POLITECNICO DI TORINO

Master's Degree in Management Engineering

Master's Degree Thesis

Quantum Amplitude Estimation Methods for Option Pricing

Supervisors

Prof. Bartolomeo MONTRUCCHIO

Dr. Edoardo GIUSTO

Dr. Emanuele DRI

Candidate

Emanuela ALLOCCA

April 2024

Summary

Quantum Computing (QC) could enable a faster resolution for certain classes of problems compared to classical computers (or even supercomputers) while consuming potentially less power. Quantum computers are particularly powerful in situations where the simultaneous analysis of different combinations is required.

Specifically, in the case of Option Pricing, quantum computers could allow for the replacement of traditional models like Monte Carlo Simulation, delivering a theoretical quadratic speedup, by leveraging the Quantum Amplitude Estimation (QAE) algorithm. This thesis aims to analyze the characteristics of different proposed methods for implementing QAE and determine which one best suits the analyzed problem.

The experiments described in this work were conducted on simple call options as a proof of concept; however, the true improvement stemming from the use of QC for option pricing lies primarily in cases involving multiple underlying assets or more complex structures for payoff calculation. Employing this method, as opposed to traditional ones, could potentially enable better risk management and investment decision-making. Nevertheless, testing quantum algorithms on more complex option pricing problems proves challenging, if not in some cases even impossible, with current technology. This is largely due to the necessity of manipulating intricate quantum states, demanding a high degree of qubit control and coherence. For instance, sophisticated quantum algorithms like Grover's, Shor's, and quantum simulation algorithms entail a significant number of qubits and quantum operations beyond the capabilities of available quantum processing units.

The current state of the art of QC is predominantly experimental, with several obstacles to actual useful applications. One major challenge to widespread quantum computing adoption is the qubit's fragile state. Existing technologies can only maintain the information in a quantum state for short periods of time, limiting the duration of calculations in practice. For this reason, errors and decoherence are inherent phenomena in present-day quantum computers. Despite the development of initial proposals for error correction techniques, error persistence still makes executing complex algorithms on a sufficient number of qubits challenging, as errors tend to dominate results.

This is why existing quantum computers are not capable of performing complex calculations at scale without producing errors and noise that can affect the quality of the results obtained. This phenomenon reflects the evolution of quantum computers and the era in which we find ourselves, defining such devices as "Noisy Intermediate-Scale Quantum" (NISQ). This is one of the reasons why the experiments conducted for this work, were executed on a simulator rather than on real quantum hardware, as even though it involves a simple call option, the calculations involved require a higher number of logical operations compared to those that quantum computers available to the general public can perform successfully. Accessing a quantum computer capable of performing complex operations is prohibitively expensive, rendering it unattainable for individuals lacking the necessary resources.

Moreover, on real machines, noise would overshadow any tangible results, stemming from errors rather than concrete outcomes. As algorithm complexity increases, error correction and distinction from final results become increasingly difficult. In general, achieving a scalable and reliable quantum computer is a formidable task, necessitating a broad array of multidisciplinary expertise: physics, engineering, computer science, and quantum theory. Consequently, resource limitations arise, with publicly available quantum computing platforms having significant constraints in terms of qubit count and available execution time, thus restricting the size of problems that can be tackled and the complexity of algorithms that can be tested.

Despite these challenges, the aim of this work is to analyze how we could replace Monte Carlo simulation with the QAE method, which is capable of performing the same task but necessitating quadratically fewer samples and to accomplish this, firstly we introduce the foundational concepts of quantum computing in Chapter 1. Then we describe those for option pricing in Chapter 2, followed by a brief presentation of the current state of the art in the pricing of options using quantum computers (Chapter 3), before delving into the experiments underlying this work in Chapter 4.

For these experiments, we considered different implementation variants of QAE and, for our analysis, three of the most promising have been chosen: *iterative*, *maximum likelihood*, and *faster* amplitude estimation. These methods exhibit different characteristics not only in the procedure by which they are implemented and the results they provide but also in the parameters provided to them to solve the problem. The objective is to estimate, through an analysis of various parameters, which of the three methods is more suitable to solve the problem related to option pricing. In doing so, various elements have been taken into consideration, including the accuracy of the result compared to the one obtained with the classical Monte Carlo simulation method. Additionally, computational complexity has been assessed by analyzing the number of qubits and the depth of the circuits used to solve the problem. Given these parameters, it becomes evident why it is necessary for the number of qubits and circuit depth to be as minimal

as possible: both due to the potential for errors and noise that can distort the obtained results and in terms of scalability, to enable the future application of these methods to more complex operations.

Taking into consideration these factors, Chapter 5 contains the final remarks. The experiments conducted show that, for our use case, despite maximum likelihood and faster methods resulting in higher accuracy in estimating the payoff and delta of an option, they require more gates compared to the iterative method. The latter, despite its lower accuracy, emerges as the method capable of resolving the posed problems with shallower depth. This is particularly crucial in terms of scalability, as in the event of using this method to address more complex issues such as path dependency, the model would be better at maintaining its performance, reliability, and functionality. As technological development progresses to a stage where the utilization of resources becomes secondary to method efficiency, this analysis of the problem will change, with maximum likelihood and faster methods seen as superior, capable of yielding the same results as those obtained with MC simulation but at significantly faster speeds.

Acknowledgements

To those who believed in from the beginning
and until the end

Table of Contents

List of Tables

List of Figures

Acronyms

AE
Amplitude Estimation

ANN
Artificial Neural Network

ASRF
Asymptotic Single Risk Factor

BSM
Black-Scholes-Merton

CDF
Cumulative Distribution Function

DAG
Directed Acyclic Graph

EAD
Exposure At Default

EPR
Einstein-Podolsky-Rosen

ES
Expected Shortfall

EV
Estimated Value

FAE

Faster Amplitude Estimation

IAE

Iterative Amplitude Estimation

IQAE

Iterative Quantum Amplitude Estimation

IRB

Internal-Rated Based

LGD

Loss Given Default

MC

Monte Carlo

ML

Maximum Likelihood

MLAE

Maximum Likelihood Amplitude Estimation

MLQAE

Maximum Likelihood Quantum Amplitude Estimation

NISQ

Noisy Intermediate-Scale Quantum

NPV

Net Present Value

PDF

Probability Density Function

QAA

Quantum Amplitude Amplification

QAE

Quantum Amplitude Estimation

QAEM

Quantum Amplitude Estimation Methods

QC

Quantum Computing

QFT

Quantum Fourier Transform

QGAN

Quantum Generative Adversal Network

WHP

With High Probability

Chapter 1

Quantum Computing

Traditional computers operate in a binary manner, leveraging bits that can hold values of 0 or 1. In contrast, quantum computing draws upon the principles of quantum mechanics [1] and introduces qubits as the fundamental units for manipulation.

Qubits can represent not only the classical states of 0 or 1 but also a combination of these states, a phenomenon known as superposition. Additionally, entanglement, a concept introduced by Erwin Schrödinger in 1935 within the framework of quantum mechanics, describes a profound interconnectedness between particles. This correlation, characterized by the mathematical construct of the wave function, allows particles to influence each other instantaneously despite physical separation. See Section 1.2 and 1.3 for more details on these phenomena.

Superposition and entanglement are key properties that enable quantum computers to process a higher volume of information than classical computers, potentially leading to increased speed and reduced energy consumption, thus promising superior performance.

Quantum computers excel particularly in scenarios requiring simultaneous analysis of multiple combinations. However, this characteristic does not inherently establish their supremacy over classical computers, as classical computers may outperform quantum ones in certain computations.

Qubits can occur naturally or be engineered, with common types including spin qubits, trapped atoms and ions qubits, photons, and qubits based on superconducting circuits.

Another critical consideration is the extreme sensitivity of quantum computers to noise. Factors such as electromagnetic fields, heat, and collisions with air molecules can cause qubits to lose their quantum properties, a phenomenon known as quantum decoherence. This effect amplifies with an increase in the number of particles involved. Consequently, quantum computers must ensure physical isolation of qubits from external interference, maintaining them at nearly absolute zero

temperatures or subjecting them to carefully controlled energy pulses. Additionally, error correction strategies, such as redundancy, may be necessary to address system errors.

1.1 Qubits

The qubit is the basic unit of the quantum computation. When the measurement has not already been executed, the qubit can be in a linear combination of the states 0 and 1, thanks to the quantum mechanical phenomenon of superposition (see Section 1.2). The geometric representation of the two possible states is defined by a vector, called *state vector*, inside Hilbert's space.
The state vector can be represented as follows:

$$|q\rangle = \alpha\,|0\rangle + \beta\,|1\rangle \tag{1.1}$$

or

$$|q\rangle = \begin{bmatrix}\alpha\\ \beta\end{bmatrix} \tag{1.2}$$

with $\alpha, \beta \in \mathbb{C}$ such that $|\alpha|^2 + |\beta|^2 = 1$.

Once it has been measured however, it can be found only in one of the two basis states $|0\rangle$ or $|1\rangle$. In fact, because they mutually exclude each other, there is no more superposition. These two basis states are represented by vectors, to distinguish them from the traditional bits.

$$|0\rangle = \begin{bmatrix}1\\ 0\end{bmatrix}, |1\rangle = \begin{bmatrix}0\\ 1\end{bmatrix} \tag{1.3}$$

The computational basis, which describes the state of the qubit, is constituted by these two vectors, which means that every quantum state can be expressed as the sum of these state vectors. For example:

$$|q_0\rangle = \frac{1}{\sqrt{2}}\,|0\rangle + \frac{i}{\sqrt{2}}\,|1\rangle \tag{1.4}$$

where

$$|q_0\rangle = \begin{bmatrix}\frac{1}{\sqrt{2}}\\ \frac{i}{\sqrt{2}}\end{bmatrix} \tag{1.5}$$

It results easy to verify that

$$\left|\frac{1}{\sqrt{2}}\right|^2 + \left|\frac{i}{\sqrt{2}}\right|^2 = 1 \tag{1.6}$$

and trace it back to the first and more general representation.

1.1.1 The Bloch sphere representation

One single qubit can be geometrically represented using the Bloch sphere, named after Felix Bloch [2], which is a sphere with a unitary radius. The poles represent the extreme states 0 on the north, and 1 on the south. While classical bits can exist only in these two states, qubits cover the entire sphere, allowing them to assume combinations of these. Each point on the sphere represents a superposition of $|0\rangle$ and $|1\rangle$. The quantum bits contain much more information than classical ones, and the Bloch sphere geometrically depicts this property:

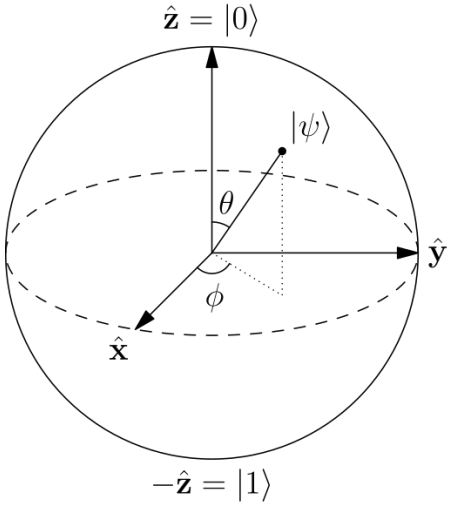

Figure 1.1: Representation of the Bloch Sphere

When the qubit is measured, it collapses to one of the two poles and this depends on which direction the arrow in the Bloch representation points to if it is closer to the north pole, there is a larger probability of collapsing to that pole (i.e. $|0\rangle$) and equally happens for the south pole (i.e. $|1\rangle$). This introduces the definition of probability in the Bloch sphere: the angle θ of the arrow with the vertical axes corresponds to that probability. If the arrow happens to point exactly at the equator, there is a 50% probability of collapse to any of the two poles upon measurement. Rotating a vector referring to the z-axis results in a phase change,

and does not affect which state the arrow will collapse to, when it is measured. This rotation is achieved by changing the ϕ variable.

As seen before, the quantum state can be represented as:

$$|q\rangle = \alpha\,|0\rangle + \beta\,|1\rangle \qquad \alpha, \beta \in \mathbb{C} \tag{1.7}$$

But, using the polar representation of complex numbers:

$$e^{i\phi} = cos\phi + isin\phi \tag{1.8}$$

we can obtain a new representation for quantum bits on the Bloch sphere:

$$|q\rangle = \alpha\,|0\rangle + \beta e^{i\phi}\,|1\rangle \tag{1.9}$$

Because the vector state as norm equal to 1 and thanks to the geometric identity, it is possible to describe the variables α and β in relationship with the angle θ:

$$\sqrt{|\alpha|^2 + |\beta|^2} = 1 \tag{1.10}$$

$$\sqrt{sin^2\theta + cos^2\theta} = 1 \tag{1.11}$$

$$\alpha = cos\frac{\theta}{2} \qquad \beta = sin\frac{\theta}{2} \tag{1.12}$$

Thanks to this, is possible to describe the state of a quantum bit using the two variables ϕ and θ:

$$|q\rangle = cos\frac{\theta}{2}\,|0\rangle + \sin\frac{\theta}{2}e^{i\phi}\,|1\rangle \tag{1.13}$$

1.1.2 Measuring a qubit

The values contained in a state vector include information about the probability of finding the qubit in a specific state, but to know exactly in which state it is, one necessarily needs to measure it.

After the measurement of a qubit in a superposition state, it switches to a pure state. This means that the qubits lost their quantumness. A pure state is a state in which if we measure again the qubit, the state obtained will be the same at 100% probability. The new resultant state must always preserve the normalization

constraint mentioned before. It is possible to start with an example to better understand how the process of measurement works.

We have a one-qubit state $|\psi\rangle$, for which we have to calculate the probability p to measure it in a state $|x\rangle$.

$$p(|x\rangle) = |\langle x|\psi\rangle|^2 \tag{1.14}$$

In this formula the probability that the quantum bit is measured in a state $|x\rangle$ is described, with $|x\rangle$ that can be every possible state of the qubit. The probability p is defined with the Bra-ket notation, also called Dirac notation after its creator, Paul Dirac [3]. It is a notation for linear algebra and linear operators on complex vector spaces together with their dual space, both in the finite-dimensional and infinite-dimensional case. Its use in quantum mechanics is quite widespread.

In the Bra-ket notation each bra $\langle v|$, which represents a vector, corresponds to one ket $|v\rangle$, which stands for a linear form. To find the probability we use the inner product, which is a product of two quantum states $\langle\psi|$ and $|x\rangle$, producing a scalar value. An inner product is also called an overlap, the overlap between quantum states.

Here we have an example:

$$|q_0\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{i}{\sqrt{2}}|1\rangle \tag{1.15}$$

$$\langle 0|q_0\rangle = \frac{1}{\sqrt{2}}\langle 0|0\rangle - \frac{i}{\sqrt{2}}\langle 0|1\rangle = \frac{1}{\sqrt{2}} \times 1 - \frac{1}{\sqrt{2}} \times 0 = \frac{1}{\sqrt{2}} \tag{1.16}$$

$$|\langle 0|q_0\rangle|^2 = \frac{1}{2} \tag{1.17}$$

In this case, we are considering the state of $|q_0\rangle$ and calculating the probability of measuring $|0\rangle$, which results to be 0.5; which is the result also in the case of the calculation of the probability of $|1\rangle$.

We considered the most complex case in which the qubit is in a state of superposition, where it can *collapse* to state $|0\rangle$ or $|1\rangle$ once measured. There is also the case in which the quantum bit is already in the state $|0\rangle$ or $|1\rangle$, and if at the beginning the qubit is in one of these states, it will be in the same state also after the measurement. This characteristic of the measurement is important because allows us to derive analogies with classical data and to manipulate such data in a quantum computer in analogy to what happens in a classical one.

1.2 Superposition

As mentioned above, the superposition is the main property that distinguishes a qubit from a classical bit. Superposition is one of the fundamental principles of quantum mechanics. A quantum state in superposition can be seen as a linear combination of other distinct quantum states. This quantum state in superposition forms a new valid quantum state.

Superposition is commonly defined as the ability of a quantum system to be in multiple states at the same time until it is measured.

This concept is correctly illustrated by the double-slit experiment carried out in 1801 by Thomas Young, an English physicist [4]. The purpose of this experiment was to prove that light consists of waves, nowadays this experiment is used to better understand the way electrons can act like waves and create interference patterns.

The experiment itself is elegantly straightforward, entailing the passage of a monochromatic light beam through a single aperture, followed by a passage through a double slit, with the resultant light being ultimately cast upon a remote screen. What Young beheld were a series of alternating luminous fringes juxtaposed with shadowy bands, an observation that he aptly construed as regions of constructive interference manifesting as the bright fringes and zones of destructive interference representing the dark bands. It is self-evident that this phenomenon would not transpire if light were to propagate linearly. To elucidate the phenomenon of diffraction, namely the perturbation of the trajectory of wave propagation when encountering an obstruction, one must invoke Huygens' principle [5], which indeed proclaims that each aperture acts as if it were an entirely fresh wellspring of luminous waves, radiating outward in all directions. This phenomenon bears resemblance to water waves that undergo diffraction as they navigate through a narrow fissure.

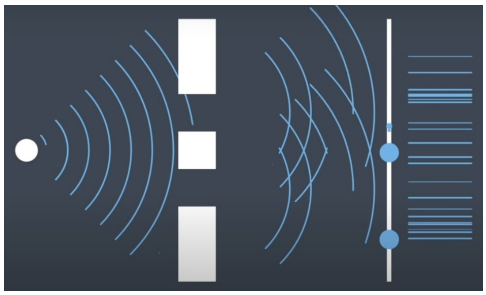

Figure 1.2: Representation of the waves traversing the slits [6]

Figure 1.3: Representation of the result of the experiment [6]

This outcome allows us to demonstrate that there is an interplay of interference among the waves traversing the slits, even though it appears that these waves should follow two separate, non-intersecting paths. Each photon, far from traversing only one of the slits, concurrently explores every conceivable trajectory on its way to the photographic plate.

To comprehend the potential occurrence of this phenomenon, alternative experiments have delved into tracing the trajectories of individual photons. The act of measurement somehow disrupts the trajectory of photons, resulting in an outcome consistent with classical physics: two bright lines on the photographic plate, perfectly aligned with the slits in the barrier. As a consequence, scientists have inferred that superposition remains elusive in direct observation; only the subsequent outcome, interference, can be witnessed.

Qubits, exhibiting the remarkable property of superposition, can simultaneously occupy both of their foundational states (again, denoted as $|0\rangle$ and $|1\rangle$). When a qubit undergoes measurement, it collapses into one of its states, and the outcome reflects that particular state.

Quantum superposition stands in stark contrast to the superposition of classical waves. In the realm of quantum computing, a system comprised of n qubits can exist in a superposition of 2^n states, ranging from $|000...0\rangle$ to $|111...1\rangle$. Conversely, when it comes to classical waves, combining, for example, n musical tones of distinct frequencies yields a superposition of n frequencies. Classical wave superposition scales linearly, while the superposition of quantum states scales exponentially. In the realm of computing, the concept of superposition carries profound implications for the future of information processing and storage, giving the possibility to increase them exponentially.

1.3 Entanglement

Entanglement was introduced by Erwin Schrödinger in 1935 within the framework of quantum mechanics [1]. The concept denotes a profound linkage between particles.

It is defined by a mathematical construct called the *wave function* of a system, which characterizes particle properties as if they constituted a singular entity, despite the considerable physical separation between the particles. This correlation permits the first particle to exert an instantaneous influence on the second, and vice versa.

Not all particles are "entangled", meaning intertwined. For this correlation to occur, i.e., for two particles to have correlated quantum states, these two particles must be simultaneously generated through a physical interaction. Even though the two particles are separated by a considerable distance, a potential alteration to the state of the first particle would instantaneously have a measurable impact on the state of the second one, thus giving rise to the phenomenon known as *"spooky action at a distance"* [7].

A typical example of a quantum state is the spin of a particle, which can take on either a positive or negative value. When dealing with "entangled" particles, those bound by this connection, the sum of the spins of the two particles adds up to zero. Therefore, if the spin of one of them is measured, the spin of the other is automatically and instantaneously known.

Niels Bohr believed that particles came into existence when observed and that only the wave function of the system was real before observation.

Albert Einstein, Boris Podolsky, and Nathan Rosen, on the other hand, were firmly convinced that particles possessed their inherent characteristics from the outset, called local realism. This conviction stemmed from the implications of relativity, which had demonstrated that no information could be transmitted instantaneously, exceeding the speed of light. This instantaneous phenomenon, entanglement, was therefore believed to be connected to hidden variables, unknown to us, that define the spin of particles even before observation. These scientists deemed quantum mechanics incomplete and voiced their criticisms in the renowned EPR paradox [8]. As became evident many decades later, such a statement must be exclusively interpreted in the context of the Theory of Relativity and cannot be deemed of general validity.

In 1964, John Bell introduced a probabilistic method known as Bell's theorem to ascertain whether the quantum state of two entangled particles was predetermined from the outset (in line with the concept proposed by Einstein, Podolsky, and Rosen) or if it only manifested as a result of observation (as per Bohr's hypothesis) [9]. His work ultimately substantiated Bohr's theory.

