

Politecnico di Torino

ScuDo
Scuola di Dottorato ~ Doctoral School
WHAT YOU ARE, TAKES YOU FAR

Doctoral Dissertation
Doctoral Program in Computer and Control Engineering (37^{th} cycle)

Quantum Computing:

Analysis and design of quantum algorithms for financial applications

Emanuele Dri

* * * * *

Supervisor
Prof. Bartolomeo Montrucchio

Politecnico di Torino
2025

This thesis is licensed under a Creative Commons License, Attribution - Noncommercial-NoDerivative Works 4.0 International: see www.creativecommons.org. The text may be reproduced for non-commercial purposes, provided that credit is given to the original author.

I hereby declare that, the contents and organisation of this dissertation constitute my own original work and does not compromise in any way the rights of third parties, including those relating to the security of personal data.

. . . . . .

Emanuele Dri

Turin, 2025

Summary

Quantum computing has the potential to become a powerful tool for solving complex problems that classical computation struggles to tackle. The financial sector presents numerous such problems, and quantitative finance is considered one of the fields where the benefits of employing quantum processing units might first become evident.

The purpose of this manuscript is to analyze quantum computing as a paradigm for solving certain computational problems in the domain of finance. These are problems for which traditional classical methods may be unsatisfactory in terms of speed or accuracy. Among these, we can certainly include tasks such as portfolio optimization, risk management, and derivative pricing. These problems have intrinsic complexity due to their combinatorial or stochastic nature. Quantum approaches have shown potential in efficiently solving some of these tasks, although significant limitations in applying these approaches on scales relevant to the industry remain. This is undoubtedly partially due to the limitations of currently available quantum hardware, which is restricted by the number of qubits and low resilience to errors.

This work begins with a brief introduction to the fundamentals of quantum computing. It presents the key characteristics that distinguish quantum bits (qubits) from their classical counterparts and introduces the concept of the Bloch sphere as an effective geometric representation of the state of a single qubit. Subsequently, the main quantum gates are briefly discussed to illustrate how they act on qubit states and manipulate their phases. The discussion is then extended to multi-qubit systems, through which fundamental concepts that characterize quantum computation, such as entanglement, are introduced.

To conclude the introductory section, several fundamental and well-known quantum algorithms are presented. The Quantum Fourier Transform is introduced, highlighting its role as the quantum counterpart of the classical discrete Fourier Transform. Building on this foundational quantum subroutine, the Quantum Phase Estimation (QPE) algorithm is then illustrated, which leverages the phase kickback phenomenon to extract the eigenvalues of unitary operators. As a direct application of QPE, Shor's factorization algorithm is presented, demonstrating its ability to efficiently decompose large integers into prime factors, thereby offering an exponential advantage over classical methods. Next, Grover's search algorithm is introduced,

showcasing its quadratic speedup for searching in unsorted databases. Finally, attention is given to quantum amplitude amplification and estimation methods, which can be seen as generalizations of Grover's algorithm to arbitrary superpositions in the case of amplitude amplification and as an extension of quantum counting in the case of amplitude estimation.

Based on these premises, the text transitions to practical applications. Specifically, some of the algorithmic subroutines already mentioned, along with newer methods, are applied to address key financial challenges. Regarding portfolio optimization, the QUBO (Quadratic Unconstrained Binary Optimization) formulation is presented as an effective tool to deal with the fundamental asset selection problem. This formulation allows for the efficient exploration of an exponentially large space of asset combinations using quantum or quantum-inspired solvers.

For Credit risk analysis, which is another domain where quantum methods are expected to have significant impact, we start by introducing how financial institutions have the need to estimate the risk of debtors' default, to determine the appropriate amount of Economic Capital needed to buffer against potential (although unlikely) losses. Then, this dissertation introduces novel quantum approaches to simulate more efficiently the stochastic behavior of asset values. By modeling both systematic and idiosyncratic risks, the quantum approach aims to generate accurate loss distributions and Value at Risk (VaR) estimates with a quadratic speedup over classical Monte Carlo methods. Such advancements could ultimately lead to improved risk management strategies in banking and finance.

The final area of analysis is derivative pricing. Increasingly, Monte Carlo methods are used to simulate future market scenarios for pricing financial derivatives such as options and swaps. These methods, while flexible, require significant computational resources, especially when addressing complex payoff structures. Therefore, Quantum Monte Carlo techniques can be advantageous also in this case and are being actively explored. The objective is to assess potential enhancements for a more efficient implementation in terms of necessary quantum resources.

In general, the main contribution of this work is the proposal of algorithmic enhancements that make specific quantum approaches more scalable, and thus more suitable for currently available hardware. Recognizing the limitations of today's noisy intermediate-scale quantum (NISQ) hardware, the dissertation outlines strategies to reduce resource overhead without sacrificing the accuracy of financial models. These optimizations represent a contribution towards the realistic deployment of quantum algorithms in operational financial settings.

Acknowledgements

I would like to express my deepest gratitude to all those who, over the course of more than a decade of education and personal growth, have accompanied, encouraged, supported (and endured) me. Those who never gave up in the face of setbacks and who celebrated each achievement along the way.

Foremost among them is my wonderful family. Alongside the immense affection I feel for them, I have also come to hold a profound admiration for their courage in navigating a turbulent world, always guided by the lantern of love.

Secondly, I am indebted to a constellation of teachers and educators (from primary school through to university) who saw potential in me even during times when I had begun to doubt my ability to truly realize it. Their belief has been instrumental in shaping the path I've taken.

I also feel the need to thank the friends who stood by my side throughout this journey, with whom I shared the carefree spirit of youth, as well as the fears and challenges that come with the adventure of growing up and stepping into adulthood.

A heartfelt thank you also goes to Costanza, who not only chose to share her life with me, but who has embraced my many flaws with grace and patience. Her presence inspires me to become a better version of myself — more open, more attentive, and more responsive to the needs of others. Love is a journey, and I feel privileged to walk that path with her by my side.

I began this journey as a boy and, despite the many limitations of which I am painfully aware, I conclude it as an adult, with the commitment to live fully rather than merely get by, to do my part and to fight for the kind of world I hope to leave as a legacy for future generations. I am deeply grateful to those who have helped me develop the skills and awareness needed to reach this point, and to those who will choose to join me in the good fight.

Two roads diverged in a wood, and I—
I took the one less traveled by,
And that has made all the difference.

- Robert Frost, *The Road Not Taken*

Contents

List of Tables IX

List of Figures X

1 Introduction 1
- 1.1 Overview . . . 1
- 1.2 Contributions . . . 2
- 1.3 Computing With Qubits . . . 3
  - 1.3.1 Wires and Gates . . . 3
  - 1.3.2 Noteworthy Notations . . . 4
  - 1.3.3 The Quantum Bit . . . 6
  - 1.3.4 One-Qubit Gate Operations . . . 7
  - 1.3.5 Handling Multiple Qubits . . . 12
  - 1.3.6 Multiple-Qubit Gate Operations . . . 14
- 1.4 Foundational Algorithms . . . 15
  - 1.4.1 Quantum Fourier Transform . . . 16
  - 1.4.2 Quantum Phase Estimation . . . 18
  - 1.4.3 Shor's Factoring Algorithm . . . 22
  - 1.4.4 Grover's Search Algorithm . . . 25
  - 1.4.5 Quantum Amplitude Amplification . . . 28
  - 1.4.6 Quantum Amplitude Estimation . . . 31
  - 1.4.7 QAE Without Phase Estimation . . . 32
- 1.5 Quantum Computing Libraries . . . 34
  - 1.5.1 Qiskit . . . 34
  - 1.5.2 Classiq and the Qmod Language . . . 38

2 Quantum Algorithms for Finance 41
- 2.1 Promising Complex Problems in Finance . . . 43
  - 2.1.1 Portfolio Optimization . . . 44
  - 2.1.2 Credit Risk Analysis . . . 45
  - 2.1.3 Derivative Pricing . . . 48
- 2.2 Near-Term Algorithms . . . 50

2.2.1 Quantum for Portfolio Optimization . . . 51
2.3 Algorithms That Require Fault-Tolerance . . . 55
2.3.1 Quantum for Risk Analysis . . . 56
2.3.2 Quantum for Pricing Derivatives . . . 59

3 Improving Quantum Credit Risk Analysis 63
3.1 Quantum Singular Value Transformation . . . 65
3.1.1 Quantum Signal Processing . . . 66
3.1.2 Chebyshev Polynomial Approximation . . . 66
3.2 Better State Preparation with QSVT . . . 67
3.3 Experiments . . . 71
3.3.1 Experimental Setup and Input Parameters . . . 71
3.3.2 Quantum Circuit and Polynomial Approximation via QSVT 71
3.3.3 Estimation of the Cumulative Distribution Function . . . 72
3.3.4 Iterative Quantum Amplitude Estimation and Bisection Search 72
3.3.5 Discussion of Results . . . 74

4 Better Quantum Derivative Pricing 77
4.1 Direct Exponential Amplitude Loading . . . 79
4.2 Integration Amplitude Loading . . . 81
4.3 Experiments . . . 83
4.3.1 Experimental Setup and Input Parameters . . . 84
4.3.2 Results . . . 84
4.4 Extension to Path-Dependent Options . . . 85

5 Conclusions and Outlook 89

A Source Code and Scripts 93
A.1 QSVT for Credit Risk . . . 93
A.1.1 MATLAB Code for Computing the Polynomial Approximation 93
A.1.2 MATLAB Code for Computing the Phases . . . 96
A.1.3 Python Code to Generate the State Preparation Circuit using QSVT . . . 98

Nomenclature 103

Bibliography 105

List of Tables

1.1 Comparison of different qubit technologies, their advantages, disadvantages, and leading industry players. Source: [31]. 8
2.1 Summary of main types of derivatives. 50
3.1 Input parameters used in the simulation experiments. 72
3.2 Comparison of circuit depth scaling with prior approaches 75
3.3 IQAE estimates across bisection search iterations, with corresponding intervals. 76
4.1 Assets' initial price. 84
4.2 Problem's parameters. 84
4.3 Estimated option price and confidence intervals for the two amplitude loading methods. 85

List of Figures

1.1 Basic quantum computation represented using the circuit model. . . 3
1.2 Bloch sphere representation of a generic state $|\psi\rangle$. 7
1.3 Bloch sphere representations before and after the application of the X gate. The X gate acts as a 180° rotation about the x–axis. . . . 9
1.4 Bloch sphere representations before and after applying the Y gate. The Y gate performs a 180° rotation about the y–axis, which in this projection inverts the vertical component of the Bloch vector. . . . 10
1.5 Bloch sphere representations before and after applying the Z gate. The Z gate corresponds to a 180° rotation about the z–axis, which in this projection flips the y component while leaving the z component unchanged. 11
1.6 Circuit representation of a controlled-NOT gate. In this instance, the top qubit will be the *control* and the bottom qubit will be the *target*. 14
1.7 Example circuit for a 3 qubit QFT implementation. 17
1.8 An apparently harmless quantum circuit with a Hadamard, a X and a controlled Z-rotation gate. 18
1.9 Representation of a general controlled unitary operation. 19
1.10 General phase kickback circuit. 19
1.11 Slightly modified phase kickback circuit. 20
1.12 Circuit to obtain a guess for the second bit of the phase θ. 21
1.13 Quantum Phase Estimation circuit. 22
1.14 Circuit for Shor's algorithm. 25
1.15 Quantum circuit implementing Grover's algorithm on n qubits. . . . 27
1.16 Visualization of the Grover subroutine 29
1.17 Circuit for the Quantum Amplitude Estimation (QAE) algorithm. The top register (first m qubits) is used for phase estimation via inverse QFT, while the bottom-most qubit acts as the *target qubit*, where the probability amplitude of interest is encoded. The operator $\mathcal{A}$ prepares a state such that measuring this qubit reflects the success probability a. 31
1.18 Overview of Qiskit's software architecture. Source: [45]. 36

1.19 Qmod code example as an external DSL used inside the Classiq platform [35]. 40
2.1 High level overview of quantum algorithms for financial problems. . 43
2.2 Graphical representation of the efficient frontier. In the image the frontier is a curve that represents the risk and return values arising from different combinations of assets. 44
2.3 Graphical illustration of Value at Risk (VaR) for a given confidence level. The figure shows the return distribution of a portfolio with the VaR threshold marked, highlighting the extreme losses that, beyond the expected loss (EL), determine the economic capital needed to buffer against unforeseen losses. 46
2.4 High-level circuit comprising all the components of the state preparation $\mathcal{A}$ used in the original implementation. Source: [28]. 56
2.5 Example of the weighted adder operator implemented in Qiskit [84]. 57
2.6 Example of the comparator operator implemented in Qiskit [43]. . . 58
2.7 Example of the uncertainty operator $\mathcal{U}$. Here $n_z = 2$ and $K = 2$. . . 58
3.1 Overview of the Credit Risk Assessment pipeline. The process starts with the calculation of exposures based on obligor data and asset correlations. Monte Carlo simulations then generate market scenarios and simulate defaults to estimate the loss distribution. Finally, key risk metrics are computed and compiled into regulatory reports. . . 64
3.2 Schematic circuit for the amplitude loading operator F. The dashed box encloses the controlled Y-rotations applied to the target qubit, which encode the loss information from the counterparty registers. . 69
3.3 Polynomial approximation of the threshold function $\mathcal{T}_\mu$ computed using QSPPACK. The high-degree polynomial ensures that the function is accurately represented for the quantum circuit implementation. 73
3.4 Bisection search process for determining the Value-at-Risk. The graph shows how the iterative quantum amplitude estimation converges toward the expected VaR, validated against classical benchmarks. 74
4.1 Schematic representation of the option pricing workflow. The first step is the definition of the underlying and relative correlations, followed by Monte Carlo simulation. Next, we have the payoff computation, risk-neutral valuation, and discounting to arrive at the final option price. 78
4.2 Circuit implementing the direct exponential amplitude loading. . . . 81
4.3 Circuit implementing the integration amplitude loading. 83
4.4 Estimated payoff (with relative confidence intervals) using the two proposed methodologies for amplitude loading and the IBM QASM simulator. The dotted line represents the classically computed expected value (20.20$). 86

Chapter 1

Introduction

1.1 Overview

The concept of leveraging quantum principles to develop a new computational paradigm forms the foundation of what we commonly refer to as *quantum computing*. This is an intriguing idea that has led to the development of a field of research currently at the frontier of modern computational science. Unlike classical computers that operate on bits with binary states, quantum computers use qubits, which can exist in a superposition of states. This capability, combined with phenomena like entanglement and interference, enables quantum systems to process and encode vast amounts of information in ways that were previously unimaginable.

This work focuses on the analysis and design of quantum algorithms tailored for financial applications. In fact, classical approaches are often not suitable for dealing with the complexity inherent to several computational problems pertaining to the domain of quantitative finance. In this opening chapter, we present the foundation of quantum computing by briefly discussing its basic set of elements: qubits, quantum gates, and the circuit model. We detail how simple operations can be orchestrated into intricate circuits that perform sophisticated computations.

Building on these premises, subsequent sections introduce key quantum algorithms that form the backbone of quantum computing. We describe techniques such as the Quantum Fourier Transform, Quantum Phase Estimation, Shor's algorithm to factor integers, and Grover's search algorithm (together with its generalizations). We show how these approaches can solve specific problems more efficiently than classical algorithms. These foundational algorithms are not only of theoretical interest; they constitute the building blocks for practical applications in finance, such as credit risk analysis, and derivative pricing.

By bridging the gap between quantum theory and financial practice, this dissertation aims to demonstrate both the potential and current challenges of implementing quantum algorithms in the financial industry. In doing so, the objective is to contribute to a growing body of research that seeks to harness the power of

quantum computing for real world problem solving.

1.2 Contributions

The primary objective of this dissertation is to investigate and enhance quantum algorithmic frameworks for key problems in quantitative finance, with a particular focus on credit risk analysis and derivative pricing. While the initial chapters provide a comprehensive overview of quantum computing and its foundational tools, the novel contributions of this work are concentrated in Chapters 3 and 4. The main contributions can be summarized as follows:

1. **Quantum Credit Risk Assessment via QSVT:** In Chapter 3, we introduce an improved method for estimating Value-at-Risk (VaR) based on Quantum Singular Value Transformation (QSVT). This approach leverages high-degree polynomial approximations to implement threshold functions on quantum circuits, enabling more precise and scalable amplitude encoding for credit loss distributions. Compared to existing quantum Monte Carlo methods, this technique maintains the accuracy of state preparation while improving depth scaling.

2. **Two Methods for Amplitude Loading in Derivative Pricing:** Chapter 4 presents and compares two distinct amplitude loading techniques to simulate payoff distributions for quantum Monte Carlo derivative pricing: (i) *Direct exponential amplitude loading*, which encodes the payoff function directly into qubit amplitudes, and (ii) *Integration-based loading*, which allows for modeling piecewise and path-dependent payoffs. Both methods are analyzed in terms of circuit structure, resource requirements, and estimation accuracy.

3. **Quantitative Evaluation on Simulated Hardware:** For both credit risk and option pricing problems, we provide simulation-based performance evaluations, including convergence behavior and comparisons with classical benchmarks to validate correctness.

4. **General Framework for Financial Quantum Algorithms:** Beyond individual algorithmic improvements, this thesis provides a modular framework that can be extended to other financial problems, such as stress testing, capital allocation, or exotic derivative pricing. The tools and strategies introduced can serve as building blocks for more advanced quantum-financial toolkits.

In summary, the contributions of this thesis lie not only in the adaptation of existing quantum techniques to finance but in the design of new, scalable, and resource-efficient quantum procedures tailored to concrete financial use cases.

1.3 Computing With Qubits

This chapter is an attempt (doomed to fail) to provide a concise and comprehensive overview of the origins and current state of quantum computing.

Quantum computing is a vast and rapidly evolving field of research; nonetheless, it seems appropriate to outline at least the fundamental principles and foundational concepts of the discipline.

1.3.1 Wires and Gates

The quantum circuit model is the standard framework for describing quantum computations, mirroring the classical circuit model while incorporating uniquely quantum mechanical phenomena such as superposition and entanglement. In this model, *wires* represent the transmission of quantum information in the form of quantum bits (defined in Section 1.3.3), while *gates* symbolize unitary operations that transform these states.

Historically, the concept of quantum circuits evolved from early ideas in quantum physics and computer science. Richard Feynman's visionary work in the early 1980s introduced the notion of simulating physical systems with quantum devices, highlighting the potential of quantum mechanics to solve problems that are intractable for classical computers [32]. David Deutsch further extended these ideas in 1985 by formalizing the quantum Turing machine and the circuit model in his seminal paper, "*Quantum Theory, the Church-Turing Principle and the Universal Quantum Computer*" [20]. This work laid the groundwork for understanding quantum computation in terms of discrete gate operations applied sequentially on qubits.

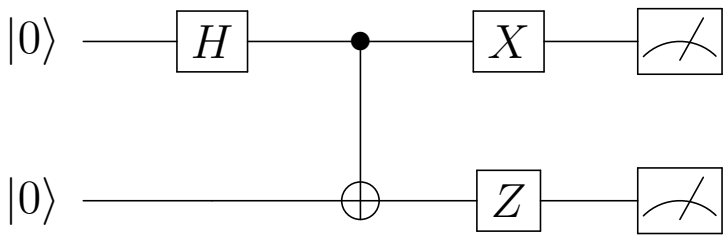


Figure 1.1: Basic quantum computation represented using the circuit model.

In a quantum circuit, wires are drawn as horizontal lines that carry qubits from the input to the output. They provide a clear visual representation of the evolution of the quantum state over time. Along these wires, various quantum gates are applied. Unlike classical gates that perform irreversible Boolean operations, quantum gates are inherently reversible and are described by unitary matrices. Figure 1.1 illustrates a simple example of a quantum circuit with two qubits. The fundamental set of gates for the circuit model undoubtedly includes multi-qubit

operations. As we will see in later sections, these gates can give rise to correlations that cannot be explained within the classical framework.

A fundamental aspect of the circuit model's is its universality. In fact, a limited set of quantum gates can be arranged to approximate any unitary operation with an arbitrarily high degree of accuracy. This universality theorem, as established by Barenco et al. [3], forms a cornerstone of quantum computation theory.

Readers who are interested in a more detailed exposition of the quantum circuit model are encouraged to consult [61], colloquially referred to as *"Mike and Ike"*. Their work offers a thorough examination of the mathematical principles underpinning quantum circuits, along with insights into their initial experimental realizations and relevant historical context.

1.3.2 Noteworthy Notations

Dirac notation, also known as the *bra-ket notation*, is often used to describe quantum states. It represents states as vectors in a complex vector space with a *bra* $\langle\phi|$ representing a row vector and a *ket* $|\psi\rangle$ representing a column vector. The inner product of two states is denoted $\langle\phi|\psi\rangle$.

The reason behind the use of vector notation lies in the fact that quantum mechanics is founded on a rigorous mathematical framework in which quantum states are represented as vectors in a *Hilbert space*. A Hilbert space is a complete, complex vector space equipped with an inner product.

In quantum computing, every qubit is represented as a vector in a two-dimensional Hilbert space, and systems of multiple qubits reside in tensor products of these spaces. Within this framework, operators act on quantum states to model physical processes and measurements. Unitary operators are of particular importance because they describe the evolution of closed quantum systems. A linear operator U is said to be unitary if it satisfies the condition:

$$U^\dagger U = I$$

where $U^\dagger$ denotes the conjugate transpose (or Hermitian adjoint) of U and I is the identity operator. Unitary operators preserve the inner product, meaning they conserve the norm of any state they act upon. This preservation of probability is analogous to a rotation in classical Euclidean space, ensuring that quantum evolution is reversible. In practice, every quantum gate in a circuit is represented by a unitary operator.

On the other hand, *Hermitian operators* (or self-adjoint operators) are defined by the equality:

$$H = H^\dagger$$

This property guarantees that all eigenvalues of a Hermitian operator are real numbers. The eigenstates of a Hermitian operator form an orthonormal basis for the Hilbert space, which means that any quantum state can be expressed as a superposition of these eigenstates. This underpins the probabilistic nature of quantum measurements, as the squared magnitudes of the coefficients in such a decomposition yield the probabilities of observing the corresponding eigenvalues.

Tensor Product

Tensor products are an essential tool for describing composite systems. If we have two quantum systems, one with Hilbert space $\mathcal{H}_A$ and another with Hilbert space $\mathcal{H}_B$, the state of the combined system is represented in the tensor product space $\mathcal{H}_A \otimes \mathcal{H}_B$.

This construction allows us to capture the full range of correlations between subsystems, including *entanglement*, a uniquely quantum phenomenon (see Section 1.3.5).

Given two states $|\psi\rangle \in \mathcal{H}_A$ and $|\phi\rangle \in \mathcal{H}_B$, their tensor product is written as $|\psi\rangle \otimes |\phi\rangle$. For example, if

$$|\psi\rangle = a|0\rangle + b|1\rangle \quad \text{and} \quad |\phi\rangle = c|0\rangle + d|1\rangle,$$

then the state of the composite system becomes

$$|\psi\rangle \otimes |\phi\rangle = ac\,|00\rangle + ad\,|01\rangle + bc\,|10\rangle + bd\,|11\rangle.$$

This result is a vector in a four-dimensional Hilbert space, even though each individual state lives in a two-dimensional space.

The tensor product is a *bilinear* operation, meaning it distributes over addition and respects scalar multiplication. This property is crucial when expressing superpositions of multi-qubit states. For instance, when combining more than two systems, tensor products are applied iteratively, allowing us to describe quantum circuits and algorithms involving several qubits.

Modular Arithmetic

We refer to modular arithmetic as a system of arithmetic for integers, where numbers "wrap around" upon reaching a certain value, the modulus. In this framework, we say that two integers a and b are congruent modulo N if their difference is an integer multiple of N. This is expressed as

$$a \equiv b \pmod{N}$$

which means that N divides $a - b$.

Modular arithmetic forms the backbone of many number-theoretic algorithms and is crucial in cryptography. We will need it when introducing Shor's algorithm, that leverages modular arithmetic to perform efficient period-finding, which is key to factoring large numbers. There, operations such as modular exponentiation:

$$a^x \mod N$$

are central to the algorithm's design and performance.

