right one in an isolated black box. Now suppose that we flip a coin with the left piece. In this configuration, if the left coin lands on tails, then by opening the black box, the right coin will be on heads and vice versa. And this without anyone having touched the right-hand piece.

1.2 Why use quantum computers?

There are many ways to understand why quantum mechanics is difficult to simulate. Perhaps the simplest is to see that quantum theory can be interpreted to mean that matter, at the quantum level, is in a multitude of possible configurations (called states). Contrary to classical probability theory, these many potentially observable configurations of the quantum state can interfere with each other. This interference prevents the use of statistical sampling to obtain quantum state configurations. Instead, we must follow all possible configurations of a quantum system if we are to understand quantum evolution.

Let's take the example of an electron system in which the electrons can be in, say, any of 40 positions. The electrons can therefore be in any of the 2^{40} configurations (since each position may or may not have an electron). To store the quantum state of electrons in a conventional computer memory, more than 130 GB of memory would be needed! This is huge, but within the reach of some computers. If we allow the particles to be in any of 41 positions, there would be twice as many 2^{41} configurations which, in turn, would require over 260 GB of memory to store the quantum state. This game of increasing the number of positions cannot be played indefinitely if we want to store the state in a conventional way, as we quickly outgrow the memory capabilities of the world's most powerful machines. At a few hundred electrons, the memory required to store the system exceeds the number

of particles in the universe. Therefore, there is no hope that our conventional computers will ever simulate their quantum dynamics. And yet, in nature, such systems easily evolve over time according to the laws of quantum mechanics, oblivious to the inability to design and simulate their evolution with conventional computing power.

This observation led those with an early view of quantum computing to ask a simple but powerful question: can we turn this difficulty into an opportunity? Specifically, if quantum dynamics is difficult to simulate, what if we built hardware that had quantum effects as its fundamental operations? Could we simulate systems of interacting particles using a system that exploits exactly the same laws that naturally govern them? Could we study tasks that are completely absent from nature, but that follow or benefit from quantum mechanical laws? These questions gave birth to quantum computing.

The core of quantum computing is to store information in quantum states of matter and to use quantum gate operations to perform computations on that information, by exploiting quantum interference and learning to "program" it.

An early example of programming interference to solve a problem considered difficult for our conventional computers was done by Peter Shor in 1994 for a problem known as factoring. Solving factorization allows computing to break many of our public key encryption schemes that underpin today's electronic commerce security, including RSA and elliptic curve encryption. Since that time, fast and efficient quantum computing algorithms have been developed for many of our challenging classical tasks: simulating physical systems in chemistry, physics, and materials science, searching an unsorted database, solving systems of linear equations, and machine learning.

Finally, problems that a quantum computer can solve more efficiently than a classical computer are called BQP (bounded error quantum polynomial time), which means that they are parameterizable by a quantum computer in polynomial time. Examples of BQP problems include the factoring problem and the search problem.

1.3 What can quantum information be used for?

Although we are still not very mature on the subject of quantum computing, the latter would bring a lot of beautiful promises in algorithmic. It would be an algorithmic with a formalism that allows to take into account the special properties of quantum materials. In particular, it can be shown that for certain types of specific tasks such as solving systems of linear equations (matrix inversions) and decomposing integers into prime factors, the execution speed of quantum algorithms is theoretically superior to that of equivalent classical algorithms. Mathematicians say that the complexity - the evolution of the number of operations as a function of the number of input data - of classical algorithms is much higher than that of equivalent quantum algorithms, when the latter exist. For example, when the number to be factorized increases by one decimal place, the duration of the classical factorization algorithm increases almost exponentially, whereas for its quantum counterpart, the computation time increases "only" quadratically, i.e. much more slowly. Thus, these new computers would make it possible to solve some extremely time-consuming algorithms - several decades - in a "reasonable" time, i.e. from a few hours to a few days.

Also, a quantum computer is not a supercomputer that can do everything faster. One of the goals of quantum computing research is to study which problems can be solved by a quantum computer faster than by a classical computer and how big the speedup is. Quantum computers are extremely effective for problems that require the computation of a large number of different possible combinations. These types of problems arise in many areas, such as quantum simulation, encryption, quantum machine learning and classification or search problems more generally.

We then see two fundamental issues intrinsic to these problems:

1.3.1 Quantum simulation

The quantum mechanism is the underlying "operating system" of our universe. It describes how the basic building blocks of fundamental nature behave. Behaviors in nature, such as chemical and biological reactions or matter formations, often involve quantum many-body interactions. To simulate intrinsically quantum mechanical systems such as molecules, quantum computing holds promise because qubits can be used to represent the natural states in question. Examples of quantum systems we can model are photosynthesis, superconductivity, and complex molecular formations.

1.3.2 Quantum accelerations

One of the goals of quantum computing research is to study the problems that can be solved by a quantum computer faster than by a classical computer and the size of the speedup. Two well-known examples are Grover's algorithm and Shor's algorithm, which produce a polynomial and exponential speedup, respectively, compared to their classical counterparts.

Shor's algorithm:

Running on a quantum computer could break classical encryption schemes such as the RSA (Rivest-Shamir-Adleman) scheme, which is widely used in electronic commerce for secure data transmission. This scheme is based on the actual difficulty of factoring prime numbers using classical algorithms. Quantum encryption promises information security by exploiting basic physics rather than complexity assumptions. Like Shor's algorithm for factoring, the hidden shift problem is a natural source of problems for which a quantum computer has an exponential advantage over the best known classical algorithms. This can potentially help solve deconvolution problems and allow us to efficiently find patterns in complex data sets. It turns out that a quantum computer can in principle compute high-speed convolutions, which in turn relies on the ability of the quantum computer to compute Fourier transforms extremely fast.

Grover's algorithm:

Gives a big boost to the processing of unstructured data searches, reducing the number of search steps in a way that no conventional algorithm has done before. Indeed, any problem that allows you to check whether a given value x is a valid solution (a "yes or no problem") can be formulated in terms of a search problem. Here are some examples:

- Boolean satisfiability problem: does the set of Boolean values x constitute an interpretation (assignment of values to variables) that satisfies the given Boolean formula?
- Traveler problem: does X describe the smallest possible loop that connects all cities?
- Database search problem: does the database table contain a record x?
- Integer factorization problem: is the fixed number N divisible by the number x?

1.3.3 Quantum Machine Learning

Machine Learning on conventional computers is revolutionizing the world of business and science. However, the high cost of model training computations is hindering the development and expansion of this field. The field of Quantum Machine Learning is exploring ways to design and implement quantum software that would enable Machine Learning to be performed faster than conventional computers.

We have previously discussed prime factor decomposition. This algorithm, emblematic of cryptography (a fortiori of cybersecurity), is indeed one of the best illustrations of a very long algorithm, even impossible to execute with a classical computer. If we look at Machine Learning algorithms, used in the framework of artificial intelligence projects, we also have to deal with computation times that are sometimes considerable: matrix inversions, gradient descent, backpropagation...

Consequently, the Machine Learning algorithms theoretically concerned by quantum acceleration are well known to Data Scientists. Among the most famous are factor analysis (e.g. PCA), most clusterings (e.g. K-Means), wide margin separators (SVM), Boltzmann machines, Bayesian inference, reinforcement learning, optimization algorithms (search for global extrema), etc. More generally, these are often algorithms whose resolution requires linear operations (or linear cost functions), since quantum computations are themselves linear. In neural networks, it is therefore more delicate because the activation functions are not linear. Nevertheless, one can use quantum gas pedals for unsupervised pre-training steps based on Boltzmann machines, for example.

Because of the brakes induced by the explosion of the volume of data, democratizing quantum computing - which is far from being an easy task - would therefore allow certain algorithms to be moved from theory to practice.

Should we then switch to the quantum processor?

From the beginning, by taking stock of the current technology related to classical computers, their intrinsic performances and their evolutionary perspectives, we can think of forcing the transition to the quantum processor in order to exceed the limits considered by Moore's law.

However, there are reasons why this transition is not yet a reality:

1. Not all Machine Learning algorithms are affected by quantum acceleration.
2. The operation of current quantum computers is far from being conclusive because the results of the algorithms contain many errors due to the great sensitivity of the physical phenomena which are present.
3. The current cost of quantum prototypes is prohibitive.

Nevertheless, although the major players such as IBM, Atos or Microsoft only market quantum "simulators" today ("classical" supercomputers capable of rapidly evaluating all the possible solutions of a quantum system), it seems appropriate for data teams to start testing quantum algorithms, or at least to "think quantum" with more traditional algorithms, on well-targeted use cases such as fraud detection, genomics or the prediction of thermo-sensitive phenomena. This early learning could indeed be a significant competitive advantage when the first real quantum processors are commercialized on a medium scale, probably in about ten years. It should be noted that the work on these so-called quantum processors is leading more and more to functional results, but for the moment they are not marketable.

Quantum theory at the service of artificial intelligence:
Issues:
Previously, we have seen that quantum computing would in principle be able to significantly increase the execution speed of certain Machine Learning algorithms. This is due to the fact that certain phenomena at the heart of such computers (superposition of states, entanglement of qubits, decoherence of superimposed states, abrupt reduction of the wave packet) make it possible to parallelize calculations by storing several pieces of information (superimposed) with the same particle. By considerably accelerating certain algorithmic treatments, we understand that quantum will allow AIs to solve new problems. Some people go even further and see a potential to make an artificial brain as efficient as the human one, able to integrate abstract concepts such as consciousness and feelings. The question that arises then is whether quantum AI would be more likely to become a "strong AI".

Before looking at the complementarities of quantum and strong AI, it is best to start by delimiting the notion of human intelligence.

1.3.4 What is human intelligence

It can be considered as a set of cognitive abilities that allow humans to learn, understand, communicate, think and act in a relevant, rational, logical or emotional way, depending on the situation they

are in.

Although there is not yet an absolute consensus on the question, the cognitive skills specific to human intelligence could be classified into broad categories (somewhat rearranged here) such as:

- Logical and quantitative intelligence: capacity of abstraction and formalization, rationality and reasoning.
- Literary and philosophical intelligence: language, reading, writing, structuring of thought and ideas.
- Social and emotional intelligence: introspection, ability to interact with others, ability to understand, perceive, feel and express emotions.
- Reactive and psychomotor intelligence: attention, coordination, parallel processing of multiple stimuli, mental endurance.
- Perceptual and artistic intelligence: visual, auditory, spatial, temporal, kinesthetic representations...
- Memory intelligence: solicitation of short, medium or long term memory depending on the task.