Due to technological constraints, it wasn't until 1982 that Alain Aspect measured the behavior of entangled photons and confirmed Bohr's theory. Einstein was, therefore, proven incorrect.

As long as the two particles remain unobserved, their spins remain undetermined, implying that both particles possess both positive and negative spins simultaneously, following the principle of superposition of states. It is only the presence of the

observer that interferes with the system and collapses it into "reality".

Entanglement provides instant knowledge of the second particle's behavior, because the two particles are a unified system governed by a single wave function. A local external disturbance, such as the arrival of a photon or an observer, not only alters the behavior of the first particle but influences the entire system. Consequently, it defines the quantum state of the second particle as well.

1.4 Quantum Gates

Quantum logic gates serve as instrumental tools for executing intricate computations, affording researchers the capability to manipulate the states of qubits, that we already described as the fundamental units of information. Each quantum gate is endowed with a specific role, such as altering the state of a qubit or creating entanglement between multiple qubits. These gates serve as the quantum equivalent of classical logic gates in traditional digital computers. What sets quantum logic gates apart is their reversibility, in contrast to many classical logic gates. Some universal classical gates, offer reversibility and can be directly adapted to quantum logic gates. In mathematical terms, quantum logic gates are represented by unitary matrices. Most frequently, quantum gates work with one or two qubits, and their behavior can be described using 2×2 or 4×4 matrices with orthonormal rows.

Quantum logic gates encompass distinct functionalities:

- The first pertains to state manipulation, wherein quantum gates possess the capability to modify the state, encompassing properties and configurations, of a qubit.

- The second involves entanglement, whereby quantum gates can establish entanglement between qubits. As we explained in the previous section, entanglement denotes a state wherein the properties of several qubits become interconnected, even when they are spatially separated. Therefore, applying an operation to one qubit instantaneously influences the state of the other qubit.

- The third function pertains to the Quantum Fourier Transform (QFT). The QFT operates on multiple qubits, altering their quantum states in a manner that encapsulates frequency-related information. It resembles a specialized mathematical operation facilitating the analysis of patterns and frequencies within quantum algorithms.

Quantum logic gates play a pivotal role in facilitating intricate computational tasks within the realm of quantum computing. They constitute the essential tools for executing operations on qubits, enabling the solution of specific problems through the utilization of quantum computers. Moreover, quantum logic gates

find application in quantum error correction methodologies, which hold utmost significance in safeguarding quantum information against inaccuracies stemming from external noise and disturbances.

As described in Section 1.1.1, every normalized pure state can be represented in the following manner:

$$|q\rangle = cos\frac{\theta}{2}|0\rangle + \sin\frac{\theta}{2}e^{i\phi}|1\rangle \tag{1.18}$$

In this context, ϕ represents the relative phase, taking values within the range [0, 2π]. Conversely, θ, within the interval [0, π], dictates the likelihood of measurement yielding $|0\rangle$ or $|1\rangle$ outcomes. As previously observed, all normalized pure states can be visually represented on the spherical surface of the Bloch sphere. The coordinates of such a state on the Bloch sphere are denoted by the Bloch vector.

$$\vec{r} = \begin{pmatrix} sin\theta cos\theta \\ sin\theta cos\phi \\ cos\theta \end{pmatrix} \tag{1.19}$$

According to this notation, the different states can be represented as following:

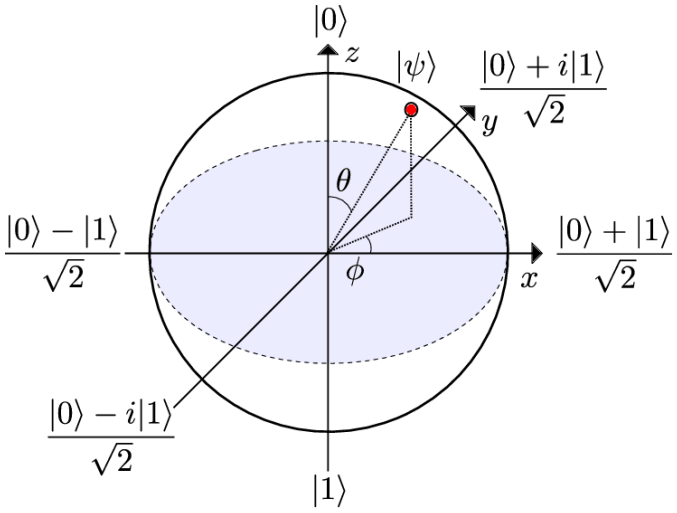

Figure 1.4: Representation of the differents states in the Bloch Sphere [10]

$$|0\rangle : \theta = 0, \phi = arb, \vec{r} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \tag{1.20}$$

$$|1\rangle : \theta = \pi, \phi = arb, \vec{r} = \begin{pmatrix} 0 \\ 0 \\ -1 \end{pmatrix} \tag{1.21}$$

$$|+\rangle : \theta = \frac{\pi}{2}, \phi = 0, \vec{r} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \tag{1.22}$$

$$|-\rangle : \theta = \frac{\pi}{2}, \phi = \pi, \vec{r} = \begin{pmatrix} -1 \\ 0 \\ 0 \end{pmatrix} \tag{1.23}$$

$$|i\rangle : \theta = \frac{\pi}{2}, \phi = \frac{\pi}{2}, \vec{r} = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \tag{1.24}$$

$$|-i\rangle : \theta = \frac{\pi}{2}, \phi = \frac{3\pi}{2}, \vec{r} = \begin{pmatrix} 0 \\ -1 \\ 0 \end{pmatrix} \tag{1.25}$$

The following representation of different gates will focus on single-qubit gates and on the CNOT gate, the building blocks of quantum computing.

1.4.1 Pauli X-gate

The X-gate is a single-qubit rotation through π radians around the x-axis, it is the quantum equivalent of the classical NOT gate. It is represented by the unitary matrix:

$$H = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} = |0\rangle \langle 1| + |1\rangle \langle 0| \tag{1.26}$$

Internally, it induces a rotation of the quantum state by π radians around the x-axis. Examining the representation of the Bloch sphere above, it is possible to observe that $|+\rangle$ and $|-\rangle$ are situated along the x-axis, which leads to conclude that $|+\rangle$ and $|-\rangle$ constitute the two eigenstates (which is a state of a quantized dynamic system in which one of the variables defining the state has a determinate

fixed value) of the X-gate. Consequently, these states remain unaltered by the Pauli X operation.

The result of this operation, at a mathematical level, starting with a $|0\rangle$ state, is:

$$X|0\rangle = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} = |1\rangle \tag{1.27}$$

The representation on the Bloch sphere and of the circuit are the following:

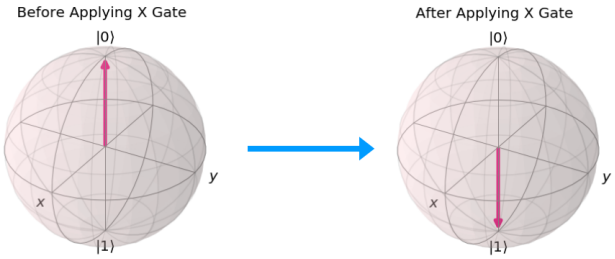

Figure 1.5: Representation of the transformation executed by the X-gate using the Bloch Sphere [11]

Figure 1.6: Pauli X-gate circuit representation [11]

1.4.2 Pauli Z-gate and Pauli Y-gate

The Z-gate is a phase flip gate that causes rotation around the z-axis by π radians.

$$Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} = |0\rangle\langle 0| - |1\rangle\langle 1| \tag{1.28}$$

Considering that $|0\rangle$ and $|1\rangle$ are situated on the z-axis, the Z-gate exerts no influence on these states. In different terms, $|0\rangle$ and $|1\rangle$ represent the two eigenstates of the Z-gate. Conversely, it inverts $|+\rangle$ to $|-\rangle$ and $|-\rangle$ to $|+\rangle$. Initiating the state vector as $|+\rangle$, this is the transformation effected by the Z gate on this vector:

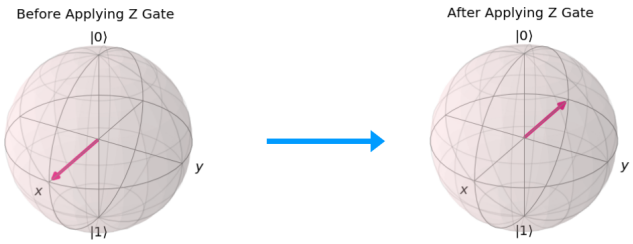

Figure 1.7: Representation of the transformation executed by the Z-gate using the Bloch Sphere [11]

The Y-gate is a phase flip gate that causes rotation around the y-axis by π radians.

$$Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} = -i\,|0\rangle\langle 1| + i\,|1\rangle\langle 0| \tag{1.29}$$

In the figure below is illustrated the Y-Gate applied on a qubit in the state of $|0\rangle$ and the resulting state, which is $|1\rangle$.

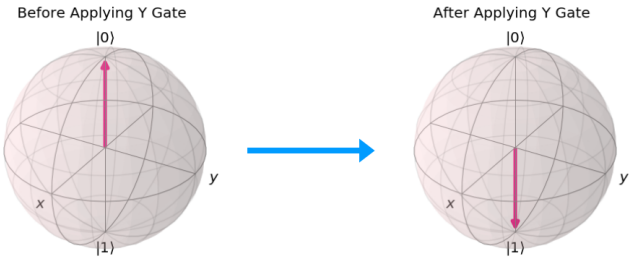

Figure 1.8: Representation of the transformation executed by the Y-gate using the Bloch Sphere [11]

1.4.3 Hadamard gate

The Hadamard gate stands as one of the most commonly employed gates in the domain of quantum computing. When the H-Gate is applied to a qubit initially in the state $|0\rangle$, it transforms the qubit into a superposition state, where the likelihood of measuring 0 equals that of measuring 1.

The effect of the H-gate can be conceptualized as a rotation around the Bloch vector [1,0,1], which represents the line connecting the x and z-axes on the Bloch sphere.

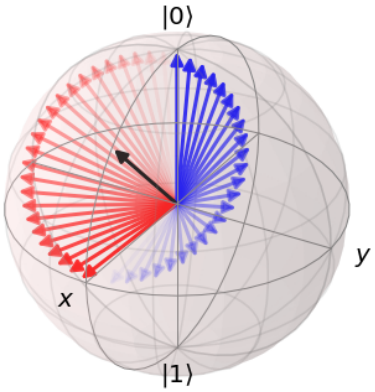

Figure 1.9: Representation of the rotation axis on the Bloch Sphere [11]

The Hadamard matrix is represented by:

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \tag{1.30}$$

If the starting point is the $|0\rangle$ state, the result applying the H-gate is the following:

$$X\,|0\rangle = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix} = |+\rangle \tag{1.31}$$

Where $|+\rangle$ is a superposition state.

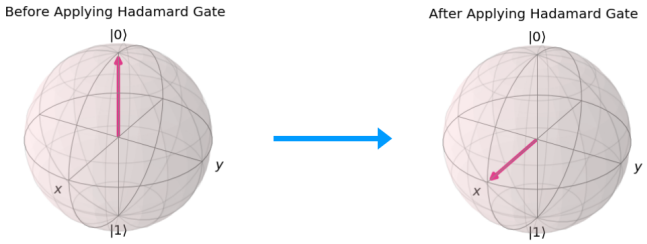

Figure 1.10: Representation of the transformation executed by the H-Gate using the Bloch sphere [11]

1.4.4 Rϕ-gate

The Rz-gate, or the Rϕ-gate, is one of the parametrized gates. The term "parametrized" means that these gates accept a parameter and execute an operation dependent

on this parameter. In this case, the accepted parameter is ϕ, and the operation conducted is a rotation around the z-axis by an angle of ϕ radians. The matrix representing this gate is provided as follows:

$$R_\phi = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\phi} \end{bmatrix} \tag{1.32}$$

Given that the states $|0\rangle$ and $|1\rangle$ are situated along the z-axis, they remain unaffected by the Rz-gate. You can experiment with the Rz-gate by initializing the state vector as $|+\rangle$ and applying it as follows:

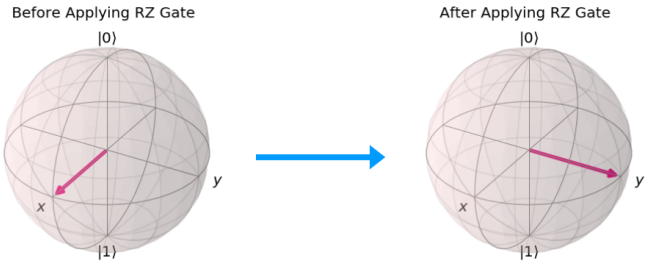

Figure 1.11: Representation of the transformation executed by the Rz-Gate using the Bloch sphere [11]

1.4.5 I, S and T-gate

The I, S, and T-gates can be viewed as specific instances of the more general Rϕ gate.

The I-gate, often known as the Identity gate, has no specific transformative effect. Its matrix corresponds to the Identity matrix itself. It holds significance due to its utility in various calculations and also for its role in specifying a "null" operation when dealing with physical hardware constraints.

The S-gate, on the other hand, is a particular case of the Rϕ gate, with ϕ equal to $\pi/2$. In other words, it induces a $\pi/2$ radians rotation around the z-axis. Unlike some other gates, the S-gate is not its inverse, although it remains unitary. The S-gate and its inverse, known as S-dagger or Sdg, are represented as follows:

$$S = \begin{bmatrix} 1 & 0 \\ 0 & e^{\frac{i\pi}{2}} \end{bmatrix}, S^\dagger = \begin{bmatrix} 1 & 0 \\ 0 & e^{-\frac{i\pi}{2}} \end{bmatrix} \tag{1.33}$$

The S-dagger gate orchestrates a clockwise rotation of the vector by $\pi/2$ radians around the z-axis. The execution of two consecutive S-gate operations is tantamount

to applying a Z-gate operation. Owing to this relationship, the S gate is also denoted as the $\sqrt{Z}$-gate.

The T-gate represents a particular instance of the Rϕ gate with ϕ equal to $\pi/4$. It exhibits analogous behavior to the S-gate. The T-gate and its inverse, referred to as T-dagger or Tdg, can be expressed as follows:

$$T = \begin{bmatrix} 1 & 0 \\ 0 & e^{\frac{i\pi}{4}} \end{bmatrix}, T^{\dagger} = \begin{bmatrix} 1 & 0 \\ 0 & e^{-\frac{i\pi}{4}} \end{bmatrix} \tag{1.34}$$

Applying four T-gate is equivalent to performing a single Z-gate.

1.4.6 U-gate

The U-gates are parametrized gates. The firsts two U-gates are U_1-gate and U_2-gate:

$$U_1 = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\lambda} \end{bmatrix}, U_2 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & -e^{i\lambda} \\ e^{i\phi} & e^{i\lambda+i\phi} \end{bmatrix} \tag{1.35}$$

The matrix for the U_1-gate is equivalent to the one for the Rϕ-gate.

In fact, every gate introduced can be represented using the U_3-gate, which is a parametrized gate and has the following matrix:

$$U_3(\theta, \phi, \lambda) = \begin{bmatrix} cos\frac{\theta}{2} & -e^{i\lambda} sin\frac{\theta}{2} \\ e^{i\phi} sin\frac{\theta}{2} & e^{i\lambda+i\phi} cos\frac{\theta}{2} \end{bmatrix} \tag{1.36}$$

From this, it is possible to observe that both U_1-gate and U_2-gate derive from U_3-gate, precisely: U_1=U_3(0,0, λ) and U_1=U_3($\pi/2, \phi, \lambda$)

1.4.7 C-NOT gate

The controlled Pauli-X gate, commonly referred to as the CNOT gate, is a ubiquitous and indispensable component in quantum circuits. This gate operates on two qubits, denoted as qubits A and B, within an n-qubit system. In this configuration, one qubit (designated as qubit B) serves as the target qubit, while the other qubit acts as the control qubit. When the control qubit A is in the state $|1\rangle$, the CNOT gate applies a Pauli-X gate (equivalent to a NOT gate) to the target qubit B, resulting in its state being flipped. Conversely, if qubit A is in the state $|0\rangle$, no operation is performed on qubit B.

It is possible to implement any arbitrary unitary operation on an n-qubit system using solely CNOT gates and single-qubit unitary gates. Numerous multi-qubit

gates are currently implemented in experimental quantum systems using CNOT gates in combination with other single-qubit gates. For instance, a controlled Pauli-Z gate can be realized through the application of two Hadamard single-qubit gates and one CNOT gate. [12]

To provide a more in-depth explanation, let's start by saying that a quantum system consisting of two qubits, denoted as A and B (also referred to as a two-qubit register), exists within a 4-dimensional Hilbert space, denoted as $\mathcal{H}_A \otimes \mathcal{H}_B$. The computational basis of this space is:

$$(|00\rangle = |0\rangle_A \otimes |0\rangle_B, |01\rangle = |0\rangle_A \otimes |1\rangle_B, |01\rangle = |1\rangle_A \otimes |0\rangle_B, |11\rangle = |1\rangle_A \otimes |1\rangle_B) \tag{1.37}$$

Let's now define U as a general unitary operator [1] acting on a single qubit, a controlled-U operator operates on the Hilbert space $\mathcal{H}_A \otimes \mathcal{H}_B$ as follows: one qubit (designated as qubit A) serves as the control qubit, while the other qubit acts as the target qubit. When the control qubit A is in the state $|1\rangle$, the operator U is applied to the target qubit B. Conversely, if qubit A is in the state $|0\rangle$, no operation is performed on qubit B. If U represents the Pauli-X operator, then the CNOT gate corresponds to the controlled-X operator (also denoted as c-X), with qubit A acting as the control and qubit B as the target. Thus, the action of the CNOT gate on the two-qubit register is described as follows:

$$CNOT\,|00\rangle = |00\rangle, CNOT\,|01\rangle = |01\rangle, CNOT\,|10\rangle = |11\rangle, CNOT\,|11\rangle = |10\rangle \tag{1.38}$$

The matrix representing the CNOT gate is:

$$CNOT_{AB} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{1.39}$$

The action of the CNOT gate can be described by:

$$\forall x, y \in \{0, 1\}, \mathrm{CNOT}\,|xy\rangle = |x, x \oplus y\rangle \tag{1.40}$$

[1] A unitary operator is a linear operator whose inverse equals its adjoint. It may be conceptualized as a linear transformation which is bijective and length-preserving.

where $\oplus$ is the XOR operator, which is the exclusive OR, this operator is a logical operation that returns true (1) if and only if one of the two input values is true, otherwise it returns false (0).

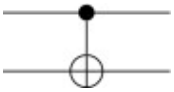

Figure 1.12: C-NOT gate representation in a circuit [13]

1.5 Quantum Circuits

The core of quantum computing is the quantum circuit, a foundational construct that orchestrates coherent quantum operations on quantum bits of information, notably qubits.

A quantum circuit is a meticulously orchestrated sequence, comprising quantum gates, measurements, and resets, potentially contingent on concurrent real-time classical computation. The universal quality of a set of quantum gates is exemplified when it can proficiently approximate any unitary transformation of qubits to a high degree of precision through a sequence of its constituent gates. Consequently, any quantum program can be articulated through a series of quantum circuits, complemented by classical computations operating in close temporal proximity.

1.5.1 Evolution of Quantum circuits

The history of quantum computing traces back to Richard Feynman's lectures on the potential advantages of utilizing quantum systems for computation in 1982, during a class at the MIT Computer Science and Artificial Intelligence Laboratory where he discussed the idea of "a universal quantum simulator". Feynman advocated for leveraging quantum mechanical phenomena to execute computations that would be unfeasible or unattainable using classical computers. Subsequently, this concept evolved into the field of quantum computing. Feynman's seminal ideas and contributions persist in shaping the landscape of quantum computing, and he is frequently regarded as one of its founding figures. In the same year, Feynman also published a paper titled "Simulating Physics with Computers," in which he underscored the necessity of quantum computers for simulating quantum systems. Feynman emphasized that quantum simulations necessitate physical quantum systems, a concept now recognized as one of the primary objectives of quantum computers. The limitations of conventional classical simulations stem from the insufficient accessibility of states, highlighting the unique capability of quantum computers to address this challenge effectively.