1.3.3 The Quantum Bit

At an abstract mathematical level, the quantum bit or *qubit* differs from the bit traditionally used in classical computation in that it is not limited to assuming one of two possible states (0 and 1). Instead, it can exist in a *superposition* of these states during the computational process and up to the final reading of the output:

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

This indicates that any possible linear combination of this two states, which form the so-called *computational basis*, constitutes a possible state of the system represented by our qubit. The superpositions are exactly these linear combination. Note that α and β are complex numbers, also called *probability amplitudes* while the states of the computational basis $|0\rangle$ and $|1\rangle$ are simply column vectors expressed using the Dirac notation described in Section 1.3.2.

However, the surprises that the qubit holds for us are not yet over. Suffice it to say that if we observe our qubit to determine the superposition it actually occupies (that is, we wish to determine the coefficients α and β in Equation 1.1), we will only observe one of the two states of the computational basis, with probabilities $|\alpha|^2$ for $|0\rangle$ and $|\beta|^2$ for $|1\rangle$, respectively. Since α and β are complex numbers and $|\alpha|^2+|\beta|^2 = 1$, it is possible to employ spherical coordinates to obtain an alternative formulation of Equation 1.1:

$$|\psi\rangle = \cos\frac{\theta}{2}\,|0\rangle + e^{i\phi}\sin\frac{\theta}{2}\,|1\rangle\,. \tag{1.2}$$

This representation is well-suited for a compact and effective visualization (although not easily generalizable to systems with multiple qubits), using what's called *Bloch sphere*. Figure 1.2 shows a generic state $|\psi\rangle$ as a vector on this sphere.

At an implementation level, realizing a qubit requires exploiting the quantum mechanical properties of physical systems with well-separated energy levels. The core idea is to isolate a two-level system whose states are energetically distinct from higher excited states, thereby minimizing unwanted transitions due to thermal excitations or environmental noise. This energy separation is crucial for preserving coherence and enabling precise control over the qubit state.

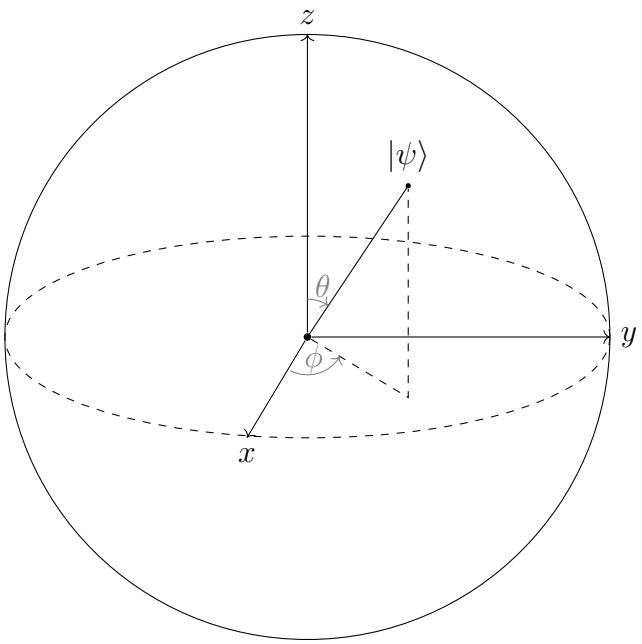


Figure 1.2: Bloch sphere representation of a generic state $|\psi\rangle$.

For instance, in *superconducting qubits*, circuits incorporating Josephson junctions create an anharmonic potential that ensures the two lowest energy states are well-separated from the rest of the spectrum. This anharmonicity allows the circuit to behave effectively as a two-level system, with microwave pulses used to manipulate the qubit state without inducing transitions to higher levels [16, 23].

Similarly, *trapped-ion qubits* exploit the internal energy levels of ions—typically hyperfine or Zeeman split states—to define the qubit. The isolation provided by electromagnetic traps and the intrinsic separation of these energy levels ensure that laser-induced transitions occur only between the desired qubit states, yielding long coherence times [9].

Other physical systems follow analogous principles. In *semiconductor quantum dots*, for example, the electron spin serves as a qubit, with an external magnetic field splitting the spin-up and spin-down states to create the necessary energy gap [49]. A minimal overview of qubit modalities is presented in Table 1.1.

For a more comprehensive treatment of qubit implementations and the importance of well-separated energy levels, readers are referred to [61] as well as the review articles by Clarke and Wilhelm [16] and Devoret and Schoelkopf [23].

1.3.4 One-Qubit Gate Operations

The usefulness and effectiveness of the representation provided by the Bloch sphere are also due to the fact that operations involving just one single qubit can

Qubit Technology	Summary	Providers
Superconducting	**Pros:** High gate speeds and fidelities. Can leverage standard lithographic processes. **Cons:** Requires cryogenic cooling, short coherence times, and microwave interconnect frequencies still not mastered.	Rigetti, Google, IBM Q, QuTech, IQM, QCI, Alice & Bob, Atlantic Quantum, Anyon Technologies
Trapped Ions	**Pros:** Extremely high gate fidelities and long coherence times. No extreme cryogenic cooling required. Ions are highly stable and consistent. **Cons:** Slow gate times. Lasers are hard to align and scale. Ultra-high vacuum required.	IonQ, AQT, Quantinuum, Oxford Ionics, Universal Quantum
Photonic	**Pros:** Extremely fast gate speeds and promising fidelities. No cryogenics or vacuums required. Small overall footprint. Can leverage existing CMOS fabs. **Cons:** Noise from photon loss. Each program requires its own chip. Photons don't naturally interact, making 2-qubit gate implementation challenging.	PsiQuantum, Xanadu, QuiX Quantum, ORCA Computing, Quandela
Neutral Atoms	**Pros:** Long coherence times. Atoms are perfect and consistent. Strong connectivity, including beyond two-qubit gates. No external cryogenics required. **Cons:** Requires ultra-high vacuums. Laser scaling is challenging. Low sampling rate.	Atom Computing, Infleqtion, QuEra, Planqc, Pasqal
Others:	**Silicon Spin, Quantum Dots, Topological, Diamond Vacancies**	Quantum Brilliance, Azure Quantum, Intel, Diraq

Table 1.1: Comparison of different qubit technologies, their advantages, disadvantages, and leading industry players. Source: [31].

usually be seen as rotations around the axes of the sphere itself.

X Gate

For instance, the well-known X gate is nothing more than a rotation of the qubit state around the x-axis of the Bloch sphere by an angle of π radians (see

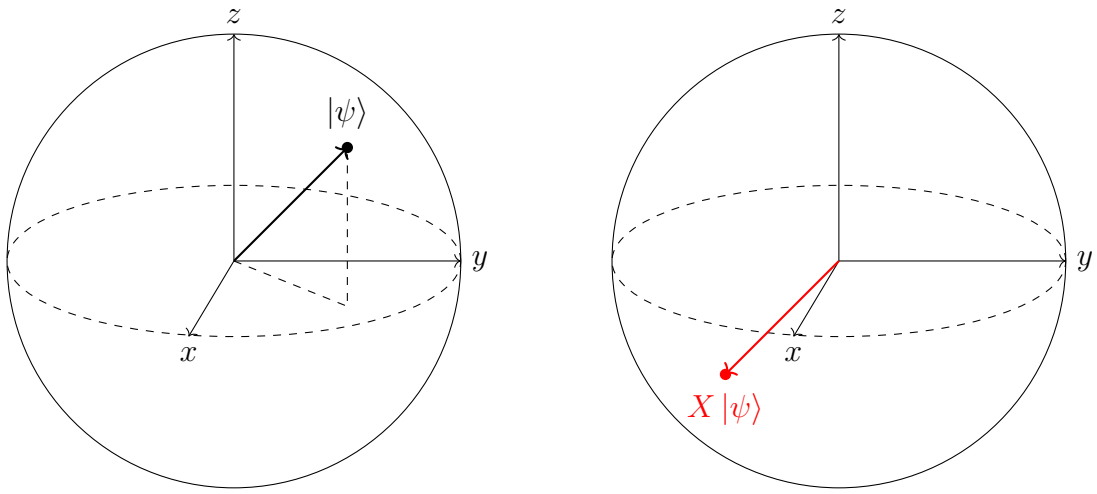
Figure 1.3). The effect of such a gate on the state described in Equation 1.1 would be to interchange the coefficients α and β:

$$X \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} \beta \\ \alpha \end{bmatrix}. \tag{1.3}$$

So that $X|0\rangle = |1\rangle$ and $X|1\rangle = |0\rangle$. This implies that X can be represented as a specific *unitary* and *Hermitian* matrix (see Section 1.3.2)

$$X \equiv \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}. \tag{1.4}$$

Two observations should become evident: firstly, the analogy with the classical *NOT* gate; secondly, the fact that the X gate is its own inverse.



Before X gate

After X gate

Figure 1.3: Bloch sphere representations before and after the application of the X gate. The X gate acts as a 180° rotation about the x–axis.

Y Gate

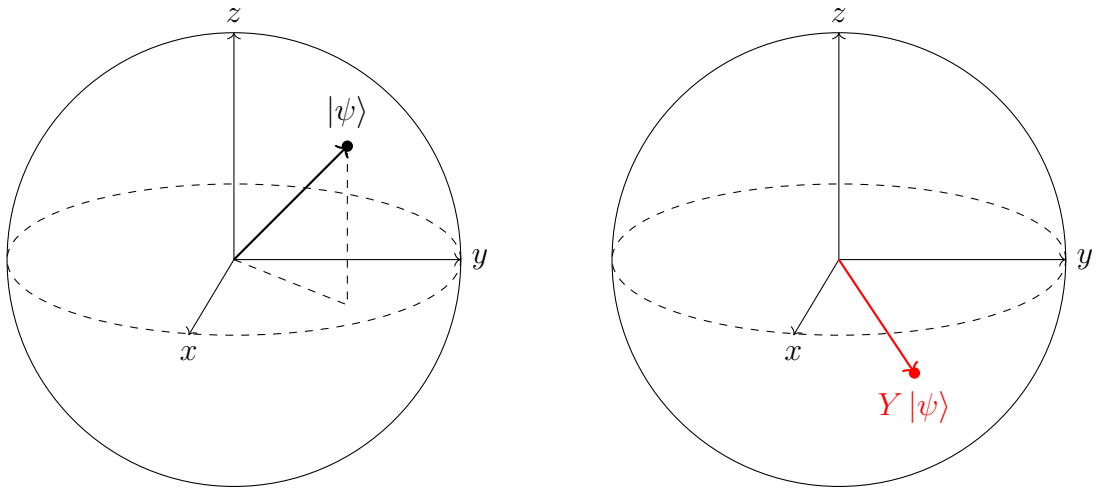
The Y gate is another important quantum gate that corresponds to a rotation of the qubit state around the y-axis of the Bloch sphere by an angle of π radians (as shown in Figure 1.4). The effect of the Y gate is to introduce a phase change and interchange the coefficients α and β, with one of them acquiring a phase factor:

$$Y \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} -i\beta \\ i\alpha \end{bmatrix}. \tag{1.5}$$

Thus, it follows that $Y|0\rangle = i|1\rangle$ and $Y|1\rangle = -i|0\rangle$. This implies that Y can be represented as a specific *unitary* matrix:

$$Y \equiv \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}. \tag{1.6}$$

Similar to the X gate, the Y gate is also its own inverse, but it differs by introducing a complex phase.



Before Y gate

After Y gate

Figure 1.4: Bloch sphere representations before and after applying the Y gate. The Y gate performs a 180° rotation about the y–axis, which in this projection inverts the vertical component of the Bloch vector.

Z Gate

Remarkably, any unitary two-by-two matrix represents a valid single-qubit gate operation. Another pertinent example is the so-called Z gate, whose unitary matrix is given by:

$$Z \equiv \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}. \tag{1.7}$$

This gate performs a rotation of the quantum state by π around the z-axis on the Bloch sphere (Figure 1.5) and, in its matrix form, is also known as the Z *Pauli operator* and denoted with σ^z. Again, Z is its own inverse and it is also referred to as *phase-flip* gate, since

$$Z \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} \alpha \\ -\beta \end{bmatrix}. \tag{1.8}$$

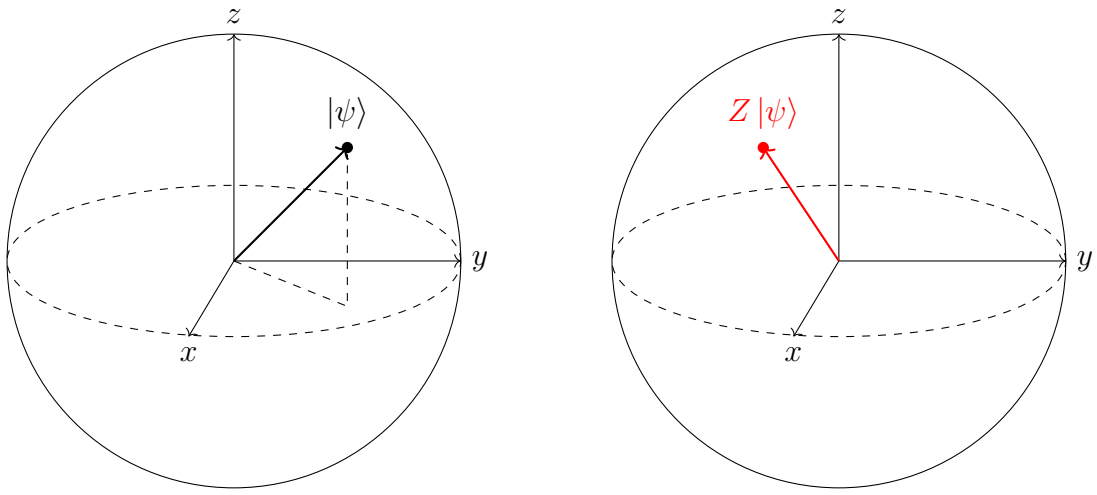
Hadamard Gate

Another notable two-by-two unitary matrix is the one representing the *Hadamard* or H gate

$$H \equiv \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}. \tag{1.9}$$

To determine the effect of a gate on the state $|0\rangle$, one needs only to examine the first column of the unitary matrix, and the second column for the effect on $|1\rangle$. In the case of the Hadamard gate, this results in the transformation of $|0\rangle$ into the superposition $\frac{|0\rangle+|1\rangle}{\sqrt{2}}$ and $|1\rangle$ into the superposition $\frac{|0\rangle-|1\rangle}{\sqrt{2}}$. These specific superpositions provide an alternative basis to the computational one, and hence the states they comprise are also denoted as

$$\begin{aligned} |+\rangle &\equiv \frac{|0\rangle + |1\rangle}{\sqrt{2}} \\ |-\rangle &\equiv \frac{|0\rangle - |1\rangle}{\sqrt{2}}. \end{aligned} \tag{1.10}$$



Before Z gate After Z gate

Figure 1.5: Bloch sphere representations before and after applying the Z gate. The Z gate corresponds to a 180° rotation about the z–axis, which in this projection flips the y component while leaving the z component unchanged.

RZ Gate

The parametrized z-rotation gate or RZ gate can be considered as a generalization from phase flip (as seen with the Z gate) to phase changing effect:

$$RZ_\phi \equiv \begin{bmatrix} 1 & 0 \\ 0 & e^{\phi i} \end{bmatrix}. \tag{1.11}$$

In this case, the effect on $|0\rangle$ will be null (as can be deduced from the first column of the matrix), while $RZ_\phi |1\rangle = e^{i\phi} |1\rangle$.

It should be noted that by varying the parameter ϕ, it is possible to implement an infinite number of different gates. However, the gates obtained with certain specific values of this parameter are frequently used and of particular significance:

$$S \equiv \begin{bmatrix} 1 & 0 \\ 0 & e^{\frac{\pi i}{2}} \end{bmatrix}. \tag{1.12}$$

$$T \equiv \begin{bmatrix} 1 & 0 \\ 0 & e^{\frac{\pi i}{4}} \end{bmatrix}. \tag{1.13}$$

RX and RY gate

Finally, an analogous concept to what was done for the RZ gate in relation to the Z gate can be considered to show the relationship between the RX and RY gates and X and Y:

$$RY_\phi \equiv \begin{bmatrix} \cos\frac{\phi}{2} & -\sin\frac{\phi}{2} \\ \sin\frac{\phi}{2} & \cos\frac{\phi}{2} \end{bmatrix} \equiv e^{-i\frac{\phi}{2}Y}. \tag{1.14}$$

$$RX_\phi \equiv \begin{bmatrix} \cos\frac{\phi}{2} & -i\sin\frac{\phi}{2} \\ -i\sin\frac{\phi}{2} & \cos\frac{\phi}{2} \end{bmatrix} \equiv e^{-i\frac{\phi}{2}X}. \tag{1.15}$$

1.3.5 Handling Multiple Qubits

A single qubit, despite its impressive properties, is insufficient to convey the potential of quantum computation. Beyond the phenomenon of superposition, it is essential to consider other phenomena that necessitate the transition to systems with multiple qubits. In particular, it is impossible to avoid talking about the *entanglement* one.

Practically, we can observe the entanglement phenomenon existing between two or more qubits when, upon measurement, we verify a correlation between the states that we are observing. However, the entanglement relationship is not limited to a mere correlation between the measurements of different qubits. To understand the phenomenon at a deeper level, it is useful to consider how we usually represent a system composed of two qubits that are in the states $|\psi_1\rangle$ and $|\psi_2\rangle$ respectively.

We use a tensor product of the two subsystems (Section 1.3.2 contains a primer on tensor products):

$$|\psi_1\psi_2\rangle \equiv |\psi_1\rangle \otimes |\psi_2\rangle \,. \tag{1.16}$$

As an example, consider two qubits that are both initialized in the state $|0\rangle$. For these two qubits, the combined state is expressed as:

$$|00\rangle = |0\rangle \otimes |0\rangle.$$

The notation $|00\rangle$ explicitly indicates that the first qubit is in the state $|0\rangle$ and the second qubit is also in the state $|0\rangle$.

In general, new quantum states can be obtained by creating linear combinations of these composite states. Notably, the resulting state often cannot be mathematically decomposed as the tensor product of two states. That is why entanglement is strictly related to the concept of superposition applied to multi-qubit systems. A good examples of this is the so-called *Bell state* or *EPR pair*:

$$\frac{|00\rangle + |11\rangle}{\sqrt{2}} \tag{1.17}$$

which can be clearly be seen as a linear combination of the states $|00\rangle$ and $|11\rangle$.

Famously, quantum entanglement allows particles to be linked regardless of the space between them. This concept is so counterintuitive that even Einstein was baffled by it. He thought that a quantum particle could secretly hide a predetermined outcome of a measurement, but that quantum mechanics fails to capture this information. This brought Einstein to believe that quantum mechanics was incomplete, meaning that it was missing some variables necessary to fully describe a system [30]. Einstein thought that these so-called hidden variables, if found, would resolve some of the seemingly paradoxical and probabilistic aspects of quantum mechanics. In contrast, Max Born argued that the probabilistic description given by quantum mechanics is all that is specified by nature [11]. The particle has no definite state before it is measured.

Then, in 1964, the physicist John Bell examined the statistical correlations between measurements of a pair of entangled particles [5]. If, prior to measurement, the states of the particles were determined by some unknown hidden variables, then Bell found that their correlations had an upper limit. On the other hand, when he calculated the correlations predicted by quantum mechanics, he found that they violated this limit. Thus, Bell's theorem provides a framework to test the predictions of quantum mechanics against the concept of local hidden variables, as proposed by Einstein. All that remained was to actually experimentally test these correlations with real particles to see whether they obeyed the hidden variable statistics or violated that rule. These tests were performed with increasing precision and sophistication over the years [2] and the violation of Bell's inequality in such experiments underscores the insufficiency of local hidden variables in accounting for

quantum phenomena, challenging Einstein's postulation of pre-determined particle states.

At a practical level, it will be useful to remember that we can write a general quantum state $|\phi\rangle$ on n qubits as

$$|\varphi\rangle = \sum_{j=0}^{2^n-1} a_j |j\rangle_n = \sum_{j=0}^{N-1} a_j |j\rangle_n. \tag{1.18}$$

Where the a_j are N amplitudes corresponding to the N states in the standard decimal notation basis.

1.3.6 Multiple-Qubit Gate Operations

It is common in quantum circuits to encounter gates that operate on more than a single qubit simultaneously. Consider, for instance, the *controlled gates*, which apply a given operation to the target qubit(s) only if the control qubits are in the state $|1\rangle$.

CX Gate

In quantum computing, the logical gate used to establish entanglement between two qubits is the so-called CX (controlled-X) gate:

$$CX \equiv \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}. \tag{1.19}$$

This gate is also known as the controlled-NOT or CNOT gate.

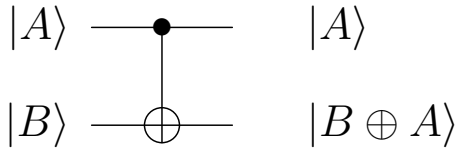


Figure 1.6: Circuit representation of a controlled-NOT gate. In this instance, the top qubit will be the *control* and the bottom qubit will be the *target*.

One of the two input qubits will serve as the target and the other as the control. Its effect is essentially to apply an X gate on the target qubit conditionally, if the control qubit is in the state $|1\rangle$. Consequently, it will not alter states such as $|00\rangle$ and $|01\rangle$, while it will apply the gate to the target qubit in cases like:

$$\begin{aligned} |10\rangle &\rightarrow |11\rangle \\ |11\rangle &\rightarrow |10\rangle . \end{aligned} \tag{1.20}$$

As can be observed in Figure 1.6, there exists an analogy between the classical XOR gate and the CNOT gate. Essentially, the target qubit (after the gate's application) stores the result of the XOR operation between the control and target qubit.

It is important to note that the CNOT gate, together with the previously introduced single-qubit gates, forms the foundation for all other multi-qubit gates, as *any multi-qubit logic gate can be constructed using a combination of CNOT and single-qubit gates*. This universality arises from two key facts:

1. Single-qubit gates can implement any rotation on an individual qubit. Mathematically, they generate the group SU(2), which means that any unitary transformation acting on a single qubit can be realized with a proper sequence of these gates.

2. The CNOT gate is the simplest two-qubit gate that can create entanglement that allows for correlations between qubits that cannot be produced by single-qubit operations alone.

Together, these properties ensure that the set consisting of all single-qubit gates plus the CNOT gate is universal. In other words, any unitary operation U acting on n qubits (i.e., any element of $\mathrm{SU}(2^n)$) can be decomposed into a sequence of single-qubit operations and CNOT gates. This result is formally established in the universality theorem for quantum gates, as demonstrated in [3]. Moreover, the Solovay-Kitaev theorem [19] guarantees that such decompositions can approximate any desired unitary to arbitrary precision using only a finite set of gates drawn from this universal set.

1.4 Foundational Algorithms

In this section, we explore the foundational quantum algorithms that have been instrumental in developing the field. Rather than merely cataloging a series of methods, we briefly present the conceptual breakthroughs that underpin these algorithms. For instance, the ability to perform a Fourier transform exponentially faster on a quantum computer than on a classical one is not just a technical improvement—it exemplifies how quantum parallelism can be harnessed to tackle problems such as integer factorization.

These algorithms serve as a bridge between the abstract mathematical structures of quantum theory and practical applications such as cryptography, optimization, and simulation. Their development has spurred a deeper understanding of quantum complexity. In essence, the foundational algorithms discussed here are not just isolated techniques; they form a somewhat cohesive framework that illustrates the transformative potential of quantum computation.

The discussion that follows will reveal how the interplay between superposition, interference, and entanglement gives rise to algorithms that are intrinsically different from classical counterparts.

1.4.1 Quantum Fourier Transform

The Quantum Fourier Transform (QFT) is a linear transformation analogous to the classical discrete Fourier transform. QFT takes as input a vector of complex numbers (the $N = 2^n$ states' probability amplitudes, where n is the amount of qubits on which we desire to perform QFT) and outputs the transformed data, still a vector of complex numbers:

$$b_j \equiv \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} a_k e^{2\pi i j k/N} \equiv \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} a_k \omega^{jk}. \tag{1.21}$$

Where $\omega = e^{\frac{2\pi i}{N}}$ is a primitive N^{th} root of unity. The Quantum Fourier Transform (QFT) operates by transitioning between two distinct bases: the computational (often referred to as the Z) basis, and the Fourier basis. It is imperative to recognize that each state within the Z basis has an equivalent state in the Fourier basis, facilitated by the QFT.