Obviously, these declinations of intelligence have overlaps and can be reorganized differently. Moreover, some of them are correlated in a subtle way, in particular with memory. This is why theoretical models tend to distinguish between so-called "fluid" and "crystallized" intelligences, in order to differentiate between processing that requires or does not require learning and knowledge that is permanently present in memory. There is a cold analogy with computers, which can use information permanently stored on hard disks or that which is temporarily present in RAM (random access memory), or even in cache.

But above all, scientists like Spearman (the father of the eponymous correlation coefficient) have long wanted to demonstrate the existence of a "g-factor" - a variable that characterizes the positive correlations that empirical research has consistently found between tests of mental ability, regardless of the content of the tests - that would consolidate all of these abilities into a single form of general intelligence. However, to date, there is no real consensus on a model.

1.3.5 Can we materially approach thought by quantum artificial intelligence?

Even without taking into account these metaphysical considerations, one feels that a Turing machine - the computational model on which traditional algorithms are based -, although able to simulate many things, will perhaps never be intrinsically capable of feeling, perceiving and reasoning like a human. This is why, in order to bypass the barrier of determinism and perception of reality, in order to realize an artificial consciousness, some would be tempted to call upon quantum computing, whose fundamental characteristic is to exploit the properties of particles to carry out intrinsically parallelized operations whose results are no longer deterministic but probabilistic. This choice is based on the fact that some scientists such as Penrose and Hameroff have put forward the hypothesis that quantum phenomena (superposition and entanglement) would be present in our brain via the spins (intrinsic magnetization) of the proteins present in the neuronal microtubules which would behave like the qubits of a quantum computer.

It may indeed seem natural to use Dirac's formalism (at the basis of the algebra modeling quantum mechanics) and qubits to describe the evolution of these cerebral micro-systems. There is for example a similar will in the field of atomistics to describe with precision the molecular orbitals. This is like using quantum computing to simulate quantum physics. It is a bit like using traditional computing to simulate the electronics of processors.

Nevertheless, this theory is highly debated because apart from photons (constituents of light), there is almost no physical system at room temperature capable of making quantum phenomena last long enough (we talk about decoherence time, generally well below a billionth of a millisecond for natural molecular aggregates) for them to have any impact on neuronal functioning.

So even if quantum computing would allow vertical AIs to solve problems that are unsolvable today, it would not seem capable of adding new properties that could only be attributed to the biological brain, whose quantum capacities are themselves debatable. In other words, to repeat the previous diagrams, for some people, quantum technology can a priori only advance AI along the performance axis.

Thus, despite the suggested inability of quantum computing to advance AI along the axes of adaptability to various tasks and consciousness, it still offers excellent prospects for optimization in general.

1.4 What is optimization?

Optimization is the process of finding the best possible option to solve a problem, given the desired result and the constraints. The best option can be the one with the lowest cost, the fastest runtime or the lowest environmental impact...etc.

Optimization is a class of computational problems that are prime candidates for future execution on quantum computers, providing a quantum advantage over classical solutions.

1.4.1 Quantum optimization

Simulating quantum effects on classical computers has led to the development of new types of quantum solutions. Quantum-inspired optimization algorithms exploit some of the advantages of quantum computing on classical hardware, which provide an acceleration over traditional approaches.

Quantum-inspired algorithms are algorithms that classically emulate the essential quantum phenomenon that produces the speedup. There are many types of quantum-inspired algorithms. One commonly used algorithm is based on a computational model called adiabatic quantum computing, which is as follows:

1. Prepare a system and initialize it to its lowest energy state.
2. Then transform this system little by little into a more complex system that describes the problem to be solved. The adiabatic model states that, as long as this transformation occurs slowly enough, the system has time to adapt and remains at this lowest energy configuration level. Once the transformations have taken place, we can solve the problem.

1.4.2 When should we consider quantum optimization?

Applying quantum-inspired optimization to real-world problems can provide companies with new insights or help them reduce costs by making their processes more efficient. Quantum-inspired optimization offers the following opportunities:

- Find a solution faster than with other optimization techniques for an established use case and consistent solution quality.
- Find a solution of higher quality than other optimization techniques for a given problem and a given time.
- Use a more realistic model than other optimization techniques by extending the problem to take into account more variables.

- The following are the necessary conditions for quantum-inspired optimization to perform well compared to other classical optimization algorithms:
- Optimization landscapes must be irregular but structured. This type of landscape is common in real problems.
- If the number of variables is small (e.g., less than a hundred), simplistic algorithms are already sufficient. For problems with hundreds of variables, quantum-inspired optimization can provide considerable improvements over some previously used methods.
- The problem parameters that affect the chosen cost metric must be represented by the variables of a cost function. It is sufficient to express the cost functions as polynomial functions over binary variables.

1.4.3 Solving problems with quantum optimization

There are various methods to find the global minimum of a cost function. One of the most efficient and widely used heuristics is simulated annealing. A heuristic is a technique for finding an approximate solution, especially in situations where finding an exact solution may take too much time. This technique is comparable to a random walk in a search space, where each walker creates a path in the optimization landscape.
With the simulated annealing method, the algorithm simulates a walker that ideally always makes a downward move, but can also make an upward move with non-zero probability. This gives the walker the ability to escape local minima and then further analyze nearby minima. Upward movements are called thermal jumps. Indeed, simulated annealing is an algorithm from physics that mimics the behavior of materials when they are slowly cooled.
Quantum-inspired optimization uses the combinatorial problem solving techniques of simulated annealing but applies quantum mechanical effects.
Quantum annealing is a quantum algorithm in the same spirit as simulated annealing, but it differs in several ways. In simulated annealing, the search space is explored by making thermal jumps from one solution to the next, whereas quantum annealing uses a quantum effect called quantum tunneling, which allows the walker to cross these energy barriers.

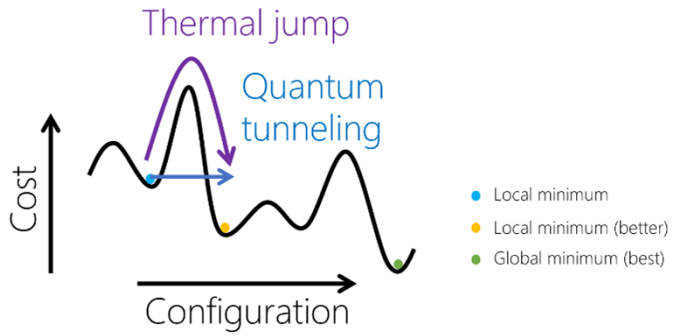


Figure 1: Quantum property

2 Fundamental concepts

2.1 Brief introduction to quantum linear algebra

In this part, we introduce some tools necessary for a good understanding of the fundamental concepts of quantum computing. An important point to underline and we will come back to it later is that it is not necessary to understand in detail the theory and the formalism in order to implement quantum programs.

2.1.1 Vectors and matrices in quantum computing

A qubit can be in state 1 or 0, or a superposition of both. In linear algebra, the state of a qubit is described by a vector and represented by a one-column matrix $\begin{bmatrix} a \\ b \end{bmatrix}$. This is the quantum state vector and it must satisfy the condition: $|a|^2 + |b|^2 = 1$.

In this condition, $|a|^2$ corresponds to the probability (weight) that the state of the qubit is reduced to 0 while $|b|^2$ corresponds to the probability that the state of the qubit is reduced to 1.

$$\begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{-1}{\sqrt{2}} \end{bmatrix}, \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{i}{\sqrt{2}} \end{bmatrix}, \begin{bmatrix} \frac{i}{\sqrt{2}} \\ \frac{i}{\sqrt{2}} \end{bmatrix}, \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{-i}{\sqrt{2}} \end{bmatrix} \; etc..$$

2.1.2 Quantum operators

Quantum operations can also be represented by a matrix. When a quantum operation is applied to a qubit, the two matrices that represent them are multiplied, and the result given in response represents the new state of the qubit after the operation.

Pauli Operator :

The operation X is represented by the Pauli matrix $X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$ which is used to invert the qubit states.

Let us consider, a qubit q of quantum state $\psi_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$, then the inverse state of ψ_1 is determined by :

$$\psi_2 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$

Hadamard Operator :

Operation H is characterized by the transformation of Hadamard $H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$ which allows to put a qubit in a state of superposition in which it is found with a probability of being reduced to 0 or 1. Let us take again the qubit q defined previously, we have then :

$$\psi_{superpositon} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

This qubit verifies the characteristic condition of any qubit except that the weight of each of the two states is identical such that $|a|^2 = |b|^2 = \frac{1}{2}$.

From these two operators, one important property is that a matrix that represents a quantum operation has a requirement: it must be a unitary matrix - a matrix whose inverse is equal to the conjugate transpose matrix - .(We explain this property in the following section.).

Conjugate transpose of matrix:

In some of the following parts we will use the notion of conjugate transpose to introduce some results. It is a simple result which states that if we consider the matrix :

$$A = \begin{bmatrix} 1 & -2-i & 5 \\ 1+i & i & 4-2i \end{bmatrix}$$

We transpose it :

$$A^t = \begin{bmatrix} 1 & 1+i \\ -2-i & i \\ 5 & 4-2i \end{bmatrix}$$

We conjugate it :

$$A^* = \begin{bmatrix} 1 & 1-i \\ -2+i & i \\ 5 & 4+2i \end{bmatrix}$$

We obtain then for our example, the result noted :

$$A^* = A^\dagger = \begin{bmatrix} 1 & 1-i \\ -2+i & i \\ 5 & 4+2i \end{bmatrix}$$

2.1.3 Tensor product

An important thing to understand before starting this part is that any Qubit is mathematically represented by a tensor - a very general object, whose value is expressed in a vector space -. Therefore, to represent the combined state of two or more qubits, it is not possible to use a simple multiplication. It is then necessary to consider the tensor product - product of each component of a tensor by each component of another tensor - noted $\otimes$.
The tensor product of the two qubit states: $\begin{bmatrix} a \\ b \end{bmatrix}$ and $\begin{bmatrix} c \\ d \end{bmatrix}$ is calculated as follows:

$$\begin{bmatrix} a \\ b \end{bmatrix} \otimes \begin{bmatrix} c \\ d \end{bmatrix} = \begin{bmatrix} a \begin{bmatrix} c \\ d \end{bmatrix} \\ b \begin{bmatrix} c \\ d \end{bmatrix} \end{bmatrix} = \begin{bmatrix} ac \\ ad \\ bc \\ bd \end{bmatrix}.$$

The result is a four dimensional matrix, where each element represents a probability. We have for example respectively :

- $|ac|^2$ represents the probability that the qubits are reduced to states 0 and 0 ;
- $|ad|^2$ represents the probability that the qubits are reduced to states 0 and 1 ;
- $|bc|^2$ represents the probability that the qubits are reduced to states 1 and 0 ;
- $|bd|^2$ represents the probability that the qubits are reduced to states 0 and 1;

The state of these two qubits combined $\begin{bmatrix} ac \\ ad \\ bc \\ bd \end{bmatrix}$ must now check the condition :

$$|ac|^2 + |ad|^2 + |bc|^2 + |bd|^2 = 1.$$

In order to represent a valid quantum state.