Feynman proposed utilizing controllable quantum environments for analog quantum computations, particularly for simulating complex systems that involve many-body interactions. While simulating a single electron is relatively straightforward, the complexity escalates exponentially with the number of electrons involved due to the increasing number of possible configurations. For instance, simulating the binding of a potential drug molecule to a receptor entails considering a vast number of combinations, rendering conventional computational methods computationally expensive. In his keynote address, Feynman explores the simulation of quantum physics using computers, considering various computing systems. He emphasizes the importance of reversible computation and introduces a rule for simulation, stating that the number of computer elements required should be proportional to the space-time volume of the physical system. Feynman outlines the concept of a new quantum mechanical computer capable of simulating any quantum system, including the physical world, and poses questions regarding the intersimulatability of quantum systems and the existence of universal quantum simulators. He suggests that linear operators on a two-state quantum system could serve as a basis for simulating any quantum system, although this remains open for further exploration. Thus, Feynman's approach highlights the potential of quantum computing to address such computational challenges efficiently. [14]

Subsequently, in 1984, Albert introduced the concept of a "self-measuring quantum automaton" capable of tasks beyond classical computing capabilities, although the machine's specifications remained largely unspecified. However, in 1985, Deutsch is credited with the development of the first clearly defined quantum computing system, albeit its feasibility was questioned. Meanwhile, Bernstein and Vazirani's work in the early 1990s opened up the field of quantum computing to the theoretical computer science community, establishing quantum complexity theory and enabling the formal study of quantum algorithms and operations akin to classical algorithms. Simon's 1993 description of an oracle problem showcased the exponential speedup of quantum computers over classical ones, while Shor's 1994 quantum algorithm for efficient factorization of large numbers ignited widespread interest in the field and concern among cryptography experts. Notably, in the early 1980s, Weisner and Bennett explored the concept of quantum key exchange, offering a potential solution to security systems compromised by computationally feasible factorization. Finally, in 1998, UC Berkeley demonstrated the first functional two-qubit nuclear magnetic resonance computer, marking significant progress in quantum computing technology. [15]

1.5.2 Properties of Quantum Circuits

In the process of constructing quantum circuits, various properties come into play, aiding in the assessment of the circuit's "magnitude" and its feasibility for execution

on a quantum device susceptible to noise. Certain properties, such as the number of qubits, offer a straightforward grasp, while others, like circuit depth and the count of tensor components, demand a more detailed elucidation. Starting from the number of qubits, it depends on the width of the circuit, however for more articulated circuits with classical registers, and classically controlled gates, this equivalence is no more satisfied.

A notably significant property of a quantum circuit is referred to as circuit depth. The circuit's depth serves as a metric for quantifying how many "layers" of quantum gates are concurrently executed, encompassing the completion of the circuit's defined computation. This property holds particular relevance since the execution of quantum gates consumes time, making the circuit's depth an approximate indicator of the time quantum computing resources will require to carry out the circuit. Therefore, the circuit's depth is a pivotal criterion employed to assess the feasibility of running a quantum circuit on a given quantum device. Mathematically, the depth of a quantum circuit can be defined as the longest path within a directed acyclic graph (DAG).

1.5.3 Quantum Circuit representation

Quantum circuits serve as direct analogs to classical computers, employing wires to connect gates that manipulate qubits. Notably, the transformations carried out by these gates are inherently reversible, distinguishing them significantly from classical computing paradigms. Circuit descriptions utilize simplified diagrams to depict connections, with single wires representing the transmission of quantum states between operations. The physical medium of transmission, whether a wire or optical channel, is of no consequence. Gates and unitary operations on qubits are represented by boxes, maintaining an equal number of input and output qubits due to the reversibility of transformations. Measurement, denoted by a box with a symbolic device, typically entails projective measurement in the computational basis, unless specified otherwise, allowing for the introduction of ancilla systems [2]. [16]

[2] an ancillary system refers to an auxiliary or supporting system utilized alongside the primary system to perform specific operations or measurements.

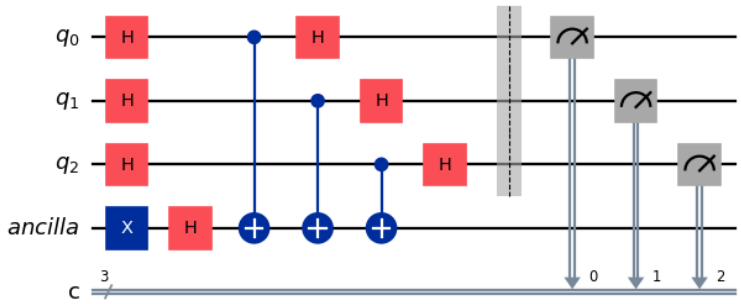

Figure 1.13: Example of a graphical representation of a quantum circuit [11]

Chapter 2

Classical methods for Option Pricing

2.1 Option Pricing

Option pricing theory involves estimating the value of an options contract, which is a derivative contract, by assigning a price, referred to as a premium. This premium is determined based on the calculated probability that the contract will finish in the money at expiration. In the money means that the exercising of the option produces value. The option pricing theory offers an assessment of an option's fair value, which traders utilize in their strategic decision-making.

The models employed for pricing options take into consideration various factors, including the current market price, strike price, volatility, interest rate, and time to expiration. Some commonly utilized models in this context include the Black-Scholes model, binomial option pricing model, and Monte Carlo simulation. These models enable traders to make informed decisions by quantifying the potential value of options based on market conditions and other relevant parameters.

2.2 Black-Scholes

The Black-Scholes model, or alternatively referred to as the Black-Scholes-Merton (BSM) model [17], conceived in 1973, continues to be esteemed as one of the most effective methods for valuing an options contract, it stands as a cornerstone in contemporary financial theory. It relies on five key variables:

- volatility
- underlying stock price

- time
- strike price
- risk-free rate

In the realm of stock market derivatives, where speculation is notably pronounced, precise option pricing plays a pivotal role in eliminating arbitrage opportunities. This model is specifically applied to determine the valuation of European options, signifying that the option can solely be executed on its expiration date.

The Black-Scholes model posits that financial instruments, like stocks or futures contracts, will exhibit a lognormal distribution of prices through a random walk characterized by constant drift and volatility. Leveraging this assumption and considering other crucial factors, the equation deduces the price of a European-style call option. As has been already said, the BSM equation necessitates five key variables: volatility, the underlying asset's price, the option's strike price, the time remaining until the option expires, and the risk-free interest rate. Armed with these variables, options sellers can theoretically establish rational prices for the options they offer.

Moreover, the model anticipates that the pricing of heavily traded assets adheres to a geometric Brownian motion marked by a consistent drift and volatility [18]. When applied to a stock option, the model integrates the stock's persistent price fluctuations, the time value of money, the option's strike price, and the time remaining until the option expires.

The Black-Scholes model is underpinned by specific assumptions:

- **No Dividends**: The option's lifespan is characterized by the absence of dividend payouts.
- **Random Markets**: Market movements are deemed unpredictable, following a stochastic process.
- **Zero Transaction Costs**: No costs are associated with purchasing the option.
- **Known and Constant Factors**: The risk-free rate and the underlying asset's volatility remains constant and are known.
- **Normal Distribution**: Returns on the underlying asset conform to a normal distribution.
- **European Option**: The option is of the European type, allowing exercise only at its expiration.

While the original Black-Scholes model initially overlooked the impact of dividends during the option's life, adaptations often incorporate dividend considerations by calculating the ex-dividend date value of the underlying stock. Moreover, many market makers are adjusting the model accounting for options that can be exercised before reaching expiration.

2.2.1 The Black-Scholes formula

The price of a call option on a dividend-free stock at time t prior to the option's expiration date can be expressed as follows:

$$C = SN(d_1) - PV(K)N(d_2) \tag{2.1}$$

Where:

$$d_1 = \frac{\ln \frac{S}{PV(K)}}{\sigma\sqrt{T}} + \frac{\sigma\sqrt{T}}{2} \tag{2.2}$$

$$d_2 = d_1 - \sigma_s\sqrt{T} \tag{2.3}$$

where:

- C is the call option price
- S is the current stock (or other underlying) price
- K is the strike price
- r is the Risk-free interest rate
- T is the number of years left to expiration
- $N(d)$ denotes the cumulative normal distribution function, represented in Figure 2.1, it represents the probability that a normally distributed random variable will assume a value less than d
- σ is the volatility
- $PV(K)$ represents the present value (or price) of a risk-free zero-coupon bond that pays the strike price K upon the expiration date of the option

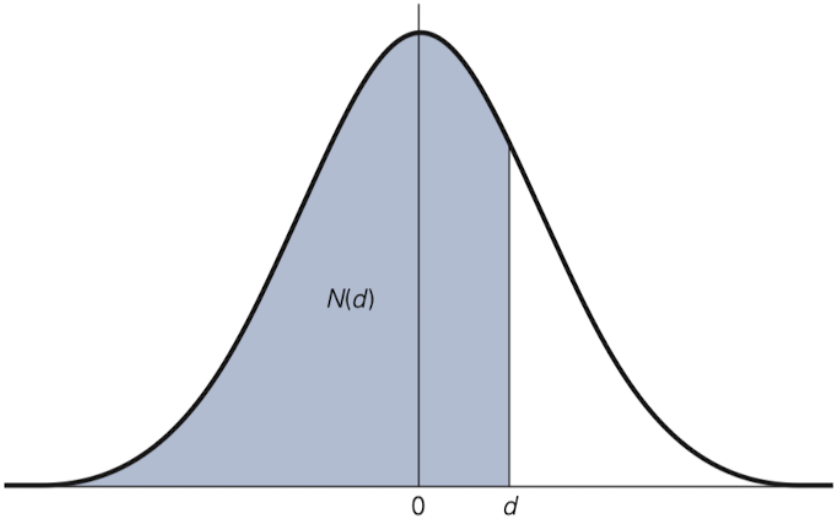

Figure 2.1: Representation of the cumulative normal distribution function for a given d [19].

The Black-Scholes Option Pricing Model requires only five input parameters to price a call option: the stock price, the strike price, the exercise date, the risk-free interest rate (for computing the present value of the strike price), and the volatility of the stock. Notably, it does not require knowledge of the probabilities as in the Binomial Option Pricing Model 2.3, nor does it necessitate knowledge of the expected return on the stock. This is because the expected return of the stock is already embedded in its current price, which is the discounted value of its future payoffs. By relying primarily on the stock's volatility, which is easier to measure and forecast than its expected return, the Black-Scholes formula can achieve high precision in option pricing. Additionally, the Black-Scholes formula applies to European options, but can also be used to price American options on non-dividend-paying stocks, as they generally have the same price as their European counterparts [19].

2.2.2 Volatility skew

The Black-Scholes model posits that stock prices adhere to a lognormal distribution and, primarily due to the inherent limitation of asset prices, they cannot be negative and are bounded by zero. In reality, asset prices often exhibit substantial right skewness and a measure of kurtosis (fat tails). This suggests that high-risk downward movements occur more frequently in the market than predicted by a normal distribution. While the assumption of lognormal underlying asset prices in

the Black-Scholes model would entail similar implied volatilities for each strike price, a shift occurred after the 1987 market crash. Implied volatilities for at-the-money options have consistently been lower than those for options further out of the money or deep in the money. This peculiar trend can be attributed to the market factoring in a heightened likelihood of significant downward volatility moves.

This phenomenon has given rise to what is known as the *volatility skew*. Graphing implied volatilities for options with the same expiration date reveals a distinctive smile or skewed shape. Consequently, the Black-Scholes model proves less efficient in accurately calculating implied volatility, prompting the use of adjustments or alternative models to better capture the intricate dynamics of market volatility.

2.2.3 Benefits of the Black-Scholes method

The Black-Scholes model stands as a beacon of success in the financial realm, embraced by numerous professionals for its manifold advantages. Below are delineated some of these merits:

- **Theoretical Framework**: The Black-Scholes model furnishes a robust theoretical foundation for option pricing. This affords investors and traders the ability to ascertain the equitable value of an option, by employing a structured and proven methodology.
- **Risk Management Capability**: Armed with the theoretical value of an option, investors leverage the Black-Scholes model to adeptly navigate and control their risk exposure across diverse assets. It proves invaluable not only for evaluating potential returns but also for discerning weaknesses within portfolios and areas of investment deficiency.
- **Portfolio Optimization**: The Black-Scholes model serves as a compass for portfolio optimization, offering insights into the expected returns and associated risks tied to different options. This empowers investors to make judicious decisions aligned with their risk tolerance and pursuit of profit.
- **Market Efficiency Boost**: The widespread use of the Black-Scholes model contributes to heightened market efficiency and transparency. Traders and investors, are well-versed in its principles and find themselves better equipped to price and trade options. This, in turn, simplifies the pricing process and engenders a more innate understanding of price derivation.
- **Pricing Consistency**: The Black-Scholes model enjoys widespread acceptance and application within the financial industry, fostering a harmonious and consistent approach to pricing across diverse markets and jurisdictions. This shared standardization streamlines the pricing landscape, promoting greater cohesion and efficiency.

2.2.4 Limitations of the Black-Scholes method

While the Black-Scholes model enjoys widespread use, it is not without its limitations. Some of these drawbacks are outlined below:

- **Limited Applicability**: The Black-Scholes model's utility is confined to pricing European options and fails to consider the possibility of early exercise for U.S. options before the expiration date, thereby limiting its overall usefulness.
- **Rigid Cashflow Handling**: The model assumes constancy in dividends and risk-free rates, a simplification that may not hold in the dynamic reality. Consequently, the Black-Scholes model may lack the flexibility to accurately depict the future cash flows of an investment due to its inherent rigidity.
- **Assumption of Constant Volatility**: The model relies on the assumption that volatility remains constant throughout the option's life. In practice, this is often not the case, as volatility tends to fluctuate with changes in supply and demand dynamics.
- **Propagation of Misleading Assumptions**: The Black-Scholes model incorporates various assumptions, including the absence of transaction costs or taxes, a constant risk-free interest rate across all maturities, permission for short selling of securities with the use of proceeds, and the absence of riskless arbitrage opportunities. Each of these assumptions has the potential to lead to pricing outcomes that deviate from actual results, introducing a level of uncertainty and potential misguidance.

2.3 Binomial Option Pricing Model

In a fiercely competitive market, ensuring the absence of arbitrage opportunities mandates that assets with identical payoff structures command identical prices. As mentioned before, the valuation of options poses a formidable challenge, and divergences in pricing create openings for arbitrage. While the Black-Scholes model persists as one of the most widely employed tools for option pricing, it is not without its limitations, as seen in the previous Section.

The binomial option pricing model assesses option values through an iterative process that incorporates multiple periods, particularly when evaluating American options [20].

The model determines option prices by assuming that, at the end of the subsequent period, the stock price can only take on two distinct values. This simplifying assumption allows us to illustrate the fundamental insight of Black and Scholes, namely that option payoffs can be perfectly replicated by creating a portfolio

consisting of a risk-free bond and the underlying stock. Additionally, we will observe that the model can offer a high degree of realism when considering stock price fluctuations over very short time horizons, the binomial model is favored in practical applications for its intuitive nature and frequent usage.

2.3.1 Binomial Option Pricing Model calculations

The binomial option model consistently applies identical probabilities of success and failure throughout each iteration until the option reaches its expiration. Assessing the value of American options and embedded options, the binomial tree proves to be an invaluable tool. While constructing the tree mechanically poses no challenge, the difficulty lies in defining the range of values the underlying asset can attain within a given timeframe. Unfortunately, the binomial tree model accommodates only two possible values, a limitation that is unrealistic since assets can, in reality, assume a multitude of values within a defined range.

Assumptions:

- The risk-free rate does not change
- There are no returns on the underlying stock
- At any given point in time, the price can only move one of two ways: either up or down (the term "binomial" refers to this)
- As a result, there are no transaction fees or taxes in today's market
- Investors are risk averse; they do not bother about taking risks
- Throughout the period, the discount factor (interest rate) remains constant

Let us consider a scenario where the present stock price is denoted as S, and in the subsequent period, the stock price can either increase to S_u or decrease to S_d. Additionally, let the risk-free interest rate be represented as r_f. We aim to calculate the price of an option, valued at C_u, if the stock price increases and C_d if the stock price decreases. To determine the value of the option today, it is necessary to find Δ which represents the number of shares of stock in the portfolio, and B, which represents the bonds, ensuring that the payoff of the replicating portfolio aligns with the payoff of the option in both upward and downward movements of the stock. To do this the following formulas are used:

$$S_u\Delta + (1 + r_f)B = C_u \tag{2.4}$$
$$S_d\Delta + (1 + r_f)B = C_d \tag{2.5}$$

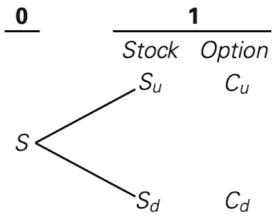

Figure 2.2: Representation of the upward-downward scenario [19]

The Replicating Portfolio formula in the Binomial Option Pricing model is represented by the following formulas:

$$\Delta = \frac{C_u - C_d}{S_u - S_d} \tag{2.6}$$

$$B = \frac{C_d - S_d\Delta}{1 + r_f} \tag{2.7}$$

where:

- C_u is the value of the option in the upward situation
- C_d is the value of the option in the downward situation
- S_u is the Stock value in the upward situation
- S_d is the Stock value in the downward situation
- Δ formula can also represent the sensitivity of the option's value to changes in the stock price

After knowing the values of Δ and B of the replicating portfolio, is possible to calculate the value of the Call option today through the following formula:

$$C = S\Delta + B \tag{2.8}$$

Even if the model appears very simple, it is possible to apply it in different situations, due to the fact that is quite powerful and could be used to evaluate any security whose payoff depends on the stock price. [19]

2.4 Monte Carlo

The Monte Carlo method is a stochastic approach involving the random sampling of inputs to address statistical problems, while a simulation serves as a virtual representation of a given problem. The Monte Carlo simulation seamlessly integrates these two concepts, creating a potent tool that enables the generation of a distribution or array of results for statistical problems. This is achieved by repeatedly sampling numerous inputs, providing a comprehensive and probabilistic perspective on the potential outcomes of the problem at hand.

A Monte Carlo simulation is a versatile method that explores a broad spectrum of possibilities, aiding in the reduction of uncertainty. Its flexibility is a key strength, enabling the variation of risk assumptions across all parameters and facilitating the modeling of a diverse range of potential outcomes. This approach allows for the comparison of multiple future scenarios, offering the ability to customize the model for different assets and portfolios under consideration. In essence, the Monte Carlo simulation provides a dynamic and comprehensive tool for assessing and managing risk by incorporating a wide array of potential outcomes and adjusting key parameters as needed.

Monte Carlo simulations are used in option pricing by generating numerous random paths for the underlying asset's price, each path associated with a specific payoff. These payoffs are subsequently discounted back to the present, and the average is computed to determine the option price [21]. This methodology is also employed in pricing fixed income securities and interest rate derivatives. Monte Carlo simulation finds its most extensive applications in portfolio management and personal financial planning. In these contexts, the simulation helps in assessing the potential performance of a portfolio by considering various factors and uncertainties. By simulating numerous possible market scenarios and their impacts on the portfolio, investors can gain insights into the range of potential outcomes and make more informed decisions. This makes the Monte Carlo simulation a valuable tool for managing risk and optimizing financial strategies in the dynamic landscape of investment and personal finance.

2.4.1 Monte Carlo simulation procedure

A broad category of pricing problems for European-style derivatives involves the evaluation of the following expectation functional.

$$E_t[f(Z(T;t,z))] \tag{2.9}$$

Where:

- Z is the stochastic process. It characterizes the price dynamics of one or more underlying financial variables, such as asset prices and interest rates, following the corresponding risk-neutral probability distribution.
- z is the value of Z at time t
- f is the function that defines the value of the derivative at the expiration time T
- t is the time
- T is the expiration time

The simulation procedure entails the generation of random variables with a specified probability density. Utilizing the law of large numbers, the average of these generated values is calculated, serving as an estimate for the expected value of the random variable. This approach allows for the approximation of complex mathematical expectations by leveraging the principles of probability and statistical sampling.

Considering a Brownian motion with drift rate μ and volatility σ:

$$dx_t = \mu dt + \sigma dB_t \tag{2.10}$$

Here B_t is a Standard Brownian motion with zero drift rate and unit variance rate.

The Euler discretized scheme is given by:

$$x_{t+\Delta t} = x_t + \mu \Delta t + \sigma(B_{t+\Delta t} - B_t) \tag{2.11}$$

Where:

- σ is a constant
- $O(\Delta t)$ is the approximation of the stochastic differential equation for x_t
- $B_{t+\Delta t} - B_t$ is a random increment. It has mean equal to 0, and variance equal to Δt. It can be simulated by random samples of $\sqrt{\Delta t}\epsilon$, where ϵ is a sample from a standard normal distribution.

The Monte Carlo procedure for the European Call option involves more specific steps. First of all, it is necessary to compute the expected payoff of the call option at expiration, based on the information on hand at time t, which is equal to:

$$E_t[max(S_T - X, 0)] \tag{2.12}$$

Then discount it to the present value at time t:

$$e^{-r(T-t)} E_t[max(S_T - X, 0)] \tag{2.13}$$

Hypothesizing that the stock price S_t follows the geometric Brownian movement, where the price dynamics, under the risk neutral measure is:

$$\frac{dS_t}{S_t} = (r - q)dt + \sigma B_t \tag{2.14}$$

or

$$d \ln S_t = (r - q - \frac{\sigma^2}{2})dt + \sigma dB_t \tag{2.15}$$

Where:

- σ is the volatility
- r is the risk free rate
- q is the dividend yield

The Euler approximation gives the following result:

$$\ln \frac{S_t + \Delta t}{S_t} = \ln S_{t+\Delta t} - \ln S_t = (r - q - \frac{\sigma^2}{2})\Delta t + \sigma\epsilon\sqrt{\Delta t} \tag{2.16}$$

The random asset price ratio is related to the standard normal random variable ϵ through:

$$\frac{S_t + \Delta t}{S_t} = e^{(r-q-\frac{\sigma^2}{2})\Delta t + \sigma\epsilon\sqrt{\Delta t}} \tag{2.17}$$

The price ratio S_T/S_t can be composed as the product of price ratios over successive time steps, where $T = N\Delta t$. If N are the time steps between the actual time t and the expiration time T, then the $\Delta t = \frac{(T-t)}{N}$. The numerical calculations procedure are repeated iteratively N times to simulate the price route from S_t to $S_T = S_{t+\Delta t}$.