An elementary demonstration of this transformation is observed in the context of a single qubit; here, $\mathrm{QFT}\,|0\rangle = |+\rangle$ and $\mathrm{QFT}\,|1\rangle = |-\rangle$. When juxtaposing these transformations with the formula referenced in 1.9, it becomes evident that, in this specific scenario, the QFT is effectively implemented by the Hadamard gate H.

More in general, from 1.18 we know that for a generic state $|\phi\rangle$ we have essentially a complex-valued function $j : \{0,1,2,\ldots,N-1\} \to a_j$. Then, the QFT of $|\phi\rangle$ is

$$\mathrm{QFT}_n\,|\phi\rangle = \sum_{j=0}^{N-1} a_j\,|j\rangle_n \mapsto \sum_{j=0}^{N-1} b_j |j\rangle_n, \tag{1.22}$$

where b_j is defined above in 1.21. Thus, for a generic N the transformation due to

QFT_n acting on the state $|x\rangle = |x_1 x_2 ... x_n\rangle$ can be derived as follows:

$$\begin{aligned} QFT_N |x\rangle &= \frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} e^{2\pi i x y / 2^n} |y\rangle \\ &= \frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} e^{2\pi i (\sum_{k=1}^{n} y_k / 2^k) x} |y_1 \dots y_n\rangle \\ &= \frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} \prod_{k=1}^{n} e^{2\pi i x y_k / 2^k} |y_1 \dots y_n\rangle \\ &= \frac{1}{\sqrt{N}} \bigotimes_{k=1}^{n} \left(|0\rangle + e^{2\pi i x / 2^k} |1\rangle \right) \\ &= \frac{1}{\sqrt{N}} \left(|0\rangle + e^{\frac{2\pi i}{2} x} |1\rangle \right) \otimes \left(|0\rangle + e^{\frac{2\pi i}{2^2} x} |1\rangle \right) \otimes \dots \otimes \left(|0\rangle + e^{\frac{2\pi i}{2^n} x} |1\rangle \right). \end{aligned} \tag{1.23}$$

It is important to note that an efficient circuit capable of implementing the Quantum Fourier Transform on n qubits is both feasible and well-established (See Figure 1.7 for the circuit implementation of the QFT on 3 qubits). Consequently, we are able to perform the QFT using $O(n^2)$ gates, which would achieve an exponential improvement compared to the best-known classical algorithm for computing the discrete Fourier transform on 2^n elements, which requires $O(n2^n)$ steps [80].

However, a significant challenge arises from the fact that the amplitudes within a quantum computer cannot be directly accessed via measurement. Thus, there is no efficient method for determining the Fourier-transformed amplitudes of the original state. Furthermore, there is generally no efficient way to prepare the original state to be transformed. Therefore, to fully leverage the power of QFT, it is more advantageous to utilize it as a subroutine within a larger quantum algorithm that benefits from internally performing the transformation, although its objective is not directly to output the transformed amplitudes. One great example is Quantum Phase Estimation.

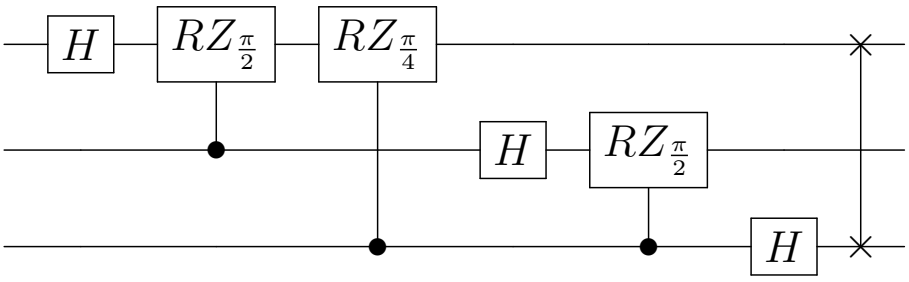


Figure 1.7: Example circuit for a 3 qubit QFT implementation.

1.4.2 Quantum Phase Estimation

Consider a unitary matrix U, which has eigenvalues of the form $e^{i\theta}$ and eigenvectors that constitute an orthonormal basis, satisfying the relation:

$$U\left|\psi\right\rangle = e^{i\theta_\psi}\left|\psi\right\rangle, \tag{1.24}$$

as prescribed by the spectral theorem. If we assume that it is possible to prepare the state $|\psi\rangle$, is there an algorithm available to extract the information contained in the phase θ_ψ, or equivalently, to estimate the eigenvalue of U?

Let us begin by noting that $0 \leq \theta < 2\pi$, thus we can instead reconsider the eigenvalue as $e^{2\pi\theta i}$ where $0 \leq \theta < 1$. Subsequently, we need to introduce a well-known and elegant technique known as phase kickback.

Phase Kickback

Suppose we have a circuit analogous to the one in Figure 1.8.

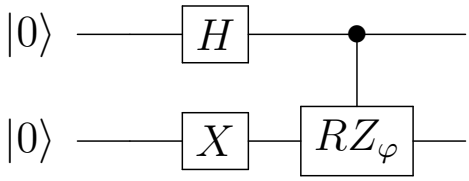


Figure 1.8: An apparently harmless quantum circuit with a Hadamard, a X and a controlled Z-rotation gate.

Prior to the application of the controlled Z-rotation (CRZ_φ), the state of the circuit can be expressed as follows:

$$\frac{\sqrt{2}}{2}(|10\rangle + |11\rangle). \tag{1.25}$$

Note that we follow little-endian convention, where qubit 0 is the least significant bit and appears on the right in tensor products. Subsequently, upon applying the controlled gate, the state transforms to:

$$\frac{\sqrt{2}}{2}(|10\rangle + e^{\varphi i}\,|11\rangle) = |1\rangle \otimes \frac{\sqrt{2}}{2}(|0\rangle + e^{\varphi i}\,|1\rangle) \tag{1.26}$$

The phase induced by the RZ gate, which is conditionally applied to the lower qubit, results in the emergence of the phase $e^{\varphi i}$ in the state of the upper qubit. Notice how the state of the upper qubit remains unaffected. This phenomenon illustrates the concept of phase kickback.

We have already mentioned how the problem of phase estimation deals with unitary operators U. Such operators can be implemented using unitary quantum circuits. We can thus generalize the concept of phase kickback by considering a controlled version of the unitary operator U, as depicted in Figure 1.9 (where the operator is viewed as a large quantum gate). Observe that a quantum circuit similar to the one illustrated can be achieved by integrating a control qubit into the circuit corresponding to the unitary operator U, followed by transforming each gate within that circuit into its controlled variant, wherein the control qubit functions as the gate's control input.

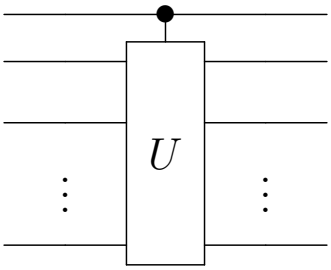


Figure 1.9: Representation of a general controlled unitary operation.

With this, we have all the ingredients for the general phase kickback circuit which is depicted in Figure 1.10.

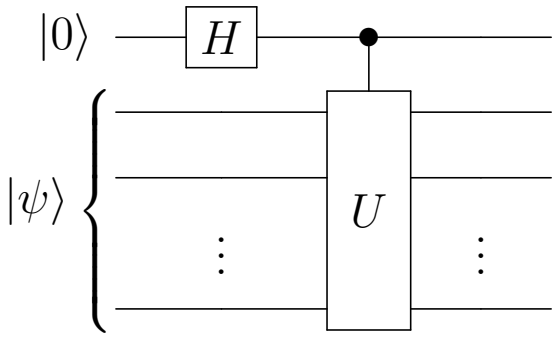


Figure 1.10: General phase kickback circuit.

Where $|\psi\rangle$ is the quantum state corresponding to an eigenvector of U. The circuit is initially in the state

$$|\psi\rangle\,|0\rangle\,. \tag{1.27}$$

But, after applying the first Hadmard, turns into

$$|\psi\rangle\,|+\rangle = \frac{\sqrt{2}}{2}(|\psi\rangle\,|0\rangle + |\psi\rangle\,|1\rangle). \tag{1.28}$$

Next, we apply the controlled U operation, resulting in the state

$$\frac{\sqrt{2}}{2}(|\psi\rangle|0\rangle + (U|\psi\rangle)|1\rangle). \tag{1.29}$$

Being $|\psi\rangle$ an eigenvector of U with eigenvalues of the form $\lambda = e^{2\pi i\theta}$, we can highlight the phase kickback effect on the state

$$\frac{\sqrt{2}}{2}(|\psi\rangle|0\rangle + e^{2\pi i\theta}|\psi\rangle|1\rangle) = |\psi\rangle \otimes \frac{\sqrt{2}}{2}(|0\rangle + e^{2\pi i\theta}|1\rangle). \tag{1.30}$$

Extracting The Phase

It is interesting to study what happens if we add an Hadamard on the control qubit and the we measure it. The circuit become the one in Figure 1.11.

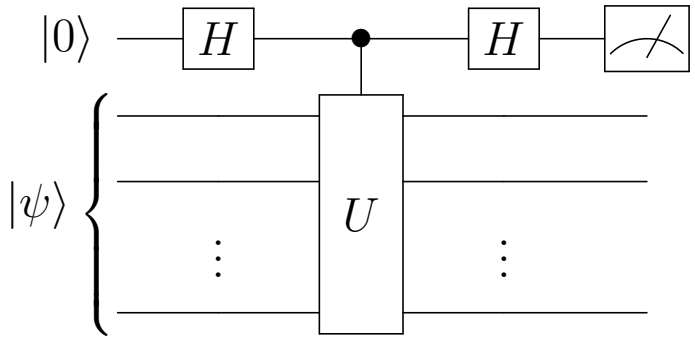


Figure 1.11: Slightly modified phase kickback circuit.

The final state of the circuit will now be

$$|\psi\rangle \otimes (\frac{1+e^{2\pi i\theta}}{2}|0\rangle + \frac{1-e^{2\pi i\theta}}{2}|1\rangle). \tag{1.31}$$

Which means that, once measured, the control qubit will yeld as outcomes 0 and 1 with probabilities:

$$p_0 = |\frac{1+e^{2\pi i\theta}}{2}|^2 = \cos^2(\pi\theta) \tag{1.32}$$

$$p_1 = |\frac{1-e^{2\pi i\theta}}{2}|^2 = \sin^2(\pi\theta). \tag{1.33}$$

We begin to gather some information regarding the phase θ. Specifically, if θ is 0, the measurement outcome will always be 0, and if it is 1/2, the measurement outcome will always be 1. Assuming that θ can only take the values 0 or 1/2, this circuit and its measurement can determine which of the two solutions is correct.

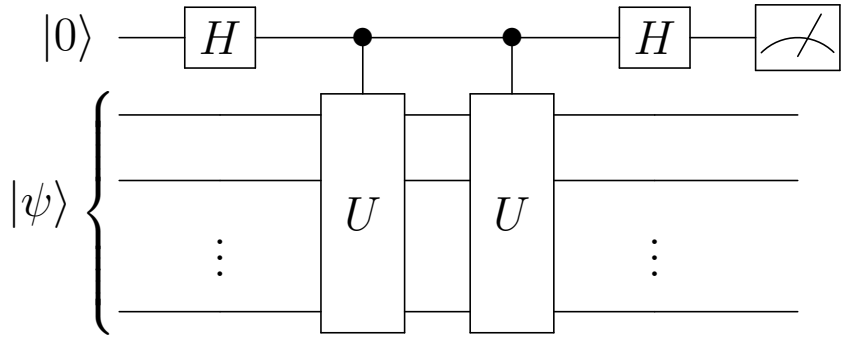


Figure 1.12: Circuit to obtain a guess for the second bit of the phase θ.

Thus, we can interpret the measurement outcome as a *guess of the phase to one (binary) bit of accuracy.*

Note how if we were to measure the state of the control qubit before the final Hadamard gets applied, we would obtain 0 and 1 each with probability $\frac{1}{2}$. The second Hadamard gate allows us to learn something about the phase through the phenomenon of *interference*, allowing θ to affect the output probabilities.

The subsequent step involves the possibility of constructing an analogous circuit in which the estimation is no longer referenced to θ but rather to 2θ. This can be achieved by applying the controlled operation U twice instead of once (or by applying the controlled operation U^2) as shown in Figure 1.12. Effectively, this introduces another guess to one bit of accuracy, but this time for a different bit in the binary representation of θ. In fact, if the binary representation of θ is $\theta = 0.a_1a_2a_3...$ then doubling ϕ shifts the binary point one position to the right: $2\theta = a_1.a_2a_3\ldots$. ($\theta = 1$ gets equated with $\theta = 0$ as we progress around the unit circle, so the bit a_1 does not influence our probabilities), thus effectively providing a guess for the second bit after the binary point.

We would like to be able to combine the information obtained from these two circuits to achieve a more precise estimate of the phase. The mainstream approach to achieve such a result consists in using multiple control qubits, forming what is commonly known as a *counting register*. Each of these qubits controlled the application of a different power of the unitary operator U, with the powers set to double for each new counting qubit. In general, after the application of the Hadamard gates to the n counting qubits, the state will be:

$$\frac{1}{(\sqrt{2})^n}(|0\rangle + |1\rangle)^{\otimes n}. \tag{1.34}$$

Afterwards, we apply the controlled power of U. Before proceeding, note:

$$U^{2^x}|\psi\rangle = U^{2^x-1}U|\psi\rangle = U^{2^x-1}e^{2\pi i\theta}|\psi\rangle = \cdots = e^{2\pi i(\theta 2^x)}|\psi\rangle \tag{1.35}$$

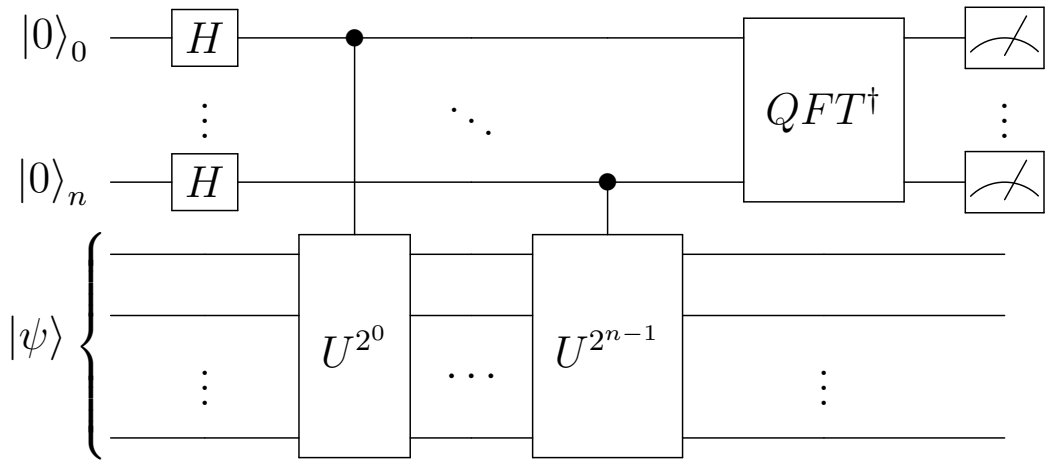


Figure 1.13: Quantum Phase Estimation circuit.

Therefore, after applying all the controlled operations it is reasonable to expect the state:

$$\frac{1}{\sqrt{N}}\left(|0\rangle + e^{2\pi i\theta 2^{n-1}}|1\rangle\right) \otimes \left(|0\rangle + e^{2\pi i\theta 2^{n-2}}|1\rangle\right) \otimes \ldots \left(|0\rangle + e^{2\pi i\theta 2^{0}}|1\rangle\right) \tag{1.36}$$

This formula is remarkably similar to the formulation of the QFT that we derived in 1.23, except for the fact that there θ is replaced by $\frac{x}{2^n}$. Thus, we can read $|2^n\theta\rangle$ in the output register of our phase estimation circuit after applying the inverse of the QFT transformation. The resulting circuit is shown in Figure 1.13.

1.4.3 Shor's Factoring Algorithm

Consider the problem of factoring a number $N = p \cdot q$, where p and q are large prime numbers, and $n = \lceil \log_2 N \rceil$ binary bits are required to represent N. Classically, this is a complex problem that typically involves methods that scale as $O(\exp[c \cdot n^{\frac{1}{3}} \cdot (\log n)^{\frac{2}{3}}])$, where c is a constant that depends on the details of the implementation.

Shor's algorithm [73], when implemented on a quantum computer, resolves this problem with a complexity that scales slightly better than $O(n^3)$. Therefore, the approach is capable of providing an exponential speedup over traditional computation methods and, for this reason, represents one of the application areas where quantum computing generates significant interest. The efficacy of this algorithm is derived from the relationship between the factoring problem and the order-finding problem, wherein the former can be transformed into the latter, which can then be addressed through Algorithm 1. Here, the *order* r of an integer a modulo N is defined as the smallest positive integer such that $a^r \equiv 1 \pmod{N}$. See Section 1.3.2 for more details.

Algorithm 1 Factorization using order finding

Require: $N = p \cdot q$ (product of two primes)
Ensure: $\{p, q\}$

Pick a such that $\gcd(a, N) = 1$
Find the order r of $a \mod N$, i.e., the smallest r such that

$$a^r \equiv 1 \pmod{N}$$

if r is even **then**
    Compute $x \equiv a^{r/2} \pmod{N}$
    if $x + 1 \not\equiv 0 \pmod{N}$ **then**
        Set $p = \gcd(x + 1, N)$, $q = \gcd(x - 1, N)$
        return $\{p, q\}$
    else
        Find another a
    end if
else
    Find another a
end if

In order to implement the quantum circuit corresponding to Shor's algorithm, we first require to define an unitary operation $\mathcal{M}_a$ as such:

$$\mathcal{M}_a |x\rangle = |ax\rangle . \tag{1.37}$$

With $\gcd(a, N) = 1$ and $x \in \{0,1, ..., N-1\}$. The intuition behind Shor's algorithm is that the eigenvalues of $\mathcal{M}_a$ are connected with the order r of a. From this, it follows that the use of phase estimation can lead to an estimation of these eigenvalues and, consequently, the order. The eigenvectors of $\mathcal{M}_a$ take the form:

$$|\psi_j\rangle = \frac{|1\rangle + \omega_r^{-j}|a\rangle + \cdots + \omega_r^{-j(r-1)} |a^{r-1}\rangle}{\sqrt{r}}, \tag{1.38}$$

for each $j \in \{0, ..., r-1\}$. Where $\omega_r = ^{2\pi i/r}$. The corresponding eigenvalues satisfy

$$\mathcal{M}_a |\psi_j\rangle = \omega_r^j |\psi_j\rangle \tag{1.39}$$

Now suppose we have a random eigenvector $|\psi_j\rangle$. We can estimate $\frac{j}{r}$ by performing phase estimation, but we cannot straightforwardly invert it since, in principle, we do not know the value of j. However, we can utilize the *continued fraction algorithm* [36] to estimate the order r. It is a well-known result that the number of qubits required for the phase estimation process, in this specific case, increases linearly with the number n of qubits needed to represent N.

Note that unfortunately j could have common factors with r. To account for this, one can draw independent samples, where $j \in \{0, ..., r-1\}$ is chosen uniformly. In this way it is possible to find r with high probability by computing the least common multiple of the different estimates.

The circuit to implement QPE using $\mathcal{M}_a$ is depicted in Figure 1.14. Notice that we start from the state $|1\rangle$ since we want a random eigenvector and, in our case:

$$|1\rangle = \frac{|\psi_0\rangle + \cdots + |\psi_{r-1}\rangle}{\sqrt{r}}. \tag{1.40}$$

$\mathcal{M}_a$ can be implemented at a cost $O(n^2)$ and more importantly there exist an efficient way to implement $\mathcal{M}_a^k$. In fact, it is sufficient to compute

$$b = a^k \pmod N$$

and then use a circuit for $\mathcal{M}_b$ instead of $\mathcal{M}_a^k$. Therefore, since we require n controlled unitary operations, each incurring a cost of $O(n^2)$, and given that we can neglect the contributions of the QFT and Hadamard operations, the overall complexity will scale as $O(n^3)$ as mentioned above.

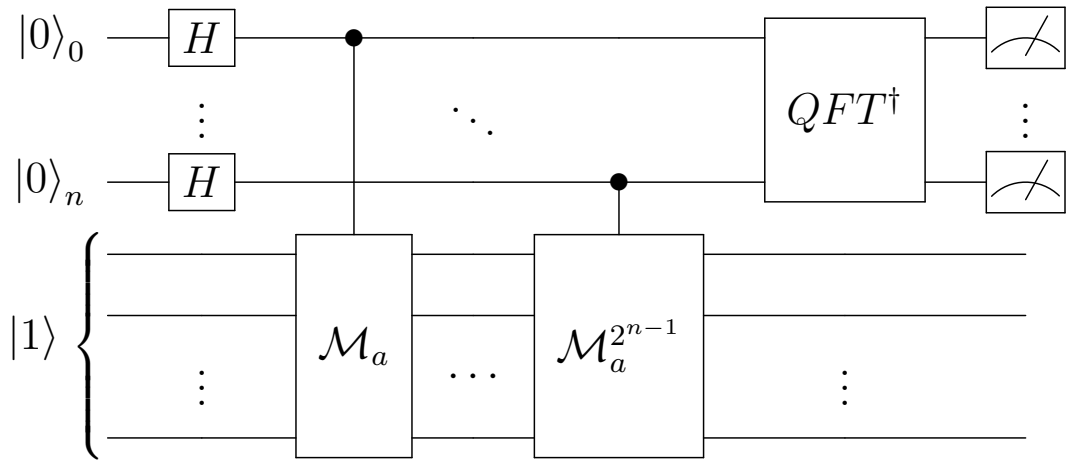


Figure 1.14: Circuit for Shor's algorithm.

1.4.4 Grover's Search Algorithm

In 1996, Lov Grover introduced a search algorithm that offers a quadratic improvement over classical counterparts [38]. This algorithm is fundamentally based on the concept of function inversion, which encompasses two specific types: phase inversion and inversion about the mean.

Consider an unsorted search space of size N. Finding an element in this space that satisfies a specific property requires $O(N)$ steps. Grover's search algorithm is capable of finding the solution with $O(\sqrt{N})$ steps.

A concrete instance of an unstructured search problem can be drawn from the task of locating a specific telephone number within a digitized version of a traditional phone book, where the entries are not ordered by number and no reverse lookup is available. Given a collection of N entries, each consisting of a name and an associated telephone number, the objective is to identify the index $i \in \{0, 1, \ldots, N-1\}$ such that the telephone number at that entry matches a fixed target value t. Crucially, the searcher does not possess any information about the name corresponding to t, and the dataset is assumed to be unsorted with respect to phone numbers, rendering indexing by value or binary search infeasible. Classically, solving this problem requires a brute-force approach that queries each entry until a match is found, resulting in a time complexity of $O(N)$. However, in the quantum setting, Grover's algorithm offers a quadratic speed-up by modeling the search as the problem of finding a unique input i such that a Boolean oracle function $f(i)$ evaluates to 1 if and only if the entry at index i contains the target number. Grover's algorithm can identify the correct index with high probability using only $O(\sqrt{N})$ evaluations of the oracle.

To understand the general functioning of the algorithm, suppose that a search

problem is characterized by a function

$$f : \{0,1,...,N-1\} \to \{0,1\}.$$

Now imagine that there exists a unique solution x_0 such that $f(x_0) = 1$. The element x_0 corresponds to the solution of our search problem. Therefore, all incorrect solutions, when evaluated through the function f, will return 0.

A fundamental component for the implementation of Grover's algorithm is a quantum oracle O capable of recognizing the correct solution to the search problem. In practice, the quantum oracle will be a unitary operator whose action allows us to evaluate a solution by observing the state of a so-called *oracle qubit*. If, following the application of the operator, the oracle qubit, once measured, returns the state $|1\rangle$, it indicates that the candidate solution evaluated by the oracle is indeed the correct one for the search problem.