2.2 The qubit in quantum computing

In the following section, we take up in more detail some concepts presented previously to explain the construction of this first fundamental tool of quantum computing.

2.2.1 Representation of a qubit

We have previously introduced the fact that if a bit, or binary digit, can have the value 0 or 1, a qubit, can have the value 0, 1 or a quantum superposition of 0 and 1.
The state of a single qubit can be described by a two-dimensional column vector of unit norm, i.e. the sum of the squares of the amplitudes of its inputs must be equal to 1 . This vector, called the quantum state vector, contains all the information necessary to describe the one-qubit quantum system just as a single bit contains all the information necessary to describe the state of a binary variable. By this construction, any two-dimensional column vector of real or complex numbers of norm 1 represents a possible quantum state contained by a qubit.

Thus, $\begin{bmatrix} \alpha \\ \beta \end{bmatrix}$ represents the state of a qubit if α and β are complex numbers satisfying :

$$|\alpha|^2 + |\beta|^2 = 1.$$

The quantum state vectors $\begin{bmatrix}1\\0\end{bmatrix}$ and $\begin{bmatrix}0\\1\end{bmatrix}$ play a special role. These two vectors form a basis for the vector space that describes the qubit state. This means that any quantum state vector can be written as a linear combination of these basis vectors. This means that for any vector $\begin{bmatrix}x\\y\end{bmatrix}$,

$$\begin{bmatrix}x\\y\end{bmatrix} = x\begin{bmatrix}1\\0\end{bmatrix} + y\begin{bmatrix}0\\1\end{bmatrix}.$$

This result is kept and these two vectors then form the basis of the calculation for the study.
In the following, we arbitrarily use a specific notation for this basis:

$$|0\rangle := \begin{bmatrix}1\\0\end{bmatrix} \qquad |1\rangle := \begin{bmatrix}0\\1\end{bmatrix},$$

Thus, out of the infinite number of possible one-qubit quantum state vectors, only two correspond to classical bit states. This is not the case for all other quantum states.

In summary, we have the following comparison between bits and qubits:

	Bits : 0 **or** 1	Qubits : 0 **and** 1
States	two exclusive possible states	two possible simultaneous states
Initialization	0 or 1	0
Intern representation	0 or 1	two-dimensional vectors
Internal dimensionality	1 binary digit	Two complex numbers
Modifications	Logic gates	Quantum gates
Interpretation	0 or 1 deterministic	0 or 1, probabilistic

Table 1: Bits and Qubits

2.2.2 Measurement of a qubit

Knowing now how to represent a qubit, we can intuitively determine what these states represent by approaching the concept of measurement. A measurement corresponds to the informal idea of "observation" of the qubit, which immediately reduces the quantum state to one of the two classical states $\begin{bmatrix}1\\0\end{bmatrix}$ and $\begin{bmatrix}0\\1\end{bmatrix}$. When a qubit given by the quantum state vector $\begin{bmatrix}\alpha\\\beta\end{bmatrix}$ is measured, we obtain the result 0 with probability $|\alpha|^2$ and the result 1 with probability $|\beta|^2$.

- If the result is 0, the new state of the qubit is $|0\rangle = \begin{bmatrix}1\\0\end{bmatrix}$.
- If the result is 1, the new state of the qubit is $|1\rangle = \begin{bmatrix}0\\1\end{bmatrix}$.

The measurement properties also mean that the overall sign of the quantum state vector does not matter. Making a vector negative is equivalent to $\alpha \to -\alpha$ and $\beta \to -\beta$. Since the measurement probability of 0 and 1 depends on the square of the amplitudes of the terms, inserting these signs does not change the probabilities. These sign changes correspond to phases called global phases. Rather than writing ± 1, these phase variations will be represented by a complex exponential $e^{i\phi}$.

A final important property of measurement is that it does not necessarily destroy all quantum state vectors. If we start with a qubit in the $\begin{bmatrix}1\\0\end{bmatrix}$ state corresponding to the classical state 0, measuring this state will always produce the result 0 and leave the quantum state unchanged. In this sense, if there are only classical bits (for example, qubits that are $\begin{bmatrix}1\\0\end{bmatrix}$ and $\begin{bmatrix}0\\1\end{bmatrix}$), then the measurement does not vary the system in any way and therefore does not damage it.

This means that we can replicate classical data and manipulate it on a quantum computer just as we could with a classical computer. However, the ability to store information in both states at once is what elevates quantum computing beyond what is possible with the classical approach and prevents quantum computers from copying quantum data in an indiscriminate manner.

2.2.3 Bloch's sphere

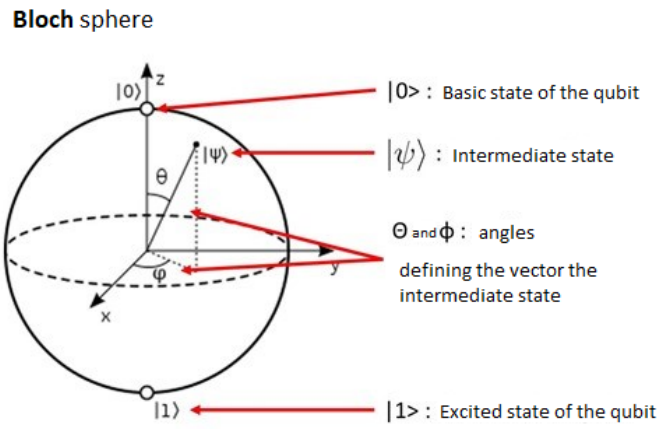


Figure 2: Bloch's Sphere representation

In the Bloch sphere, the $|0\rangle$ state of a two-state quantum is represented by the position of a vector of length 1 running from the center of the sphere to the north pole of the sphere and the $|1\rangle$ state is a vector running from the center of the sphere to its south pole. The intermediate states are represented by vectors from the center of the sphere that are always of length 1 with an angle θ with respect to the vertical z and an angle ψ with respect to the x-axis located going from the center of the sphere to its equator and around the z-axis.
The probabilistic model established for the characterization of the qubit was developed by Max Born in 1926. It gives the square of the modulus of the wave function of a quantum the meaning of a probability density of the presence of an elementary particle. This is related to the fact that the state $|0\rangle$ and the state $|1\rangle$ do not correspond to a precise position of a particle but are represented by the Schrödinger wave function which describes the probabilistic distribution of the quantum state in time and space. It is applied here in space.
The set of information given by the bloch sphere can be formalized as follows:

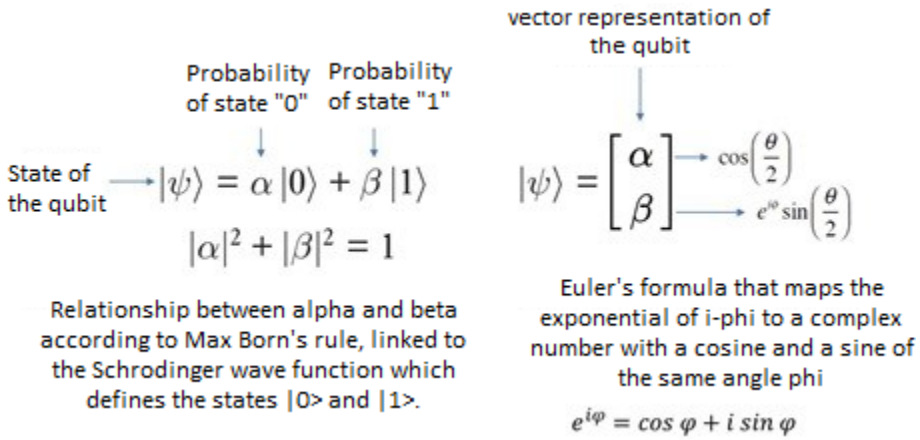


Figure 3: States explained

Thus, the Bloch sphere offers a way to describe a one-qubit quantum state (a complex two-dimensional vector) as a real-valued vector in three dimensions. This aspect is important because we can then visualize one-qubit states and thus develop a valuable reasoning for understanding multi-qubit states (whose representation unfortunately does not work with the Bloch sphere). We can give an example based on the previous considerations to better understand this tool:

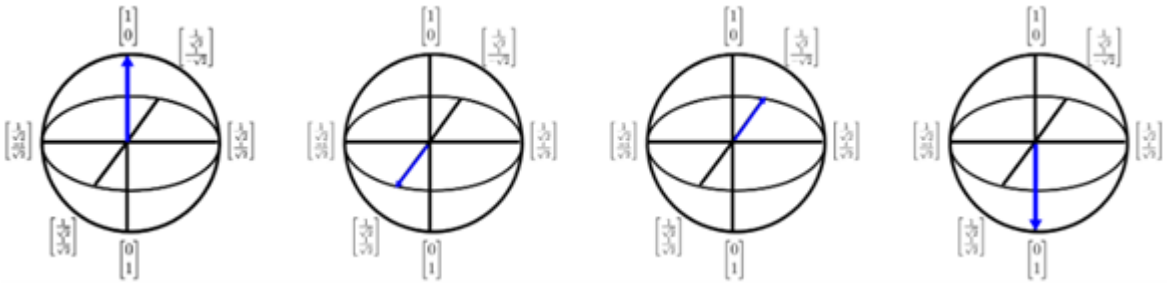


Figure 4: Bloch's Sphere representation

The arrows in this diagram indicate the direction of the quantum state vector, and each transformation of the arrow can be thought of as a rotation about one of the cardinal axes. Viewing a quantum computation as a sequence of rotations provides excellent intuitive reasoning, which is, however, difficult to use to design and describe algorithms.