The i^{th} result from the simulation of the call values c_i, which coincides with the terminal asset price $S_T^{(i)}$ is obtained with the discounted expectation approach under the risk neutral measurement:

$$c_i = e^{-r(T-t)} max(S_T^{(i)} - X,0) = \tag{2.18}$$

$$= e^{-r(T-t)} max(S_t e^{(r-q-\frac{\sigma^2}{2})(T-t)+\sigma\epsilon\sqrt{\Delta t}} \sum_{j=1}^{N} \epsilon_j^{(i)} - X,0) \tag{2.19}$$

After conducting the simulation multiple times, as described above, and with a sufficiently large number of runs, the expected call value is determined by calculating the sample average of the simulated call values obtained in the sample simulation. [13]

Chapter 3

Option Pricing using Quantum Computers

Here, we delineate the fundamental elements required to price options on a gate-based quantum computer. The essential components consist of:

1. Representing the probability distribution, denoted as $\mathbb{P}$, which characterizes the evolution of random variables (which represents the price of the option's underlying asset) $\mathbf{X} = \{X_1, X_2, \ldots, X_N\}$ within the quantum computer;
2. Creating the circuit for computing the payoff function, $f(\mathbf{X})$;
3. Determining the expectation value of the payoff, $\mathbb{E}_{\mathbb{P}}[f(\mathbf{X})]$.

We use Quantum Amplitude Estimation [22] for computing the expectation value of a function involving random variables. After introducing QAE, we elaborate on the process of loading probability distributions into a quantum register. This is followed by the construction of the circuit for computing the payoff and the configuration of Amplitude Estimation to gauge the expected value of the payoff. With these steps completed, we'll have the necessary components to price options on a quantum computer.

3.1 Quantum Amplitude Estimation

The advantage of the Quantum Amplitude Estimation in option pricing is related to the fact that it offers an asymptotic quadratic speed-up in comparison to the classical Monte Carlo simulation [23]. Monte Carlo methods in option pricing typically demand substantial computational resources to yield accurate estimates, especially for intricate options. Given the widespread use of options in the finance

industry, enhancing the convergence of these methods can significantly influence the operational efficiency of a financial institution.

3.1.1 Quantum Amplitude Amplification

In quantum computing, amplitude amplification is a technique akin to boosting the probability of success in classical randomized algorithms. The probabilistic approach enhances the likelihood of success by a consistent increment on each iteration, whereas amplitude amplification boosts the amplitude of success by a comparable margin with each iteration.

Suppose a unitary operator $\mathcal{A}$ acting on a register of $(n+1)$ qubits as:

$$\mathcal{A}\left|0\right\rangle_{n+1} = \sqrt{1-a}\left|\psi_0\right\rangle_n\left|0\right\rangle + \sqrt{a}\left|\psi_1\right\rangle_n\left|1\right\rangle \tag{3.1}$$

Where $a \in [0,1]$ is unknown and $\left|\psi_0\right\rangle_n$ and $\left|\psi_1\right\rangle_n$ are normalized states. The amplitude estimation enables to estimate a, with high probability. Let consider a Boolean function $\chi : X \to 0,1$, in which x is considered *good* if $\chi(x) = 1$; a represents the probability of producing a good element when measuring $\mathcal{A}\left|0\right\rangle$. If we iterate the process of executing $\mathcal{A}$, measuring the output, and validating the result using ψ_0, we can anticipate an average of $1/a$ repetitions before a solution is discovered.

Amplitude amplification is a method facilitating the discovery of a favorable solution x within an anticipated number of iterations of $\mathcal{A}$ and its inverse, which scales proportionally to $1/\sqrt{a}$. This expectation holds under the condition that algorithm $\mathcal{A}$ does not involve any measurements.

This concept represents an extension of Grover's search algorithm, where $\mathcal{A}$ was constrained to generate an equal superposition of all elements in X, with the guarantee that a single specific x existed satisfying $\chi(x) = 1$.

Since amplitudes are related to the square roots of probabilities, it is adequate to iterate the amplitude amplification process approximately $1/\sqrt{a}$ times to achieve success with a high probability. Analogous to how repeating a classical algorithm multiple times increases the likelihood of success, amplitude amplification aims to enhance the probability of being in a desired subspace of a Hilbert space [22].

Let's denote the Hilbert space representing the quantum system's state space as $\mathcal{H}$. Each Boolean function $\chi : Z \to \{0,1\}$ partitions $\mathcal{H}$ into two subspaces: a "good" subspace spanned by basis states $\left|x_i\right\rangle \in \mathcal{H}$ where $\chi(x) = 1$, and a "bad" subspace which is the orthogonal complement of the good subspace.

Every pure state $\left|\Upsilon_i\right\rangle$ in $\mathcal{H}$ can be uniquely decomposed into a sum of two projections: $\left|\Upsilon_{1i}\right\rangle$ onto the good subspace and $\left|\Upsilon_{0i}\right\rangle$ onto the bad subspace. The probability of measuring a "good" state from $\left|\Upsilon_{1i}\right\rangle$ is denoted as a_Υ, and the

probability of measuring a "bad" state is denoted as b_Υ. The probabilities are defined by the following formulas:

$$a_\Upsilon = \langle \Upsilon_1 | \Upsilon_1 \rangle \tag{3.2}$$
$$b_\Upsilon = \langle \Upsilon_0 | \Upsilon_0 \rangle \tag{3.3}$$

Due to the fact that $|\Upsilon_1\rangle$ and $|\Upsilon_0\rangle$ are orthogonal, we obtain that $a_\Upsilon + b_\Upsilon = 1$.

Now, consider a quantum algorithm $\mathcal{A}$ acting on $\mathcal{H}$ without any measurements, transforming the initial zero state $|0\rangle$ into $|\Psi_i\rangle = \mathcal{A}|0\rangle$. The process of amplification involves repeatedly applying a unitary operator $\mathcal{Q}$, defined as:

$$\mathcal{Q} = \mathcal{Q}(\mathcal{A}, \psi_0) = \mathcal{A}\mathcal{S}_0\mathcal{A}^\dagger\mathcal{S}_{\psi_0} \tag{3.4}$$

Here, the operator S_{ψ_0} selectively changes the sign of the amplitudes of "good" states:

$$|x_i\rangle \to \begin{cases} -|x_i\rangle & \text{if } \chi(x) = 1 \\ |x_i\rangle & \text{if } \chi(x) = 0 \end{cases} \tag{3.5}$$

while S_0 changes the sign of the amplitude only if the state is the zero state $|0\rangle$. Notably, $\mathcal{Q}$ is well-defined because it's assumed that $\mathcal{A}$ has an inverse, since it does not use any measurement. The effectiveness of the operator $\mathcal{Q}$ lies in its straightforward impact on the subspace $\mathcal{H}_\Psi$, spanned by the vectors $|\Psi_{1i}\rangle$ and $|\Psi_{0i}\rangle$. In essence, $\mathcal{Q}$ operator serves as a mechanism to iteratively enhance the probability amplitudes associated with the desired subspace, akin to how classical algorithms repeat to boost success probabilities.

$$\mathcal{Q}|\Psi_1\rangle = (1-2a)|\Psi_1\rangle - 2a|\Psi_0\rangle \tag{3.6}$$
$$\mathcal{Q}|\Psi_1\rangle = 2(1-a)|\Psi_1\rangle + (1-2a)|\Psi_0\rangle \tag{3.7}$$

Let's now consider the orthogonal complement $\mathcal{H}_\Psi^\perp$ of $\mathcal{H}_\Psi$ within $\mathcal{H}$. Given that the operator $\mathcal{A}\mathcal{S}_0\mathcal{A}^{-1}$ acts as the identity on $\mathcal{H}_\Psi^\perp$, the operator $\mathcal{Q}$ behaves as $\mathcal{S}_{\Psi_0}$ within $\mathcal{H}_\Psi^\perp$. Consequently, $\mathcal{Q}^2$ functions as the identity on $\mathcal{H}_\Psi^\perp$, and every eigenvector of $\mathcal{Q}$ in $\mathcal{H}_\Psi^\perp$ possesses eigenvalues of either $+1$ or -1. Consequently, comprehending the impact of $\mathcal{Q}$ on an arbitrary initial vector $|\Upsilon\rangle$ within $\mathcal{H}$ merely requires assessing its effect on the projection of $|\Upsilon\rangle$ onto $\mathcal{H}_\Psi$.

Since the operator $\mathcal{Q}$ is unitary, it features an orthonormal basis comprising two eigenvectors of $\mathcal{Q}$:

$$|\Psi_{\pm}\rangle = \frac{1}{2}(\frac{1}{\sqrt{a}}|\Psi_1\rangle \pm \frac{\imath}{\sqrt{1-a}}|\Psi_0\rangle) \tag{3.8}$$

given $0 < a < 1$ and $\imath = \sqrt{-1}$. The corresponding eigenvalues are as follows:

$$\lambda_{\pm} = \exp^{\pm \imath 2\theta_a} \tag{3.9}$$

in this formula the θ_a angle is defined as follow:

$$\sin^2(\theta_a) = a = \langle\Psi_1|\Psi_1\rangle \tag{3.10}$$

where $0 \leq \theta_a \leq \pi/2$.

We can now express $|\Psi\rangle = \mathcal{A}|0\rangle$ in terms of the eigenvector basis, something that will result significantly useful later.

$$\mathcal{A}|0\rangle = |\Psi\rangle = \frac{-\imath}{\sqrt{2}}(\exp^{-\imath\theta_a}|\Psi_+\rangle - \exp^{-\imath\theta_a}|\Psi_-\rangle) \tag{3.11}$$

3.1.2 Adding Quantum Phase Estimation

The standard QAE uses Quantum Phase Estimation (QPE), which is a pivotal algorithm in quantum computing, integral to estimating the phase of a unitary operator acting on a quantum state. It leverages the unique ability of quantum computers to manipulate superpositions of states. By utilizing quantum gates, QPE estimates the phase, providing a quantum representation that can be translated into a classical one. This algorithm finds significance in various quantum computational tasks, including integer factorization and optimization problems. QPE's essence lies in its capability to extract phase information efficiently, underscoring its importance in advancing quantum algorithms and applications.

To estimate a, we make good use of the properties of the operator $\mathcal{Q} = \mathcal{Q}(\mathcal{A}, \psi_0) = \mathcal{A}\mathcal{S}_0\mathcal{A}^{\dagger}\mathcal{S}_{\psi_0}$. The amplitudes of $|\Psi_1\rangle$ and $|\Psi_0\rangle$ as functions of the number of applications of $\mathcal{Q}$ (which are also called *quantum samples* or oracle queries), are sinusoidal functions, both of period π/θ_a. Recall that a = $\sin^2(\theta_a)$, and thus an estimate for θ_a also gives an estimate for a.

To estimate this period, we use Fourier analysis like Shor does for a classical function in his factoring algorithm [24]. This approach can also be viewed as an eigenvalue estimation and is best analyzed based on the eigenvectors of the operator at hand. As explained above, the eigenvalues of $\mathcal{Q}$ on the subspace spanned by $|\Psi_1\rangle$ and $|\Psi_0\rangle$ are $\lambda_+ = e^{i2\theta_a}$ and $\lambda_- = e^{-i2\theta_a}$. Thus we can estimate a simply

by estimating one of these two eigenvalues. Errors in the estimation of $\tilde{\theta_a}$ for θ_a translate into errors in the estimation of $\tilde{a} = \sin^2(\tilde{\theta_a})$ for a.

In this case, m ancilla qubits are used, after being set in equal superposition through Hadamard Gates (H), to represent the outcome, it establishes the number of quantum samples as $M = 2^m$ and employs geometrically increasing powers of $\mathcal{Q}$ controlled by the ancillas. After applying $F^\dagger$, an inverse Quantum Fourier Transform (QFT), their state is measured, yielding an integer $y \in \{0, \ldots, M-1\}$. This integer is then mapped to an angle $\widetilde{\theta}_a = y\pi/M$ that results in a evaluation of a:

$$\tilde{a} = \sin^2\left(\widetilde{\theta}_a\right)^2 \in [0, 1]. \tag{3.12}$$

The complete circuit for Amplitude Estimation is illustrated in Figure 3.1. The estimator $\tilde{a}$, with a probability of at least $8/\pi^2$, meets the condition:

$$|a - \tilde{a}| \leq \frac{2\pi\sqrt{a(1-a)}}{M} + \frac{\pi^2}{M^2} = \mathcal{O}(M^{-1}) \tag{3.13}$$

Where $\mathcal{O}(M^{-1})$ is ϵ, which is the estimation error. The probability of success can be significantly enhanced, nearing 100%, through iterative repetitions of the process, and by employing the median estimate. This denotes a quadratic enhancement compared to the convergence rate of classical Monte Carlo methods, which typically follows $\mathcal{O}(\sqrt{M})$. These estimates, denoted as $\tilde{a}$, are constrained within the grid $\left\{\sin^2\left(\frac{y\pi}{M}\right) : y = 0, \ldots, \frac{M}{2}\right\}$ across the potential measurement outcomes of y. Alternatively, it is feasible to employ Maximum Likelihood Estimation (MLE) on the observations for y. For a given θ_a, the probability of observing $|y_i\rangle$ when measuring the ancilla qubits is given by:

$$\mathbb{P}[|y\rangle] = \frac{\sin^2(M\Delta\pi)}{M^2\sin^2(\Delta\pi)} \tag{3.14}$$

Here, Δ represents the minimum distance on the unit circle between the angles θ_a and $\pi\widetilde{y}/M$, where $\widetilde{y}$ is defined as $\widetilde{y} = y$, and $y \leq M/2$ and $\widetilde{y} = M/2 - y$. By utilizing a set of y-measurements, this can be leveraged within a MLE framework to derive an estimate of θ_a that is not confined to discrete grid points. Moreover, it facilitates the utilization of the likelihood ratio method to establish confidence intervals. In these evaluations, likelihood ratio-based confidence intervals consistently demonstrated superior reliability compared to alternative methods, such as the observed Fisher information. Consequently, we refer to the canonical QAE incorporating MLE

on y-measurements for enhanced estimate precision and likelihood ratio-based confidence intervals. [25]

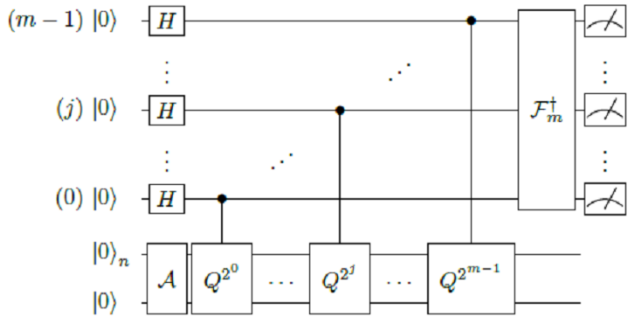

Figure 3.1: Quantum Circuit for Amplitude Estimation [22]

QAE without QPE

In recent years, a method to perform QAE without relying on quantum phase estimation has been proposed, thereby reducing the required number of qubits and circuit depth while maintaining a quadratic speed-up [26]. We can use the expression:

$$\mathcal{Q}^k\mathcal{A}\left|0\right\rangle_n\left|0\right\rangle = cos((2k+1)\theta_a)\left|\psi_0\right\rangle_n\left|0\right\rangle + sin((2k+1)\theta_a)\left|\psi_1\right\rangle_n\left|1\right\rangle, \tag{3.15}$$

where $\mathcal{Q}^k\mathcal{A}\left|0\right\rangle$ denotes the state resulting from applying the operator $\mathcal{Q}^k\mathcal{A}$ to the initial state $\left|0\right\rangle\left|0\right\rangle$. By measuring $\mathcal{Q}^k\mathcal{A}\left|0\right\rangle$ for $k = 2^0, \ldots, 2^{m-1}$ for a given value of m and employing maximum likelihood estimation, an approximation for θ_a (and thus a) can be recovered. If we define $M = 2^m - 1$ as the total number of $\mathcal{Q}$-applications and consider N shots for each experiment, empirical evidence shows that the resulting estimation error scales as $\mathcal{O}(1/(M\sqrt{N}))$, achieving quadratic speed-up in terms of M. This idea is the basis of the methods described in Section 4.1.

QAE for the pricing of options

In the context of the options contracts under consideration, the random variables represent the potential values S_i that the underlying asset can assume, along with the corresponding probabilities p_i of those values being realized. For an option with payoff f, the $\mathcal{A}$ operator generates the state.

$$\sum_{i=0}^{2^n-1}\sqrt{1-f(S_i)}\sqrt{p_i}\left|S_i\right\rangle\left|0\right\rangle + \sum_{i=0}^{2^n-1}\sqrt{f(S_i)}\sqrt{p_i}\left|S_i\right\rangle\left|1\right\rangle \tag{3.16}$$

By comparing this with Equation 3.1, we observe that

$$a = \sum_{i=0}^{2^n-1} f(S_i)p_i = \mathbb{E}[f(\mathbf{S})], \tag{3.17}$$

This implies that Amplitude Estimation (AE) enables the computation of the undiscounted price of an option, provided there exists a method to represent the option's payoff as a quantum circuit and generate the state described by Equation 3.16. We will delineate then the essential components required to accomplish this task.

3.2 Distribution loading

The initial component of our option pricing model involves a circuit tasked with taking a probability distribution implied for potential future asset prices and transferring it into a quantum register. This process ensures that each basis state within the register denotes a feasible asset price, while its amplitude signifies the associated probability. In simpler terms, given an n-qubit register and sets of asset prices $\{S_i\}$ for i in the range of $\{0, \ldots, 2^n - 1\}$ alongside their respective probabilities $\{p_i\}$, the distribution loading module generates the following state:

$$|\psi\rangle_n = \sum_{i=0}^{2^n-1} \sqrt{p_i}\,|S_i\rangle_n\,. \tag{3.18}$$

The analytical formulas employed for pricing options in the Black-Scholes-Merton (BSM) model rely on the assumption that the underlying stock prices at maturity adhere to a log-normal distribution with constant volatility. Log-concave probability distributions, such as the log-normal distribution utilized in the BSM model, can be effectively loaded onto gate-based quantum computers [27]. The process of loading the pertinent probability distributions onto quantum registers does not necessitate excessively complex procedures.

However, the simplified assumptions made in the BSM model often fail to encompass crucial market dynamics, thereby restricting the model's relevance in real-world scenarios. Consequently, accurately estimating the intrinsic value of option contracts necessitates the proper capture of the market-implied probability distribution of the underlying asset. To tackle these limitations, Artificial Neural Networks (ANN) are gaining traction as a method to capture the authentic dynamics of relevant market parameters without relying on simplified underlying models. Thus, it becomes crucial to efficiently represent distributions of financial parameters on a quantum computer, especially those lacking explicit analytical representations.

The process of loading arbitrary states into quantum systems typically demands an exponentially large number of gates, rendering it inefficient to model arbitrary distributions as quantum gates. However, since the distributions of interest often exhibit a special structure, this limitation can be mitigated by employing quantum Generative Adversarial Networks (qGAN) [28]. These networks enable us to load a distribution using a polynomial number of gates. A qGAN can effectively learn the underlying random distribution X from the observed data samples $\{x^0, \ldots, x^{k-1}\}$ and directly encode it into a quantum state. This generative model involves the interaction of a discriminator, which is typically a neural network, and a quantum generator represented by a parametrized quantum circuit. Specifically, the training of a qGAN involves alternating optimization steps for the discriminator's parameters and the generator's parameters θ. Following the training process, the output of the generator produces a quantum state.

$$|\psi(\theta)\rangle_n = \sum_{i=0}^{2^n-1} \sqrt{p_i(\theta)}\, |i\rangle_n \,, \tag{3.19}$$

that represents the target distribution. The n-qubit state $|i\rangle = |i_{n-1}, \ldots, i_0\rangle$ encodes the integer $i = 2^{n-1}i_{n-1} + \ldots + 2i_1 + i_0 \in \{0, \ldots, 2^n - 1\}$ with $i_k \in \{0, 1\}$ for $k = 0, \ldots, n-1$. The probabilities $p_i(\theta)$ approximate the random distribution underlying the training data. It's important to note that the outcomes of a random variable X can be mapped to the integer set $\{0, \ldots, 2^n - 1\}$ using an affine mapping. Moreover, this approach can be straightforwardly extended to handle multivariate data, where a separate register of qubits is employed for each dimension. Another advantageous aspect of using qGANs for loading probability distributions is the flexibility to tailor the qGAN variational form to construct circuits with short depths while maintaining an acceptable level of accuracy. This capability enables the evaluation of the performance of option pricing quantum circuits on near-term quantum hardware, where resources are still limited. While the use of Artificial Neural Networks (ANNs) to represent probability distributions incurs a training cost in both classical and quantum models, this cost can be efficiently amortized during common business practices such as overnight risk assessment of large portfolios comprising millions of option contracts. It's also worth noting that qGAN training tends to perform better when the initial distribution closely resembles the target distribution. Therefore, as new market data becomes available and needs to be incorporated into the probability distribution (e.g., for overnight risk assessment, where many of the same options as the previous day need to be priced but with an additional day's worth of market data), the previously trained qGAN can be utilized as the initial distribution, leading to faster convergence.