In this context it is common to apply a Hadamard gate on the above-mentioned oracle qubit to obtain the state $\frac{|0\rangle-|1\rangle}{\sqrt{2}}$. Then is possible to apply the oracle, whose action can be described as:

$$|x\rangle \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right) \xrightarrow{O} (-1)^{f(x)} |x\rangle \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right). \tag{1.41}$$

The oracle alters the final state only if the bitstring x is the solution to the search problem, effectively swapping $|0\rangle$ and $|1\rangle$. In essence, the oracle **flips** the phase for the correct solution.

It is important to note that the oracle does not need to know the correct solution a priori; however, it must be capable of recognizing the correct solution once it is provided as input.

As previously mentioned, the oracle serves as the fundamental component of Grover's algorithm, however it is not the only one. The algorithm requires a register of n qubits (n being the smallest positive number in $\mathbf{Z}$ such that $N \leq 2^n$), which are initially in the state:

$$|\psi_0\rangle = |0\rangle^{\otimes n}.$$

Subsequently, a Hadamard gate (also known as the Hadamard transform) is applied to each qubit in the register.

$$|\psi_s\rangle = \frac{1}{N^{1/2}} \sum_{x=0}^{N-1} |x\rangle \tag{1.42}$$

This operation creates an equal superposition of all bitstrings that represent possible solutions to the search problem. The following step consists in obtaining the circuit in Figure 1.15 by applying $O(\sqrt{N})$ times the *Grover operation* (Q) which is composed of the two inversion operator mentioned in the opening of this section.

The first one, known as the *phase-inversion* operator is exactly the oracle O described above. Following its application, we need to apply the *inversion about*

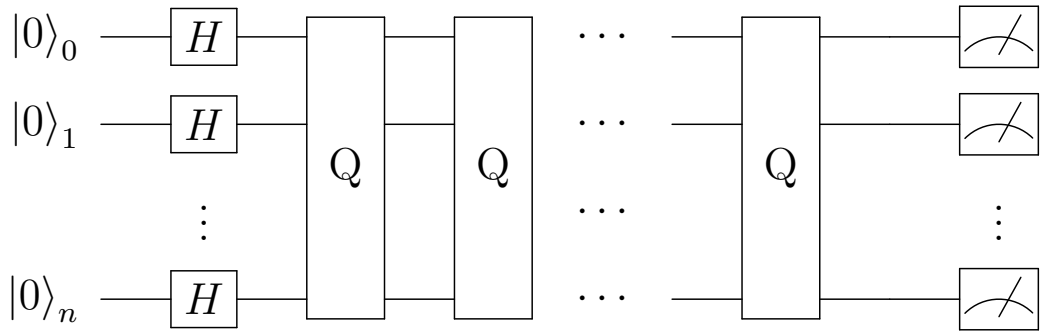


Figure 1.15: Quantum circuit implementing Grover's algorithm on n qubits.

the mean operator U_i, which requires firstly another Hadamard transform applied on the n qubits, followed by the following operation:

$$2\left|0\right\rangle\left\langle 0\right| - I$$

Here is an example of his action for one single qubit:

- $|0\rangle$ will be the basis state $\begin{pmatrix}1\\0\end{pmatrix}$.
- $\langle 0|$ is thus the conjugate transpose of $|0\rangle$, i.e., $\begin{pmatrix}1 & 0\end{pmatrix}$.
- $|0\rangle\langle 0|$: projection operator onto the state $|0\rangle$, defined as:

$$|0\rangle\langle 0| = \begin{pmatrix}1\\0\end{pmatrix}\begin{pmatrix}1 & 0\end{pmatrix} = \begin{pmatrix}1 & 0\\0 & 0\end{pmatrix}.$$

- First, compute $2|0\rangle\langle 0|$:

$$2|0\rangle\langle 0| = 2 \cdot \begin{pmatrix}1 & 0\\0 & 0\end{pmatrix} = \begin{pmatrix}2 & 0\\0 & 0\end{pmatrix}.$$

  The identity matrix I is:

$$I = \begin{pmatrix}1 & 0\\0 & 1\end{pmatrix}.$$

- Now subtract I from $2|0\rangle\langle 0|$:

$$2|0\rangle\langle 0| - I = \begin{pmatrix}2 & 0\\0 & 0\end{pmatrix} - \begin{pmatrix}1 & 0\\0 & 1\end{pmatrix} = \begin{pmatrix}1 & 0\\0 & -1\end{pmatrix}.$$

- Final Result:

$$2|0\rangle\langle 0| - I = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}.$$

Eventually, we apply a final Hadamard transform to conclude our Grover iteration.

The impact of repeated applications of the operator Q highlights the efficacy of the algorithm. This can be effectively illustrated through a useful graphical visualization, which can be observed in Figure 1.16 that illustrates the effect of applying once the Grover iteration routine.

In the figures, we observe two axes corresponding to two states of our system. The vertical axis represents the state $|\omega\rangle$, which we assume to be the state corresponding to the correct solution of our problem. The horizontal axis, on the other hand, represents the state perpendicular to the former, which we denote as $|s'\rangle$.

Figure 1.16a illustrates the effect of applying the oracle to the initial uniform superposition (Equation 1.42). As can be observed, this effect corresponds graphically to a reflection with respect to $|s'\rangle$, inverting the phase exclusively for the state (within the superposition) that corresponds to the correct solution, i.e., $|\omega\rangle$. In Figure 1.16b, we observe the action of U_i, the second operator that constitutes Q. In this case, the effect is a reflection with respect to $|\psi\rangle_s$ itself.

Overall, the combined effect of these two operators leads to the result shown in Figure 1.16c, where it is evident that the final state has undergone a rotation, moving closer to the correct solution state. Therefore, the repeated application of Grover's operator, Q, progressively brings the state closer to $|\omega\rangle$.

It can be shown that the amount of repetitions of Q needed to approach the correct solution depends on N and grows as $O(\sqrt{N})$ [38].

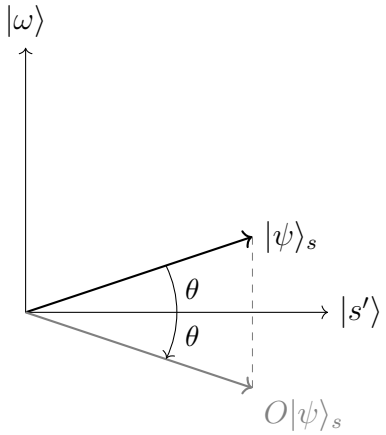
1.4.5 Quantum Amplitude Amplification

It is possible to generalize Grover's algorithm approach for problems where there are multiple correct solutions and the initial superposition is not necessarily uniform. This initial superposition is therefore not obtained using a Hadamard transform, but by applying an arbitrary transformation resulting from the application of a general operator $\mathcal{A}$. In this case, the fundamental insight lies in the quantum analogue of increasing the probability of finding a correct solution. In quantum, this increase can be obtained *amplifying* the probability amplitude of being in the specific subspace of the Hilbert space that "contains" the correct solutions.

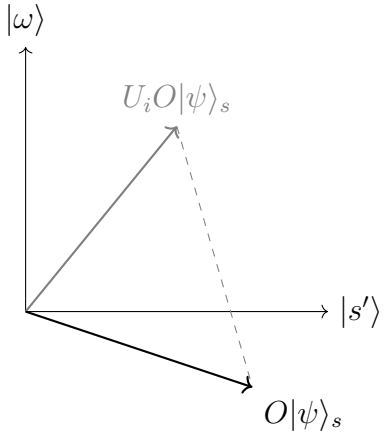
Even in this case, we can consider a function analogous to f, which we will call

$$\chi : \{0,1,...,N-1\} \to \{0,1\}.$$

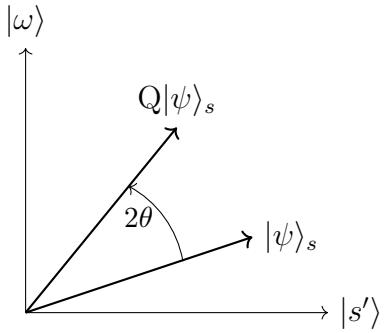
We can partition the set of input into two subsets: one of *bad* elements ($\chi(x) = 0$) and one of *good* elements ($\chi(x) = 1$). $\mathcal{A}$ can be thought of as a quantum algorithm



(a) Effect of applying the oracle O on the initial uniform superposition.



(b) Effect of applying U_i on the state $O\left|\psi\right\rangle_s$.



(c) Global effect of the Grover iteration Q on the initial superposition.

Figure 1.16: Visualization of the Grover subroutine

such that

$$\mathcal{A}|0\rangle = \sum_{j=0}^{2^n-1} \alpha_j |j\rangle_n \tag{1.43}$$

prepares a general quantum state on n qubits (see Equation 1.18). Now let a denote the probability that a good solution gets observed once $\mathcal{A}|0\rangle$ is measured and $\mathcal{H}$ denote the Hilbert space were we find the states of our system. We can see our function χ as dividing the Hilbert space $\mathcal{H}$ into a *good* subspace and a *bad* subspace. Therefore, every pure state $|\Upsilon\rangle$ can be decomposed as $|\Upsilon\rangle = |\Upsilon_1\rangle + |\Upsilon_0\rangle$ which stands for the sum of its projection onto the good subspace and the one onto the bad subspace. In this context, $a = \langle\Upsilon_1|\,\Upsilon_1\rangle$ is indeed the probability of obtaining a good state when measuring $|\Upsilon\rangle$ [12]. Lastly, let's assume that $\mathcal{A}$ has an inverse.

Now, we can amplify a by applying powers of a specific (and probably familiar by now) unitary operator to $\mathcal{A}|0\rangle$. In particular, the operator is as such:

$$\mathbf{Q} = \mathcal{A}S_0\mathcal{A}^\dagger S_\chi \tag{1.44}$$

Where S_0 changes the sign of the amplitude whenever the state is the zero state (what we called *inversion about the mean* operator in Section 1.4.4), while S_χ (that we called *phase-inversion* operator in the same Section) flips the sign of the amplitudes of the good states.

It is common to operate a change of basis to better demonstrate the amplification effects of applying $\mathbf{Q}$ to $|\Psi\rangle = \mathcal{A}|0\rangle$: the subspace $\mathcal{H}_\Psi$ has an orthonormal basis consisting of two eigenvectors of $\mathbf{Q}$,

$$|\Psi_\pm\rangle = \frac{1}{\sqrt{2}}\left(\frac{1}{\sqrt{a}}|\Psi_1\rangle \pm \frac{\imath}{\sqrt{1-a}}|\Psi_0\rangle\right), \tag{1.45}$$

with eigenvalues $\lambda_\pm = e^{\pm\imath 2\theta_a}$ so that $a = \sin^2\theta_a$ (with $0 \leq \theta_a \leq \frac{\pi}{2}$). It is now trivial to show how one can use the operator $\mathbf{Q}$ to boost the probability a of observing a good solution. In fact if we start with

$$|\Psi\rangle = \frac{-\imath}{\sqrt{2}}\left(e^{\imath\theta_a}|\Psi_+\rangle - e^{-\imath\theta_a}|\Psi_-\rangle\right) \tag{1.46}$$

and apply the operator $\mathbf{Q}$ for a number of times equal to some integer $j \geq 0$, the final state will be

$$\begin{aligned}\mathbf{Q}^j|\Psi\rangle &= \frac{-\imath}{\sqrt{2}}\left(e^{(2j+1)\theta_a}|\Psi_+\rangle - e^{-(2j+1)\theta_a}|\Psi_-\rangle\right)\\ &= \frac{1}{\sqrt{a}}\sin\left((2j+1)\theta_a\right)|\Psi_1\rangle + \frac{1}{\sqrt{1-a}}\cos\left((2j+1)\theta_a\right)|\Psi_0\rangle\,.\end{aligned} \tag{1.47}$$

Therefore, measuring $\mathbf{Q}^j|\Psi\rangle$ will yeld a good state with a boosted probability given by $\sin^2\left((2j+1)\theta_a\right)$.

1.4.6 Quantum Amplitude Estimation

Now consider the task of estimating the number of correct solutions (the number of inputs for which the function χ takes the value 1). It is easy to observe that this problem is essentially equivalent to estimating the fraction of correct solutions with respect to the total number of possible solutions, and thus, adopting a frequentist approach, essentially to estimating $a = \langle \Psi_1 | \Psi_1 \rangle$ in

$$\mathcal{A}|0\rangle = \sqrt{1-a}\,|\Psi_0\rangle\,|0\rangle + \sqrt{a}\,|\Psi_1\rangle\,|1\rangle. \tag{1.48}$$

The objective is to exploit the properties of the operator $\mathbf{Q}$ as defined in the previous section. It is particularly interesting to highlight how Equation 1.47 clearly shows that the amplitudes of the good and bad states, as a function of the number of applications of $\mathbf{Q}$, are sinusoidal functions whose period is related to θ_a, and therefore to a.

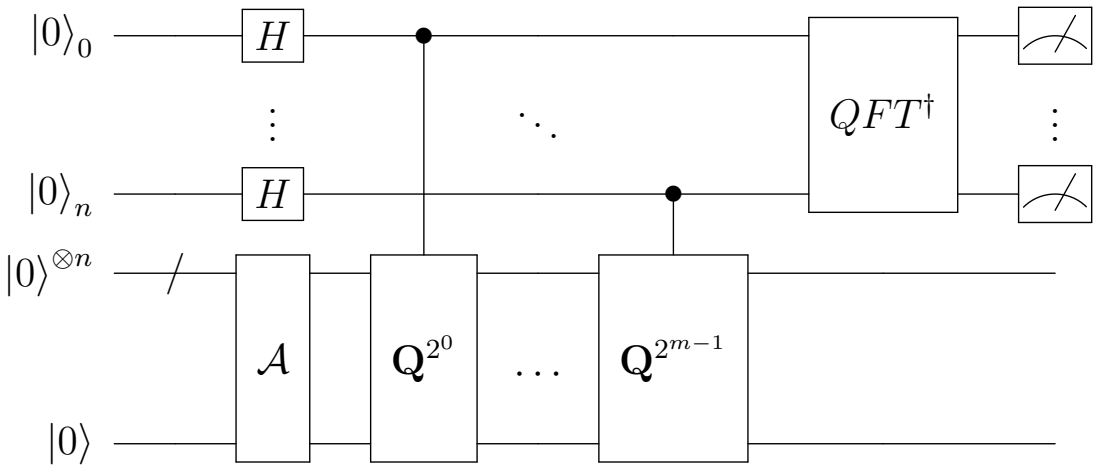


Figure 1.17: Circuit for the Quantum Amplitude Estimation (QAE) algorithm. The top register (first m qubits) is used for phase estimation via inverse QFT, while the bottom-most qubit acts as the *target qubit*, where the probability amplitude of interest is encoded. The operator $\mathcal{A}$ prepares a state such that measuring this qubit reflects the success probability a.

We have already seen quantum algorithms for period estimation, particularly the application of the Quantum Fourier Transform in Shor's algorithm and for eigenvalue estimation using QPE. This latter application provides an alternative perspective on the problem: as mentioned above the eigenvalues of Q on the subspace spanned by $|\Psi_1\rangle$ and $|\Psi_0\rangle$ are $\lambda^+ = e^{i2\theta_a}$ and $\lambda^- = e^{-i2\theta_a}$. Consequently, the use of quantum phase estimation (QPE) allow us to estimate θ_a (and thus a). The resulting circuit is displayed in Figure 1.17. In the circuit depicted, the target

qubit (the first from the bottom) plays a central role in encoding and extracting the amplitude information that is critical for QAE.

as shown in Equation (1.48). Here, the target qubit (the qubit in the rightmost register) is used to distinguish between the "good" outcomes (encoded in $|\Psi_1\rangle$) and the "bad" outcomes (encoded in $|\Psi_0\rangle$).

After the state preparation, the algorithm applies controlled operations of the amplitude amplification operator $\mathbf{Q}$ to the target qubit. These controlled $\mathbf{Q}$ operations—applied with powers $2^0, 2^1, \ldots, 2^{m-1}$—induce the phase kickback effect. In other words, while the target qubit retains the information regarding the superposition of the "good" and "bad" components, it also facilitates the imprinting of the phase (related to θ_a) onto the ancillary qubits.

1.4.7 QAE Without Phase Estimation

Recently, numerous studies have introduced simplified models of QAE that exclusively employ Grover iterations, thereby eliminating the requirement for Quantum Phase Estimation. These approaches offer a significant reduction in resource demands, particularly with respect to the number of qubits and the overall circuit complexity, a key advantage for implementation on NISQ devices.

QPE typically requires a deep quantum circuit, additional ancilla qubits, and a quantum Fourier transform, all of which impose considerable overhead on current quantum hardware.

The alternative approaches circumvent these challenges by employing iterative Grover operations coupled with classical post-processing. In these methods, the quantum circuit is simplified to consist primarily of the state preparation operator $\mathcal{A}$ and the Grover operator $\mathbf{Q} = \mathcal{A}S_0\mathcal{A}^\dagger S_\chi$. Instead of applying a controlled-$\mathbf{Q}$ operation in a QPE framework, the algorithm repeatedly applies $\mathbf{Q}$ and measures the state after various numbers of iterations. The measurement statistics are then processed using techniques such as maximum likelihood estimation or iterative refinement to estimate the amplitude a.

This approaches, like Iterative Quantum Amplitude Estimation (IQAE) [37] and Maximum Likelihood Amplitude Estimation (MLAE) [77], have been shown to achieve comparable accuracy to traditional QAE while significantly reducing circuit depth and qubit requirements. Although the latter does not provide rigorous proofs of the correctness of MLAE, the former (IQAE) provably has the desired asymptotic behavior up to a multiplicative factor that depends on the target accuracy and the resulting confidence level. However, IQAE requires iterative queries to the QPU and cannot be parallelized. In contrast, MLAE allows for the parallel execution of the different queries. Regardless, the potential parallelization speedup is limited. In fact, in MLAE the length of the queries is doubled in each iteration. Thus, the speedup is at most a factor of two. For these reasons, we will use IQAE for the amplitude estimation subroutine that is part of the algorithms proposed in

Algorithm 2 Iterative Quantum Amplitude Estimation. Source: [37].

Require: ϵ, α, N_{shots}, ci {ci is a chosen confidence interval method.}

$i \leftarrow 0$ {initialize iteration count}
$k_i \leftarrow 0$ {initialize power of $\mathcal{Q}$}
$\text{up}_i \leftarrow$ True {keeps track of the half-plane}
$[\theta_l, \theta_u] \leftarrow [0, \pi/2]$ {initialize confidence interval}
$T \leftarrow \lceil \log_2(\pi/8\epsilon) \rceil$ {maximum number of rounds}
Calculate $L_{\max}$ {max error on every iteration; depends on ϵ, α, N_{shots}, and the choice of confidence interval}
while $\theta_u - \theta_l > 2\epsilon$ **do**
  $i \leftarrow i + 1$
  $[k_i, \text{up}_i] \leftarrow \text{FindNextK}(k_{i-1}, \theta_l, \theta_u, \text{up}_{i-1})$
  Set $K_i \leftarrow 4k_i + 2$
  if $K_i > \lceil L_{\max}/\epsilon \rceil$ **then**
    $N \leftarrow \lceil N_{\text{shots}}\, L_{\max}/(\epsilon\, K_i/10) \rceil$ {No-overshooting condition}
  else
    $N \leftarrow N_{\text{shots}}$
  end if
  Approximate $a_i = \mathbb{P}[|1\rangle]$ for the last qubit of $\mathbf{Q}^{k_i}\mathcal{A}\,|0\rangle_n\,|0\rangle$ 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 result
  end if
  if ci = "Chernoff-Hoeffding" **then**
    $\epsilon_{a_i} \leftarrow \sqrt{\frac{1}{2N} \log\left(\frac{2T}{\alpha}\right)}$
    $a_i^{\max} \leftarrow \min\left(1,\, a_i + \epsilon_{a_i}\right)$
    $a_i^{\min} \leftarrow \max\left(0,\, a_i - \epsilon_{a_i}\right)$
  end if
  if ci = "Clopper-Pearson" **then**
    $a_i^{\max} \leftarrow I^{-1}\left(\frac{\alpha}{2T};\, Na_i,\, N(1-a_i)+1\right)$
    $a_i^{\min} \leftarrow I^{-1}\left(1-\frac{\alpha}{2T};\, Na_i+1,\, N(1-a_i)\right)$
  end if
  Calculate $[\theta_i^{\min}, \theta_i^{\max}]$ for $\{K_i\theta_a\}_{\text{mod } 2\pi}$ from $[a_i^{\min}, a_i^{\max}]$ and flag up_i by inverting $a = (1 - \cos(K_i\theta))/2$
  $\theta_l \leftarrow \frac{\lfloor K_i\theta_l \rfloor_{\text{mod } 2\pi} + \theta_i^{\min}}{K_i}$
  $\theta_u \leftarrow \frac{\lfloor K_i\theta_u \rfloor_{\text{mod } 2\pi} + \theta_i^{\max}}{K_i}$
end while
$[a_l, a_u] \leftarrow [\sin^2(\theta_l), \sin^2(\theta_u)]$
return $[a_l, a_u]$

Chapters 3 and 4.

We now outline the main ideas IQAE relies on. The method is formally described in [37] using Algorithm 2, while its main sub-routine `FindNextK` is outlined through Algorithm 3. We start with a confidence interval $[\theta_l, \theta_u] \subseteq [0, \pi/2]$ for θ_a and a power k of $\mathbf{Q}$ as well as an estimate for $\sin^2((2k+1)\theta_a)$. Through exploiting trigonometric identities, the estimates for $\sin^2((2k+1)\theta_a)$ can be turned into estimates for $\cos((4k+2)\theta_a)$. Here, the cosine is only invertible without ambiguity if we know the argument is restricted to either $[0, \pi]$ or $[\pi, 2\pi]$. Thus, we want to find the largest k such that the scaled interval $[(4k+2)\theta_l, (4k+2)\theta_u]_{\bmod 2\pi}$ is fully contained either in $[0, \pi]$ or $[\pi, 2\pi]$. Once this is achieved, it is possible to invert $\cos((4k+2)\theta_a)$ and improve the estimate for θ_a with high confidence. The algorithm's core is the procedure that finds the next k given $[\theta_l, \theta_u]$ (Algorithm 3). It can be proved that for this algorithm, supposing a confidence level $1-\alpha \in (0,1)$ and a target accuracy $\epsilon > 0$:

$$N_{\text{oracle}} < \frac{50}{\epsilon} \log\left(\frac{2}{\alpha} \log_2\left(\frac{\pi}{4\epsilon}\right)\right). \tag{1.49}$$

which is the desired bound for the total number of $\mathbf{Q}$-applications. For a rigorous proof we refer the reader to the original paper [37].

1.5 Quantum Computing Libraries

The development of quantum computing algorithms requires appropriate software tools. Over the years, numerous libraries have been created for this purpose. One of the most established is undoubtedly Qiskit (Quantum Software Development Kit), an open-source library for quantum information science. Nevertheless, we must also mention CirQ [21] (associated with Google) and PennyLane [6], originally developed by the quantum company Xanadu, as well as Q# (Microsoft) [56] and the QMod language [81] developed by Classiq.

For the experimental part of this work, Qiskit and the Classiq platform were utilized. Therefore, it seems appropriate to present these two important software tools in more detail.

1.5.1 Qiskit

Qiskit was initiated in 2017 by IBM [45]. To date, it has been employed in over 2,000 scientific publications. Figure 1.18 illustrates the primary components constituting the Qiskit software architecture.

Typically, the user begins by outlining a high-level computational problem and identifying a quantum algorithm to resolve it using quantum circuits. At this stage, the resulting circuits are often quite abstract, comprised of high-level mathematical operators. Qiskit subsequently generates concrete circuits from these through

Algorithm 3 Procedure for finding k_{i+1}. Source: [37].

Require: $k_i,\ \theta_l,\ \theta_u,\ \mathrm{up}_i,\ r = 2$

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

a process called *transpilation* that translates the abstract circuits to the target instruction set architecture (ISA) using its standard library of gates. Such gates are representable using OpenQASM [18] or other intermediate quantum languages.

The ISA often includes a universal set of quantum gates and additional operations such as *measure* and *reset*.