2.3 Several qubits

The true power of quantum computing only becomes apparent as you increase the number of qubits. Single-qubit gates have some counterintuitive features, such as the ability to be in multiple states at once. However, if all you had in a quantum computer were single-qubit gates, then a calculator and certainly a classical supercomputer would dwarf its computing power.
Part of the power of quantum computing is that the dimension of the vector space of quantum state vectors increases exponentially with the number of qubits. This means that while a single qubit can be modeled very simply, simulating a 50-qubit quantum computation will likely exceed the limits of existing supercomputers. Increasing the size of the single qubit computation doubles the memory that is needed to store the state, and more or less doubles the computation time. This rapid doubling of computational power is the reason why a quantum computer with a relatively small number of qubits can vastly outperform the most powerful supercomputers of today, tomorrow, and even beyond, for certain computational tasks.

2.3.1 Representation of two qubits

If we consider two distinct qubits :

- q1 of state $\psi_1 = \begin{bmatrix} \alpha \\ \beta \end{bmatrix}$.
- q2 of state $\psi_2 = \begin{bmatrix} \delta \\ \gamma \end{bmatrix}$.

Then the corresponding two-qubit state is given by :

$$\psi_1 \otimes \psi_2 = \begin{bmatrix} \alpha \\ \beta \end{bmatrix} \otimes \begin{bmatrix} \delta \\ \gamma \end{bmatrix} = \begin{bmatrix} \alpha \begin{bmatrix} \delta \\ \gamma \end{bmatrix} \\ \beta \begin{bmatrix} \delta \\ \gamma \end{bmatrix} \end{bmatrix} = \begin{bmatrix} \alpha\delta \\ \alpha\gamma \\ \beta\delta \\ \beta\gamma \end{bmatrix}.$$

Therefore, given the two single-qubit states ψ and ϕ , each with 2 dimensions, the corresponding state of the two qubits $\psi \otimes \phi$ will have 4 dimensions. Moreover, the tensor represents a quantum state if :

$$|\alpha\delta|^2 + |\alpha\gamma|^2 + |\beta\delta|^2 + |\beta\gamma|^2 = 1$$

More generally, you can see that the quantum state of n qubits is represented by a unit vector $v_1 \otimes \cdots \otimes v_n$ of dimension $2 * 2 * ... * 2 = 2^n$ using this construction. As with single qubits, the multi-qubit quantum state vector contains all the information necessary to describe the behavior of the system.
The dimension 4 computational basis for two-qubit states is formed by the tensor products of the elements of the dimension 2 basis for single qubits.

$$00 := \begin{bmatrix}1\\0\end{bmatrix} \otimes \begin{bmatrix}1\\0\end{bmatrix} = \begin{bmatrix}1\\0\\0\\0\end{bmatrix} \qquad 01 := \begin{bmatrix}1\\0\end{bmatrix} \otimes \begin{bmatrix}0\\1\end{bmatrix} = \begin{bmatrix}0\\1\\0\\0\end{bmatrix}$$

$$10 := \begin{bmatrix}0\\1\end{bmatrix} \otimes \begin{bmatrix}1\\0\end{bmatrix} = \begin{bmatrix}0\\0\\1\\0\end{bmatrix} \qquad 11 := \begin{bmatrix}0\\1\end{bmatrix} \otimes \begin{bmatrix}0\\1\end{bmatrix} = \begin{bmatrix}0\\0\\0\\1\end{bmatrix}$$

Note that although one can always take the tensor product of two single-qubit states to form a two-qubit state, not all two-qubit quantum states can be written as the tensor product of two single-qubit states. For example, there are no :

$$\psi = \begin{bmatrix}\alpha\\\beta\end{bmatrix} \textit{ and } \phi = \begin{bmatrix}\gamma\\\delta\end{bmatrix} \textit{ such that } \psi \otimes \phi = \begin{bmatrix}\alpha\delta\\\alpha\gamma\\\beta\delta\\\beta\gamma\end{bmatrix} = \begin{bmatrix}\frac{1}{\sqrt{2}}\\0\\0\\\frac{1}{\sqrt{2}}\end{bmatrix}$$

This type of two-qubit state, which cannot be written as the tensor product of single-qubit states, is called an "intricate state". The two qubits are said to be intricate. To summarize, since the quantum state cannot be written as a tensor product of single-qubit states, the information that the state contains is not restricted to either qubit. Instead, the information is stored non-locally, in the correlations that exist between the two states. The fact that information is not stored locally is one of the main features of quantum computing. Furthermore, this feature is essential for a number of quantum protocols, including quantum teleportation and quantum error correction.

2.3.2 Measuring states with two qubits

Measuring two-qubit states is very similar to measuring single-qubit states. Indeed, measuring the state: $\begin{bmatrix}\alpha_{00}\\\alpha_{01}\\\alpha_{10}\\\alpha_{11}\end{bmatrix}$ gives:

- 00 with a probability $|\alpha_{00}|^2$;
- 01 with a probability $|\alpha_{01}|^2$;
- 10 with a probability $|\alpha_{10}|^2$;
- 11 with a probability $|\alpha_{11}|^2$;

After the measure,

- If the result is 00 , then the quantum state of the two-qubit system has been reduced and now has the value $00 := \begin{bmatrix}1\\0\\0\\0\end{bmatrix}$
- If the result is 01 , then the quantum state of the two-qubit system has been reduced and now has the value $01 := \begin{bmatrix}0\\1\\0\\0\end{bmatrix}$

- If the result is 10 , then the quantum state of the two-qubit system has been reduced and now has the value $10 := \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$
- If the result is 11 , then the quantum state of the two-qubit system has been reduced and now has the value $11 := \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$

It is also possible to measure a single qubit in a two-qubit quantum state. If you measure only one of the qubits, the impact of the measurement will be slightly different, because the entire state is not reduced to the state of a computational basis, but rather to that of a single subsystem. In other words, in such cases, the measurement of a single qubit reduces only one of the subsystems.

Let's take a look at an example to such a process. Let two qubits at the state $|0\rangle$. The state vector given by : $H^{\otimes 2}\left(\begin{bmatrix} 1 \\ 0 \end{bmatrix} \otimes \begin{bmatrix} 1 \\ 0 \end{bmatrix}\right)$ verifies :

$$H^{\otimes 2}\left(\begin{bmatrix} 1 \\ 0 \end{bmatrix} \otimes \begin{bmatrix} 1 \\ 0 \end{bmatrix}\right) = \frac{1}{2}\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}\begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \frac{1}{2}\begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \mapsto \begin{cases} \text{outcome } = 0 & \frac{1}{\sqrt{2}}\begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{bmatrix} \\ \text{outcome } = 1 & \frac{1}{\sqrt{2}}\begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \end{bmatrix} \end{cases}.$$

Thus, in this case, if the outcome is 0, then the measured qubit state is $\frac{1}{\sqrt{2}}\begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{bmatrix}$ and if the outcome is 1, then the measured qubit state is $\frac{1}{\sqrt{2}}\begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}$. We can understand this by analogy to the previously defined quantum calculation base for single qubits.

2.4 Dirac Notation

2.4.1 Why should we consider a specific notation ?

Dirac notation is a mathematical notation used to describe quantum states and quantum operations in quantum mechanics. It was developed by the British physicist Paul Dirac and is widely used in quantum physics and quantum computing.

In Dirac notation, quantum states are represented by vectors in a complex state space. Quantum operations are represented by operators acting on these states.

While column vector notation is common in linear algebra, it's often cumbersome in quantum computing, especially when dealing with multiple qubits. For example, when you define ψ to be a vector it's not explicitly clear whether ψ is a row or a column vector. Thus, if ϕ and ψ are vectors, then it's equally unclear if $\phi\psi$ is even defined, because the shapes of ϕ and ψ may be unclear in the context. Beyond the ambiguity about the shapes of vectors, expressing even simple vectors using linear algebra's notation can be cumbersome. For example, if you wish to describe an n -qubit state where each qubit

takes the value 0 , then you would formally express the state as

$$\begin{bmatrix}1\\0\end{bmatrix} \otimes \cdots \otimes \begin{bmatrix}1\\0\end{bmatrix}.$$

Evaluating this tensor product is impractical because the vector lies in an exponentially large space. As such, this notation is, in fact, the best description of the state that can be given using the previous notation.

2.4.2 Classical notation

There are two types of vectors in Dirac notation: the bra vector and the ket vector, so named because when put together they form a "braket" or inner product. If ψ is a column vector, then you can write it in Dirac notation as $|\psi\rangle$, where the $|.\rangle$ indicates that it is a unit column vector. Similarly, the row vector ψ is expressed as $\langle\psi|$. In other words, $\psi^\dagger$ is obtained by applying an input complex conjugation to the elements of the transpose of ψ. The bra-ket notation directly implies that $\langle\psi|\psi\rangle$ is the inner product of the vector ψ with itself, which is by definition 1.
More generally, if ψ and ϕ are quantum state vectors, then their inner product is $\langle\phi|\psi\rangle$. This inner product implies that the probability of measuring the state $|\psi\rangle$ to be $|\phi\rangle$ is $|\langle\psi|\psi\rangle|^2$.
It is this formalism that introduces the notation that we have used previously and by which we consider that :

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

2.4.3 Tensor product

Dirac notation also includes an implicit tensor product structure. This structure is important because in quantum computing, the state vector described by two uncorrelated quantum registers is the tensor products of the two state vectors.
The tensor product structure implies that we can write $\psi \otimes \phi$ for any two quantum state vectors ϕ and ψ as $|\psi\rangle \otimes |\phi\rangle$. The $\otimes$ notation convention is not mandatory, we can simply write it $|\psi\rangle|\phi\rangle$ which is equal to $|\psi\phi\rangle$. For example, we have :

$$|0\rangle \otimes |0\rangle = |0\rangle|0\rangle = |00\rangle = \begin{bmatrix}1\\0\end{bmatrix} \otimes \begin{bmatrix}1\\0\end{bmatrix} = \begin{bmatrix}1\\0\\0\\0\end{bmatrix}.$$

In the same way, any natural number k can be represented by the dirac notation by considering its binary representation. For example :

$$|1\rangle|0\rangle|1\rangle = |101\rangle = |5\rangle.$$

Within this notation, $|0\rangle$ need not refer to a single-qubit state but rather to a qubit register that stores a binary encoding of 0 . The differences between these two notations is usually clear from the context. This convention is useful for simplifying the first example which can be written in any of the following ways:

$$\begin{bmatrix}1\\0\end{bmatrix} \otimes \cdots \otimes \begin{bmatrix}1\\0\end{bmatrix} = |0\rangle \otimes \cdots \otimes |0\rangle = |0\ldots0\rangle = |0\rangle^{\otimes n}\ (Tensor\ product\ of\ n\ \ |0\rangle)$$

2.4.4 Ket-bra product

The final item worth discussing in Dirac notation is the ketbra or outer product. The outer product is represented within Dirac notations as $|\psi\rangle\langle\phi|$, and sometimes called ketbras because the bras and kets occur in the opposite order as brakets. The outer product is defined via matrix multiplication as $|\psi\rangle\langle\phi| = \psi\phi^\dagger$ for quantum state vectors ψ and ϕ. The simplest, and arguably most common example of this notation, is :

$$|0\rangle\langle 0| = \begin{bmatrix}1\\0\end{bmatrix}\begin{bmatrix}1 & 0\end{bmatrix} = \begin{bmatrix}1 & 0\\0 & 0\end{bmatrix} \qquad |1\rangle\langle 1| = \begin{bmatrix}0\\1\end{bmatrix}\begin{bmatrix}0 & 1\end{bmatrix} = \begin{bmatrix}0 & 0\\0 & 1\end{bmatrix}$$

2.4.5 Back to the measurement of the amplitude of a state

Previously, we have shown that a qubit was characterized by a quantum state vector of the form

$$\psi = \begin{bmatrix} a \\ b \end{bmatrix} = a\,|0\rangle + b\,|1\rangle$$

Where :

1. $|a|^2$ is the probability (or amplitude) that the vector ψ is reduced to the state $|0\rangle$.
2. $|b|^2$ is the probability (or amplitude) that the vector ψ is reduced to the state $|1\rangle$.