3.3 Computing Payoff

We derive the expectation value of a linear function f of a random variable X using QAE by constructing the operator $\mathcal{A}$, such that $a = \mathbb{E}[f(X)]$, as illustrated in Equation 3.17. Once $\mathcal{A}$ is implemented, we proceed to prepare the state outlined in Equation 3.1 and define the $\mathcal{Q}$ operator, which relies solely on $\mathcal{A}$ and generic quantum operations. In this section, we present the procedure for creating a close counterpart of $\mathcal{A}$ and demonstrate how to integrate it with AE to compute the expectation value of f. The payoff function for the option contracts under consideration exhibits a piece-wise linear structure. Consequently, we only need to consider linear functions $f : \{0, \ldots, 2^n - 1\} \to [0, 1]$, which can be expressed as $f(i) = f_1 i + f_0$. We can effectively construct an operator that performs.

$$|i\rangle_n |0\rangle \to |i\rangle_n (cos[f(i)] |0\rangle + sin[f(i)] |1\rangle) \tag{3.20}$$

Controlled Y-rotations are utilized for implementing the linear term of $f(i)$. In this approach, each qubit j (where $j \in \{0, \ldots, n-1\}$) in the $|i\rangle_n$ register serves as a control for a Y-rotation with an angle of $2^j f_1$ applied to the ancilla qubit. Meanwhile, the constant term f_0 is implemented by rotating the ancilla qubit without any controls, as illustrated in Figure 3.2. These controlled Y-rotations can be realized using CNOT and single-qubit gates.

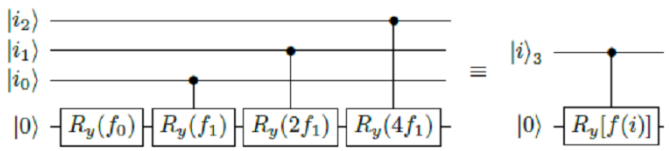

Figure 3.2: Quantum circuit that creates the state in Eq. 3.20.

We proceed to outline the method for computing $\mathbb{E}[f(X)]$ for a linear function f of a random variable X, which is associated with integer values $i \in \{0, \ldots, 2^n - 1\}$ each occurring with a probability of p_i. To accomplish this, we employ the procedure detailed previously to generate the operator responsible for mapping $\sum_i \sqrt{p_i} |i\rangle_n |0\rangle$ to

$$\sum_{i=0}^{2^n-1} \sqrt{p_i} |i\rangle_n [cos(c\tilde{f}(i) + \frac{\pi}{4}) |0\rangle + sin(c\tilde{f}(i) + \frac{\pi}{4}) |1\rangle] \tag{3.21}$$

and $\tilde{f}(i)$ is a scaled version of $f(i)$

$$\tilde{f}(i) = 2\frac{f(i) - f_{min}}{f_{max} - f_{min}} - 1, \tag{3.22}$$

With $f_{min} = min_i f(i)$ and $f_{max} = max_i f(i)$, and $c \in [0, 1]$ being an additional scaling parameter, the relation in Equation 3.22 is selected to ensure that $\tilde{f}(i) \in [-1, 1]$. Consequently, $sin^2[c\tilde{f}(i) + \pi/4]$ is an anti-symmetric function around $f(i) = 0$. With these definitions, the probability of finding the ancilla qubit in the state $|1\rangle_i$, namely

$$P_1 = \sum_{i=0}^{2^n-1} p_i sin^2\left(c\tilde{f}(i) + \frac{\pi}{4}\right), \tag{3.23}$$

approximated, for small values of $c\tilde{f}(i)$, by

$$P_1 \approx \sum_{i=0}^{2^n-1} p_i \left(c\tilde{f}(i) + \frac{1}{2}\right) = c\frac{2\mathbb{E}[f(X)] - f_{min}}{f_{max} - f_{min}} - c + \frac{1}{2}. \tag{3.24}$$

To obtain this result we made use of the approximation

$$sin^2\left(c\tilde{f}(i) + \frac{\pi}{4}\right) = c\tilde{f}(i) + \frac{1}{2} + \mathcal{O}(c^3\tilde{f}^3(i)). \tag{3.25}$$

With this first-order approximation, the convergence rate of AE is $\mathcal{O}(M^{-2/3})$ when c is appropriately chosen, it is already faster than classical Monte Carlo methods. We can restore the $\mathcal{O}(M^{-1})$ convergence rate of AE by utilizing higher orders implemented with quantum arithmetic. However, the resulting circuits entail more gates. This trade-off also provides a formula specifying the optimal value of c to minimize the estimation error incurred when using AE. From Equation 3.25, we can recover $\mathbb{E}[f(X)]$ since AE enables us to efficiently retrieve P_1 and because we possess the values of f_{min}, f_{max}, and c.

Chapter 4

Experiments and Results

This chapter illustrates the experimental part of this thesis. It has been shown how Amplitude Estimation (AE) is fundamental in the pricing of options using quantum computers, now we will analyze how the different methods to implement the amplitude estimation impact the computation of the payoff in terms both of precision and quantum resources required. In this experimental phase, three of the most relevant implementations have been analyzed:

- the Iterative Quantum Amplitude Estimation,
- the Maximum Likelihood Quantum Amplitude Estimation
- and the Faster Quantum Amplitude Estimation.

The goal of this experiment is to determine which one fits best the problem of computing the payoff of an option. To implement a model able to perform the following computations, the utilization of the Qiskit library was fundamental. Qiskit enables regular users to develop, simulate, and execute quantum applications across different platforms, providing a wide range of tools and functionalities for quantum programming and experimentation [11].

4.1 QAE implementations

QAE has demonstrated superiority over traditional Monte Carlo methods in numerical integration, particularly when tackling high-dimensional integration challenges. Leveraging QAE enables the estimation of integrals with comparable precision while extracting fewer samples, thereby substantially curtailing the computational resources demanded for numerical integration. The efficacy of Monte Carlo methods is inherently constrained by their reliance on the number of samples employed in the simulation for approximation accuracy. This methodology grapples with the

challenge of producing robust sampling from an arbitrary probability density function. Consequently, Monte Carlo methods often entail significant computational overhead for achieving high-precision integration, rendering them less expedient for specific applications.

Recent efforts aim to simplify its quantum counterpart (QAE) by eliminating its dependence on QPE, with approaches such as Maximum Likelihood Quantum Amplitude Estimation (MLQAE) and Iterative QAE (IQAE) showing promise. While these variants offer theoretical advantages, their practical feasibility and performance warrant empirical validation and comparison.

4.1.1 Iterative Quantum Amplitude Estimation

For IQAE, a quantum computer is used to approximate $P[|1\rangle] = \sin^2((2k+1)\theta_a)$ for the last qubit in $Q_k |0\rangle \langle 0|$, with $|0\rangle$ for different powers k. The rationale behind IQAE is formally presented in Algorithm 2 in Appendix A while its main subroutine, the one to compute the next value k, is delineated in Algorithm 1.

Given a confidence interval $[\theta_l, \theta_u] \subseteq [0, \pi/2]$ for θ_a and a power k of $\mathcal{Q}$, along with an estimate for $\sin^2((2k+1)\theta_a)$, we exploit the trigonometric identity $\sin^2(x) = (1 - \cos(2x))/2$ to translate our estimates for $\sin^2((2k+1)\theta_a)$ into estimates for $\cos((4k+2)\theta_a)$. However, unlike in Kitaev's Iterative QPE, we cannot estimate the sine alone, and the cosine alone is only invertible without ambiguity if we know the argument is restricted to either $[0, \pi]$ or $[\pi, 2\pi]$, i.e., the upper or lower half-plane. Thus, the aim is to find the largest k such that the scaled interval $[(4k+2)\theta_l, (4k+2)\theta_u]_{mod 2\pi}$ is fully contained either in $[0, \pi]$ or $[\pi, 2\pi]$. If this condition is met, we can invert $\cos((4k+2)\theta_a)$ and enhance our estimate for θ_a with high confidence [25]. This implies an upper bound of k, and the core of the algorithm lies in the procedure used to find the next k given $[\theta_l, \theta_u]$, as formally introduced in Algorithm 1.

In essence, the key aspect of the algorithm - the FindNextK subroutine - enables to maximize Fisher Information I on a given iteration in a greedy manner. This is evident from the observation that any summand of I is proportional to $N_{\text{shots}} K^2$, where $K := 4k + 2$ [25].

4.1.2 Maximum Likelihood Quantum Amplitude Estimation

MLQAE consists of a Grover-like circuit and a Maximum Likelihood estimation on a classical processor. The fundamental concept behind this algorithm entails crafting a likelihood function derived from measurements conducted on multiple quantum states, each subjected to the amplitude amplification process. While the probability of measuring favorable or unfavorable states hinges on the number

of amplitude amplification operations, it is noteworthy that the outcomes are interrelated, as each amplified probability is contingent upon a singular parameter [26].

The MLQAE algorithm approximates $P[|1\rangle]$ for powers k of $\mathcal{Q}$, where $k = 2^j$ and $j = 0, 1, 2, \ldots, m-1$, for a given m, using N_{shots} measurements from a quantum computer for each j. In total, $\mathcal{Q}$ is applied $N_{\text{shots}}(M-1)$ times, where $M = 2^m$. Previous studies have demonstrated that the corresponding Fisher information scales as $O(N_{\text{shots}}M^2)$, implying a lower bound of the estimation error scaling as $\Omega(1/(\sqrt{N_{\text{shots}}M}))$. Confidence intervals can be derived from the measurements using methods such as the likelihood ratio approach.

4.1.3 Faster Quantum Amplitude Estimation

The faster quantum amplitude estimation is an algorithm without phase estimation that reaches the Heisenberg scaling and the constant factor proportional to $1/\epsilon \ln 1/\epsilon$. Despite the algorithm's ability to diminish circuit depth compared to the quantum phase estimation algorithm, the necessary depth still scales as $O(1/\epsilon)$, and noise effects cannot be disregarded. Therefore, investigating methods to mitigate noise within the algorithm becomes crucial for evaluating its practicality, a task that remains open for future endeavors [29].

Similar to previous literature, we employ the quantum amplitude amplification technique 3.1.1. This technique enables us to estimate the values of $\cos(2(2m+1)\theta)$ for each non-negative integer m directly through measurements, with resulting estimation errors of $\cos(2(2m+1)\theta)$ being of the order $O(\sqrt{s})$ with high probability when s measurements are executed for each m.

The algorithm iteratively estimates the values of $\cos(2(2m+1)\theta)$ for each $m = 2^{j-1}$ $(j = 1, 2, \ldots, l)$. Similar to Kitaev's iterative phase estimation, if $2(2^j+1)\theta_{mod2\pi}$ are estimated and the estimation errors are within approximately $\pi/2$, then the value of θ can be iteratively estimated with an error of $O(1/2^l)$, achieving Heisenberg scaling. However, determining $2(2^j+1)\theta|_{mod2\pi}$ solely from the estimate of $\cos(2(2^j+1)\theta)$ is ambiguous because it is uncertain whether $2(2^j+1)\theta|_{mod2\pi}$ lies in $[0, \pi]$ or $[\pi, 2\pi]$.

To resolve this ambiguity, the algorithm employs a two-stage method. In the first stage, when $2(2^j+1)\theta < \pi$, $2(2^j+1)\theta|_{mod2\pi}$ can be obtained unambiguously from the estimate of $\cos(2(2^j+1)\theta)$ using the inverse cosine function. In the second stage, when the estimate of $2(2^{j_0}+1)\theta$ is approximately $\pi/2$ at iteration $j = j_0$, $2(2^j+1)\theta$ might be larger than π, hence $2(2^j+1)\theta|_{mod2\pi}$ cannot be determined solely from the measurements of $\cos(2(2^j+1)\theta)$. However, by combining measurements of $\cos(2(2^j+2^{j_0}+1)\theta)$ with those of $\cos(2(2^j+1)\theta)$, the value of $\sin(2(2^j+1)\theta)$ can be estimated using the trigonometric addition formula, enabling determination of $2(2^j+1)\theta|_{mod2\pi}$ without ambiguity. As a result, the algorithm can estimate the

value of θ with an error less than $O(1/2^l)$ [29].

4.2 The setting of the experiment

For this experiment we are going to take into consideration a standard European Call Option, with a strike price K, a stock price that at maturity is equal to S_T, which is given by a random distribution. The payoff is given by:

$$max(S_T - K; 0) \tag{4.1}$$

The quantum model is constructed to estimate the expected payoff, through usage of Amplitude Estimation.

$$\mathbb{E}[max(S_T - K; 0)] \tag{4.2}$$

4.2.1 Uncertainty model

The first step to set the ground for the experiment is the definition of certain fixed variables. The first one is a certain number of qubits which represents the uncertainty in the price of the underlying asset. The higher the number of qubits, the higher the accuracy in the representation of uncertainty. In this case, the number of qubits is equal to 3. Then all the fixed variables already present in all the standard models used to price options and fundamental to construct the random distribution: the initial stock price, the volatility (which measures the dispersion of the returns), the annual interest rate, and the days to maturity, considering an annual base.

Variables	**Values**
Number of qubits that represents uncertainty	3
Stock price (S)	2.0
Volatility (vol)	0.4
Annual interest rate (r)	0.05
Days to maturity (T)	40/365

Table 4.1: Uncertainty model fixed variables

The parameters for the log-normal distribution are subsequently defined: the mean, the standard deviation, and all the values correlated to them, for example,

the variance. To load this distribution on a quantum circuit, the lowest and highest values are set, considering in the middle an equidistant discretization.

The last step of this part is to build up the $\mathcal{A}$ for the Amplitude Estimation, which is the fundamental element to compute the payoff. The fixed number of qubits in the constructed circuit reflects the inherent uncertainty of the model. Utilizing this predetermined number of qubits along with the previously computed value of the log-normal distribution, the circuit is systematically constructed.

4.2.2 Payoff computation

The payoff function, as defined before, is represented by 0 for each value of S_T for which is valid the inequality $S_T \leq K$, and then increases linearly. To represent and control the linearity is used a comparator, that flips an ancilla qubit from $|0\rangle$ to $|1\rangle$.

This is the approximation of the linear part [30]:

$$\sin^2(y + \frac{\pi}{4}) \approx y + \frac{1}{2} \quad \text{for small} \quad |y| \tag{4.3}$$

Hence, when considering a particular scaling factor for approximation, denoted as $c_{approx} \in [0,1]$, and $x \in [0,1]$, we contemplate the following:

$$\sin^2(\frac{\pi}{2} * c_{approx} * (x - \frac{1}{2}) + \frac{\pi}{4}) \approx \frac{\pi}{2} * c_{approx} * (x - \frac{1}{2}) + \frac{1}{2}, \text{ for small } c_{approx} \tag{4.4}$$

We can readily devise an operator that functions as:

$$|x\rangle |0\rangle \rightarrowtail |x\rangle (\cos(a * x + b) |0\rangle + \sin(a * x + b) |1\rangle) \tag{4.5}$$

using controlled Y-rotations. The concern lies in determining the likelihood of measuring $|1\rangle$ in the final qubit, which corresponds to $\sin(a * x + b)$. Coupled with the aforementioned approximation, this facilitates the approximation of the values of interest. The smaller the choice of c_{approx}, the more precise the approximation becomes. This is estimating a property scaled by c_{approx} so the number of qubits m needed for the evaluation, must be adapted consequently. The fixed variables necessary for this part are the strike price and the approximation scaling c_{approx} for the payoff function.

Variables	**Values**
Strike price (K)	1.896
Approximation (c_{approx})	0.25

Table 4.2: Payoff fixed variables

Then we set up the breakpoints, slopes, offsets, f_{min} and f_{max} that are used to construct the $\mathcal{A}$ operator for the Amplitude Estimation and define the number of qubits of this circuit. Finally, the exact expected value is estimated through the classical Monte Carlo simulation.

4.2.3 Expected payoff computation

The computation of the expected payoff is the most important part of the experiment, because it is the one, with the delta evaluation, in which the different kinds of amplitude estimation are implemented. The fixed variables defined for this problem are the epsilon ϵ, which represents the maximum error accepted, and alpha α, which determines the confidence interval of our model.

In this case, to construct the circuit representing the problem, the Qiskit Finance module called *EuropeanCallPricing* was used [31], giving it the following parameters: the number of qubits that express the uncertainty, the strike price, the rescaling factor, which regulates the interval of uncertainty to maximize the efficiency of the algorithm, the bounds, which define the interval in which the price of the underlying asset is going to fluctuate, and the uncertainty model used to modulate the behavior of the asset over time.

The goal of this experiment is to implement different kinds of Amplitude Estimation with different characteristics and try to determine which is the best one for this kind of problem analyzing how the results change, changing the way they are obtained.

IQAE is the most commonly used. Its computation requires the following parameters: *epsilon target* which is the target precision of the algorithm and *alpha* which determines the confidence interval.

Moreover, the number of shots represents the number of times the circuit is run to obtain the statistic of the results, and a seed is used to generate random numbers during the execution of the program, useful to guarantee that the results of the simulation are deterministic.

MLQAE instead, to be implemented needs an evaluation schedule, that defines the number of evaluations to be done during the execution of the algorithm.

For his part, the Faster Quantum Amplitude Estimation (FQAE) needs the following parameters to be implemented: Δ, which represents the maximum allowed difference between the real amplitude estimation and the estimated one, and the maximum number of iterations that are permitted during the execution of the algorithm.

4.2.4 Delta Evaluation

The delta of a call option is a derived measure used in finance to quantify the sensitivity of the option's price to changes in the price of the underlying asset and for assessing the associated trading risks and opportunities. It is defined as the change in the option price relative to a unit change in the price of the underlying asset.

For this kind of option, the delta ranges between 0 and 1. A delta equal to 0 indicates that the option has little or no intrinsic value when the price of the underlying asset is near the strike price, while a delta of 1 indicates that the option perfectly tracks the price of the underlying asset, and its value is directly linked to the price of the underlying asset.

The delta measures the probability that a call option will be in-the-money at expiration, which means the option has a positive intrinsic value. For instance, a delta of 0.6 indicates a 60% probability that the option will be in the money at expiration, assuming all other variables remain constant.

The Delta evaluation is constructed using a comparator circuit and an ancilla qubit to identify the cases in which the $S_T > K$, that are the cases in which the call option is exercised. Thanks to the environment chosen to develop the experiment, it is possible to use the library that contains the *EuropeanCallDelta* class [32] to define the quantum circuit associated. After the construction of the circuit and the setting of *epsilon* and *alpha*, the amplitude estimation implementation to estimate delta is defined. As in the case of the computation of the payoff, in this experiment was analyzed how the three different kinds of QAE behave.

4.3 Results

The "exact" (baseline) values obtained using the uncertainty model defined before and the classical MC simulation[1] are:

Table 4.3: EXACT VALUES

Exact Expected Value	Exact Delta Value
0.1688	0.8098

To analyze the methods more effectively, it was decided to set the total program executions to 50 runs for each method. To be precise, since the number of shots for each run is equal to 100, this entails analyzing a sample of 5000 executions.

[1] 2 million classical samples were taken

For the circuit that is used to solve the payoff problem, the number of qubits is equal to 7, instead for the one solving the delta value the number of qubit used is 6. These values remain the same for the method and the different situations analyzed.