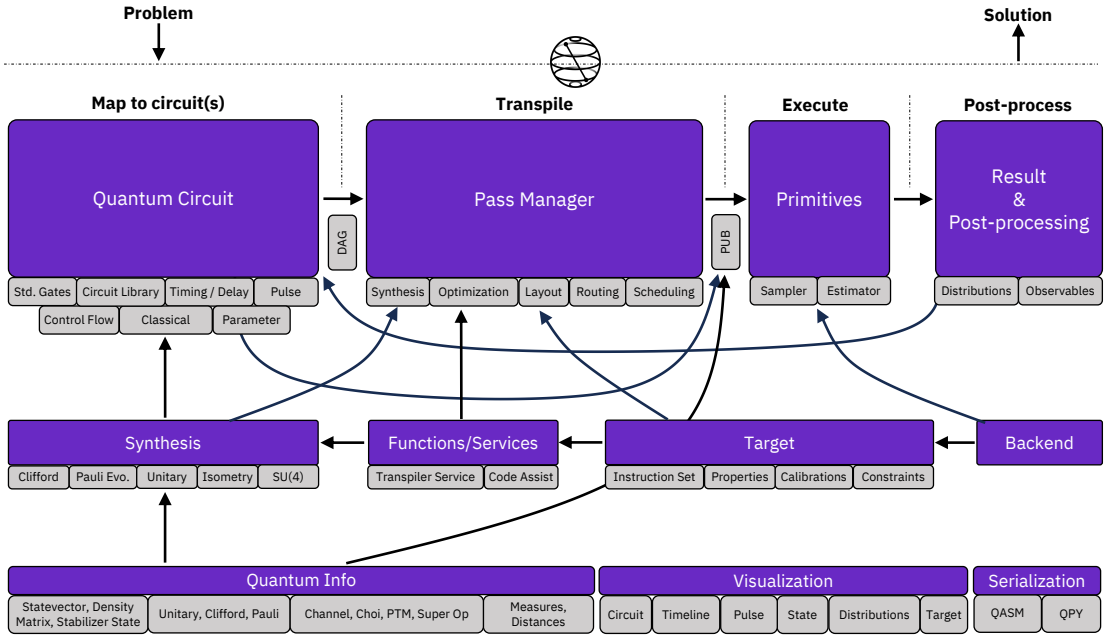


Figure 1.18: Overview of Qiskit's software architecture. Source: [45].

In [45], the authors emphasize the role of modularity and extensibility as driving forces behind the software design in Qiskit. This is evident when considering the plug-in interface that facilitates the integration and extension of Qiskit with additional methods, circuits, and optimizations.

Another strength of qiskit lies in its ability to make circuits compatible with diverse quantum platforms (with different ISA).

```
from qiskit import QuantumCircuit
from qiskit.quantum_info import SparsePauliOp
from qiskit.transpiler import generate_preset_pass_manager
from qiskit_ibm_runtime import EstimatorV2 as Estimator
from qiskit_ibm_runtime import QiskitRuntimeService

# Create a new circuit with two qubits
qc = QuantumCircuit(2)

# Add a Hadamard gate to qubit 0
qc.h(0)

# Perform a controlled-X gate on qubit 1, controlled by qubit 0
```

```
qc.cx(0, 1)

# Return a drawing of the circuit using MatPlotLib ("mpl"). This is the last line of the cell, so the drawing appears in the cell output.
display(qc.draw("mpl"))

# Set up six different observables.

observables_labels = ["IZ", "IX", "ZI", "XI", "ZZ", "XX"]
observables = [SparsePauliOp(label) for label in observables_labels]

service = QiskitRuntimeService()

backend = service.least_busy(simulator=False, operational=True)

# Convert to an ISA circuit.
pm = generate_preset_pass_manager(backend=backend, optimization_level=1)
isa_circuit = pm.run(qc)

# Construct the Estimator instance.
estimator = Estimator(mode=backend)
estimator.options.resilience_level = 1
estimator.options.default_shots = 5000

mapped_observables = [observable.apply_layout(isa_circuit.layout) for observable in observables]

# One pub, with one circuit to run against five different observables.
job = estimator.run([(isa_circuit, mapped_observables)])

# Use the job ID to retrieve your job data later
print(f">>> Job ID: {job.job_id()}")
```

Listing 1.1: Qiskit code example. Source: [41].

This is achieved thanks to its `Target` class that defines the instructions and the properties of a specific quantum hardware.

Listing 1.1 illustrates a practical example of a quantum routine realized through Qiskit. The example pertains to the estimation of multiple observables from a simple two-qubit circuit executed on an IBM quantum backend. It demonstrates how Qiskit translates a high-level quantum algorithm into hardware-specific instructions via transpilation and optimization, while integrating runtime services to perform resilient estimation of expectation values.

1.5.2 Classiq and the Qmod Language

Qmod is an a high-level quantum programming language whose aim is to deal with the issue of describing non-trivial quantum logic at the circuit level [81]. This language was developed by Classiq Technologies[1], which incorporates it into its platform for the design and synthesis of quantum circuits [35].

The strength of the Qmod language lies in its ability to support quantum numeric variables and expressions. Most notably, Qmod supports native expressions that combine quantum variables, classical variables, and literal constants using familiar arithmetic, relational, and bitwise operators [81]. These expressions can be evaluated in multiple modes, encoding their results either in the computational basis or in the phase or amplitude of basis states.

Qmod accepts two different input formats:

- an external domain-specific language (DSL) with its dedicated syntax (see Figure 1.19 for an example),
- an embedded DSL with Python as host language (an example is represented by the code in Listing 1.2).

Both let programmers express complex numeric computations in high-level functional terms.

[1] https://www.classiq.io/

```python
from classiq import (
    qfunc,
    Output,
    QBit,
    CReal,
    synthesize,
    PauliTerm,
    Pauli,
    RX,
    allocate,
)
from classiq.execution import ExecutionSession


@qfunc
def main(x: Output[QBit], t: CReal):
    allocate(1, x)
    RX(t, x)


qprog = synthesize(main)

with ExecutionSession(qprog) as execution_session:
    hamiltonian = [
        PauliTerm(pauli=[Pauli.I], coefficient=1),
        PauliTerm(pauli=[Pauli.Z], coefficient=2),
    ]

    estimate_result = execution_session.estimate(
        hamiltonian, {"t": 0.5}
    )
    batch_estimate_result = execution_session.batch_estimate(
        hamiltonian, [{"t": 0.5}, {"t": 0.6}]
    )
    estimate_job = execution_session.submit_estimate(
        hamiltonian, {"t": 0.5}
    )
    batch_estimate_job = execution_session.submit_batch_estimate(
        hamiltonian, [{"t": 0.5}, {"t": 0.6}]
    )
```

Listing 1.2: Qmod code example as an embedded DSL with Python as host language.

The resulting quantum programs are expressed in terms of a high-level model together with constraints and objectives. This is possible thanks to the principles and methods of well-proven electronic design automation (EDA) [85] on which the Classiq framework is based.

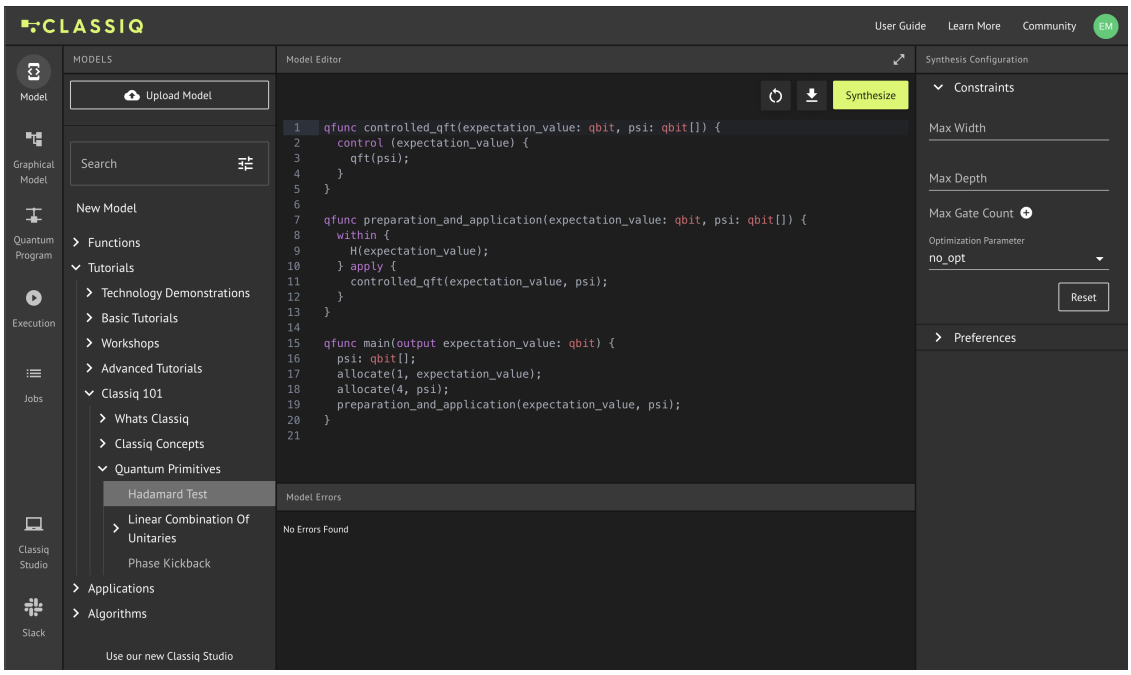


Figure 1.19: Qmod code example as an external DSL used inside the Classiq platform [35].

Chapter 2

Quantum Algorithms for Finance

The financial sector is anticipated to be a major beneficiary of the advancements enabled by quantum computing [62, 29, 42]. The primary reason lies in the abundant number of complex problems that arise within the field. The highly competitive nature of the sector is undoubtedly an additional incentive towards investment and research in what is shaping up to be the promising domain of *quantum finance*.

According to a McKinsey & Company 2023 analysis [40], quantum computing applications in the financial sector could generate up to $622 billion in value by 2035. The distribution of this economic impact varies across different financial domains, with corporate banking expected to benefit the most, contributing approximately $190 billion. Risk and cybersecurity follow closely, with a projected value of $160 billion. Retail banking, payments, and asset and wealth management are estimated to generate $90 billion, $80 billion, and $80 billion, respectively. Investment banking stands to gain around $20 billion, while operations and finance are expected to see a comparatively modest impact of $2 billion [40]. These projections highlight the significant potential of quantum computing in optimizing financial operations.

Financial problems with significant complexity include forecasting tasks such as pricing, risk estimation, anomaly detection in transactions, and analyzing customer preferences. In these cases, stochastic processes are commonly used as a tool. The central idea is to model phenomena by representing the descriptive quantities with random variables. What characterizes the application of these methods in the financial domain is the objective (for example, aiding in making investment decisions) and the nature of these quantities of interest, which are related to the description of market conditions. Often, this model leverages stochastic differential equations (SDEs) that need to be solved to obtain expectation values for the variables of interest. In a few simple cases, it is possible to compute analytical solutions to solve the SDEs; more often, numerical approaches are required. These approaches can be high-precision and non-randomized methods which, however, have a strong dependence on the problem's dimension, or methods with weaker dependence at

the expense of variance, such as Monte Carlo Integration (MCI).

Another avenue of significantly complex problems is that of optimization for tasks such as optimal portfolio selection and hedging. Typically, in finance, one encounters problems that are highly constrained, linear or quadratic. These problems can involve continuous or discrete variables, and sometimes both. Quantum methods have demonstrated polynomially better complexity dependence on certain dimensions of the problem.

In [42], the authors list three main tools for solving these financial problems:

- stochastic modeling,
- optimization,
- machine learning.

In stochastic modeling, quantum Monte Carlo integration and quantum partial differential equation (PDE) solvers are proposed for pricing and risk modeling, with potential advantages such as quadratic speedup, though challenges like model loading and pre-/post-processing remain. In particular, to achieve the capability to address problems with commercially relevant specifications, it is necessary to reduce the resources required for the quantum embedding of classical input data and to optimize the readout process to prevent diluting the polynomial advantage provided by the quantum model. Thus, these applications are expected to have long-term applicability.

Optimization problems, including portfolio management, financing, and resource allocation, can benefit from quantum-enhanced solvers, which may provide quadratic speedup but face challenges such as costly QBLAS (quantum basic linear algebra subroutine) limitations and oracle complexity. Quantum heuristics, however, show partial evidence of near-term applicability, despite parameter tuning challenges.

In machine learning, quantum-enhanced and quantum-native models hold promise for forecasting, anomaly detection, and recommendation systems, though their applicability varies. While quantum-enhanced machine learning faces data-loading constraints and is considered a long-term prospect, quantum-native approaches may have near-term potential despite issues with parameter tuning and sampling complexity [42]. Thus, the literature is clear both on the promise and on the significant technical challenges that must be addressed before quantum computing can be fully leveraged in financial applications. In particular, it is worth also emphasizing the reliance of quantum methods on classical computation concerning the identification of optimal initial conditions and/or the implementation of error mitigation schemes.

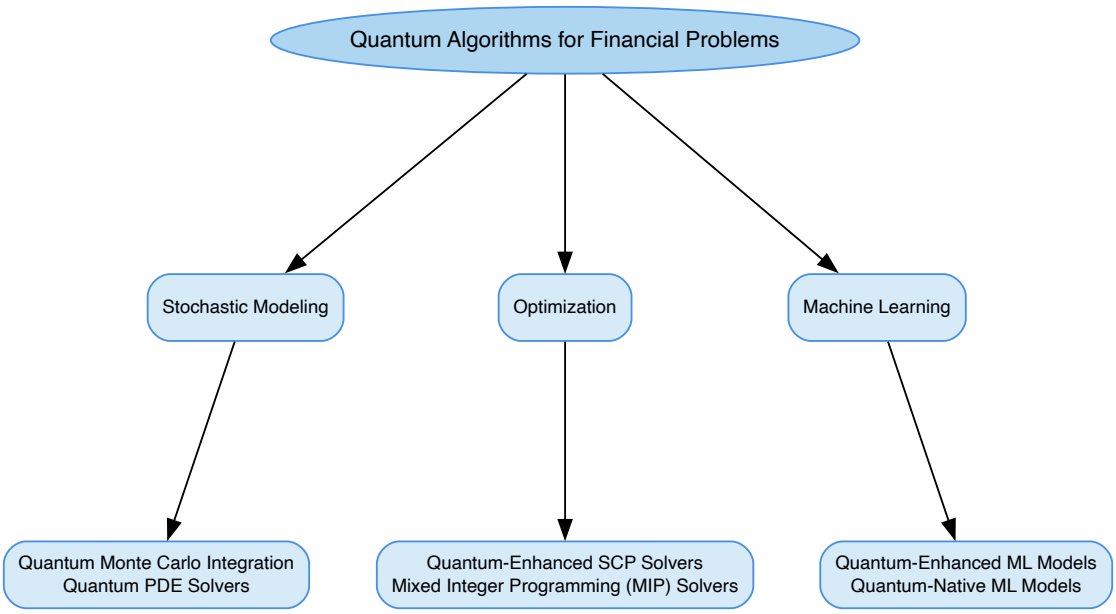


Figure 2.1: High level overview of quantum algorithms for financial problems.

2.1 Promising Complex Problems in Finance

As previously mentioned, there exist specific classes of problems within the financial domain that are difficult to solve. In particular, this work will focus on three of the most relevant and well-known: *portfolio optimization*, *credit risk analysis* and *derivative pricing*. These will also be categorized based on the nature of the quantum algorithms proposed to address them: so-called *near-term* algorithms, and those that assume the possibility of execution on *fault-tolerant* QPUs, thus not amenable to useful implementation using currently available technology.

Fault-tolerant QPUs refer to quantum processors that incorporate robust error correction techniques to mitigate the effects of decoherence, gate errors, and other sources of noise. These devices typically require quantum error-correcting codes to encode logical qubits into many physical qubits, thereby enabling the correction of errors during computation [61]. This approach is essential for executing long and complex quantum algorithms with high fidelity. However, the overhead is significant; fault-tolerant architectures often require thousands of physical qubits to reliably represent a single logical qubit. Consequently, while fault-tolerant QPUs promise to unlock the full potential of quantum computing, they remain a long-term goal, as current hardware is limited to noisy intermediate-scale quantum (NISQ) devices.

In contrast, *near-term* quantum algorithms are specifically designed to operate within the constraints of existing quantum hardware. These algorithms aim to extract useful computational results despite the absence of full error correction and

the presence of operational noise [64]. While such approaches may not fully exploit the exponential speedups predicted for fault-tolerant quantum computers, they provide a practical pathway for demonstrating quantum advantage in real-world applications today. This categorization will be helpful later in delineating which financial problems might benefit from current NISQ-era technologies versus those that will require the advent of fault-tolerant QPUs to be effectively addressed.

2.1.1 Portfolio Optimization

The objective of optimal portfolio selection is to build portfolios that aim to maximize expected returns while aligning with each investor's acceptable risk thresholds [63]. The most traditional framework for addressing the problem of optimizing the mix of the components within a financial portfolio is usually considered to be the one proposed by Markowitz in 1952 [50] and known as *modern portfolio theory*. In this context, risk minimization (for a given target return) is achieved

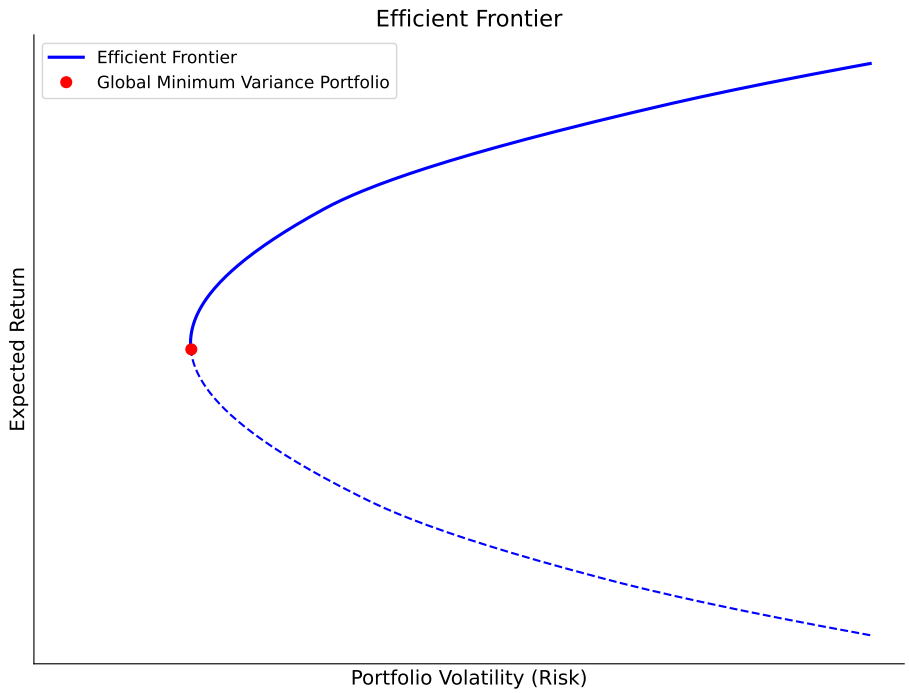


Figure 2.2: Graphical representation of the efficient frontier. In the image the frontier is a curve that represents the risk and return values arising from different combinations of assets.

through the minimization of variance (covariance in the case of multiple assets),

which is associated with risk in the model in question. This results in a quadratic mathematical optimization problem which, classically, is often addressed through the application of operations research techniques. Moreover, we are faced with a convex optimization problem, where market constraints often require that portfolio positions be either integers or multiples of a fixed increment. This necessitates the use of discrete optimization methods or, more generally, mixed-integer programming (MIP) [7]. A particularly powerful class of algorithms for solving MIPs is the branch-and-bound (B&B) method [47], which constitutes the core of most commercial solvers utilized in the field. Essentially, the algorithm constructs a tree of sub-problems, employing heuristic techniques to identify sub-problems that can be efficiently solved while pruning branches of the tree whenever possible to enhance computational efficiency.

Clearly, as the number of possible asset combinations increases, the selection of the optimal portfolio quickly becomes a complex problem. Classical optimization techniques therefore prove to be inefficient, since the tree can be exponentially large in problem size. Thus, heuristic techniques can provide an improvement over simpler methods, but with the risk of being too slow or suboptimal.

2.1.2 Credit Risk Analysis

Credit risk analysis is undoubtedly one of the fundamental tasks for the risk management department of any reputable financial institution. The objective is to estimate the amount of capital that must be set aside to withstand rare, yet possible, adverse conditions arising from events that impact the ability of the institution's debtors to repay their loans.

To quantify this credit risk we will refer to two basic measures:

- Probability of Default (PD):
- Loss Given Default (LGD)

The higher these quantities are, the greater the Expected Loss (EL) will be. The latter is obtained by multiplying the elementary risk measures mentioned above. With reference to a single exposure i:

$$EL_i = PD_i \cdot LGD_i \tag{2.1}$$

Given that EL is an additive measure, portfolio EL is equal to the sum of single exposure's expected loss.

The expected loss at an aggregate level is the first component in estimating what is known as *Economic Capital* (EC):

$$EC = VaR_\alpha - EL \tag{2.2}$$

The purpose of Economic Capital is precisely to estimate the amount of equity that a financial institution must hold to withstand the risk of credit losses in its portfolio. The other ingredient is Value at Risk (VaR) which is defined for a given confidence level $\alpha \in [0, 1]$ as the smallest total (aggregated) loss that still has a probability greater than or equal to α to occur. Figure 2.3 depicts this concept. In 2.2, the EL is subtracted from Value at Risk (VaR) because expected losses are

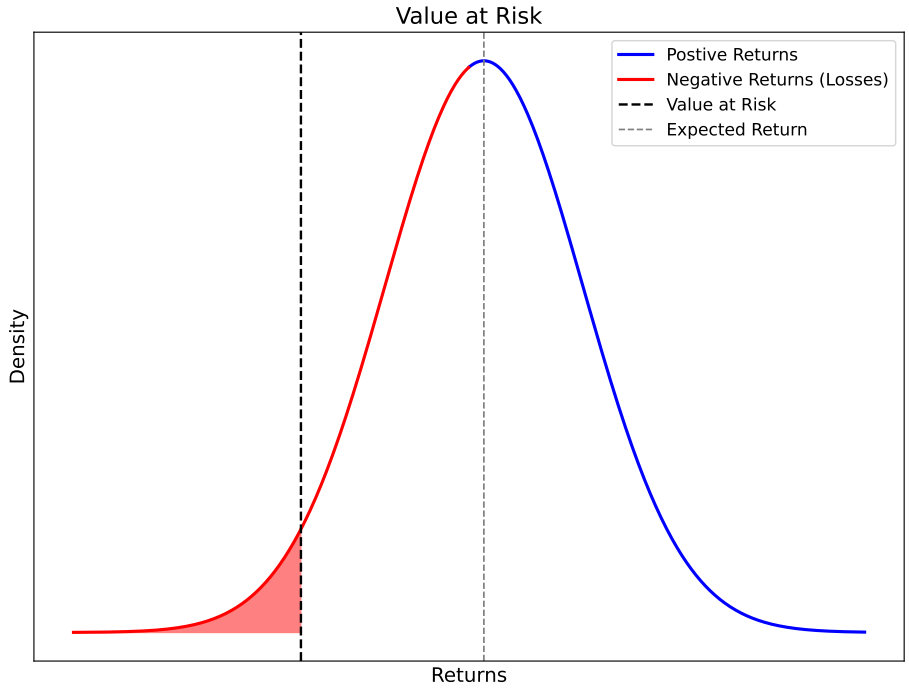


Figure 2.3: Graphical illustration of Value at Risk (VaR) for a given confidence level. The figure shows the return distribution of a portfolio with the VaR threshold marked, highlighting the extreme losses that, beyond the expected loss (EL), determine the economic capital needed to buffer against unforeseen losses.

already accounted for as a distinct item in the institution's financial statements. Thus, Economic Capital does not represent an average loss, as EL does, but rather captures extreme or unforeseen losses.

Unlike EL, VaR is not an additive measure: at a portfolio level it is necessary to consider the joint distribution of individual losses, which depends on the correlation between them. In case of unitary correlations, there won't be any benefit of diversification and the Economic Capital would be equal to the sum of capitals of single exposures.