This amplitude is actually characterized by an inner product, so we have for any state ψ_1 :

$$Prob(|\psi_1\rangle\, in\, |\psi\rangle) = |\,\langle\psi_1|\psi\rangle\,|^2$$

is the probability that ψ is reduced to the state ψ_1. Thus, we have :

1. Prob($|0\rangle$ in $|\psi\rangle$) $= |\,\langle 0|\psi\rangle\,|^2 = |a\,\langle 0|0\rangle + b\,\langle 0|1\rangle\,|^2 = |a|^2$
2. Prob($|1\rangle$ in $|\psi\rangle$) $= |\,\langle 1|\psi\rangle\,|^2 = |a\,\langle 1|0\rangle + b\,\langle 1|1\rangle\,|^2 = |b|^2$

3 One-qubit and multi-qubit operations: Logic gates and quantum circuits

3.1 Gates

Logic gates are electronic components that allow basic Boolean logic operations to be performed. They are used in many electronic circuits to perform basic operations such as (AND), (OR) and (NOT).

Each logic gate takes one or more data bits as input and produces an output bit depending on the Boolean logic operation it performs. For example, an AND gate takes two bits as input and produces an output bit that is 1 if both input bits are 1, and 0 otherwise.

Logic gates are used to build more complex logic circuits, such as computational circuits, memory circuits and control circuits. They are also used in many computer applications, including processors and data storage devices.

In quantum computing, logic gates are also used to perform basic Boolean logic operations on qubits (quantum bits). However, quantum logic gates are generally different from classical logic gates, as they have to take into account the quantum nature of the qubits and their interaction.

In this subsection we will cover gates. Due to the number of gates and the similarities between them, we selected a short list of the most important ones. We have included a few digressions to introduce important ideas at appropriate places throughout the chapter

3.1.1 Hadamard Gate

The Hadamard (H-gate) is a fundamental quantum gate which operates on a single qubit. It maps the basis state $|0\rangle$ to $\frac{|0\rangle + |1\rangle}{\sqrt{2}}$ and $|1\rangle$ to $\frac{|0\rangle - |1\rangle}{\sqrt{2}}$.
It allows to move away from the poles of the Bloch sphere and create a superposition of $|0\rangle$ and $|1\rangle$. The Hadamard gate is equivalent to the combination of two rotations : one of 90^{o} rotation around the Z-axis, followed by a 180^{o} rotation around the Y-axis on the Bloch sphere as can be seen in Figure 1.

The Hadamard gate is represented by the Hadamard matrix :

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

This transformation can be written :

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

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

It is this consideration that helps to explain why the specific states $|0\rangle$ *and* $|1\rangle$ explained above are often called a basis for computation: every quantum state can always be expressed as sums of computational basis vectors and these sums are easily expressed using Dirac notation. Thus, we have :

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

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

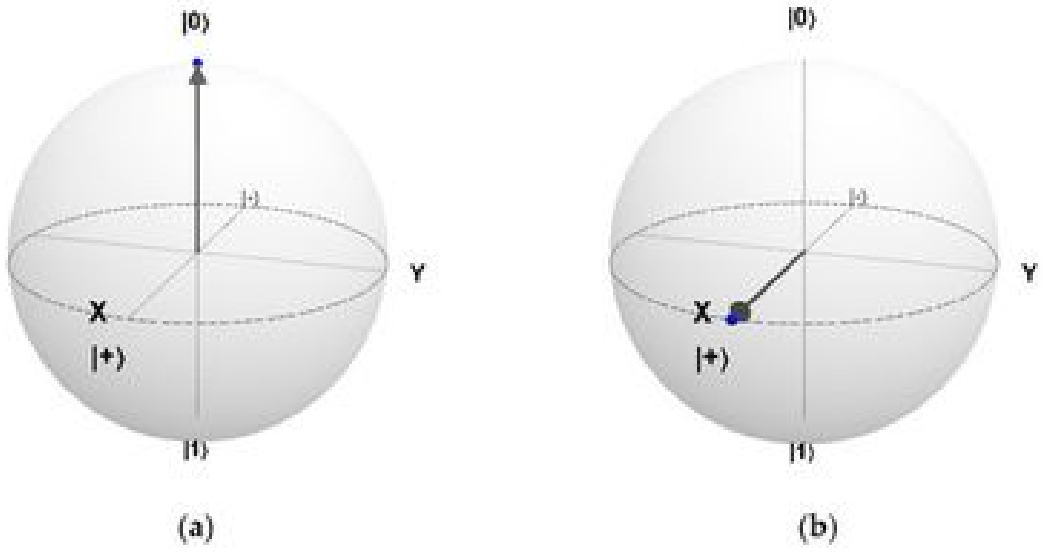


Figure 5: The Block sphere representing a qubit in the state of: (a) $|0\rangle$ and (b) after applying a Hadamard gate to (a).

3.1.2 Pauli Gate

The Pauli gates (X,Y,Z) are the three Pauli matrices ($\sigma_x, \sigma_y, \sigma_z$) and act on a single qubit. The Pauli X, Y and Z equate, respectively, to a rotation around the x, y and z axes of the Bloch sphere by π radians.

3.1.3 X-gate

The X gate is equivalent to a classical NOT operation, or logical negation. The computation basis states are interchanged, so that $|0\rangle$ becomes $|1\rangle$ and $|1\rangle$ becomes $|0\rangle$:

$$X|0\rangle = |1\rangle$$

$$X|1\rangle = |0\rangle$$

It is represented by the Pauli-X matrix :

$$\mathrm{X} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} = |0\rangle\langle 1| + |1\rangle\langle 0|$$

3.1.4 Y-gate

Similarly, the Pauli-Y maps $|0\rangle$ to $i|1\rangle$ and $|1\rangle$ to $-i|0\rangle$:

$$Y|0\rangle = +i|1\rangle$$

$$Y|1\rangle = -i|0\rangle$$

It is represented by the Pauli-Y matrix :

$$\mathrm{Y} = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} = -i|0\rangle\langle 1| + i|1\rangle\langle 0|$$

3.1.5 Z-gate

The Pauli-Z leaves the basis state $|0\rangle$ unchanged and maps $|1\rangle$ to $-|1\rangle$:

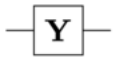
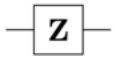
$$Y|0\rangle = +|0\rangle$$

$$Y|1\rangle = -|1\rangle$$

It is represented by the Pauli-Y matrix :

$$\mathrm{Z} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} = |0\rangle\langle 0| - |1\rangle\langle 1|$$

Here are the Single Qubit Gates we just saw above :

Operator	Gate(s)
Pauli-X (X)	
Pauli-Y (Y)	
Pauli-Z (Z)	
Hadamard (H)	

3.1.6 CNOT Gate

CNOT or Conditional NOT is a really important operation which operates on a pair of bits. One bit is designated to be the "control" bit and the other one is the "target" bit. If the control bit value is is $|1\rangle$, the gate performs the NOT operation on the second bit. It means that it flips the value of the target bit. If the control bit is $|0\rangle$ then the target bit is unchanged. The control bit is always unchanged. With respect to the basis $|00\rangle$, $|01\rangle$ $|10\rangle$ $|11\rangle$, it is represented by the Hermitian unitary matrix:

$$\mathrm{CNOT} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

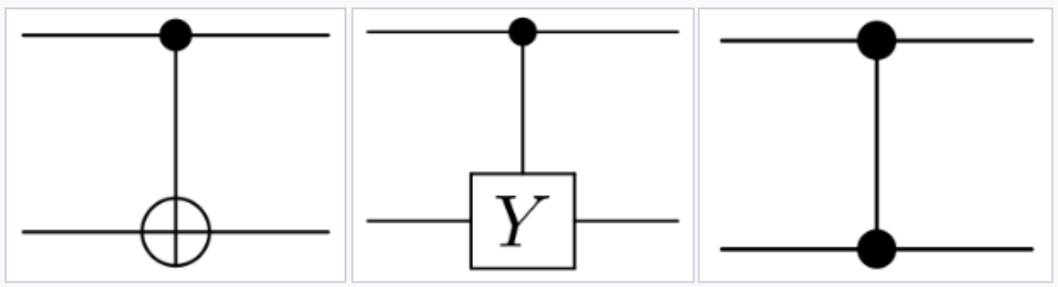


Figure 6: Circuit diagrams of controlled Pauli gates (from left to right): CNOT (or controlled-X), controlled-Y and controlled-Z.