Table 4.4: ITERATIVE method results: expected payoff calculated with 100 shots and 50 runs

	Estimated Value	ΔEV	$\%\ \Delta EV$	Depth of the circuit
Average	0.1654	0.0066	4.07%	4008
St.Deviation	$7.846\ 10^{-3}$	$5.193\ 10^{-3}$	$3.200\ 10^{-2}$	522

Table 4.5: MAXIMUM LIKELIHOOD method results: expected payoff calculated with 100 shots and 50 runs

	Estimated Value	ΔEV	$\%\Delta EV$	Depth of the circuit
Average	0.1684	0.0069	4.24%	5171
St.Deviation	$4.126\ 10^{-3}$	$2.521\ 10^{-3}$	$1.553\ 10^{-2}$	0

Table 4.6: FASTER method results: expected payoff calculated with 100 shots and 50 runs

	Estimated Value	ΔEV	$\%\Delta EV$	Depth of the circuit
Average	0.1687	0.0065	4.02%	4545
St.Deviation	$3.852\ 10^{-3}$	$3.634\ 10^{-3}$	$2.239\ 10^{-2}$	0

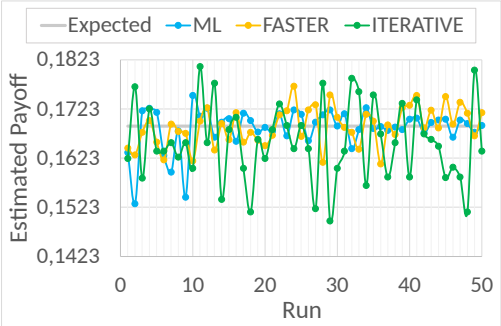

Figure 4.1: Estimated Payoff run sequence - scenario with 100 shots and 50 runs

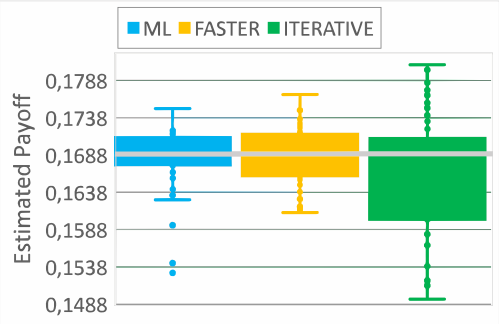

Figure 4.2: Estimated Payoff box aggregation - scenario with 100 shots and 50 runs

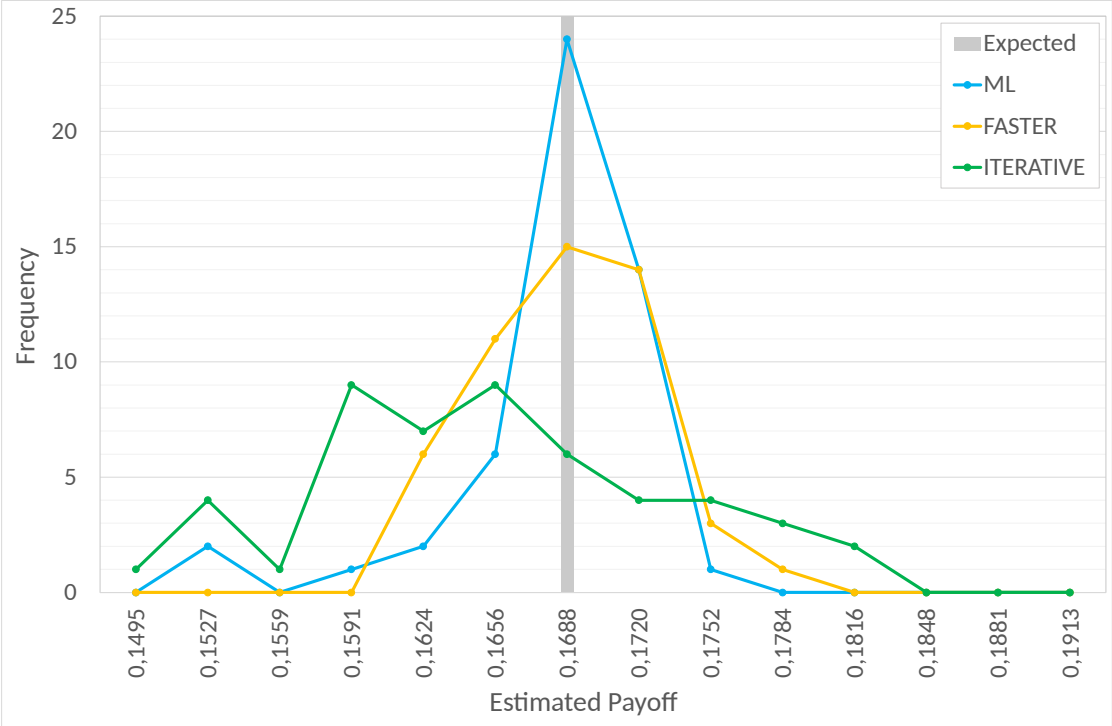

Figure 4.3: Estimated Payoff distribution - scenario with 100 shots and 50 runs

As discerned from the graph in Figure 4.3, which has on the x-axes the estimated payoff and on the y-axes the frequency, the ML and faster methods manifest a mean value that most closely converges with the estimate derived from the classical Monte Carlo simulation method, in contrast to the other methodology (IQAE) that slightly deviates from the exact defined value, which serves as a reference. Regarding standard deviation and variance, the iterative method yields a variance

sufficient to encompass the exact value within the estimated values, albeit notably high as we can clearly see in Figure 4.2, leading to results that diverge from the exact value. On the other hand, the ML and faster methods, manage to encompass the estimated value obtained from the Monte Carlo simulation but with a lower variance than the iterative case, resulting in the ML method being the best one if we just consider the estimated payoff accuracy and the variance associated.

In Figure 4.1, the graph distinctly highlights how the values attained through the iterative method oscillate considerably more, deviating from the mean value.

Conversely, in the case of the ML and faster methods, a substantially lesser oscillation is discernible, corroborated by the values of their standard deviations. Here, the iterative method exhibits a standard deviation practically double that of the faster method. The differences between the values obtained with MC and those obtained with the different QAE methods are approximately the same for all three methods, but in the case of the iterative method, there is a higher variance.

From Table 4.4, it becomes apparent that in the iterative method, the average depth remains consistently lower compared to the other three methods employed. However, its standard deviation is significant with a value of 522 , indicating a greater variability and uncertainty relative to the mean value. Despite the ML method being the best in terms of performance, since it possesses the highest depth value, it turns out to be the worst in terms of scalability.

Table 4.7: ITERATIVE method results: delta value calculated with 100 shots and 50 runs

	Estimated Value	ΔEV	$\%\ \Delta EV$	Depth of the circuit
Average	0.8084	0.0051	0.62%	1461
St.Deviation	$6.844\ 10^{-3}$	$4.768\ 10^{-3}$	$5.888\ 10^{-3}$	195

Table 4.8: MAXIMUM LIKELIHOOD method results: delta value calculated with 100 shots and 50 runs

	Estimated Value	ΔEV	$\%\Delta EV$	Depth of the circuit
Average	0.8088	0.0018	0.23%	1901
St.Deviation	$2.134\ 10^{-3}$	$1.450\ 10^{-3}$	$1.790\ 10^{-3}$	0

Table 4.9: FASTER method results: delta value calculated with 100 shots and 50 runs

	Estimated Value	ΔEV	$\%\Delta EV$	Depth of the circuit
Average	0.8097	0.0024	0.29%	1901
St.Deviation	$2.908\ 10^{-3}$	$1.650\ 10^{-3}$	$2.037\ 10^{-3}$	0

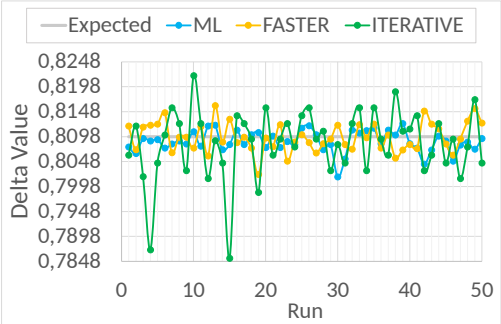

Figure 4.4: Estimated Delta run sequence - scenario with 100 shots and 50 runs

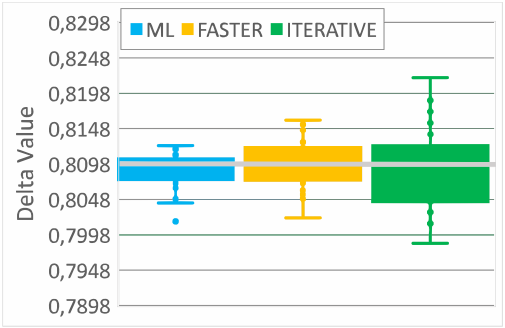

Figure 4.5: Estimated Delta box aggregation - scenario with 100 shots and 50 runs

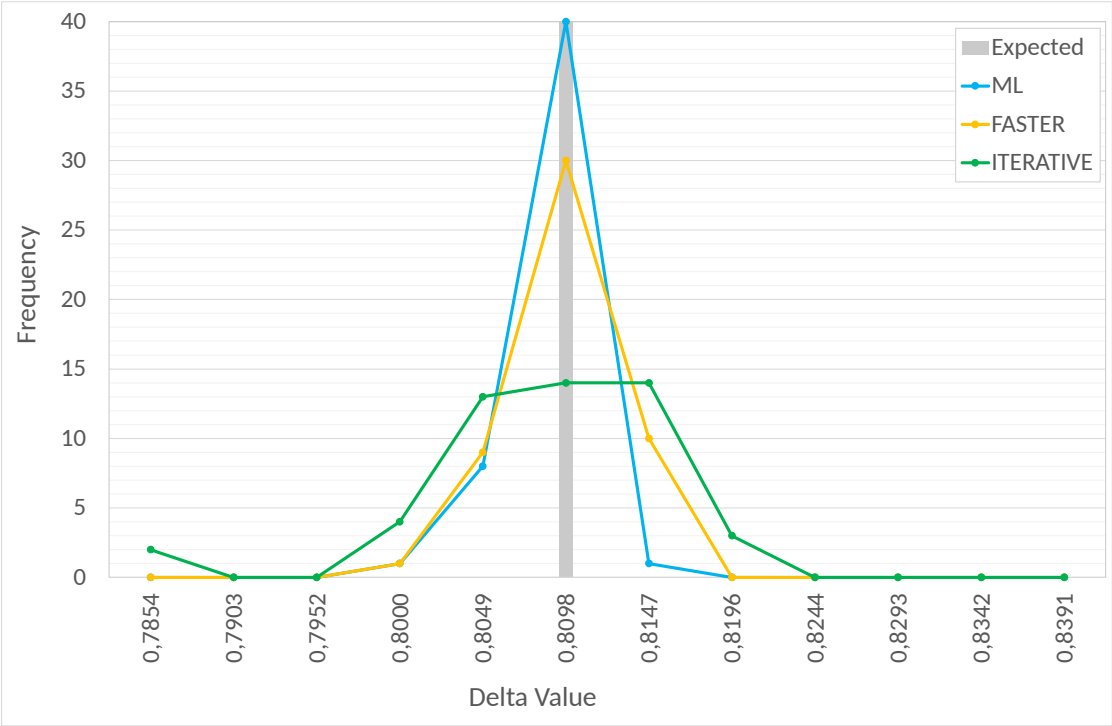

Figure 4.6: Estimated Delta distribution - scenario with 100 shots and 50 runs

Regarding the delta estimates, all three mean values calculated are very close to what we defined as "exact", this is evident in the estimated delta distribution graph in Figure 4.6. However, the one computed with the faster method turns out to be the closest to it. Taking into account the standard deviation, ML method emerges as the one with the lowest standard deviation, while the iterative method, once again, ranks as the method with the highest standard deviation,

with a value that is almost three times higher than the others. This aspect is clearly observable in the estimated delta box aggregation graph depicted in Figure 4.5. These statements are further confirmed by the average percentage differences calculated: the iterative method has the highest percentage, reaching 0.62%, nearly double the value computed with both other methods. This reflects the significant variability of this method, stemming from its iterative calculation approach.

4.4 Effectiveness of the quantum method

To analyze the obtained data and its precision, both an absolute and a percentage distance between the "exact" values and the estimated ones were calculated. With "exact" value, we refer to the value, both in terms of expected payoff and the delta of the option, calculated through the classical Monte Carlo simulation.

Meanwhile, the "estimated" value corresponds to that computed using the quantum method. Regarding the utilization of resources to achieve the desired outcome, the depth (in terms of sequential quantum gates) and number of qubits of the circuit employed were computed. In defining the efficacy of a quantum method using these parameters, we must consider three criteria: computational complexity, quantum error, and scalability.

Computational complexity informs us that, all else being equal in terms of result precision, a method that utilizes fewer resources, hence fewer qubits and a lower depth, is preferable.

Quantum error, on the other hand, underscores how, despite a method appearing superior in accuracy, if it exhibits significantly higher depth than a method of comparable precision, it may not be deemed the better method. This is because higher depth increases the likelihood of errors, particularly in the presence of noise and decoherence. Therefore, it is crucial to consider a method's structure relative to quantum errors and hardware imperfections.

Scalability, instead, focuses on the ability to scale the utilized method to larger quantum systems. When a method requires fewer qubits and lower depth, it becomes easier to scale while maintaining good performance.

From the reported data, it is evident that in terms of accuracy, the faster implementation emerges as the superior method, followed closely by ML with a little increase in standard deviation. Conversely, the iterative method exhibits lower precision compared to the former two. Despite utilizing the same number of qubits, the three methods demonstrate varying depths. The iterative method boasts an average depth of 4008 with a standard deviation of 522. On the other hand, the faster method, despite proving more accurate, showcases a higher depth. As mentioned earlier, this may negatively impact scalability, computational complexity, and quantum error. The MLQAE method emerges as the weakest of the three, as

it exhibits the highest depth.

When we examine the estimated value, we can observe minimal differences among the three methods. However, what significantly impacts the analysis is the standard deviation, which is lower in the case of the faster method.

Moreover, we collected data also for the delta value. As mentioned above, the delta of an option gauges how much the option price will change for every unit increase in the underlying asset price and it can be seen also as the probability that the stock price will be higher than the strike price, letting be the option in the money. Therefore, the delta is valuable in assessing the risk associated with the option. It can be regarded as an index of the precision of estimating the option price concerning variations in the underlying asset price. A higher delta signifies a greater sensitivity of the option price to changes in the underlying asset price, whereas a lower delta indicates less sensitivity. In this instance, the Monte Carlo simulation estimates a delta of 0.8098, indicating that if the underlying asset price were to increase by \$1, then the option price would increase by \$0.8098. The calculated delta appears to have a relatively high value, implying greater risk for the investor but also the potential for higher rewards, as fluctuations in the asset price have a more significant impact on the payoff, both positively and negatively. Therefore, it is crucial to accurately estimate the delta, as it allows for assessing the level of risk one is willing to undertake.

For the delta values, the faster method confirms itself as the most accurate in terms of precision, boasting a commendable depth as well. The ML method positioned itself in the middle ground, displaying good precision and a depth on par with that of the faster method. Conversely, the iterative method proved to be the least precise, albeit with a markedly lower average depth compared to the others (1461 instead of 1901).

Upon analyzing the obtained mean values, the faster method emerges as the one that yields the result closest to the exact value, coupled with a low standard deviation, thereby indicating higher precision for both of the problems analyzed.

4.5 Results with different number of shots

In this instance, it was decided to maintain a consistent number of runs at 50, while incrementing the number of shots taken for each run to 1024. Studies conducted have examined that increasing the number of shots, one approaches more closely the current value [33].

Table 4.10: ITERATIVE method results: expected payoff calculated with 1024 shots and 50 runs

	Estimated Value	ΔEV	$\%\ \Delta EV$	Depth of the circuit
Average	0.1679	0.0077	4.75%	4058
St.Deviation	$8.814\ 10^{-3}$	$6.998\ 10^{-3}$	$4.312\ 10^{-2}$	478

Table 4.11: MAXIMUM LIKELIHOOD method results: expected payoff calculated with 1024 shots and 50 runs

	Estimated Value	ΔEV	$\%\Delta EV$	Depth of the circuit
Average	0.1687	0.0064	3.92%	5171
St.Deviation	$1.890\ 10^{-3}$	$1.890\ 10^{-3}$	$1.165\ 10^{-2}$	0

Table 4.12: FASTER method results: expected payoff calculated with 1024 shots and 50 runs

	Estimated Value	ΔEV	$\%\Delta EV$	Depth of the circuit
Average	0.1688	0.0067	4.14%	4545
St.Deviation	$4.400\ 10^{-3}$	$4.120\ 10^{-3}$	$2.\ 539\ 10^{-2}$	0

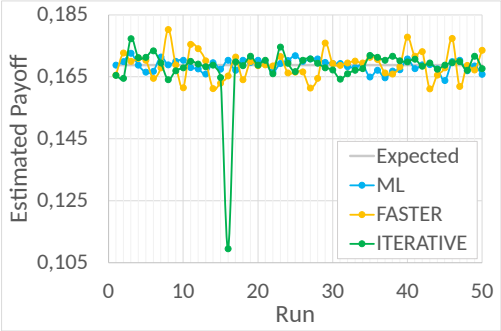

Figure 4.7: Estimated Payoff run sequence - scenario with 1024 shots and 50 runs

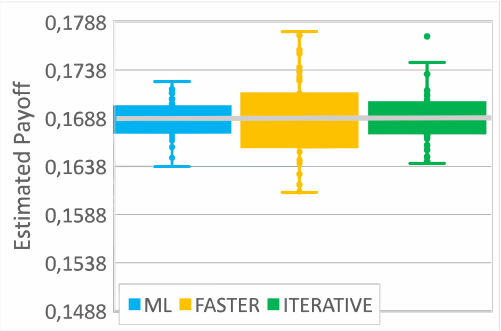

Figure 4.8: Estimated Payoff box aggregation - scenario with 1024 shots and 50 runs

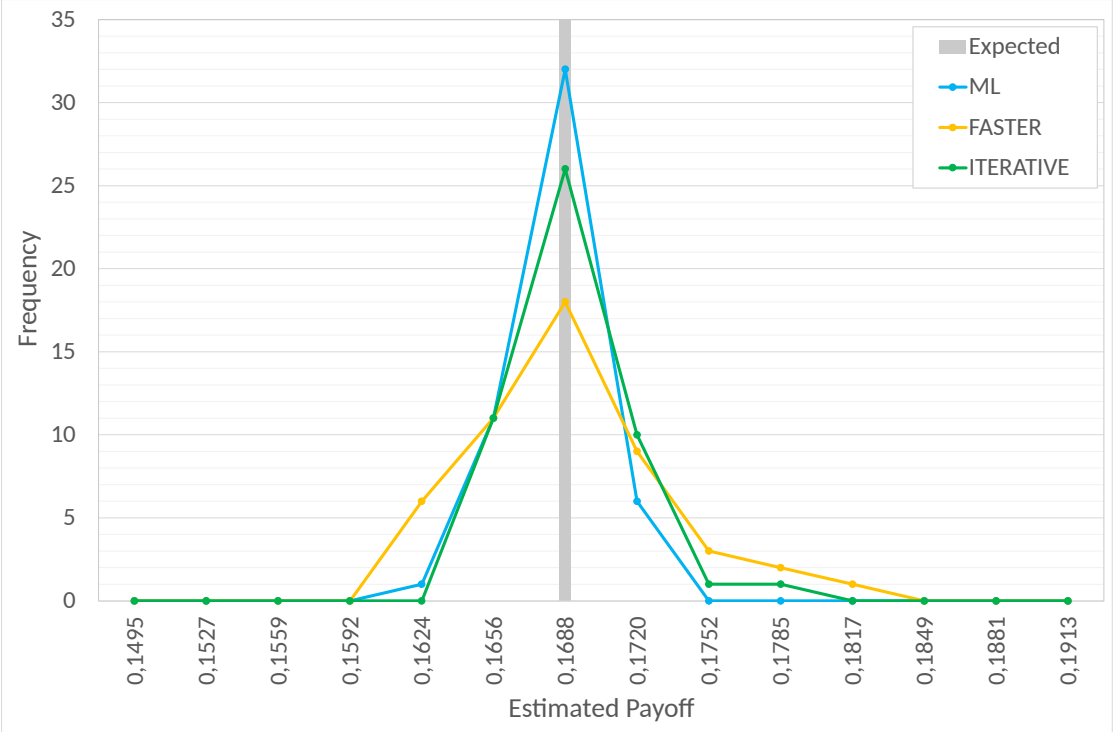

Figure 4.9: Estimated Payoff distribution - scenario with 1024 shots and 50 runs

In the case of the iterative method, the mean value is 0.1679, featuring the presence of an outlier as the minimum value, recorded at 0.1095, and a maximum value of 0.1773.These aspects are readily appreciable in the estimated payoff run sequence graph shown in Figure 4.13. As evident, both the maximum and minimum values encompass the exact value, albeit the mean deviates a bit from it. The faster method instead, with a mean of 0.1688, represents exactly the MC result,

with a lower standard deviation compared to the iterative method. Conversely, the maximum likelihood method does not exactly represent the MC result, but it is very close to it, reducing further the standard deviation. This is further confirmed by the percentage distance, which is the lowest among the three methods employed.

With 1024 shots, the distribution tightens, reducing the variance. However, it is possible to see an alignment among the outcomes of the three algorithms.

In figure 4.8 it is possible to see how the higher standard deviation in the faster method becomes apparent, resulting in the exact value being encompassed within the range of values. Conversely, in the other two methods with a lower standard deviation, this does not occur, as reflected in the sample values. The iterative method stands out as the only one with an outlier, with a value significantly distant from the mean at 0.1095, while both other methods tend to yield results clustered around their mean values.

Compared to the previously analyzed scenario with a lower number of shots, the standard deviation increases in the case of the iterative and faster methods, while it significantly decreases in the case of maximum likelihood estimation. This ensures a less variable outcome compared to the previous two cases, while still maintaining an average in line with the other two methods and an average percentage difference that is the lowest among the three methods analyzed.

Regarding the circuit depth, as analyzed previously, that of the iterative method turns out to be the lowest, while in the other two methods, it remains constant even when changing the number of shots. This cannot be said for the iterative method: in this case, where the number of shots increases and the number of executions remains the same, this method presents a higher average depth and a lower standard deviation, indicating a lower variability of this value compared to the previously analyzed case. Focusing on this aspect is crucial, especially in terms of errors and model scalability, which are essential elements to consider for our analysis.