The diversification of credit risk in a loan portfolio is determined by two factors:

- the granularity of the portfolio, which equates to the number of loans in the portfolio and the existence of single loans which are material in the context of the portfolio as a whole;
- the correlation of the risk characteristics of the debtors within the portfolio.

Typically, these two factors are defined as *specific risk* and *systematic risk*. Specific risk is a function of the granularity of the portfolio, whilst systematic risk is determined by the correlation of the characteristics of the debtor risks. In most banks' portfolios, credit risk is principally a function of systematic risk (arising from the correlation between the characteristics of the debtor risks), whilst specific risk is only significant where there are very large exposures.

Most of the methodologies in credit risk analysis are based on the causal default model developed by Robert Merton [55] and known as the *Asset Based Default Model*. This model assumes that a company will become insolvent if the overall market value of its assets is below the value of its current liabilities. If we know the range of possible asset values over a given time period (typically one year) we can therefore calculate the Probability of Default with reference to the probability that the value of the assets will drop below the value of the liabilities over the time period under consideration.

Continuing with the Merton approach, one can assume a normal distribution for the log of assets value and no drift. Thus, it is possible to introduce a random variable X_i with normal standard distribution: when its value falls below the value of its current liabilities the counterparty i defaults. The following equation represents the dynamics of X_i for counterparty i as the combination of a systemic component Y and an idiosyncratic one γ_i (i.e. a specific risk of counterparty i, as mentioned above):

$$X_i = rY + \sqrt{1 - r^2} \cdot \gamma_i \sim \Phi(0,1) \tag{2.3}$$

In the previous equation, r is the parameter which expresses the linear correlation between each asset and the systemic risk. Y represents the systematic risk, determined as the linear combination of S independent and normally distributed factors:

$$Y = \sum_{k=1}^{S} \alpha_k Z_k \quad \text{with} \quad \sum_{k=1}^{S} \alpha_k^2 = 1; \quad Z_k \sim \Phi(0,1) \text{ i.i.d.} \tag{2.4}$$

where the weightings α_k are derived from the factorisation (using the Cholesky or the spectral decomposition into eigenvalues and eigenvectors) of the *asset correlation matrix*.

The distribution of total portfolio losses, taking into account deterministic LGDs, can be expressed as follows:

$$L(Y) = \sum_{i}^{N_c} LGD_i \cdot 1_{\{X_i < \Phi^{-1}(PD_i)\}} \tag{2.5}$$

where N_c represents the total number of counterparties, LGD_i is the Loss Given Default for counterparty i. If, as a result of adverse development in the Y systematic factor, X_i is caused to fall below the threshold $\Phi^{-1}(PD_i)$, as obtained by inverting the standard normal distribution corresponding to the Probability of Default PD_i, then the counterparty i will default and the second factor in L will go to 1. To detrmine PD_i conditioned on Y we can use the so called *Gaussian Conditional Independence Model*:

$$PD_i|Y = \Phi\left(\frac{\Phi^{-1}\left(PD_i^0\right) - r_i Y}{\sqrt{1-r_i^2}}\right). \tag{2.6}$$

This can be interpreted as can the default scenario in a perfectly granular hypothesis, i.e. when N_c tends towards infinity or is at least a very large number.

The distribution of portfolio losses is numerically projected using a Monte Carlo simulation: in every scenario, the Z factors relating to idiosyncratic risk are generated randomly. According to the model algorithm, for each scenario j, the original PD of each counterparty is thereby transformed into a conditional probability of default $PD_i|Z_j$. As a result, conditional PDs are used as inputs to simulate a binomial random variable to determine how many of the components will default in scenario j, thereby causing the conditional loss $L|Z_j$. The distribution of the losses is thereby produced as function of the conditional portfolio losses by scenario.

If the conditional losses are sorted in ascending order, the credit VaR for a given confidence level (cl), is conventionally calculated as follows:

$$\mathrm{VaR}_{cl}(L) := q_{cl}(L) = \inf\{l \in \mathbb{R} \mid \mathbb{P}[L \leq l] \geq cl\} \tag{2.7}$$

As a reminder, Economic Capital can be calculated as VaR minus expected losses:

$$EC_{cl} = \mathrm{VaR}_{cl}(L) - EL \tag{2.8}$$

The predetermined confidence level is defined each year as part of the process of determining the bank's risk appetite. For example, Economic Capital estimated with a cl of 99.93% is estimated to be capable of absorbing Group credit losses in 99.93% of cases.

2.1.3 Derivative Pricing

Financial derivatives are instruments whose values are contingent on the performance of underlying assets (stocks, bonds, indices, commodities). These instruments are frequently employed by investors to transfer various types of financial risk to other market participants, for whom the (compensated) incremental risk is deemed more attractive and manageable. Derivative pricing is essentially the task of determining the appropriate monetary value to assign to the transfer of such risk.

A core tenet in pricing these instruments is the no-arbitrage or arbitrage-free principle. If a derivative's price deviates from the cost of a portfolio engineered to mimic its cash flows, a risk-free profit opportunity emerges. To prevent this, the pricing mechanism ensures that the derivative's market price aligns exactly with that of its synthetic counterpart. As soon as any brief imbalance appears, traders quickly act on it, driving the prices back into equilibrium so that no risk-free profit can be made.

Among the most frequently traded derivative instruments are *call and put options, futures, forwards,* and *swaps*. See Table 2.1 for some context on these instruments. Options, in particular, are contracts that give the holder the right, but not the obligation, to purchase (*call option*) or sell (*put option*) a certain underlying asset at a specified price (referred to as the *strike price*) at the end of or within a given time frame. It is important to highlight that the strike price can be a fixed value but also a function of several market parameters.

For the simplest types of option, an analytical solution is possible. A pertinent example consists of European options, which can be priced using the renowned Black-Scholes options pricing model. The purpose of the model is to estimate the fair price for a call option under a given set of conditions. The model assumes no arbitrage, a geometric random walk for price evolution (with constant drift and variance), no dividends, a specified risk-free rate, and no fees or costs.

However, the simplifying assumptions required for the model to provide closed-form solutions are a limitation to its applicability. As a result, it is common, particularly for more exotic options, to employ Monte Carlo simulations. This method involves simulating a large number of paths for the time evolution of prices. For each of these paths, it is possible to compute the future payoff and finally average the payoffs across all paths to determine the expected value of the simulated distribution. The option price can then be obtained by discounting this expected value. Monte Carlo methods offer the flexibility to incorporate extreme value functions as well as economic shocks into the analysis. It should also be emphasized that classical Monte Carlo methods require extensive computational resources, making them a suitable candidate for exploring the potential of quantum computing.

Example of payoff function

Consider a European call option with strike price K and maturity T. The payoff function for this option is defined as

$$\pi(S_T) = \max(S_T - K, 0), \tag{2.9}$$

where S_T is the underlying asset price at maturity. This function returns the intrinsic value of the call option: if S_T is greater than K, the option is exercised and the payoff is $S_T - K$; otherwise, if $S_T \leq K$, the payoff is zero. The fair price C_0 of the option is then computed by taking the discounted expected payoff:

$$C_0 = e^{-rT}\mathbb{E}\left[\max(S_T - K, 0)\right], \tag{2.10}$$

with r denoting the risk-free interest rate. In Monte Carlo simulations, many independent realizations of S_T are generated according to the underlying asset's dynamics, the payoff is calculated for each path using the function $\pi(S_T)$, and the average is taken before discounting.

Similarly, for a European put option, the payoff function is given by

$$\pi(S_T) = \max(K - S_T, 0), \tag{2.11}$$

which reflects the right to sell the asset at the strike price K if S_T falls below K.

Derivative Type	**Definition**	**Key Characteristics**
Call Option	Right to buy an asset at a predetermined price	Holder benefits if asset price rises above strike price
Put Option	Right to sell an asset at a predetermined price	Holder benefits if asset price falls below strike price
Futures	Obligation to buy/sell an asset at a future date	Standardized contracts traded on exchanges
Forwards	Obligation to buy/sell an asset at a future date	Customized contracts traded over-the-counter (OTC)
Swaps	Agreement to exchange cash flows based on underlying asset	Commonly used for interest rate and currency risk management

Table 2.1: Summary of main types of derivatives.

2.2 Near-Term Algorithms

The advent of noisy intermediate-scale quantum (NISQ) devices has spurred the development of algorithms specifically designed to work within the constraints of current quantum hardware. In this regime, quantum processors have a limited number of qubits, restricted coherence times, and are susceptible to various noise sources, making the execution of deep, fault-tolerant quantum circuits impractical. To address these challenges, researchers have turned to *variational quantum algorithms* [13].

Variational algorithms combine quantum and classical computation in a hybrid framework. They rely on a parameterized quantum circuit (or *ansatz*) to prepare a quantum state that encodes the solution to a problem. A classical optimization loop is then used to adjust the parameters, minimizing a cost function defined in terms of a Hamiltonian operator that represents the problem. This approach is particularly well-suited to NISQ devices because it generally requires shallower circuits and can tolerate a degree of noise through iterative refinement.

Well-known examples of variational algorithms include the Variational Quantum Eigensolver (VQE) and the Quantum Approximate Optimization Algorithm (QAOA). These methods have shown promise in a variety of applications ranging from quantum chemistry to combinatorial optimization.

In the following subsection, we focus on *Quantum for Portfolio Optimization*, where variational approaches leverage NISQ devices to find near-optimal solutions by encoding the portfolio selection problem into a cost function and optimizing the corresponding quantum circuit parameters via a classical algorithm.

2.2.1 Quantum for Portfolio Optimization

The task of optimally selecting assets for portfolio construction can be formulated as a Quadratic Unconstrained Binary Optimization (QUBO) problem. The quadratic nature of this problem is evident from the presence of mixed terms in its characteristic equation:

$$V = w^T \Sigma w, \tag{2.12}$$

which relates the portfolio variance V to the vector w of proportional investments in each stock, and to the variance-covariance matrix Σ. Moreover, the model is unconstrained since no additional restrictions are imposed during the optimization process. In addition, it exhibits a combinatorial nature (the decision variables can be discrete), as the objective is to identify a discrete set of assets to then build with those the optimal portfolio. In general, problems of this type are non-convex and cannot be solved in polynomial time using classical methods.

Within the field of quantum computing, various proposals suggest the use of variational algorithms for solving optimization problems formulated as QUBO. In particular, heuristic approaches based on the Variational Quantum Eigensolver (VQE) [79] and the Quantum Approximate Optimization Algorithm (QAOA) [10] can be executed on devices belonging to the NISQ era. It is important to emphasize that, although these algorithms are inefficient to simulate on a classical computer, there is no theoretical guarantee that they provide a significant speedup over classical methodologies for solving QUBO problems, despite the presence of promising empirical results.

To understand why variational approaches are particularly suited for solving QUBO problems, it is important to note that every QUBO can be transformed into an Ising model [44]:

$$\min_s E(s) = -\sum_i h_i s_i - \sum_{i,j} J_{ij} s_i s_j, \quad s_i \in \{-1,1\} \tag{2.13}$$

with a specific Hamiltonian that consists of a combination of Z Pauli operators (see Section 1.3.4)

$$\mathcal{H}_P = -\sum_i h_i \sigma_i^z - \sum_{i<j} J_{ij} \sigma_i^z \sigma_j^z, \tag{2.14}$$

where σ_i^z is the Pauli matrix Z acting on qubit i (while, in Equation 2.13, s_i represents the spin), h_i is the external longitudinal magnetic field on qubit i, and J_{ij} defines the coupling strength between qubits i and j.

This conversion is possible thanks to a one-to-one mapping of the binary variables to spin variables:

$$x_i = \frac{1+s_i}{2}. \tag{2.15}$$

Given the Ising formulation of the QUBO, it is possible to use VQE (detailed in Algorithm 4) to solve the optimization problem.

Analogously, the QAOA algorithm can be employed to achieve the same result. In this case, it is necessary to define two sets of rotation parameters:

$$\boldsymbol{\beta} = (\beta_1, ..., \beta_p),$$
$$\boldsymbol{\gamma} = (\gamma_1, ..., \gamma_p).$$

Furthermore, in addition to the Ising Hamiltonian matrix $\mathcal{H}_C \in \mathbb{C}^{n\times n}$, the algorithm also requires a mixing Hamiltonian $\mathcal{H}_B \in \mathbb{C}^{n\times n}$, which is constructed as a sum of Pauli X operators. For further details, we refer the reader to Algorithm 5.

Going back to portfolio optimization, the primary objective is to solve a combinatorial optimization problem characterized by the following formulation:

$$\min_{\boldsymbol{w}} \boldsymbol{w}^T \Sigma \boldsymbol{w} \qquad \text{s.t.} \sum_{w_i \in \boldsymbol{w}} w_i = 1 \text{ and } \boldsymbol{\mu}^T \boldsymbol{w} = r, \tag{2.16}$$

where r represents the target return for the portfolio, $\boldsymbol{\mu}$ denotes the vector of expected returns for the assets, and the constraint $\sum_{w_i \in \boldsymbol{w}} w_i = 1$ signifies that the entire budget must be allocated.

Since we need to work with binary variables, we approximate each continuous variable w_i using a finite set of binary variables $x_{i,a}$:

$$w_i = \sum_{a=1}^{k} 2^{-a} x_{i,a}, \quad x_{i,a} \in \{0,1\}. \tag{2.17}$$

This ensures that each weight w_i is represented using k binary variables, with a resolution controlled by k.

Algorithm 4 Variational Quantum Eigensolver (VQE)

Require: Hamiltonian $\mathcal{H}$, parameterized ansatz $|\psi(\theta)\rangle$, classical optimizer
Ensure: Approximate ground state energy $E_{\min}$ and optimal parameters θ^*

Initialize: Choose an initial set of parameters $\theta = \theta_0$
repeat
  Step 1: Prepare Quantum State
  Apply the parameterized quantum circuit $U(\theta)$ to initialize the ansatz state:

$$|\psi(\theta)\rangle = U(\theta)|0\rangle^{\otimes n}$$

  Step 2: Measure Expectation Value
  Measure the expectation value of the Hamiltonian:

$$E(\theta) = \langle\psi(\theta)|\mathcal{H}|\psi(\theta)\rangle$$

  Estimate $E(\theta)$ using multiple measurements
  Step 3: Classical Optimization
  Update the parameters θ using a classical optimizer:

$$\theta \leftarrow \theta - \eta\nabla E(\theta)$$

  where η is the learning rate (for gradient-based methods)
until Convergence criteria met (e.g., energy difference below threshold ϵ)

return Approximate ground state energy $E_{\min} = E(\theta^*)$ and optimal parameters θ^*

Algorithm 5 Quantum Approximate Optimization Algorithm (QAOA)

Require: Problem Hamiltonian $\mathcal{H}_C$, Mixing Hamiltonian $\mathcal{H}_B$, depth parameter p, classical optimizer

Ensure: Approximate solution state $|\psi(\boldsymbol{\beta}, \boldsymbol{\gamma})\rangle$ and optimal parameters $(\boldsymbol{\beta}^*, \boldsymbol{\gamma}^*)$

Initialize: Choose initial parameters $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_p)$ and $\boldsymbol{\gamma} = (\gamma_1, \ldots, \gamma_p)$

Prepare the initial state:

$$|\psi_0\rangle = |+\rangle^{\otimes n}$$

repeat

Step 1: Apply Problem Hamiltonian Evolution

Apply unitary evolution with $\mathcal{H}_C$:

$$|\psi_1\rangle = e^{-i\gamma_1 \mathcal{H}_C} |\psi_0\rangle$$

Step 2: Apply Mixing Hamiltonian Evolution

Apply unitary evolution with $\mathcal{H}_B$:

$$|\psi_2\rangle = e^{-i\beta_1 \mathcal{H}_B} |\psi_1\rangle$$

Repeat steps 1 and 2 for p layers:

$$|\psi(\boldsymbol{\beta}, \boldsymbol{\gamma})\rangle = \prod_{j=1}^{p} e^{-i\beta_j \mathcal{H}_B} e^{-i\gamma_j \mathcal{H}_C} |+\rangle^{\otimes n}$$

Step 3: Measure Expectation Value

Measure the expectation value:

$$E(\boldsymbol{\beta}, \boldsymbol{\gamma}) = \langle \psi(\boldsymbol{\beta}, \boldsymbol{\gamma}) | \mathcal{H}_C | \psi(\boldsymbol{\beta}, \boldsymbol{\gamma}) \rangle$$

Estimate $E(\boldsymbol{\beta}, \boldsymbol{\gamma})$ using multiple measurements

Step 4: Classical Optimization

Update the parameters $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ using a classical optimizer:

$$(\boldsymbol{\beta}, \boldsymbol{\gamma}) \leftarrow (\boldsymbol{\beta}, \boldsymbol{\gamma}) - \eta \nabla E(\boldsymbol{\beta}, \boldsymbol{\gamma})$$

where η is the learning rate (for gradient-based methods)

until Convergence criteria met

return Approximate solution state $|\psi(\boldsymbol{\beta}^*, \boldsymbol{\gamma}^*)\rangle$ and optimal parameters $(\boldsymbol{\beta}^*, \boldsymbol{\gamma}^*)$

Now we need to introduce the QUBO objective and convert the constraints into penalty functions:

$$Q = \sum_{i,j} \Sigma_{i,j} w_i w_j + \lambda_1 \left(\sum_i \mu_i w_i - r \right)^2 + \lambda_2 \left(\sum_i w_i - 1 \right)^2. \tag{2.18}$$

Expanding the squared constraints and eliminating constants:

$$Q = \sum_{i,j} \Sigma_{i,j} w_i w_j + \lambda_1 \left(\sum_i \mu_i w_i \right)^2 - 2\lambda_1 r \sum_i \mu_i w_i + \lambda_2 \left(\sum_i w_i \right)^2 - 2\lambda_2 \sum_i w_i. \tag{2.19}$$

Replacing w_i with its binary expansion:

$$\begin{aligned} f(x) = &\left(\sum_i \sum_a \mu_i 2^{-a} x_{i,a} \right)^2 - 2r \sum_i \sum_a \mu_i 2^{-a} x_{i,a} + r^{-2} \left(\sum_i \sum_a 2^{-a} x_{i,a} \right)^2 \\ &- 2 \sum_i \sum_a 2^{-a} x_{i,a} + \lambda_3 \sum_{i,j} \sum_{a,b} \Sigma_{i,j} 2^{-a-b} x_{i,a} x_{j,b}. \end{aligned} \tag{2.20}$$

This final expression represents the optimization problem in a discrete quadratic form, that we can use to build the Ising Hamiltonian that represents the input for QAOA and VQE. In particular, an Ising Hamiltonian has the form seen in Equation 2.13.

To convert Equation 2.20 into an Ising Hamiltonian, we need to map the binary variables $x_{i,a} \in \{0,1\}$ to spin variables $s_{i,a} \in \{-1,+1\}$ using the transformation 2.15. Now, this substitution would lead to a new expression entirely in terms of spin variables $s_{i,a}$, which would then be suitable for minimization with our variational algorithms using QPUs or quantum annealing platforms [67].

2.3 Algorithms That Require Fault-Tolerance

Quantum algorithms promising exponential speedups, such as Shor's factoring algorithm or advanced quantum Monte Carlo methods, often assume the availability of large-scale, fault-tolerant quantum processors. Fault-tolerance in quantum computing refers to the system's ability to perform reliable computations even in the presence of noise and errors. This is achieved by implementing quantum error correction (QEC) schemes, which encode a logical qubit into a larger number of physical qubits using error-correcting codes (e.g., the surface code, color codes, or concatenated codes) [22]. QEC works by detecting and correcting errors without collapsing the quantum state, thereby preserving superposition and entanglement—the key resources for quantum computation.

Fault-tolerant protocols involve careful design of quantum circuits such that errors can be systematically identified and corrected throughout the computation.

Although the overhead is significant (often requiring hundreds to thousands of physical qubits per logical qubit), achieving fault-tolerance is essential for executing deep quantum circuits and complex algorithms reliably.

2.3.1 Quantum for Risk Analysis

Recently, there has been a growing interest in the application of quantum computing to credit risk analysis [74]. Specifically, the focus is on the Monte Carlo process used to estimate fundamental metrics such as the Value at Risk.

Primarily, this interest stems from the proposal of an algorithm called quantum Monte Carlo [57]. This algorithm is based on the quantum amplitude estimation subroutine and, as such, offers a quadratic speedup compared to the traditional classical simulation method.

For comprehensiveness, however, we must mention that there are also proposals to maintain the classical simulation process but enhance it with the use of quantum computers (or other quantum hardware) to leverage the inherently random nature of quantum phenomena in order to replace the pseudorandom processes underlying the classical Monte Carlo method.

Our focus will primarily be on quantum Monte Carlo, which is employed in [86, 28] for credit risk analysis. This implementation uses three registers in addition to a target qubit, whose amplitude estimation is mapped to the probability that the aggregate loss (resulting from the sum of the LGD of all defaulted counterparties) is less than or equal to a given threshold.

The three registers are shown in Figure 2.4.

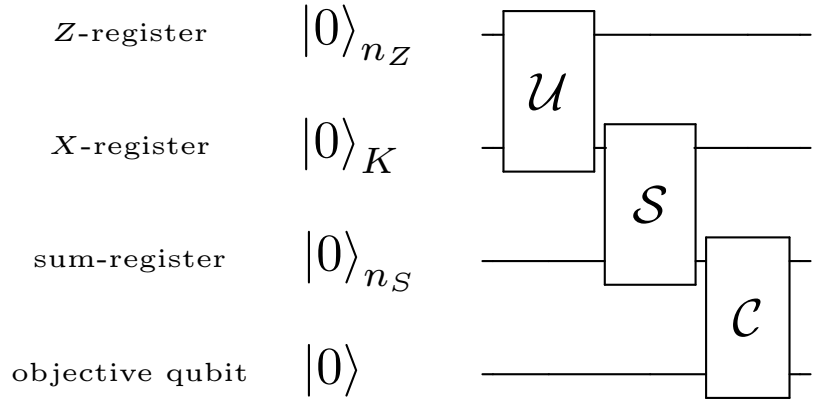


Figure 2.4: High-level circuit comprising all the components of the state preparation $\mathcal{A}$ used in the original implementation. Source: [28].

- The Z register is designated to encode a single systemic risk factor, represented by a normally distributed random variable (appropriately truncated and discretized to be approximated by the binary register).

- The X register contains K qubits corresponding to the number of counterparties. Each qubit assumes a value of 1 if the respective counterparty is in default, or 0 if there is no default event for that counterparty. The probability of observing a 'one' upon measurement corresponds to the default probability of each counterparty, as this is conditioned on the realization of the systemic risk variable. Controlled rotations by the qubits in the Z register, targeting the qubits in the X register, will encode this probability conditioned on the state of the Z register. An example can be observed in Figure 2.7.

- The sum register will contain the weighted sum of the values in the counterparties register: essentially, it will consider the qubits corresponding to defaulted counterparties, and each will be associated with its own LGD. These are then summed to record the total aggregate loss in the register (for that specific *quantum sample*). The weighted summation operation is performed by the sum operator $\mathcal{S}$ (an example is depicted in Figure 2.5). For details on its original implementation we refer to [28].

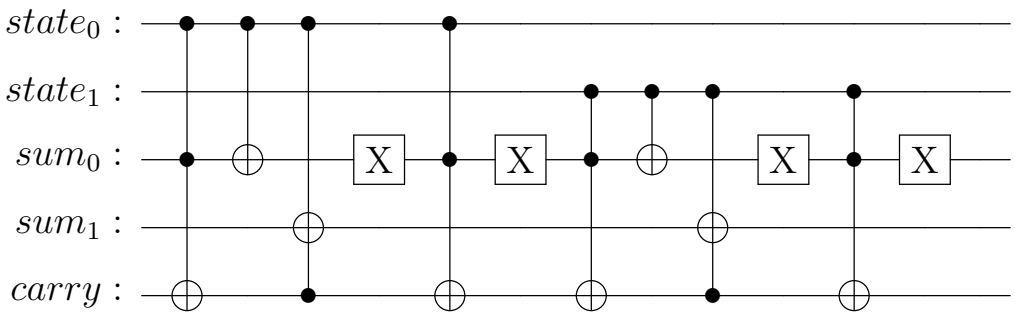


Figure 2.5: Example of the weighted adder operator implemented in Qiskit [84].