3.1.7 Other Quantum Gates

There are a number of other quantum. Many of them operate on several qubits at a time, leading to matrices with complex-numbered elements. For example we have:

- Toffoli gateFredkin gate
- Deutsch gate
- Swap gate (and swap-gate square root)
- NOT-gate square root

We will not go into more detail because they sometimes require important matrix calculation skills. Any classical computation can be done with a combination of NOT + OR = NOR gates or AND + NOT= NAND gates. The list of quantum gates can be reduced to a simple set of universal quantum gates. As a sum up we have :

Operator	Gate(s)	Matrix
Pauli-X (X)	X	$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$
Pauli-Y (Y)	Y	$\begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$
Pauli-Z (Z)	Z	$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$
Hadamard (H)	H	$\frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$
Phase (S, P)	S	$\begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}$
$\pi/8$ (T)	T	$\begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{bmatrix}$
Controlled Not (CNOT, CX)		$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$
Controlled Z (CZ)	Z	$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}$
SWAP		$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
Toffoli (CCNOT, CCX, TOFF)		$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$

3.2 Quantum Circuits

So far, we have seen various single-qubit and multi-qubit gates. We can use these gates in concert with other components to build quantum circuits. Before implementing quantum algorithms on real quantum computers, it is important to know what a quantum circuit is, as we will be building quantum circuits to implement these algorithms.

3.2.1 What is a Quantum Circuit ?

Quantum circuits are collections of quantum gates interconnected by quantum wires. It is a model for quantum computation in which a computation is an ordered sequence of quantum gates, measurements, initializations and resets. The minimum set of actions that a circuit needs to be able to perform on the qubits to enable quantum computation is known as DiVincenzo's criteria.

3.2.2 Quantum Circuit Diagram Convention

In a circuit diagram, each solid line depicts a qubit, or more generally, a qubit register. By convention, the top line is qubit register 0 and the remainder are labeled sequentially. Circuits are written such that the horizontal axis is time, starting at the left hand side and ending at the right. Horizontal lines are qubits, doubled lines represent classical bits.

4 Quantum Hardware Challenges

Quantum computers available today are called NISQ (Noisy intermediate scale quantum) devices. It is a technology that is developed since many years driven by industry and by academia. Due to its complexity, quantum hardware face challenges to use quantum computers easier without any problem. Significant scientific and engineering efforts are needed to improve and optimize these devices. We identify four main challenges which are noise, connectivity, gate speed and native gates. The technical challenges of NISQ devices have a major impact on the current state of quantum algorithms. The algorithms can be split into two parts: (1) near-term algorithms designed to run on NISQ devices and (2) algorithms that have a theoretically proven advantage for when hardware advances enough but require many qubits with extremely high-fidelity quantum gates. One should keep in mind that arguably none of the discussed algorithms implemented on NISQ devices provide a decisive advantage over classical algorithms yet.

4.1 Noise

Noise describes all the things that can cause a quantum computer to disfunction. A quantum computer is susceptible to noise from different sources like electromagnetic signals from wi-fi or mobile phones, cosmic rays or disturbances in the earth's magnetic field. Due to the noise, qubits lose their desired quantum state or decoherence over the time. That means the information of qubits could be randomized or even erased which is clearly bad when you compute. Quantum processors are very sensitive to noise compared to classical computers. For example, now we can only perform dozens of operations before noise causes a fatal error. To avoid this problem, theorists created an algorithm called quantum error correction (QEC) than can identify and fixe errors in the hardware. However, to implement this algorithm you have to spread the information of one qubit over lots of qubits which is not optimal. Another way to reduce decoherence is to use a quantum firmware working with dynamic stabilization. You need to constantly rotate qubits in just the right way to immune them from noise. This allows to run more algorithms before errors arise and reduce the number of qubits needed to use quantum error correction.

4.2 Connectivity

Quantum circuits must be optimally mapped to the topology of a quantum device to minimize errors and total run time. Actually, qubits can communicate only with neighbouring subsets of other qubits. The objective is to create a network that can connect quantum computers over large distances to use it like internet nowadays. For existing superconducting processors, the positioning of the qubits cannot

change. As a result, two-qubit quantum operations between remote qubits must be mediated by a chain of additional two-qubit SWAP gates via the qubits connecting them. Moreover, the two-qubit gate error of current NISQ devices is high. Therefore, quantum circuit optimizers and quantum hardware must be developed with these limitations in mind.

4.3 Gate Speed

Gate speed also need to be considered because we need fast gates to achieve quantum advantage with NISQ devices in the quantum circuit model. Nevertheless, some quantum computers are particularly slow such as trapped-ion quantum processors but they have lower gate error rates. Speed is crucial especially when you use algorithms that require many circuits executions repetitions. Moreover, error rates increase clearly if the gate time is reduced below a certain duration. Thus, we need to find the right balance between space, speed and fidelity, which often requires calibrations and fine-tuning.

4.4 Native Gates

The last challenge concerns the native gates. The native gate set is the set of quantum gates that are physically executed on IonQ hardware by addressing ions with resonant lasers via stimulated Raman transitions. We currently expose two single-qubit gates and one two-qubit entangling gate. The existence of a universal set of native gates is crucial to design short-depth high-fidelity quantum circuits. Therefore, it is important to Develop advanced quantum compilers for efficiently mapping general quantum gates to the native gates of a particular device.

5 Fundamental algorithms

5.1 Phase Estimation Algorithm

Every quantum algorithm consists of three major steps: initialization, applying some sequence, and then readout of the result. Although this final step appears to be straightforward, it is not always trivial how to read out the results of all the qubits in your algorithm.

The concept of quantum phase kickback, is a powerful tool in extracting information out of your system. Relative phases and global phases, are essential building blocks for the quantum phase kickback to work and then output the result of the quantum algorithm. Quantum phase estimation is an algorithm to estimate these relative phases. The phase can be considered as an eigenvalue of an eigenvector of a unitary operator. More precisely, given a unitary matrix and a quantum state $|\psi\rangle$ such that $U|\psi\rangle = e^{2\pi i\theta}|\psi\rangle$, the algorithm estimates the value of the phase θ with high probability within additive error ε, using $O(\log(1/\varepsilon))$ qubits and $O(1/\varepsilon)$ controlled-U operations. A more detailed explanation of the algorithm and what happens when the eigenvalue phases cannot be perfectly represented with the specified precision, ε, is provided by Nielsen and Chuang.

5.2 Variational Quantum Algorithms

The Variational Quantum Algorithms (VQAs) is based on the variational method which consists in finding low energy states of a quantum system. The idea of this method is that one defines a trial wave function (sometimes called an ansatz) depending on one or more parameters, and then ones finds the values of these parameters for which the expectation value of the energy is the lowest possible. The wavefunction obtained by fixing the parameters to such values is then an approximation to the ground state wavefunction, and the expected value of the energy in that state is an upper bound to the ground state energy. In the last few years, it has been realized that quantum computers can reproduce this technique with certain advantages. In particular, when we apply the classical variational method to a system of n qubits, an exponential number of complex numbers is necessary to generically represent the wave function of the system. However, with a quantum computer, we can directly produce this state using a parameterized quantum parameterized quantum circuit with less than exponential parameters, and then use repeated measurements to estimate the expectaion value of the energy. Of course this approach is not just limited to finding low energy eigenstates, but minimizing any objective function that can be expressed as a quantum observable. It is applicable to optimization problems, and both linear and nonlinear problems. Furthermore, problems using VQAs are seen as one of the

leading potential applications of near-term quantum computing. So, VQAs have now been proposed for essentially all applications that researchers have envisioned for quantum computers, and they appear to the best hope for obtaining quantum advantage. Nevertheless, challenges remain including the trainability, accuracy, and efficiency of VQAs.

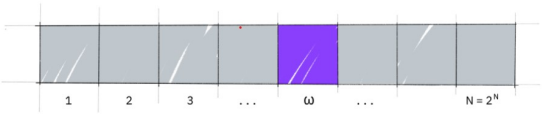
5.3 Grover's Algorithm (Quantum Unstructured Search)

Grover's algorithm is a quantum algorithm for unstructured search. Over a classical computer, this algorithm can speed up an unstructured search problem quadratically. Those problems in classical computation cannot be solved in fewer than $O(N)$ evaluations, over $O(\sqrt{N})$ with Grover's algorithm. Beyond that, it can be used to obtain quadratic run time improvements for a variety of other algorithms. The algorithm takes as input an oracle, which determines whether an element satisfies the search criterion. The oracle is defined by a function $f : \{0, 1, ..., N-1\} \rightarrow \{0, 1\}$ which value is 1 when x satisfies the search criterion. Using the oracle, the algorithm applies an amplification operator that amplifies the target state and decreases the other states. The state can then be measured to obtain the target state with near certainty. Grover's algorithm can be used to invert functions. From a function $y = f(x)$, it allows to find x when given y. Grover algorithm can also be used for classical search problems such as if we assume we want to solve $x^2 - 1 = 0$.

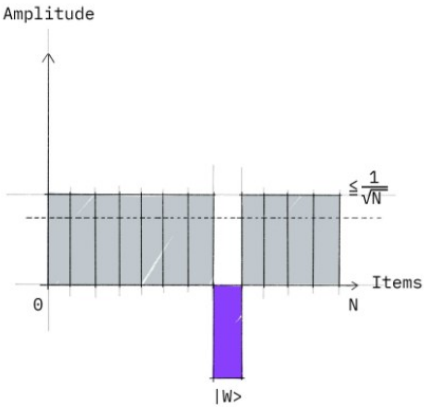
$$f : [1, N] \rightarrow \{0, 1\} \begin{cases} f(x) = 1 \; if \; \mathrm{x}^2 = 1 \\ f(x) = 0 \; if \; \mathrm{x}^2 \neq 1 \end{cases}$$

5.3.1 Intuition behind

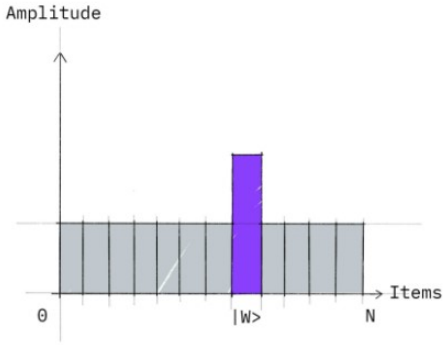
Suppose you are given a large list of N items. Among these items there is one item with a unique property that we wish to locate; we will call this one the winner w. Think of each item in the list as a box of a particular color. Say all items in the list are gray except the winner w, which is purple.



If we work with a classical algorithmic structure, the idea would be to scan and test element by element each quantum state coefficient and to identify the one we are looking for. In terms of time complexity, this is one of the points on which grover intervenes insofar as in this algorithm, all states are tested simultaneously. A global operation is applied and the quantum state coefficient searched changes sign. The average amplitude is then calculated. This is represented graphically by the horizontal dotted lines.



Finally, we apply a symmetry with respect to the mean. Thus, the states slightly above the average are weakened while the sought coefficient (whose sign has been changed) is strongly amplified.