Table 4.13: ITERATIVE method results: delta value calculated with 1024 shots and 50 runs

	Estimated Value	ΔEV	$\%\ \Delta EV$	Depth of the circuit
Average	0.8100	0.0014	0.17%	1489
St.Deviation	$1.218\ 10^{-3}$	$9.147\ 10^{-4}$	$1.131\ 10^{-3}$	177

Table 4.14: MAXIMUM LIKELIHOOD method results: delta value calculated with 1024 shots and 50 runs

	Estimated Value	ΔEV	$\%\Delta EV$	Depth of the circuit
Average	0.8097	0.0009	0.11%	1901
St.Deviation	$6.417\ 10^{-4}$	$5.357\ 10^{-4}$	$6.622\ 10^{-4}$	0

Table 4.15: FASTER method results: delta value calculated with 1024 shots and 50 runs

	Estimated Value	ΔEV	$\%\Delta EV$	Depth of the circuit
Average	0.8103	0.0030	0.37%	1901
St.Deviation	$3.280\ 10^{-3}$	$1.906\ 10^{-3}$	$2.357\ 10^{-3}$	0

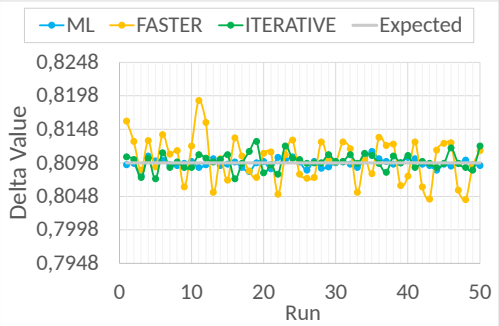

Figure 4.10: Estimated Delta run sequence - scenario with 1024 shots and 50 runs

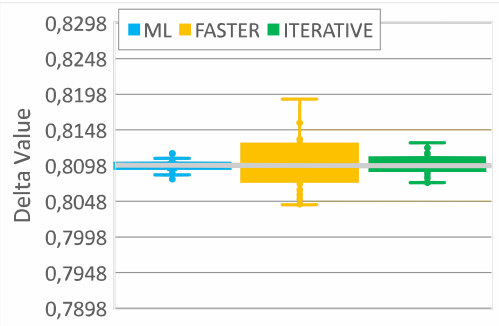

Figure 4.11: Estimated Delta box aggregation - scenario with 1024 shots and 50 runs

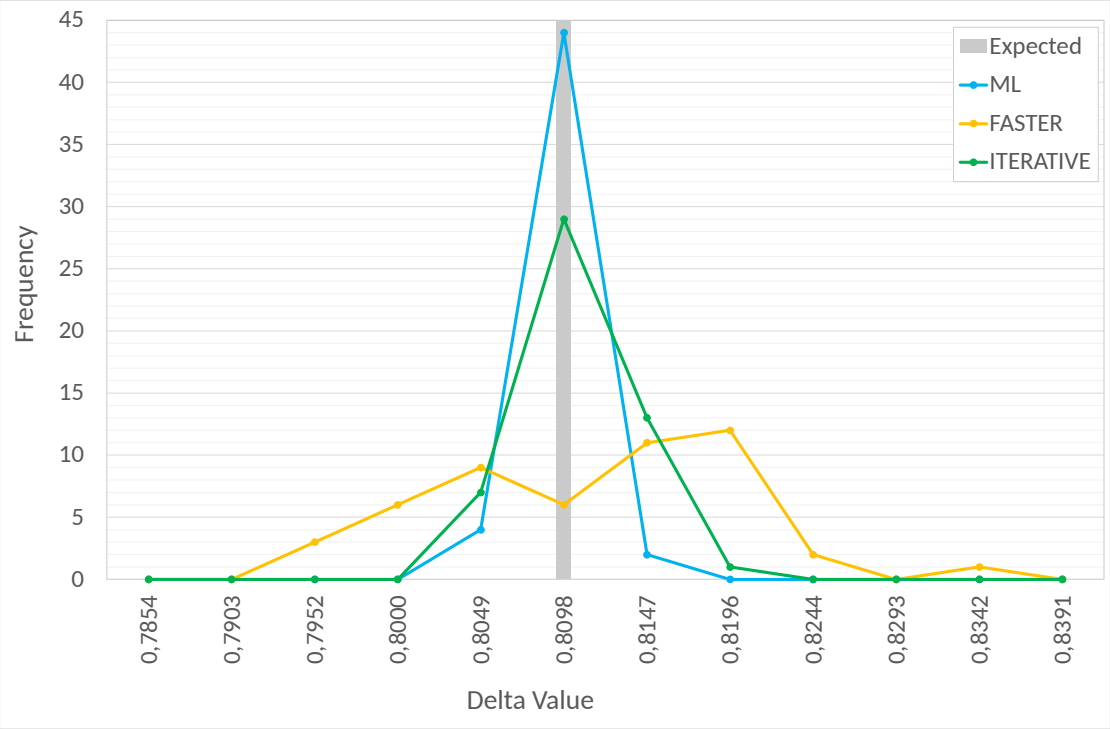

Figure 4.12: Estimated Delta distribution - scenario with 1024 shots and 50 runs

As for the delta value, the maximum likelihood method emerges as the best in terms of both the estimated value, which closely approximates the "exact" value calculated with Monte Carlo simulation, and the standard deviation, as it has the lowest value. Therefore, having a mean value very close to the classically estimated one, a small standard deviation is desirable. These assertions are also supported by the average percentage difference, which turns out to be the lowest among

the analyzed methods. The circuit depth consistently shows the lowest value for the iterative method with a low standard deviation. However, even though the ML method has a higher depth value, it's not so high as to render the precision advantages negligible. This is because the iterative method may have a lower depth but with a mean value further from the exact one and a higher standard deviation.

However, currently, it's challenging to successfully run a circuit with a depth of 1901 or even 1489 on real quantum hardware due to the issues related to noise and decoherence mentioned earlier. Hence, for such analysis, it is preferable to consider elements that assess the accuracy of the method, even if methods with lower depth will certainly be executable earlier on quantum hardware.

4.6 Results with different volatility

In the latest case under analysis, the retention of 50 runs and 1024 shots per run remains constant. However, it was decided to alter the volatility from 0.4 to 0.8, a crucial parameter in option pricing calculation, and observe how elevating it affects the behavior of our methods.

Table 4.16: EXACT VALUES calculated with a volatility equal to 0.8

Exact Expected Value	Exact Delta Value
0.2739	0.4393

Table 4.17: ITERATIVE method results: expected payoff calculated with 1024 shots, 50 runs and a volatility equal to 0.8

	Estimated Value	ΔEV	$\%\ \Delta EV$	Depth of the circuit
Average	0.2739	0.0135	5.17%	3709
St.Deviation	$5.628\ 10^{-3}$	$5.430\ 10^{-3}$	$2.085\ 10^{-2}$	469

Table 4.18: MAXIMUM LIKELIHOOD method results: expected payoff calculated with 1024 shots, 50 runs and a volatility equal to 0.8

	Estimated Value	ΔEV	$\%\Delta EV$	Depth of the circuit
Average	0.2727	0.0126	4.84%	5071
St.Deviation	$4.702\ 10^{-3}$	$3.361\ 10^{-3}$	$1.290\ 10^{-2}$	0

Table 4.19: FASTER method results: expected payoff calculated with 1024 shots, 50 runs and a volatility equal to 0.8

	Estimated Value	ΔEV	$\%\Delta EV$	Depth of the circuit
Average	0.2752	0.0149	5.72%	4457
St.Deviation	$8.533\ 10^{-3}$	$8.244\ 10^{-3}$	$3.165\ 10^{-2}$	0

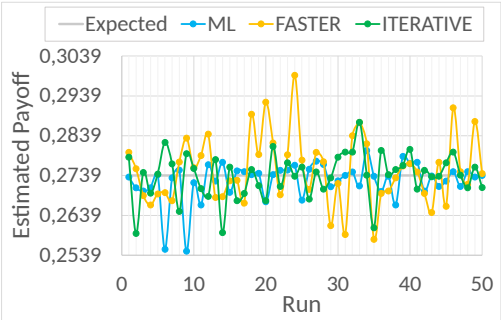

Figure 4.13: Estimated Payoff run sequence - scenario with 1024 shots, 50 runs and a volatility equal to 0.8

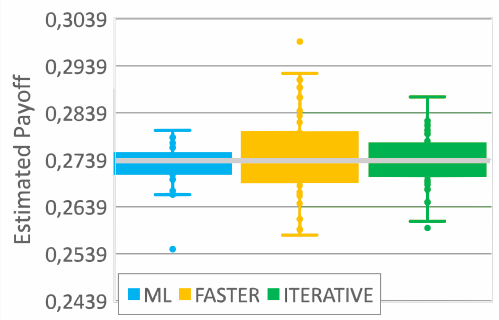

Figure 4.14: Estimated Payoff box aggregation - scenario with 1024 shots, 50 runs and a volatility equal to 0.8

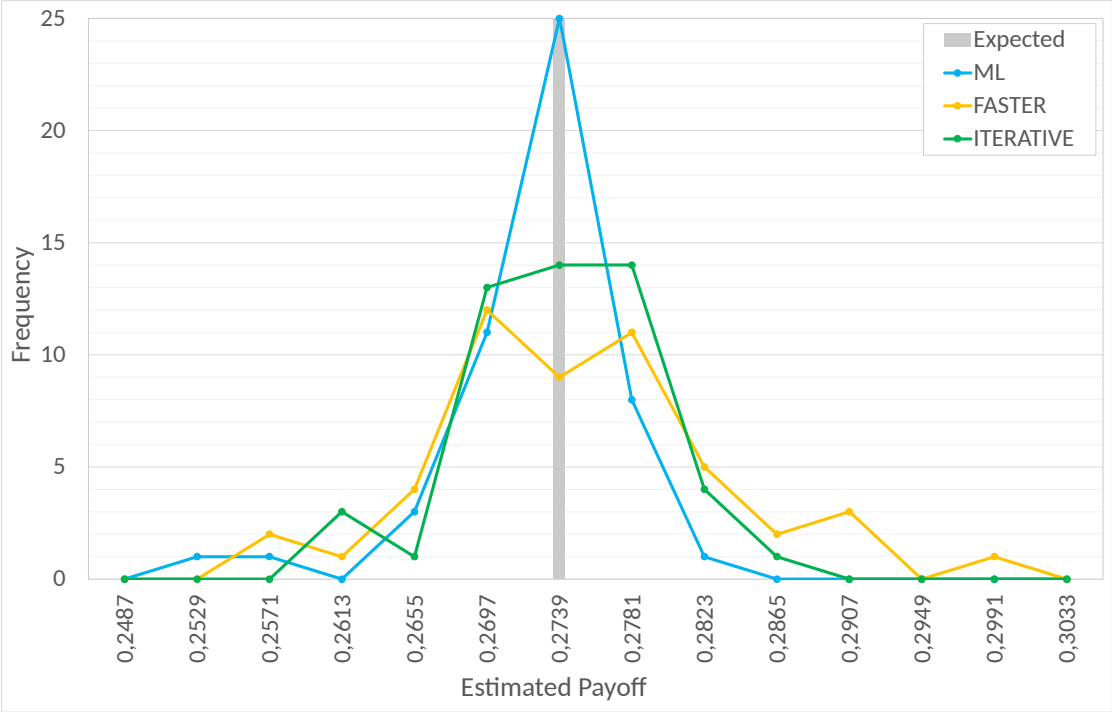

Figure 4.15: Estimated Payoff distribution - scenario with 1024 shots, 50 runs and a volatility equal to 0.8

Table 4.20: ITERATIVE method results: delta value calculated with 1024 shots, 50 runs and a volatility equal to 0.8

	Estimated Value	ΔEV	% ΔEV	Depth of the circuit
Average	0.4387	0.0020	0.45%	1335
St.Deviation	$2.523\ 10^{-3}$	$1.625\ 10^{-3}$	$3.699\ 10^{-3}$	171

Table 4.21: MAXIMUM LIKELIHOOD method results: delta value calculated with 1024 shots, 50 runs and a volatility equal to 0.8

	Estimated Value	ΔEV	$\%\Delta EV$	Depth of the circuit
Average	0.4392	0.0007	0.17%	1831
St.Deviation	$9.188\ 10^{-4}$	$5.563\ 10^{-4}$	$1.266\ 10^{-3}$	0

Table 4.22: FASTER method results: delta value calculated with 1024 shots, 50 runs and a volatility equal to 0.8

	Estimated Value	ΔEV	$\%\Delta EV$	Depth of the circuit
Average	0.4393	0.0012	0.28%	1831
St.Deviation	$1.556\ 10^{-3}$	$9.232\ 10^{-4}$	$2.101\ 10^{-3}$	0

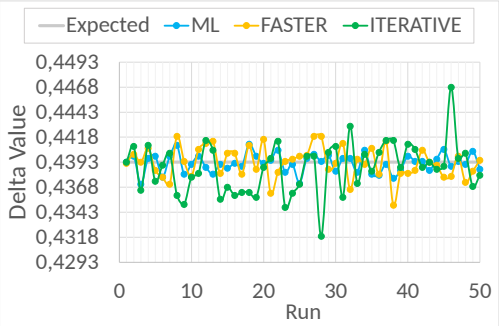

Figure 4.16: Estimated Delta run sequence - scenario with 1024 shots, 50 runs and a volatility equal to 0.8

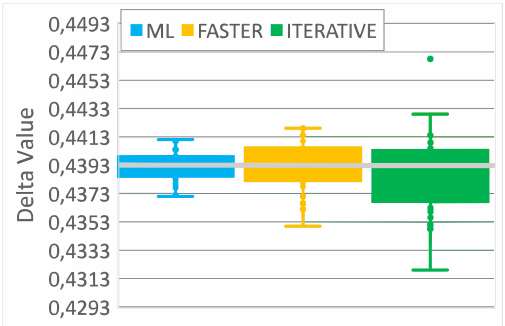

Figure 4.17: Estimated Delta box aggregation - scenario with 1024 shots, 50 runs and a volatility equal to 0.8

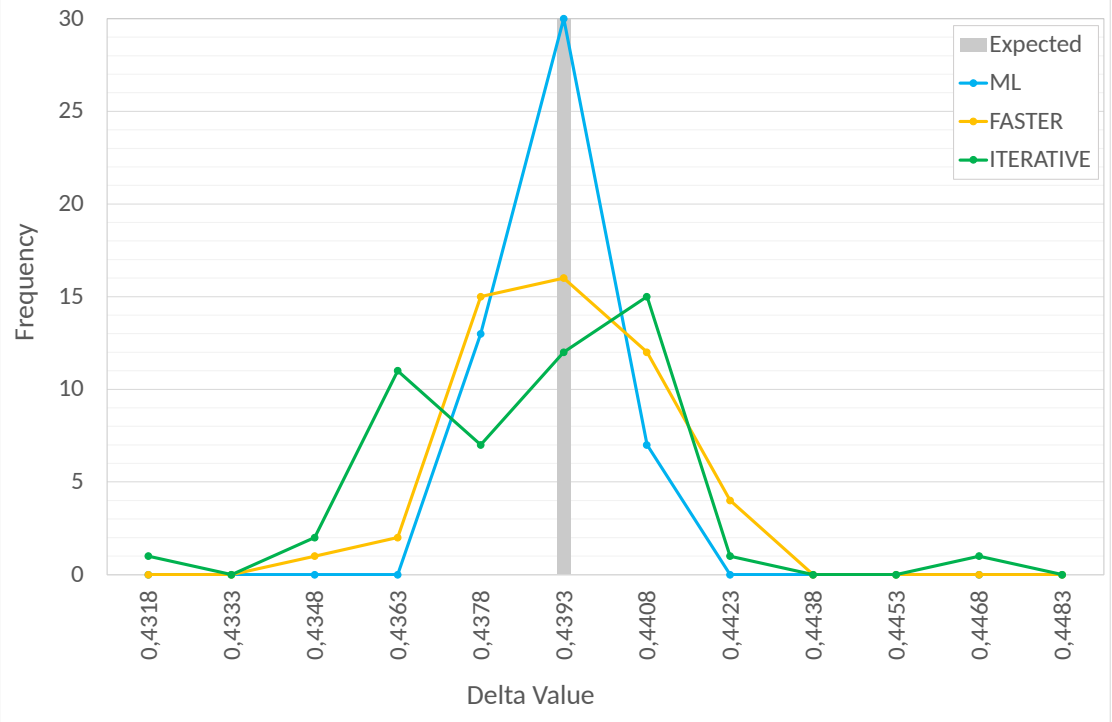

Figure 4.18: Estimated Delta distribution - scenario with 1024 shots, 50 runs and a volatility equal to 0.8

In this scenario, it was decided to modify the value of volatility and increase it, as it is the parameter that can exert the most stress on our experiment, representing the fluctuation in the price of the underlying asset over time. Greater volatility can lead to a higher potential payoff in the future, as the value of the stock may increase more significantly compared to a stock with lower volatility. This is evident from

the result obtained through the Monte Carlo simulation, which reports a payoff of 0.2739, significantly higher than the value calculated with half the volatility. However, while the payoff nearly doubles, the delta is halved, as it represents the precision of the calculated payoff. With high volatility, the stock's value fluctuates more, both positively and negatively, which can also lead to a decrease in its value and thus a loss, but as previously analyzed in the Black-Scholes model, higher volatility has a positive effect on the option price. We are examining a scenario where the strike price and the initial stock price are close, so by increasing the volatility, we significantly raise the probability that the option ends up out of the money and therefore not exercised. This is why the delta decreases.

In this case, with a delta of approximately 0.5, we can infer that the option exhibits a moderate sensitivity to changes in the underlying asset price. Given that the option in question is a call, this implies that the option price will vary roughly half as much as the underlying asset price. Generally, a delta of 0.5 suggests a balanced position between risk and potential gain concerning the movement of the underlying asset. Conversely, in the previous scenario with lower volatility, we had a delta of approximately 1, indicating that the option price would vary exactly in tandem with the underlying asset price. However, this entails a higher exposure to risk and price fluctuations.

The iterative method presents an average value of 0.2739, the faster method at 0.2752, and the ML method at 0.2727; except for the first one, which matches the value estimated with MC, the last two diverge considerably from it, considering the previously analyzed values.

Nevertheless, as depicted in 4.13, the variation in estimated values is greater compared to the case with lower volatility, except for the iterative method. This is because the values tend to oscillate more and deviate further from the calculated mean for each method. However, the minimum and maximum values of all three methods tend to encompass what is the exact value.

Regarding the circuit depth for calculating the estimated payoff, it consistently remains at a very high value for all three methods but is lower compared to the scenario with lower volatility. The mean depth of the iterative method always ranks lower than the others, even though it exhibits a significantly higher standard deviation when compared to the previous cases.

The average delta calculated using the different QAE methods is very close to the one defined as exact, and in the case of the faster method, this mean value coincides with it, with a relatively low standard deviation as well. The circuit depth used in the faster method is equal to that of the maximum likelihood method but greater than that of the iterative method. Despite having lower values compared to the case with lower volatility, they still exhibit a very high value to be executed on hardware.

Chapter 5

Conclusion

The Monte Carlo method represents a pivotal technique employed to probabilistically assess the properties of a system by generating statistical samples of the system's realizations. Within the realm of finance, Monte Carlo simulations play a crucial role in modeling the impact of uncertainties influencing financial instruments, whether they are individual stocks, portfolios, or options. This stochastic approach allows for a comprehensive examination of the potential outcomes under various scenarios, aiding in risk assessment, decision-making, and strategy formulation within the financial domain. When aiming to derive the most likely outcome from a broad distribution or to minimize the associated error to a negligible level, the demand for Monte Carlo simulations can escalate significantly. This challenge is particularly pronounced in scenarios such as stock market simulations, where the simulations can extend over entire trading days due to the need for an extensive number of iterations to achieve precise results.

In Option Pricing, leveraging a quantum speedup could significantly alter the landscape. Pioneering steps in this direction were taken by Brassard, Hoyer, Mosca, and Tapp [22]. Initially, they expanded Grover's search algorithm to develop an algorithm for Quantum Amplitude Amplification (QAA). By starting with a target state with a probability p, Brassard et al. demonstrated the capability to amplify this probability to nearly one within $1/\sqrt{p}$ operations, essentially achieving a quadratic acceleration compared to the most efficient classical algorithm. In the subsequent segment of their study, Brassard et al. introduced their Quantum Amplitude Estimation (QAE) algorithm. QAE plays a crucial role in numerous advanced quantum algorithms and notably facilitates a potential quadratic enhancement in computing expectation values through Monte Carlo sampling.

Montanaro demonstrated that using QAE, Monte Carlo simulations can be executed on a quantum computer, achieving equivalent accuracy, but with nearly quadratically fewer samples, denoted as $\mathcal{O}(\sigma/\epsilon)$ samples (with polylogarithmic adjustments) [23]. This acceleration stems from the QAE algorithm, which lies at

the core of efficiently estimating the distribution's mean [34].