- Finally, the objective qubit is the one in which the probability of $|1\rangle$ corresponds to $a = \langle\Psi_1|\Psi_1\rangle$. This is achieved using a comparator operator $\mathcal{C}$, which compares the total aggregate loss in the sum register with a given reference value. If the value is less than or equal to the reference value, the objective qubit is flipped to 1.

Note how the resulting circuit effectively estimates the value of the cumulative distribution function (CDF) for a given value of the aggregate total loss distribution. Using this circuit as the state preparation operator $\mathcal{A}$ in a QAE subroutine we can estimate CDF for a given target precision with quadratically fewer samples with respect to traditional Monte Carlo. The resulting circuit can be executed iteratively (estimating the CDF for different values of the aggregated loss distribution) in a

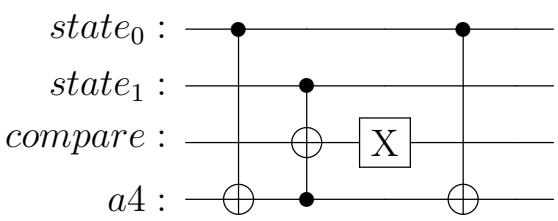


Figure 2.6: Example of the comparator operator implemented in Qiskit [43].

bisection search process to optimally converge to an estimate for VaR (defined in Equation 2.7) given a predefined confidence level.

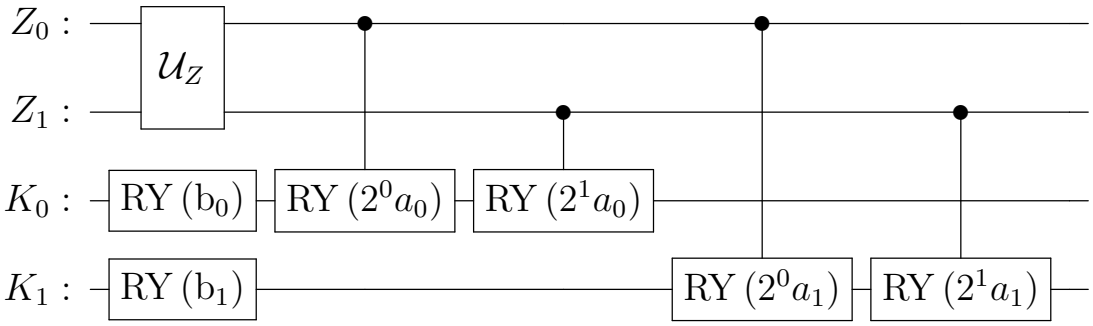


Figure 2.7: Example of the uncertainty operator $\mathcal{U}$. Here $n_z = 2$ and $K = 2$.

A crucial component is the so-called uncertainty operator $\mathcal{U}$, which is responsible for encoding the uncertainty model within the circuit. It essentially first encodes the normally distributed random variable into the qubits of the Z register, note that any discretized and truncated log-concave distribution can be efficiently represented in a quantum register using controlled rotations (hidden inside $\mathcal{U}_Z$ in Figure 2.7). Then, it reads the same register and, conditional on the register's value, applies a specific combination of RY rotations to the qubits representing the counterparts. Additionally, a rotation on the same axis is performed on these qubits, which is not controlled by any qubit. This is used to encode the intrinsic default probability of each counterparty, which does not depend on the risk factor. An example is presented in Figure 2.7.

The depth of the circuit implementing U scales as $O(n_z K)$ and thus is linear in the number of counterparties. The total number of qubits will scale like $O(K)$, since each debtor (exposure) is represented with a single qubit.

2.3.2 Quantum for Pricing Derivatives

Another key task in finance for which numerous quantum implementations have been recently proposed is option pricing. As already mentioned, in scenarios extending beyond the Black-Scholes-Merton (BSM) model [8], analytical approaches are often too simplistic. Therefore, the dominant approach for such pricing involves utilizing Monte Carlo evaluation techniques. Monte Carlo simulations entail extracting M samples from the input distributions. These samples are then employed to estimate the metric of interest, achieving a confidence interval that scales as $O(1/\sqrt{M})$. As mentioned in Section 2.3.1, proposals exist to obtain a quadratic improvement in the amount of samples needed exploiting QAE [57]. Since the use of options is quite extensive in the finance industry, improving their pricing methodology represents a desirable achievement for financial entities.

Building on this premise and starting from the proposals in [70], which introduced the earliest quantum algorithm for valuing European and Asian options, several works dealt with the complexities of quantum option pricing [75, 51, 14, 15].

Summarizing the literature, we can identify three key components for option pricing with quantum computers:

1. The ability to represent the probability distribution describing the evolution of a multivariate random variable $W = \{w_1, w_2, \ldots, w_N\}$ corresponding to the possible values w_i the underlying assets can take at maturity, and the associated probabilities p_i that those values will be realized. This model has to be loaded into a quantum register such that each basis state represents a possible value and its amplitude the corresponding probability. The resulting encoding for an $n-$qubits register would be:

$$|\psi\rangle_n = \sum_{i=0}^{2^n-1} \sqrt{p_i}\,|w_i\rangle_n \tag{2.21}$$

   From the literature [14], two strategies for state preparation emerge: while the first is based on loading the distribution of asset prices, the second loads the distribution of log-returns. Moving from the *price space* representation to the *return space* one presents a notable advantage, since, when asset prices obey a log-normal distribution, then the log-returns are distributed normally. Therefore switching between price space and return space changes from log-normal distribution loading to normal distribution loading, which is considerably easier [69]. The latter can be loaded by preparing several standard normals distributions in parallel and then applying affine transformations to get the means and assets' covariance for each Gaussian [14].

2. Obtain the circuit which computes a scaled version $\tilde{f}$ of the payoff function

f inside the amplitude of a target qubit:

$$\sum_{i=0}^{N-1} \sqrt{(1-\tilde{f}\,(w_i))p_i}\,|w_i\rangle\,|0\rangle + \sum_{i=0}^{N-1} \sqrt{\tilde{f}\,(w_i)\,p_i}\,|w_i\rangle\,|1\rangle \tag{2.22}$$

3. Calculate the expectation value of the payoff $\mathbb{E}[f]$ using QAE. Remember that Amplitude Estimation aims at efficiently estimating a in Equation 1.48 using an operator $\mathbf{Q} = \mathcal{A}S_0\mathcal{A}^\dagger S_\chi$. Considering $\mathcal{A}$ as the operator that creates the state in Equation 2.22, the value a to estimate is:

$$a = \sum_{i=0}^{N-1} \tilde{f}\,(w_i)\,p_i = \mathbb{E}[\tilde{f}] \tag{2.23}$$

It is then possible to reconstruct the expected payoff $\mathbb{E}[f]$ by post-processing the output.

As noted in [14], the pricing of options through quantum computers faces challenges due to the resource requirements for path loading and payoff calculations.

As illustrated in [42], to achieve a practical quantum advantage, it is necessary to obtain an efficient implementation of $\mathbf{Q}$: the time required to obtain a *quantum sample* by applying $\mathbf{Q}$ should be comparable to that required to obtain a classical sample.

For this reason, many works have focused on developing techniques for the efficient implementation of $\mathbf{Q}$. For instance, several proposals concentrate on the efficient encoding of generalized normalized L_1 or L_2 functions [39, 72, 82, 83, 4]. The idea is that the target state is prepared with a success probability that is then boosted using QAA. However, these algorithms prove to be not particularly efficient when the target distribution is concentrated around a few values [42].

It is also worth mentioning the existence of methods that load polynomial approximations of functions [54] and of variational methods trained to load distributions, such as those required for derivative pricing [88, 60].

Note that, in the case of tabulated data, one resorts to generic state-loading techniques [87, 58, 1] that scale linearly with the number of input data points. For this reason, it may sometimes be more advantageous not to load the complete payoff distribution but instead to load the joint distribution over path increments [14]. In this approach, the linear scaling is with respect to the number of time steps, and consequently, costly quantum arithmetic is required.

In [42], the authors also mention a deterministic classical method known as quasi-MCI, which, for low-dimensional problems, achieves an error scaling comparable to that of QMCI. However, this benefit comes at the expense of an exponential dependence on the dimension—a dependence that, in practice, does not appear for certain financial problems [34].

Given the improvement in query complexity that QMCI provides for the widely employed Monte Carlo method, the community has begun to develop algorithms for pricing various types of derivatives, such as European options [68], Asian options [75], and autocallable and target accrued redemption forwards (TARFs) [14].

Again, note that analytic forms can simplify path loading in cases like path-independent derivatives. In such cases, only the final underlying price distribution is necessary. For example, the Black-Scholes model provides a log-normal distribution that can be computed analytically and loaded into quantum states. However, having an analytic form means the derivative is typically easy to price classically. Thus, towards the final part of Chapter 4, our focus will shift to path-dependent derivatives that require a superposition over paths.

Chapter 3

Improving Quantum Credit Risk Analysis

As noted earlier, a crucial component of executing credit risk analysis (depicted schematically in Figure 3.1) on quantum computers is the state preparation operator $\mathcal{A}$. Specifically, it must be sufficiently sophisticated to encode realistic models that effectively correspond to those used by major financial institutions (for instance, it would be desirable to encode models with more than one systemic risk factor). At the same time, it must scale reasonably with respect to the problem size, ensuring that the resulting circuit depth remains manageable.

In [26, 27], the authors illustrate how it is indeed possible to obtain the encoding of an uncertainty model that considers multiple normally distributed random variables, whereby the probability of default is conditioned on a linear combination of the realizations of these variables. More interestingly, this implementation also avoids some of the costly arithmetic necessary to execute the algorithm according to the original implementation presented in [28]. However, the proposal in [26, 27] results in an increase in the number of gates required, which scales as $O(2^K)$, where K is the number of counterparties. This is because each scenario of counterparty loss, i.e., every possible subset of the set of counterparties, necessitates a specific rotation controlled by the entire counterparties register. As a result, the circuits rapidly attain infeasible execution depths, particularly for realistic cases.

A potential solution to mitigate this problem involves utilizing the Quantum Singular Value Transformation (QSVT) [33] and taking inspiration from [76] to combine the benefit of avoiding costly arithmetic with scaling that remains linear in the number of counterparties.

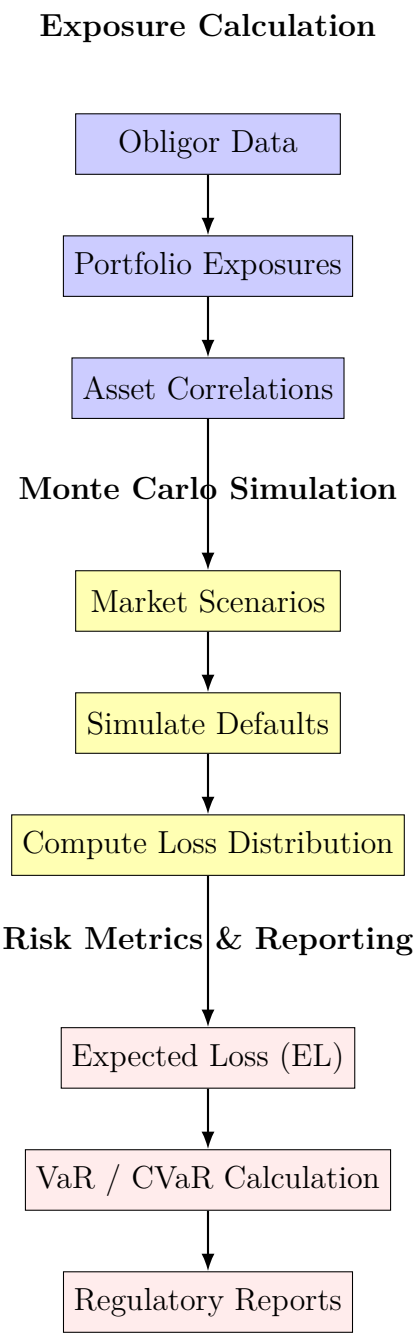


Figure 3.1: Overview of the Credit Risk Assessment pipeline. The process starts with the calculation of exposures based on obligor data and asset correlations. Monte Carlo simulations then generate market scenarios and simulate defaults to estimate the loss distribution. Finally, key risk metrics are computed and compiled into regulatory reports.

3.1 Quantum Singular Value Transformation

QSVT is a fundamental framework in quantum computing that generalizes a wide array of quantum algorithms by enabling polynomial transformations of the singular values of block-encoded matrices. Introduced by Gilyén et al. [33], QSVT generalizes qubitization and quantum signal processing (QSP) [52], providing a systematic method for designing quantum algorithms that manipulate structured linear algebraic problems. The aim of this Section is to provide a basic understanding of QSVT and to show its potential and the principles underlying it.

The methodology is based on the possibility to manipulate the singular values of a matrix A using a polynomial function $P(A)$ expressed in terms of unitary transformations. Thus, given a matrix A with a singular value decomposition (SVD) $A = U\Sigma V^\dagger$, it is possible to construct a quantum operator that selectively acts on the singular values of A, without the need for the explicit computation of the spectral decomposition.

QSVT operates on matrices that are block-encoded within a larger unitary matrix. A matrix $A \in \mathbb{C}^{m\times n}$ is said to be block-encoded in a unitary $U_a \in \mathbb{C}^{(m+p)\times(n+p)}$ if there exists a normalization factor α such that:

$$U_a = \begin{bmatrix} A/\alpha & * \\ * & * \end{bmatrix}, \tag{3.1}$$

where the unspecified entries represent additional degrees of freedom that ensure U_a is unitary. The parameter α determines the effective condition number and query complexity in QSVT-based algorithms.

Thus, given a block-encoded A, QSVT enables transformations of the form:

$$P(A) = \sum_i P(\sigma_i)|u_i\rangle\langle v_i|, \tag{3.2}$$

where σ_i are the singular values of A, and $P(\cdot)$ is the polynomial transformation applied directly to them. This transformation is achieved via controlled phase modulation techniques that act on a quantum system embedding A. Specifically, the polynomial function $P(A)$ can be built using sequences of parametrized unitary transformations with adequately chosen angles. In particular, we can express $P(A)$ as a combination of reflection and rotation operators. Let's define two foundamental operators:

$$U_A = \begin{bmatrix} A & \sqrt{I - A^2} \\ \sqrt{I - A^2} & -A \end{bmatrix},$$

$$U_B = \begin{bmatrix} I & 0 \\ 0 & -I \end{bmatrix}.$$

Using such operators, we can build the above mentioned sequence of unitary transformations which implement the desired polynomial function on the singular value of the matrix.

Therefore, we obtain a general protocol for the application of arbitrary transformations to the singular values of a matrix. One of the key features of this technique is its ability to selectively modify the eigenvalues without altering the overall structure of the operator. This proves particularly useful in applications such as solving linear systems, spectral filtering, and quantum data compression.

The practical implementation of QSVT requires a careful optimization of the circuits' depth, especially if the execution has to be conducted on NISQ hardware, where noise and decoherence can hinder the fidelity of quantum operations. To mitigate these issues, researchers are developing error-correction and error-mitigation techniques [22, 78].

3.1.1 Quantum Signal Processing

A core component of QSVT is Quantum Signal Processing (QSP) [52], which provides a mechanism for implementing polynomial transformations through phase modulation. The primary idea of QSP is that a sequence of unitary transformations can be decomposed into a phase-modulated product of a given unitary matrix and simple phase rotations. Specifically, for a unitary operator U with eigenvalues $e^{i\theta}$, QSP constructs a polynomial transformation $P(\cos\theta)$ by applying a sequence of alternating phase-modulated operations:

$$U(\theta) = e^{i\phi_0} \prod_{k=1}^{d} \left(W_{\phi_k} U\right) W_{\phi_k}, \tag{3.3}$$

where the phase rotations are given by:

$$W_{\phi_k} = \begin{bmatrix} e^{i\phi_k} & 0 \\ 0 & e^{-i\phi_k} \end{bmatrix}. \tag{3.4}$$

This sequence allows the construction of any even or odd polynomial transformation $P(x)$ within bounded intervals, constrained by Chebyshev approximation theory.

3.1.2 Chebyshev Polynomial Approximation

QSVT relies on the decomposition of arbitrary functions into Chebyshev polynomials. Given a function $f(x)$ that we wish to approximate, we express it as a Chebyshev series:

$$f(x) \approx \sum_{k=0}^{d} c_k T_k(x),$$

where $T_k(x)$ are Chebyshev polynomials of the first kind, defined recursively as:

$$T_0(x) = 1, \quad T_1(x) = x, \quad T_{k+1}(x) = 2xT_k(x) - T_{k-1}(x).$$

By leveraging the optimal properties of Chebyshev polynomials in minimizing uniform approximation error, QSVT ensures that the polynomial transformation remains within quantum resource constraints while achieving near-optimal accuracy.

The degree of the polynomial $P(x)$ directly influences the number of quantum queries required to implement a QSVT-based transformation. The query complexity of QSVT is determined by the degree d required to approximate the target function within a desired precision ϵ. Given a function $f(x)$ with bounded derivatives, the asymptotic query complexity scales as:

$$O(d) = O(\log(1/\epsilon) \cdot \kappa),$$

where κ is an effective condition number that depends on the transformation domain. This scaling highlights the near-optimal performance of QSVT in comparison to classical polynomial approximation methods.

To implement QSVT efficiently, the controlled evolution of the block-encoded matrix must be realized via a sequence of selective phase rotations and controlled reflections. The standard approach is to construct a sequence of alternating applications of the block-encoding unitary U_A and phase-modulated reflections:

$$V = e^{i\phi_0} \prod_{k=1}^{d} (U_A W_{\phi_k}).$$

The exact choice of phase angles $\{\phi_k\}$ is determined by solving a system of trigonometric equations derived from the Chebyshev expansion. This phase selection process is computationally nontrivial but can be efficiently precomputed for a broad class of polynomial transformations.

A crucial property of QSVT is that it preserves the spectral structure of the transformed operator while modifying only the singular values. Given a block-encoded operator A, the transformed matrix $P(A)$ retains the original left and right singular vectors $|u_i\rangle$ and $|v_i\rangle$, ensuring that:

$$P(A)|v_i\rangle = P(\sigma_i)|u_i\rangle.$$

This property is essential for many quantum algorithms that require controlled conditioning of singular values without perturbing vector orientations.

3.2 Better State Preparation with QSVT

In [26] the authors show that in a multivariate context, it is possible to build a circuit whose final state models the possible default scenarios. It is indeed possible to model the counterparties as binary variables, where the probability of assuming the value 1 corresponds to the probability of default. In this way, a specific scenario is represented by a specific configuration of this register of binary variables.

As an example, let us consider having three counterparties and reading the value of the register of these three binary variables, obtaining a given bitstring. We will have 8 possible scenarios:

- *000*
- *001*
- *010*
- *011*
- *100*
- *101*
- *110*
- *111*

Each bitstring corresponds to a different default scenario: the first represents the case where no counterparties default, while the last represents the scenario where all three counterparties default. The intermediate cases correspond to all possible combinations of individual defaults.

To construct the crucial state preparation operator $\mathcal{A}$, an operator U is needed to associate each scenario with its specific probability of occurrence. In [27] such an operator is proposed for the multivariate case in which we would like to model the systemic risk with a linear combination of normally distributed random variables.

Regardless of the amount of risk factors, for every $i \in \{1, \ldots, n\}$, we can denote with l_i the loss given default of the i-th counterparty. For each configuration j of possible defaults, represented by binary variables, let j_i be the i-th binary digit (0 or 1) of $j \in \{0, \ldots, 2^K - 1\}$. The total loss for that configuration, L_j, is then given by:

$$L_j = \sum_{i=1}^{K} j_i l_i.$$

We'll call L_M The maximal loss, namely L_{2^K-1}.

In order to avoid costly arithmetic, we must encode the aggregated loss for a given configuration into the amplitude of an additional qubit. Recently, in [76], the authors proposed the following encoding:

$$F \left|j\right\rangle_K \left|0\right\rangle = \sqrt{\frac{L_j}{L_M}} \left|j\right\rangle_K \left|1\right\rangle + \sqrt{1 - \frac{L_j}{L_M}} \left|j\right\rangle_K \left|0\right\rangle \tag{3.5}$$

so that the probability of obtaining 1 on the additional qubit equals the expected loss scaled by L_M. This solution can provide results equivalent to algorithms implemented using a sum register but requires something acting analogously to the comparator operator, but on amplitudes. This is why we need QSVT [76].

What we would like as a result is to obtain an effective state preparation operator $\mathcal{A}$ with linear scaling (for the use case under analysis) and implement it.

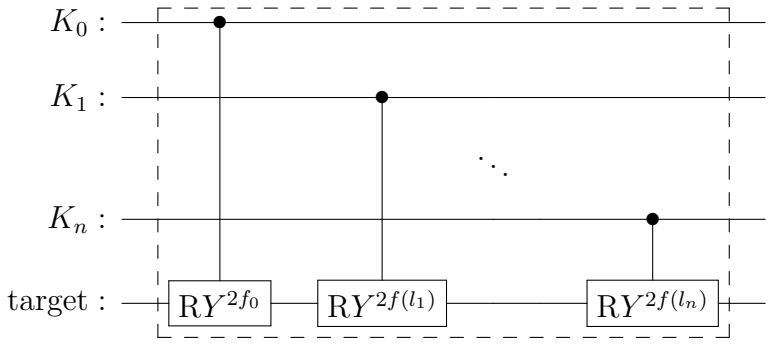


Figure 3.2: Schematic circuit for the amplitude loading operator F. The dashed box encloses the controlled Y-rotations applied to the target qubit, which encode the loss information from the counterparty registers.

Let f be an additive real function. We construct a circuit F (see Figure 3.2) acting on the counterparties register and on the additional qubit by applying for every $i \in \{1, \ldots, n\}$ a rotation about y to the additional qubit of an angle $2f(l_i)$, controlled by the i-th qubit of the counterparties register. Since f is additive, we have

$$\sum_{i=1}^{n} j_i f(l_i) = f\left(\sum_{i=1}^{n} j_i l_i\right) = f(L_j)$$

and therefore this implementation of F is such that

$$F\left|j\right\rangle_K\left|0\right\rangle = \cos\left(\theta_0 + f(L_j)\right)\left|j\right\rangle_K\left|0\right\rangle + \sin\left(\theta_0 + f(L_j)\right)\left|j\right\rangle_K\left|1\right\rangle \tag{3.6}$$

and, importantly, linearly scales with respect to the number of counterparties.

The other needed ingredient relates to the threshold function $\mathcal{T}_\mu(x)$, that is

$$\mathcal{T}_\mu(x) = \begin{cases} 1 & \text{when } |x| \leq \mu \\ 0 & \text{when } |x| \geq \mu \end{cases}.$$

For some real number $\mu \in (0,1)$. Since the threshold function is even and has only two discountinuities in μ and $-\mu$, for any ϵ and δ positive real numbers, we can find an *even* polynomial P such that

$$|P(x) - \mathcal{T}_\mu(x)| < \epsilon$$

for every $x \in (-1,1)$ such that $||x| - \mu| > \delta$. Naturally, as the values of δ and ϵ decrease, the degree of the polynomial required will become larger.

We want to apply QSVT using P, but unfortunately we cannot directly apply it. In fact, there exists the possibility for the polynomial approximations to exceed 1 in absolute value. However, we can easily bypass this limitation by approximating $\mathcal{T}_\mu(x)$ with $c\mathcal{T}_\mu(x)$ for some proper positive $c < 1$. We will show below that this workaround does not affect the final solution.

So, let us apply QSVT to $\mathcal{A}$ using:

- an even polynomial of degree d approximating $c\mathcal{T}_\mu(x)$ in $(-1,1)$;
- $\Pi = |0\rangle\langle 0|$ and $\tilde{\Pi} = |1\rangle\langle 1|$ as projectors.