By doing so, as it was explained that the state measurement is based on a probabilistic model, we amplify the probability that the state vector is reduced to a specific state.

5.3.2 In practice

Considering the previous explanations, the oracle of the grover algorithm is defined as follows :

$$U_w |x\rangle = \begin{cases} |x\rangle \ \ if \ \ |x\rangle \neq w \\ -|x\rangle \ \ if \ \ |x\rangle = w \end{cases}$$

the oracle corresponds to a diagonal matrix which lists the different possible state vectors of the quantum basis (for example, $\{000, ..., 111\}$ $ie:$ $\{1, .., 7\}$). Then, if we consider for example that w = 111, we have :

$$U_w = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \end{bmatrix}$$

Thus, the oracle can be expressed as a function of f (it's general definition) such that :

$$U_w = \begin{bmatrix} (-1)^{f(0)} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & (-1)^{f(1)} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & (-1)^{f(2)} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & (-1)^{f(3)} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & (-1)^{f(4)} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & (-1)^{f(5)} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & (-1)^{f(6)} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & (-1)^{f(7)} \end{bmatrix}$$

And more generally :

$$U_w = \begin{bmatrix} (-1)^{f(0)} & 0 & 0 & 0 \\ 0 & (-1)^{f(1)} & 0 & 0 \\ 0 & 0 & ... & 0 \\ 0 & 0 & 0 & (-1)^{f(2^n-1)} \end{bmatrix}$$

In summary, here are the steps of the amplification of amplitude:

1. Step 1 : The amplitude amplification procedure starts out in the uniform superposition $|state\rangle$, which is easily constructed from $|state\rangle = H^{\otimes n} |0\rangle^n$. This is the simplest initialization since the Hadamard transform generates a superposition state with an identical amplitude for each quantum state.

2. Step 2 : We apply the oracle reflection U_f to the state $|s\rangle$: $|NewState\rangle = U_f |state\rangle$. (Note: we note this time U_f because the state to be amplified is determined by the function f).

3. Step 3: We now apply an additional reflection U_s about the state $|state\rangle$: $U_s = 2|s\rangle\langle s| - I$. This transformation maps the state to $U_s U_f |state\rangle$ and completes the transformation.

5.4 Quantum Amplitude Estimation

Quantum amplitude estimation is a quantum algorithm that allows one to estimate the probability amplitudes of a quantum state. It is a generalization of classical algorithms for estimating probabilities, such as the Monte Carlo method, to the quantum setting. To understand how quantum amplitude estimation works, we have to remember that the square of the absolute value of the probability amplitude of a particular state is the probability that the system will be found in that state when it is measured.

In quantum computing, a quantum computer is used to perform calculations on these probability amplitudes. One way to do this is to use the quantum Fourier transform, which is a quantum algorithm that allows one to perform a Fourier transform on a quantum state. By using the quantum Fourier transform and other quantum operations, it is possible to perform a variety of calculations on the probability amplitudes of a quantum state.

Quantum amplitude estimation is a quantum algorithm that allows one to estimate the probability amplitudes of a quantum state by repeatedly applying the quantum Fourier transform and other quantum operations. It works by first preparing a quantum state that is an equal superposition of all possible states, and then applying the quantum Fourier transform to this state. This allows one to obtain a probability distribution over the possible states, which can then be used to estimate the probability amplitudes of the original quantum state.

Overall, quantum amplitude estimation is a powerful tool for estimating the probability amplitudes of quantum states, and has a variety of applications in quantum computing and other areas of physics and computer science.

5.5 Quantum Amplitude Amplification

Foreword: An NP-complete problem is a type of decision problem for which there is a proof that an algorithm does not exist that would solve the problem in polynomial time (i.e., in a number of computational steps that grows polynomially with the size of the input).
NP-complete problems are considered particularly difficult to solve, as they can only be solved efficiently on a quantum computer or using approximate solution algorithms.

Quantum Amplitude Amplification is a quantum computational technique that allows to highlight quantum states with a probability higher than the one they have in nature. It was developed by Gilles Brassard and Peter Høyer in 1996.

The technique is based on the use of a Grover operator, which allows to "mix" the states of a quantum system so that the desired states have a higher probability of being measured. This makes it possible to solve certain solution search problems in fewer computational steps than would be necessary with a classical algorithm.

Quantum Amplitude Amplification is used in a variety of applications, including finding solutions to NP-complete problems, such as the Boolean satisfiability problem, and factoring integers into products of primes, which is important for the security of encryption systems.
Amplitude amplification is a technique in quantum computing which is a generalization of Grover's algorithm. In a quantum computer, amplitude amplification can be used to obtain a quadratic speedup over several classical algorithms.
The Grover algorithm uses Hadamard gates to create the uniform superposition of all states at the

beginning of the Grover operator Q. If some information about the states being searched for is available, it may be useful not to start in a uniform superposition but to initialize only specific states. This generalized version of the Grover algorithm is called *Amplitude Amplification*. Amplitude amplification allows to measure specific states with a high probability.

5.6 Quantum Linear System Algorithms

The quantum linear systems problem (QLSP) is defined as follows. Given an invertible Hermitian matrix $A \in \mathbb{R}^{N \times N}$ and a vector $\vec{b} \in \mathbb{R}^N$, find the solution vector $\vec{x}$ satisfying $A\vec{x} = \vec{b}$.
The Harrow–Hassidim–Lloyd (HHL) algorithm is an algorithm solving the QLSP. It provides an exponential speedup in the system size N for well-conditioned matrices, over all known classical algorithms for a simulation of this problem. Several conditions must be satisfied to make the algorithm work: the given linear system must have a low condition number κ, and the matrix A must be s-sparse and Hermitian. It should also be taken in account that while the classical algorithm returns the full solution, the HHL algorithm can only approximate functions of the solution vector.
Due to the ubiquity of linear systems in virtually all areas of science and engineering, the quantum algorithm for systems of linear equations has the potential to be widely applied and could be used for the calibration of financial models or in machine learning.

6 Machine Learning

6.1 Classification

More and more data is used and must be managed on a daily basis. That's why it is more and more important to find innovative methods of machine learning. Supervised machine learning algorithms infer an input-output relation from large sets of training data that consist of "correct" examples of mappings. In other words, the computer learns from experience how to treat new inputs.
Classification is the process of placing objects into predefined groups. This area of machine learning can be used effectively in risk management and large data processing. In Quantum computing, classification is a task where a quantum algorithm is used to determine the class or category of a given input. The input could be a quantum state, a set of classical data, or a combination of both.

6.1.1 Quantum Support Vector Machine

Support Vector Machine (SVM) is one of the most efficient machine learning algorithms, which is mostly used for pattern recognition. Indeed, it is a supervised learning model used to analyse data for classification and regression analysis. Here, we will show that the support vector machine can be implemented on a quantum computer, with complexity logarithmic in the size of the vectors and the number of training examples whereas classical sampling algorithms require polynomial time. Given a set of points $\{\overrightarrow{x}\}$ that are in y={-1,1}, The task for the SVM is to classify a vector into one of two classes. So, the objective here is to find a maximum margin hyperplane with normal vector that separates the two classes.
The margin is defined as the distance between the hyperplane and the closest data points from each class, also known as support vectors. The maximum margin hyperplane is the one that maximizes this distance, meaning that it is the one that is the farthest away from any data point. This line can be linear, but it can also be much more complex, that's why we need the concept of kernel to overcome these difficulties.
A kernel is a mathematical tool used in classification methods to separate data that cannot be separated by a straight line or hyperplane in its original space. This is done by using a non-linear function called a feature map that changes the data into a new space where it can be separated. The classification is then done based on how close the data points are to each other in this new space. Instead of computing the feature map for each data point, only the inner product of each pair of data points in the new space is calculated. This inner product is called the kernel. The advantage of using a kernel is that it can make it possible to classify data even if the feature map is difficult to compute.

Let's go back to the goal of the SVM which is to find the maximum margin hyperplane. To find

the maximum margin hyperplane, we use a mathematical function called the Lagrangian dual formulation which maximizes the function

$$L(\alpha) = \sum_{j=1}^{M} y_j \alpha_j - \frac{1}{2} \sum_{j,k=1}^{M} \alpha_j K_{jk} \alpha_k$$

where $\overrightarrow{\alpha} = (\alpha_1, ..., \alpha_2)^T$, subject to the constraints $\sum_{j=1}^{M} \alpha_j = 0$ and $y_j \alpha_j \geq 0$. The kernel matrix $K_{j,k} = k(\overrightarrow{x_j}, \overrightarrow{x_k}) = \overrightarrow{x_j} \cdot \overrightarrow{x_k}$ is used to evaluate the dot products of the data points in the new feature space, this is done to find the optimal parameters α_j. This process is called quadratic programming and takes O(M3) in the non-sparse case.
A quantum SVM was first proposed by Rebentrost and it showed that a quantum SVM can be implemented with O(log MN) runtime in both training and classification stages. This is done by assuming that there are oracles for the training data that return quantum states, the norms and the labels of the data are given, and the states are constructed by using a specific technology called qRAM. One of the main challenges is the need for quantum memory (qRAM) to store and access the training data in a quantum state. This technology is still in development, and it's not yet clear if it will be feasible to implement in practice. The core of this quantum classification algorithm is a technique called non-sparse matrix exponentiation which is used to efficiently perform a matrix inversion of the training data inner-product (kernel) matrix. This technique helps to speed up the process of finding the optimal hyperplane and classifying the data. In addition, classical SVMs using quantum enhanced feature spaces to construct quantum kernels have been proposed by Havlíček.
One benefit of quantum kernels is that there exist metrics for testing for potential quantum advantage. Additionally, there have been techniques proposed to enable generalization in quantum kernel methods. Generalization means that a model can perform well on unseen data, and it has been observed to be an issue with standard quantum-kernel methods. These proposed techniques aim to overcome this issue and make quantum kernels more reliable and useful in practice.