How QAE is implemented can impact the algorithm's outcome, prompting an analysis of various implementations and scenarios to discern the most effective method. This work aimed at identifying the optimal implementation approach to ensure enhanced efficiency and accuracy in the quantum estimation process.

5.1 Comments on results

The MLQAE (Maximum Likelihood Quantum Amplitude Estimation) and IQAE (Iterative Quantum Amplitude Estimation) demonstrate comparable accuracy but they exhibit contrasting characteristics. IQAE operates under control by specifying an error bound, providing a distinct advantage over MLQAE, which lacks direct control over the error bound, leading to ambiguity regarding its upper error bound. The capability to regulate the error endows IQAE with significant superiority. Conversely, IQAE's circuit depth is not always controllable due to occasional occurrences of excessively large powers of Q, presenting a limitation absent in MLQAE. [35]

In this experiment, such assertion found validation in the implementation of the methods themselves: within the iterative method, we defined an *epsilon* parameter dictating the error, along with an *alpha* parameter facilitating the specification of the confidence interval. Conversely, for the maximum likelihood estimation, such control was not attainable, as the only parameter that could be defined was the number of iterations to be executed. Additionally, in terms of depth, the literature's assertions were corroborated: while in the iterative method, depth fluctuates with each run, in the ML method, this variability is absent, with the depth being controlled and maintaining a consistent value throughout.

The pivotal parameter governing the application of MLQAE is the evaluation schedule, which denotes the number of $\mathcal{Q}$ applications. For consistency with other employed methods, it was set to 4; however, empirical testing [33] revealed that this value introduces noise in the results due to decoherence and gate errors. For our use case, a value equal to 3 emerges as the threshold where results are efficiently computed on a real-world device while minimizing the adverse effects of aforementioned errors.

This is one of the reasons why the experiment was conducted on a simulator rather than real hardware, as only noise would have been observed on the latter. This phenomenon reflects the current stage of quantum computing evolution known as "Noisy Intermediate-Scale Quantum". This phase delineates a period wherein devices lack a high number of qubits and sufficient power to execute various algorithms without succumbing to quantum and calibration errors, more commonly referred to as noise. Naturally, during this phase, until more powerful computers

are constructed, efforts primarily revolve around mitigating the error's impact through various error-mitigation techniques.

For the IQAE method, as mentioned, the crucial parameter is *epsilon*, which indirectly defines the power of $\mathcal{Q}$. Naturally, the lower the value of *epsilon*, the greater the computational difficulty of the result. For the sake of coherence, epsilon is set to 0.03, a value higher than the standard value of 0.01 (and that corresponded on average to 4 applications of $\mathcal{Q}$) yet not excessively high to yield significantly divergent results from the exact value.

The faster quantum amplitude estimation method, similarly to the two previously outlined approaches, tackles the AE problem without employing phase estimation. This implementation without phase estimation also allows for a reduction in circuit depth, albeit still at the $O(1/\epsilon)$ depth requirement. Despite the constant depth, noise effects persist. In terms of actual values, the depth of the faster method is only lower compared to the ML method. However, when comparing it to the iterative method, the average depth, with a standard deviation of 522, suggests that this method isn't always able to achieve a lower depth for solving the problem.

The analysis of the collected data has further revealed how both the ML and faster methods consistently demonstrate higher accuracy compared to the iterative method across multiple contexts. However, the significantly lower depth of the latter method gives it a predominant position over the others. Considering the concepts of error, computational complexity, and scalability, the iterative method, with its considerably lower depth, emerges as preferable to the other methods, particularly in a context where available quantum computers have not yet reached a level of development where this aspect is negligible concerning method accuracy.

5.2 Future works

This analysis of QAE can be extended not only including put options, but also by varying other parameters such as stock price, strike price, and expiration date, and can be applied to more complex classes of options beyond simple European puts and calls, such as basket options or path-dependent options. For this work, as highlighted, such experiments have been conducted solely on a simulator and not on hardware, because currently available quantum computers are unable to support such computations without producing noise that makes result analysis impossible. Naturally, progress will enable the execution of increasingly complex calculations; however, methods analyzed requiring fewer quantum resources will have the advantage of being verifiable on hardware earlier than those with higher depth.

New algorithms and methods of implementing QAE can further contribute to enhancing what is already a functioning algorithm, allowing for faster MC execution

with practically equal accuracy, but with additional margins for improvement. However, the true innovation brought by QAE lies in terms of convergence speed, particularly evident in path-dependent options, which are financial derivatives whose value depends not only on the price of the underlying asset at the expiration date but also on the path followed by that price over time, where an MC simulation method requiring a large number of random samples and considerable time to find a result can be replaced by a method that guarantees accurate results at a significantly increased speed. Furthermore, using this methodology is also advantageous in terms of evaluating the risk associated with the option, which is a significant factor in managing the options portfolio owned.

Finally, in future investigations, a fine-tuning process of hyperparameters could be conducted within the context of QAE and its variants. This process entails the exploration and adjustment of parameters aimed at maximizing the algorithm's performance and effectiveness in executing its amplitude estimation tasks.

Appendix A

IQAE Subroutines

Algorithm 1 Procedure for finding k_{i+1}

function FindNextK($k_i, \theta_l, \theta_u, \mathrm{up}_i, r = 2$):
  $K_i = 4k_i + 2$ // current θ-factor
  $\theta_i^{\min} = K_i\theta_l$ // lower bound for scaled θ
  $\theta_i^{\max} = K_i\theta_u$ // upper bound for scaled θ
  $K_{\max} = \left\lfloor \frac{\pi}{\theta_u - \theta_l} \right\rfloor$ // set an upper bound for θ-factor
  $K = K_{\max} - (K_{\max} - 2)_{mod4}$ // largest potential candidate of the form $4k+2$
  while $K \geq rK_i$ **do**
    $q = K/K_i$ // factor to scale $[\theta_i^{\min}, \theta_i^{\max}]$
    if $\{q \cdot \theta_i^{\max}\}_{\mathrm{mod}\ 2\pi} \leq \pi$ and $\{q \cdot \theta_i^{\min}\}_{\mathrm{mod}\ 2\pi} \leq \pi$ **then**
      // $\left[\theta_{i+1}^{\min}, \theta_{i+1}^{\max}\right]$ is in upper half-plane
      $K_{i+1} = K$
      $\mathrm{up}_{i+1} = \mathrm{True}$
      $k_{i+1} = (K_{i+1} - 2)/4$
      return $(k_{i+1}, \mathrm{up}_{i+1})$
    if $\{q \cdot \theta_i^{\max}\}_{\mathrm{mod}\ 2\pi} \geq \pi$ and $\{q \cdot \theta_i^{\min}\}_{\mathrm{mod}\ 2\pi} \geq \pi$ **then**
      // $\left[\theta_{i+1}^{\min}, \theta_{i+1}^{\max}\right]$ is in lower half-plane
      $K_{i+1} = K$
      $\mathrm{up}_{i+1} = \mathrm{False}$
      $k_{i+1} = (K_{i+1} - 2)/4$
      return $(k_{i+1}, \mathrm{up}_{i+1})$
    $K = K - 4$
  return $(k_i, \mathrm{up_i})$ // return old value

Algorithm 2 Iterative Quantum Amplitude Estimation

function IQAE($\epsilon, \alpha, N_{\text{shots}}$, ci) :

// ci is a chosen confidence interval method, which can be either Clopper-Pearson or Chernoff-Hoeffding.

$i = 0$ // initialize iteration count

$k_i = 0$ // initialize power of $\mathcal{Q}$

$\text{up}_i = \text{True}$ // keeps track of the half-plane

$[\theta_l, \theta_u] = [0, \pi/2]$ // initialize conf. interval

$T = \lceil \log_2(\pi/8\epsilon) \rceil$ // max. number of rounds

calculate $L_{\max}$ // max. error on every iteration; depends on $\epsilon, \alpha, N_{\text{shots}}$ and choice of confidence interval

while $\theta_u - \theta_l > 2\epsilon$ **do**

$i = i + 1$

k_i, up $_i = \text{FindNextK}\left(k_{i-1}, \theta_l, \theta_u, \text{up}_{i-1}\right)$

set $K_i = 4k_i + 2$

if $K_i > \lceil L_{\max}/\epsilon \rceil$ **then**

$N = \lceil N_{\text{shots}}\, L_{\max}/\epsilon/K_i/10 \rceil$ // No-overshooting condition

else

$N = N_{\text{shots}}$

approximate $a_i = \mathbb{P}[|1\rangle]$ for the last qubit of $\mathcal{Q}^{k_i}\mathcal{A}|0\rangle_n|0\rangle$ by measuring N times

if $k_i = k_{i-1}$ **then**

combine the results of all iterations $j \leq i$ with $k_j = k_i$ into a single results, effectively increasing the number of shots

if ci = "Chernoff-Hoeffding" **then**

$\epsilon_{a_i} = \sqrt{\frac{1}{2N} \log\left(\frac{2T}{\alpha}\right)}$

$a_i^{\max} = \min\left(1, a_i + \epsilon_{a_i}\right)$

$a_i^{\min} = \max\left(0, a_i - \epsilon_{a_i}\right)$

if ci = "Clopper-Pearson" **then**

$a_i^{\max} = I^{-1}\left(\frac{\alpha}{2T}; Na_i, N\left(1 - a_i\right) + 1\right)$

$a_i^{\min} = I^{-1}\left(1 - \frac{\alpha}{2T}; Na_i + 1, N\left(1 - a_i\right)\right)$

calculate the confidence interval $[\theta_i^{\min}, \theta_i^{\max}]$ for $\{K_i\theta_a\}_{\bmod 2\pi}$ from $[a_i^{\min}, a_i^{\max}]$ and boolean flag up u_i by inverting $a = \left(1 - \cos\left(K_i\theta\right)\right)/2$

$\theta_l = \frac{\lfloor K_i\theta_l \rfloor_{\bmod 2\pi} + \theta_i^{\min}}{K_i}$

$\theta_u = \frac{\lfloor K_i\theta_u \rfloor_{\bmod 2\pi} + \theta_i^{\max}}{K_i}$

$[a_l, a_u] = \left[\sin^2\left(\theta_l\right), \sin^2\left(\theta_u\right)\right]$

return $[a_l, a_u]$

Bibliography

[1] E. Schrödinger. «Discussion of Probability Relations between Separated Systems». In: *Mathematical Proceedings of the Cambridge Philosophical Society* 31.4 (Oct. 1935), pp. 555–563. ISSN: 1469-8064. DOI: 10.1017/s0305004100013554. URL: http://dx.doi.org/10.1017/S0305004100013554 (cit. on pp. 1, 7).

[2] F. Bloch. «Nuclear Induction». In: *Physical Review* 70.7–8 (Oct. 1946), pp. 460–474. ISSN: 0031-899X. DOI: 10.1103/physrev.70.460. URL: http://dx.doi.org/10.1103/PhysRev.70.460 (cit. on p. 3).

[3] P. A. M. Dirac. «A new notation for quantum mechanics». In: *Mathematical Proceedings of the Cambridge Philosophical Society* 35.3 (July 1939), pp. 416–418. ISSN: 1469-8064. DOI: 10.1017/s0305004100021162. URL: http://dx.doi.org/10.1017/S0305004100021162 (cit. on p. 5).

[4] Thomas Young. In: *Philosophical Transactions of the Royal Society of London* 94 (Dec. 1804), pp. 1–16. ISSN: 2053-9223. DOI: 10.1098/rstl.1804.0001. URL: http://dx.doi.org/10.1098/rstl.1804.0001 (cit. on p. 6).

[5] David A. B. Miller. «Huygens's wave propagation principle corrected». In: *Optics Letters* 16.18 (Sept. 1991), p. 1370. ISSN: 1539-4794. DOI: 10.1364/ol.16.001370. URL: http://dx.doi.org/10.1364/OL.16.001370 (cit. on p. 6).

[6] *Superposition and Entanglement*. URL: https://www.quantum-inspire.com/kbase/superposition-and-entanglement/ (visited on 03/12/2024) (cit. on pp. 6, 7).

[7] J. S. Bell and Alain Aspect. *Speakable and Unspeakable in Quantum Mechanics: Collected Papers on Quantum Philosophy*. Cambridge University Press, June 2004. ISBN: 9780511815676. DOI: 10.1017/cbo9780511815676. URL: http://dx.doi.org/10.1017/CBO9780511815676 (cit. on p. 8).

[8] A. Einstein, B. Podolsky, and N. Rosen. «Can Quantum-Mechanical Description of Physical Reality Be Considered Complete?» In: *Physical Review* 47.10 (May 1935), pp. 777–780. ISSN: 0031-899X. DOI: 10.1103/physrev.47.777. URL: http://dx.doi.org/10.1103/PhysRev.47.777 (cit. on p. 8).

[9] J. S. Bell. «On the Einstein Podolsky Rosen paradox». In: *Physics Physique Fizika* 1.3 (Nov. 1964), pp. 195–200. ISSN: 0554-128X. DOI: 10.1103/physicsphysiquefizika.1.195. URL: http://dx.doi.org/10.1103/PhysicsPhysiqueFizika.1.195 (cit. on p. 8).

[10] Anton Frisk Kockum and Franco Nori. «Quantum Bits with Josephson Junctions». In: *Springer Series in Materials Science*. Springer International Publishing, 2019, pp. 703–741. ISBN: 9783030207267. DOI: 10.1007/978-3-030-20726-7_17. URL: http://dx.doi.org/10.1007/978-3-030-20726-7_17 (cit. on p. 10).

[11] Qiskit contributors. *Qiskit: An Open-source Framework for Quantum Computing*. 2023. DOI: 10.5281/zenodo.2573505 (cit. on pp. 12–15, 21, 43).

[12] Marc Bataille. *Quantum circuits of CNOT gates*. 2020. DOI: 10.48550/ARXIV.2009.13247. URL: https://arxiv.org/abs/2009.13247 (cit. on p. 17).

[13] Kwok Yue-Kuen. *Lecture notes in Computational Methods for Pricing Structured Products*. URL: https://www.math.hkust.edu.hk/~maykwok/courses/MAFS5250/lecture%20notes/MAFS5250_Topic_5.pdf (cit. on pp. 18, 32).

[14] Richard P. Feynman. «Simulating physics with computers». In: *International Journal of Theoretical Physics* 21.6–7 (June 1982), pp. 467–488. ISSN: 1572-9575. DOI: 10.1007/bf02650179. URL: http://dx.doi.org/10.1007/BF02650179 (cit. on p. 19).

[15] Michael Demmer, Rodrigo Fonseca, and Farinaz Koushanfar. «RICHARD FEYNMAN: SIMULATING PHYSICS WITH COMPUTERS». In: (Jan. 2008) (cit. on p. 19).

[16] Elsevier, 2014. ISBN: 9780128009536. DOI: 10.1016/c2013-0-19170-2. URL: http://dx.doi.org/10.1016/C2013-0-19170-2 (cit. on p. 20).

[17] Fischer Black and Myron Scholes. «The Pricing of Options and Corporate Liabilities». In: *Journal of Political Economy* 81.3 (May 1973), pp. 637–654. ISSN: 1537-534X. DOI: 10.1086/260062. URL: http://dx.doi.org/10.1086/260062 (cit. on p. 22).

[18] Sheldon Ross. «Brownian Motion and Stationary Processes». In: *Introduction to Probability Models*. Elsevier, 2014, pp. 607–644. DOI: 10.1016/b978-0-12-407948-9.00010-4. URL: http://dx.doi.org/10.1016/B978-0-12-407948-9.00010-4 (cit. on p. 23).

[19] Jonathan Berk and Peter DeMarzo. *Corporate Finance, Global Edition*. en. 3rd ed. The Pearson series in finance. London, England: Pearson Education, Apr. 2013 (cit. on pp. 25, 29).

[20] John C. Cox, Stephen A. Ross, and Mark Rubinstein. «Option pricing: A simplified approach». In: *Journal of Financial Economics* 7.3 (Sept. 1979), pp. 229–263. ISSN: 0304-405X. DOI: 10.1016/0304-405x(79)90015-1. URL: http://dx.doi.org/10.1016/0304-405X(79)90015-1 (cit. on p. 27).

[21] Phelim P. Boyle. «Options: A Monte Carlo approach». In: *Journal of Financial Economics* 4.3 (May 1977), pp. 323–338. ISSN: 0304-405X. DOI: 10.1016/0304-405x(77)90005-8. URL: http://dx.doi.org/10.1016/0304-405X(77)90005-8 (cit. on p. 30).

[22] Gilles Brassard, Peter Høyer, Michele Mosca, and Alain Tapp. «Quantum amplitude amplification and estimation». In: *Quantum Computation and Information* (2002), pp. 53–74. ISSN: 0271-4132. DOI: 10.1090/conm/305/05215 (cit. on pp. 33, 34, 38, 68).

[23] Ashley Montanaro. «Quantum speedup of Monte Carlo methods». In: *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* 471.2181 (Sept. 2015), p. 20150301. ISSN: 1471-2946. DOI: 10.1098/rspa.2015.0301. URL: http://dx.doi.org/10.1098/rspa.2015.0301 (cit. on pp. 33, 68).

[24] Peter W. Shor. «Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer». In: *SIAM Review* 41.2 (Jan. 1999), pp. 303–332. ISSN: 1095-7200. DOI: 10.1137/s0036144598347011. URL: http://dx.doi.org/10.1137/S0036144598347011 (cit. on p. 36).

[25] Dmitry Grinko, Julien Gacon, Christa Zoufal, and Stefan Woerner. «Iterative quantum amplitude estimation». In: *npj Quantum Information* 7.1 (Mar. 2021). ISSN: 2056-6387. DOI: 10.1038/s41534-021-00379-1. URL: http://dx.doi.org/10.1038/s41534-021-00379-1 (cit. on pp. 38, 44).

[26] Yohichi Suzuki, Shumpei Uno, Rudy Raymond, Tomoki Tanaka, Tamiya Onodera, and Naoki Yamamoto. «Amplitude estimation without phase estimation». In: *Quantum Information Processing* 19.2 (Jan. 2020). ISSN: 1573-1332. DOI: 10.1007/s11128-019-2565-2. URL: http://dx.doi.org/10.1007/s11128-019-2565-2 (cit. on pp. 38, 45).

[27] Lov Grover and Terry Rudolph. «Creating superpositions that correspond to efficiently integrable probability distributions». In: (2002). DOI: 10.48550/ARXIV.QUANT-PH/0208112. URL: https://arxiv.org/abs/quant-ph/0208112 (cit. on p. 39).

[28] Christa Zoufal, Aurélien Lucchi, and Stefan Woerner. «Quantum Generative Adversarial Networks for learning and loading random distributions». In: *npj Quantum Information* 5.1 (Nov. 2019). ISSN: 2056-6387. DOI: 10.1038/s41534-019-0223-2. URL: http://dx.doi.org/10.1038/s41534-019-0223-2 (cit. on p. 40).

[29] Kouhei Nakaji. «Faster amplitude estimation». In: *Quantum Information and Computation* 20.13 and 14 (Nov. 2020), pp. 1109–1123. ISSN: 1533-7146. DOI: 10.26421/qic20.13-14-2. URL: http://dx.doi.org/10.26421/QIC20.13-14-2 (cit. on pp. 45, 46).

[30] Stefan Woerner and Daniel J. Egger. «Quantum risk analysis». In: *npj Quantum Information* 5.1 (Feb. 2019). ISSN: 2056-6387. DOI: 10.1038/s41534-019-0130-6. URL: http://dx.doi.org/10.1038/s41534-019-0130-6 (cit. on p. 47).

[31] *EuropeanCallPricing - Qiskit Finance 0.4.1*. URL: https://qiskit-community.github.io/qiskit-finance/stubs/qiskit_finance.applications.EuropeanCallPricing.html (visited on 03/27/2024) (cit. on p. 48).

[32] *EuropeanCallDelta*. en. URL: https://docs.quantum.ibm.com/api/qiskit/0.19/qiskit.finance.components.uncertainty_problems.EuropeanCallDelta (visited on 03/27/2024) (cit. on p. 49).

[33] Pooja Rao, Kwangmin Yu, Hyunkyung Lim, Dasol Jin, and Deokkyu Choi. «Quantum amplitude estimation algorithms on IBM quantum devices». In: *Quantum Communications and Quantum Imaging XVIII*. Ed. by Keith S. Deacon. SPIE, Aug. 2020. DOI: 10.1117/12.2568748. URL: http://dx.doi.org/10.1117/12.2568748 (cit. on pp. 56, 69).

[34] Román Orús, Samuel Mugel, and Enrique Lizaso. «Quantum computing for finance: Overview and prospects». In: *Reviews in Physics* 4 (Nov. 2019), p. 100028. ISSN: 2405-4283. DOI: 10.1016/j.revip.2019.100028. URL: http://dx.doi.org/10.1016/j.revip.2019.100028 (cit. on p. 69).

[35] Kwangmin Yu, Hyunkyung Lim, Pooja Rao, and Dasol Jin. *Comparison of Amplitude Estimation Algorithms by Implementation*. 2020. DOI: 10.48550/ARXIV.2005.05300. URL: https://arxiv.org/abs/2005.05300 (cit. on p. 69).