6.1.2 Quantum Nearest-Neighbors Algorithm

The K-nearest neighbors (KNN) is a type of machine learning algorithm that is used for classification and regression tasks. The basic idea behind KNN is to determine the class of a given input by looking at the class labels of the k-nearest training examples.
In quantum computing, there have been proposals for implementing KNN using quantum algorithms. The main advantage of using a quantum algorithm for KNN is that it can potentially provide exponential speedup over classical algorithms for large datasets. This is because quantum algorithms can perform many calculations in parallel, and they can also perform certain operations, such as searching and comparing, more efficiently than classical algorithms.
However, there are also challenges and difficulties in implementing KNN using quantum computing. One of the main challenges is dealing with the high-dimensional feature space. In classical KNN, the feature space is usually low-dimensional, but in quantum KNN, the feature space is high-dimensional. This means that the quantum algorithm needs to be able to handle high-dimensional data, which can be difficult. Another challenge is the measurement of the quantum state. In classical KNN, the distance between the input and the training examples is calculated using a distance metric. In quantum KNN, the distance is calculated using the inner product of the quantum states, which can be difficult to measure.
The quantum nearest-neighbor classification algorithm was proposed by Wiebe, who used the Euclidean distance and the inner product as distance metrics. The distance between two points is encoded in the amplitude of a quantum state. They used amplitude amplification to amplify the probability of creating a state to store the distance estimate in a register, without measuring it.

6.1.3 Benefits of quantum classification

Quantum machine learning classification can be useful in quantitative finance by offering the following benefits:

1. Fraud detection: Quantum algorithms can quickly analyze large amounts of financial data and detect anomalies, potentially uncovering fraudulent activity.

2. Risk management: Quantum algorithms can classify financial data into different risk categories, allowing financial institutions to manage risk more effectively.

3. Trading strategy: Quantum algorithms can analyze market data and make predictions about future market trends, enabling more effective trading strategies.

4. Customer segmentation: Quantum algorithms can classify customers into different segments based on financial behavior, allowing for more targeted marketing efforts.

5. Improved accuracy: Quantum algorithms can lead to more accurate predictions and decisions in finance, as they can handle complex relationships in data than classical algorithms.

6. Real-time analysis: Quantum algorithms can process financial data in real-time, enabling faster decision-making in fast-paced financial markets.

6.2 Clustering

Clustering, or cluster analysis, is an unsupervised machine learning task. It explores and discovers the grouping structure of the data. In finance, cluster analysis can be used to develop a trading approach that helps investors build a diversified portfolio. It can also be used to analyze different stocks such that the stocks with high correlations in returns fall into one basket.

6.2.1 Quantum k-means clustering

The K-means clustering algorithm creates clusters of training data based on the distances of each data item from centroids initially chosen randomly and then optimized at each iteration. K-Means works as follows:

1. Choose initial values for the k centroids randomly or by some chosen method,

2. Assign each vector to the closest centroid,

3. Recompute the centroids for each cluster.

We then repeat these steps to refine the position of the centroids at each iteration until we obtain the best possible clusters. Determining the optimal clusters for the data is an NP-hard problem. As a result, the search for a solution can be computationally very expensive. Quantum computing can be leveraged to accelerate a single step of the k-means. At each iteration, if we consider M the number of data points and N the dimension of the vector, the algorithm estimates the distance to the centroids in $O(Mlog(MN))$, while classical algorithms take $\mathrm{O}(M^2N)$. With the same technique used for the quantum nearest-neighbor algorithm described the previous part, a step for k–means can be performed by using a number of queries that scales as $\mathrm{O}(M\sqrt{k}log(k)/\epsilon)$. While a direct classical method requires O(kMN), the quantum solution is substantially better if kN $\gg$ M. Q-means, a new quantum algorithm for clustering is a quantum version of a robust k-means algorithm, with similar convergence and precision guarantees. The q-means algorithm has a running time that depends polylogarithmically on the number of data points.

6.2.2 Benefits of quantum clustering

When considering an advance, it is necessary to be able to situate the usefulness of the latter. Moreover, the question that inevitably arises is the difference in terms of performance. Thus, the advantages of quantum clustering over its predecessor for classical computing are :

1. Scalability: Quantum clustering can handle larger data sets than classical clustering algorithms.

2. Speed: Quantum algorithms can be faster than classical algorithms for some tasks.

3. High-dimensional data: Quantum clustering is better suited for high-dimensional data than classical clustering.

4. Robustness: Quantum algorithms are less sensitive to noise and outliers than classical algorithms.

5. Non-linearity: Quantum clustering can find clusters in non-linearly separated data.

6. Global optimization: Quantum algorithms can find the global optimum solution, rather than getting stuck in a local optimum as classical algorithms often do.

Finally, beyond the theoretical aspect of machine learning, the usefulness of this new quantum technology is particularly evident in the fields of quantitative finance, such as :

1. Portfolio optimization: Quantum algorithms can optimize portfolios more efficiently, taking into account complex relationships between assets.

2. Risk management: Quantum clustering can help identify correlations and dependencies in financial data, aiding in risk management.

3. Financial forecasting: Quantum algorithms can analyze large amounts of financial data to make predictions about market trends and asset prices.

4. High-frequency trading: Quantum algorithms can be used to process vast amounts of financial data in real-time, enabling more effective high-frequency trading.

5. Anomaly detection: Quantum algorithms can detect unusual patterns and anomalies in financial data, potentially uncovering fraudulent activity.

6. Improved accuracy: Quantum algorithms can lead to more accurate predictions and decisions in finance, as they can handle more complex relationships in data than classical algorithms.

6.3 Quantum Neural Networks

Quantum neural networks (QNNs) are a variant of neural networks that are developed using quantum computing. They are a cutting-edge area of artificial intelligence research that aims to harness the power of quantum mechanics to create more powerful machine learning models. Researchers have developed different approaches to QNNs, including quantum feedforward neural networks, quantum convolutional neural networks, and quantum graph neural networks. These approaches involve initializing a network architecture, specifying a learning task, implementing a training algorithm, and simulating the learning task. These QNNs have varying degrees of proximity to classical neural networks, but they have the potential to speed up operations used by classical neural networks and to have a quantum-enhanced feature map.
QNNs are typically developed as feed-forward networks, similar to classical neural networks, which intake input from one layer of qubits, evaluate the information, and pass the output to the next layer. The layers do not have to be of the same width, meaning they don't have to have the same number of qubits as the layer before or after it. The structure is trained on which path to take, similar to classical artificial neural networks. For example, in the field of machine learning, QNNs can perform certain types of pattern recognition and data classification tasks exponentially faster than CNNs. They are also believed to be very useful in solving optimization problems, especially when the classical algorithms are computationally expensive. QNNs are typically developed as feed-forward networks, similar to classical neural networks, which intake input from one layer of qubits, evaluate the information, and pass the output to the next layer. The layers do not have to be of the same width, meaning they don't have to have the same number of qubits as the layer before or after it. The structure is trained on which path to take, similar to classical artificial neural networks.
There are three main categories of QNNs: quantum computers with classical data, classical computers with quantum data, and quantum computers with quantum data. Examples of QNNs include quantum perceptrons, which attempt to find a quantum equivalent for the perceptron unit from which neural nets are constructed, and quantum networks, which attempt to generalize neural networks to the quantum setting by using unitary gates to control interactions between neurons. Current proposals for QNNs are being developed on near-term devices using parameterized quantum circuits, and they are being analysed for their expressive power and potential benefits when used in conjunction with classical neural networks.
As of today, the implementation of these quantum neural networks can be done as variational quantum algorithms (VQAs). They work in a hybrid way where the quantum algorithms are executed

on a quantum computer, but the optimisation process is performed on a classical computer. A benefit of using variational quantum algorithms is that they can be successfully executed on noisy and intermediate-scale quantum computers. Using these algorithms limits the number of quantum gates within one quantum circuit which prevents noise entering with each quantum gate from altering the performance of the algorithm. Quantum neural networks do not require too many quantum gates and can therefore be run on early quantum computers.
In conclusion, QNNs are a promising new area of artificial intelligence research that has the potential to revolutionize the field of machine learning. However, it is important to note that QNNs are still in the early stages of development, and it will take time to fully understand their capabilities and limitations. Although they can be trained on early quantum computers and noisy devices, the execution of these quantum algorithms remains challenging, due to the limited number of qubits in today's computers and by the high noise levels that can lead to limits.

6.4 Natural Language Processing

Natural language processing (NLP) refers to the branch of artificial intelligence that gives computers the ability to understand text and spoken words in the same way as humans. It focuses on the understanding, manipulation and generation of natural language by machines. Thus, NLP is really at the interface between computer science and linguistics. NLP plays a growing role in enterprise solutions that help streamline business operations, increase employee productivity, and simplify mission-critical business processes. Designing quantum algorithms for NLP tasks is a sub-field of quantum computing now called quantum natural language processing (QNLP). It started with the Distributional Compositional Category (DisCoCat) model : a model that combines embedded words along the grammatical structure of a sentence to encode its meaning. The algorithm which is proposed by the authors to implement this model has an exponential speed-up when implemented on a quantum computer. For the moment, the area of QNLP is still mainly at stage of proofs of concepts. There is a need to scale to larger problems and to apply these methods to real industry cases. Generally, QNLP faces the same limitations as the rest of the quantum computing field: implementation of quantum memory doesn't exist, limitation in the number of qubits and absence of fault tolerant universal quantum machine.

7 Conclusion

Quantum computing is an evolving field that has the potential to dramatically change the way we work with data. Quantum computers use quantum properties such as superposition and entanglement to process data significantly faster and more efficiently than classical computers. This enables applications such as complex system simulation, pattern recognition, secure cryptography, and finding optimal solutions for complex problems.

However, there are also significant challenges in the adoption of quantum computing. First, quantum computers require highly controlled environmental conditions, such as very low temperatures and minimal noise, to operate reliably. In addition, quantum algorithms are often very different from those used on classical computers, which can make it difficult to migrate existing applications to the quantum environment.

Ultimately, quantum computing is a very promising field that offers significant opportunities to improve computer performance and data security. However, it is still in development and requires significant investment in research and development to realize its full potential. It is therefore important to monitor advances in this field and understand the opportunities and challenges.

Regarding quantitative finance, quantum computing can be used to improve financial simulation models and automated trading algorithms. Quantum computers can process massive amounts of financial data in real time, which can improve the accuracy of price predictions and trading decisions. In addition, the use of quantum computing can enable better financial risk management by simulating many potential scenarios and assessing the consequences for portfolios.

Nevertheless, there are also significant challenges in using quantum computing in finance. First, quantitative models can be very complex and difficult to understand, which can make informed decision making difficult. In addition, the use of quantum computing can lead to market fragmentation, as players who have access to this technology may have a competitive advantage over those who do not.

At the end of the day, quantum computing can play an important role in improving the accuracy and efficiency of financial decisions. However, it is important to understand the opportunities and challenges associated with its use and to ensure that this technology is used ethically and responsibly.