Observe that $\tilde{\Pi}F\Pi$ is a diagonal matrix, and hence, with an appropriate selection of f and θ_0, it is already expressed in terms of its singular values. Thus, applying QSVT we obtain an operator Q acting on the counterparties registers, the target qubit and on an auxiliary qubit:

$$\begin{aligned}
|0\rangle\langle 0|\,\Pi Q \Pi\,|0\rangle\langle 0| = \\
&= \sum_{j=0}^{2^n-1} P\sin(\theta_0 + f(L_j))\,|j\rangle\langle j|\,|0\rangle\langle 0|\,|0\rangle\langle 0| \\
&\approx \sum_{j=0}^{2^n-1} c\mathcal{T}_\mu \sin(\theta_0 + f(L_j))\,|j\rangle\langle j|\,|0\rangle\langle 0|\,|0\rangle\langle 0| \\
&= \sum_{j \text{ s.t. } |\sin(\alpha_j)|<\mu} c\,|j\rangle\langle j|\,|0\rangle\langle 0|\,|0\rangle\langle 0| \qquad (3.7)
\end{aligned}$$

where $\alpha_j = \theta(L_j) + \theta_0$ for every $j \in \{0,\dots,2^n-1\}$. Concatenating Q and the U operator described in [26] we obtain

$$\begin{aligned}
QU\,|0\rangle_{n_Z}\,|0\rangle_K\,|0\rangle_{target}\,|0\rangle \approx \\
&\approx \sum_{j \text{ s.t. } |\sin(\alpha_j)|<\mu} c\sqrt{p_j}\,|\psi_j\rangle_{n_Z}\,|j\rangle_K\,|0\rangle_{target}\,|0\rangle + \dots \qquad (3.8)
\end{aligned}$$

where the implicit remaining part is orthogonal to the subspace characterized by state $|0\rangle$ in the auxiliary and target qubits. Thus, the probability of measuring a state in such subspace is

$$\sum_{j \text{ s.t. } |\sin(\alpha_j)|<\mu} c^2 p_j. \qquad (3.9)$$

When f and θ_0 are properly chosen, for some $L_T \in (0, L_M)$, formula (3.9) equals the cumulative distribution function evaluated in L_T, scaled by a factor c^2.

3.3 Experiments

In order to validate the proposed methodology, we conducted an extensive simulation campaign that implements the full algorithm, from initial state preparation through to the estimation of VaR. Our experiments were designed not only to demonstrate the end-to-end performance of the algorithm but also to analyze intermediate outputs, such as the cumulative distribution function for given aggregated loss values. This stepwise validation is crucial for confirming that the quantum circuit behaves as expected and that each component contributes correctly to the overall computation.

3.3.1 Experimental Setup and Input Parameters

The experiments are based on a realistic financial model that incorporates multiple assets, risk factors, and default probabilities. The resulting circuits were executed using quantum simulations performed on exact shot-based simulators [65] available through the Qiskit library [45]. Table 3.1 lists the input parameters used throughout the study. These parameters include the number of qubits allocated per Gaussian distribution (used in the state preparation of risk factors), the truncation bounds for these Gaussians, the number of assets in the portfolio, intrinsic probabilities of default, sensitivities to risk factors, loss given default values, and the correlation matrix linking risk factors. Additional settings, such as the number of shots for quantum circuit measurements and the confidence level for VaR calculations, are also specified. These details are included for reproducibility reasons.

3.3.2 Quantum Circuit and Polynomial Approximation via QSVT

An important component of our approach is the Quantum Singular Value Transformation (QSVT) step. In this phase, we compute the coefficients for the polynomial approximation of a threshold function $\mathcal{T}_\mu$, which plays a key role in the estimation of risk measures. To determine these coefficients and to estimate the associated phases for the operators in the circuit, we employed the QSPPACK library[1] that uses MATLAB. Appendix A.1 contains the code functions used to generate the results shown here. For our experiments, the degree of the polynomial was set to $d = 1000$. Figure 3.3 shows the resulting polynomial approximation, demonstrating that the threshold function is captured with high fidelity across the required domain.

[1] Available at https://qsppack.gitbook.io/

Parameter	Value
# of qubits per Gaussian	2
Gaussian truncation values	± 2
Number of assets	4
Intrinsic probability of default	[0.10, 0.20, 0.05, 0.15]
Sensitivity to risk factors ($\boldsymbol{\rho}$)	[0.10, 0.20, 0.10, 0.80]
Loss given default (LGD)	[50000, 20000, 33000, 12000]
Correlation with risk factors	$\begin{bmatrix} 0.10 & 0.05 \\ 0.25 & 0.10 \\ 0.10 & 0.05 \\ 0.15 & 0.05 \end{bmatrix}$
Number of shots (n_{shots})	2048
Confidence level (α_{var})	0.05

Table 3.1: Input parameters used in the simulation experiments.

3.3.3 Estimation of the Cumulative Distribution Function

Before proceeding to the final risk measure estimation, we analyzed the outputs of the quantum circuit at an intermediate stage. In this phase, the circuit is used to estimate the CDF for a given aggregated loss. This verification step involves comparing the quantum-estimated CDF with a classical Monte Carlo estimation. By sampling from the state prepared by the operator $\mathcal{A}$, we obtain a distribution of loss outcomes. The target value, as determined through classical simulation, serves as a benchmark to assess whether the quantum circuit exhibits the expected probabilistic behavior.

3.3.4 Iterative Quantum Amplitude Estimation and Bisection Search

For the amplitude estimation component, we adopted an iterative variant of Quantum Amplitude Estimation (IQAE). Unlike traditional QAE methods that rely on a counting register for phase estimation, IQAE streamlines the process by directly specifying the precision parameter ϵ and the confidence level α for the final estimate. This variant significantly reduces the quantum resources required, making it better-suited for NISQ devices.

After having obtained a quantum circuit that accurately estimates the CDF, the algorithm is integrated into an iterative bisection search process. The bisection

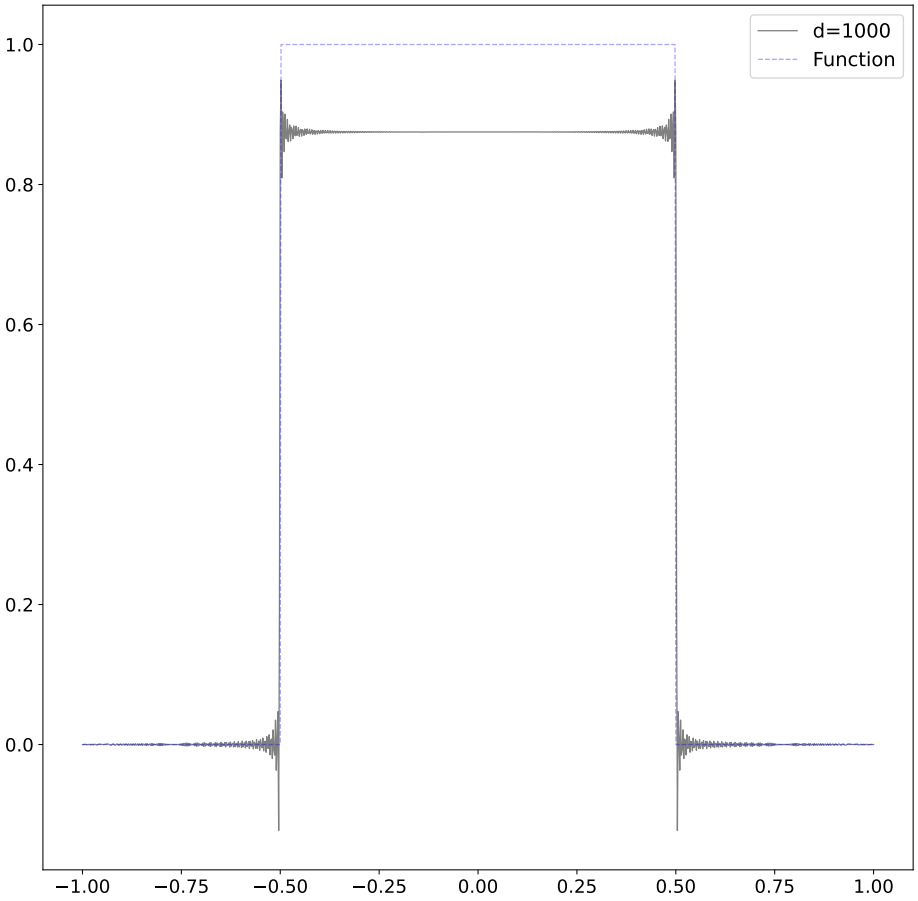


Figure 3.3: Polynomial approximation of the threshold function $\mathcal{T}_\mu$ computed using QSPPACK. The high-degree polynomial ensures that the function is accurately represented for the quantum circuit implementation.

search leverages the quantum-estimated CDF to locate the VaR threshold. Figure 3.4 illustrates the results of the bisection search steps, where the algorithm converges toward the expected VaR value as determined by classical calculations. This convergence validate our quantum approach in its ability to provide reliable risk metrics.

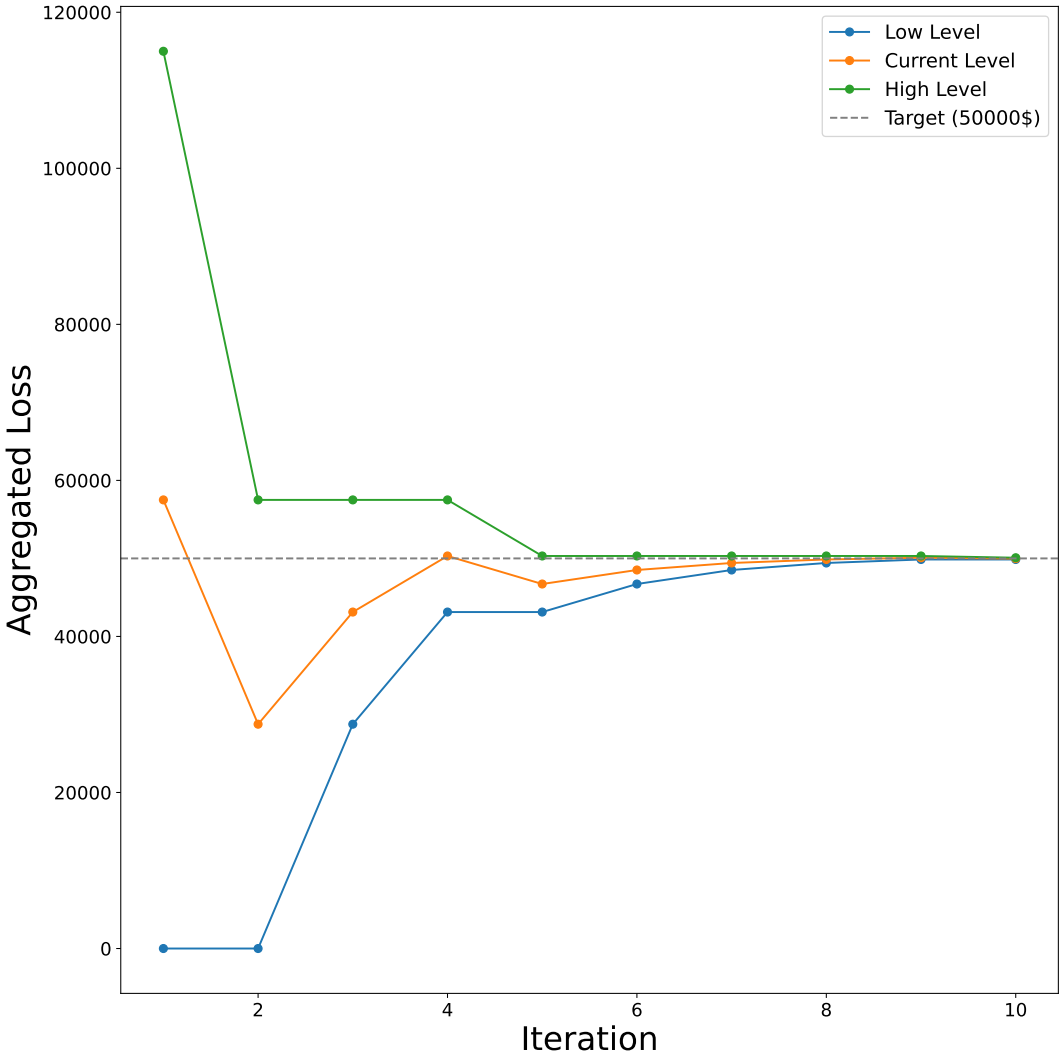


Figure 3.4: Bisection search process for determining the Value-at-Risk. The graph shows how the iterative quantum amplitude estimation converges toward the expected VaR, validated against classical benchmarks.

3.3.5 Discussion of Results

The experimental results confirm that the proposed end-to-end algorithm (from state preparation and QSVT-based polynomial approximation to IQAE and bisection search) successfully computes risk measures such as VaR. The quantum circuit's output for the CDF matches closely with that obtained from classical Monte Carlo simulations, indicating that the circuit behaves as theoretically predicted. Furthermore, the iterative bisection search effectively converges towards

the expected result.

These results show how, for a small-scale example it is indeed possible to obtain a suitable circuit implementation using QSVT. The efficient scaling remains the crucial aspect for further developments. In this context, the proposed implementation should be well positioned to contribute greatly given the fact that it solves the problem of avoiding costly quantum arithmetic while maintaining a linear scaling of the circuit depth with respect to the amount of counterparties.

Comparison with Prior Work

Previous quantum approaches to credit risk modeling without costly quantum arithmetic, were implemented using Qiskit's `LinearAmplitudeFunction` class [26, 27]. Notably, they incur circuit depths that scale exponentially with the number of counterparties involved, i.e., $\mathcal{O}(2^K)$ where K is the number of counterparties. This is due to the need to encode piecewise-linear payoff functions or discretized distributions using controlled operations over an exponentially large Hilbert space.

In contrast, the state preparation technique proposed in this work, based on QSVT and polynomial approximations, reduces the circuit depth to $\mathcal{O}(K)$ (as can be seen in 3.2) while maintaining comparable accuracy. This improvement makes the method significantly more scalable and eliminates the need for ancilla qubit overhead required by prior amplitude encoding strategies [26, 27].

Table 3.2: Comparison of circuit depth scaling with prior approaches

Method	**Depth scaling**
Previous work [26, 27]	$\mathcal{O}(2^K)$
Proposed QSVT-based approach	$\mathcal{O}(K)$

Error Analysis and Limitations

To evaluate the accuracy and reliability of the proposed quantum approach for credit risk estimation, we provide a quantitative error analysis based on the bisection search outputs.

Table 3.3 summarizes the evolution of the CDF estimates, along with the width of the estimation interval Δ = higher bound − lower bound.

Table 3.3: IQAE estimates across bisection search iterations, with corresponding intervals.

Iteration	**Lower Bound**	**CDF Estimate**	**Upper Bound**	**Δ**
1	0.000	0.936	1.000	1.000
2	0.936	0.999	1.000	0.064
3	0.936	0.975	0.999	0.063
4	0.936	0.959	0.975	0.039
5	0.959	0.969	0.975	0.016
6	0.959	0.969	0.975	0.016
7	0.959	0.969	0.975	0.016
8	0.949	0.969	0.975	0.026
9	0.949	0.969	0.984	0.035
10	0.949	0.969	0.984	0.035

As observed, the width Δ of the interval initially decreases with the number of iterations, reflecting convergence of the bisection process. From iteration 5 onward, the estimated amplitude stabilizes at 0.969, matching the classically computed value with negligible error. The maximum possible estimation error is upper bounded by $\Delta/2$ at each step, which at the final iteration corresponds to less than ± 0.018.

These results confirm the robustness of the proposed approach in simulation and validate the correctness of the quantum amplitude estimation process. However, it is worth noting that all experiments were carried out in a noise-free environment. When applied to real quantum hardware, decoherence, gate errors, and limited qubit connectivity might significantly degrade performance. Future work should explore error mitigation techniques and hybrid verification schemes to make the method more resilient in practical, noisy intermediate-scale quantum (NISQ) settings.

Chapter 4

Better Quantum Derivative Pricing

As emerges from Equation 2.22, the correct functioning of the pricing algorithm under our analysis requires the loading of the payoff function f (or, more precisely, its discretized version $\tilde{f}$) within the amplitude of a target qubit. Subsequently, this will make it possible to use the QAE algorithm to estimate $\tilde{f}$ with quadratically fewer samples. In fact, the quantum Monte Carlo [57] aim to replicate the flexibility of classical Monte Carlo simulation for pricing financial derivatives, while offering such improvements in computational complexity. Classical MC methods are widely used for derivative pricing due to their generality and ease of implementation, particularly for high-dimensional or path-dependent payoffs. However, their main limitation lies in their convergence rate: achieving an estimation error ε typically requires $\mathcal{O}(1/\varepsilon^2)$ samples, making them computationally expensive for high-precision pricing tasks.

In contrast, QAE and its derivatives, such as IQAE, achieve a quadratic speedup in convergence, requiring only $\mathcal{O}(1/\varepsilon)$ queries to obtain the same precision. This speedup arises from QAE's ability to exploit quantum superposition and interference: instead of sampling outcomes one-by-one as in classical Monte Carlo, QAE prepares a quantum state whose amplitude encodes the probability of interest and uses quantum phase estimation to extract that amplitude with fewer repetitions. By effectively concentrating information through amplitude amplification and phase resolution, the algorithm avoids the need for repeated sampling and averaging. This results in significantly fewer iterations and sample evaluations, especially relevant in scenarios where complex payoff structures or rare-event modeling would lead to high variance in classical simulations.

It is crucial that this process of amplitude loading is carried out as efficiently as possible, so as not to undermine the potential quadratic advantage offered by QAE. In this regard, in [15], the authors propose two novel schemes for amplitude loading, which we will now illustrate. They use as an example the case of a rainbow options,

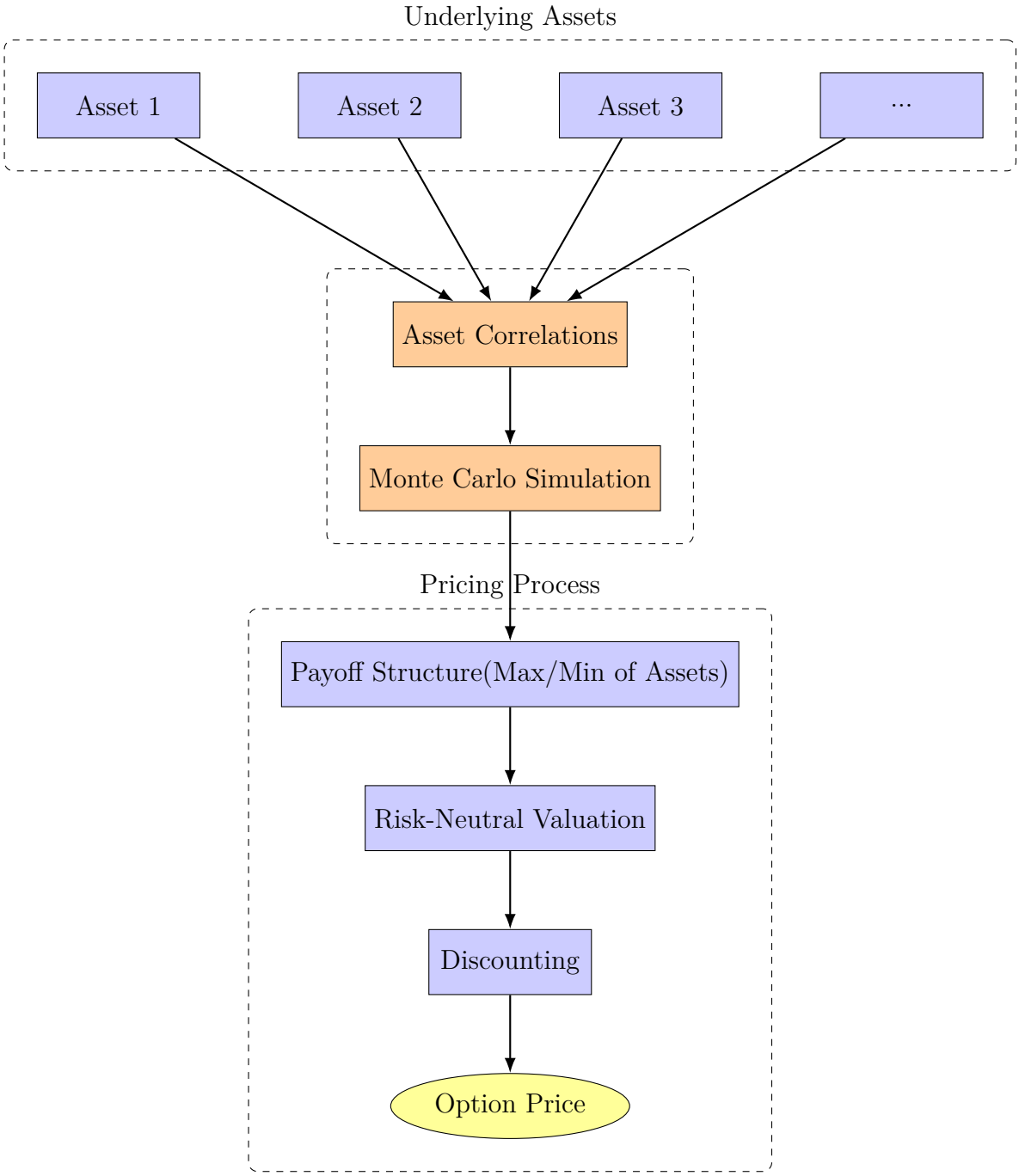


Figure 4.1: Schematic representation of the option pricing workflow. The first step is the definition of the underlying and relative correlations, followed by Monte Carlo simulation. Next, we have the payoff computation, risk-neutral valuation, and discounting to arrive at the final option price.

a path-independent derivative product, although their method could be applied also to more interesting options as we briefly discuss in Section 4.4.

Rainbow options are multi-asset derivatives whose payoff depends on the performance of two or more underlying assets. They typically offer a payout based on the best or worst performer among the assets. A schematic view of the pricing process for such options is given in Figure 4.1. Given the above, in price space and for a given strike price K, one could typically have a payoff in the form:

$$f(\vec{S}^t) = \begin{cases} \max(\vec{S}^t) - K, & \text{if } \max(\vec{S}^t) \geq K \\ 0, & \text{if } \max(\vec{S}^t) < K \end{cases}. \tag{4.1}$$

First, in order to achieve the amplitude loading required to implement this payoff function, both proposed methods need a $z \geq \frac{\log(K)-b'}{b}$ comparator in order to compare the maximum module's output, with $\frac{\log(K)-b'}{b}$. All quantum registers have the same domain and units. Thus, the authors assumed the same register size k for each of them. The Toffoli-depth of a comparator between a quantum register and another quantum register or a constant [25] is $2\log_2(k)+5$. Given the analysis in [48] we assume an unlimited number of qubits, and multiply the Toffoli-depth by a factor of 3 T-depth per Toffoli gate.

$$\Delta^T_{comp}(k) = \Delta^{\text{Toffoli}}_{comp}(k) \cdot 3 = 6\log_2(k) + 15 \tag{4.2}$$

Both methods require a controlled rotation when the condition is not verified. The necessary controlled-RY gate is obtained using two CX gates and two RY gates. Following the Solovay–Kitaev theorem, each RY gate has a T-depth of:

$$\Delta^T_{CRY}(\epsilon_{RY}) = 3\log_2(1/\epsilon_{RY}) \tag{4.3}$$

For details on ϵ_{RY} we refer the reader to [71]. Following these initial general considerations, in the next sections, a detailed analysis of the two methods will be reported.

4.1 Direct Exponential Amplitude Loading

Now, for reasons explained in Section 2.3.2, we would like to move to the return space. There, to identify the asset with the maximum value the authors exploit the fact that, due to the monotonically increasing nature of the exponential function:

$$\arg\max(\bar{S}_0 e^{\bar{R}_t}) = \arg\max(\log(\bar{S}_0) + \bar{R}_t) \tag{4.4}$$

Therefore, we can define

$$Z_t = \max(\log(\bar{S}_0) + \bar{R}_t), \tag{4.5}$$

and, given Equation 4.5, the payoff in return space becomes:

$$f(Z_t) = \begin{cases} e^{Z_t} - K, & \text{if } Z_t \geq \log(K) \\ 0, & \text{if } Z_t < \log(K) \end{cases}, \tag{4.6}$$

that, when taking into account the rescaling detailed in [15], takes the form

$$f(x) = \begin{cases} e^{bz}, & \text{if } z \geq \frac{\log(K)-b'}{b} \\ Ke^{-b'}, & \text{if } z < \frac{\log(K)-b'}{b} \end{cases}. \tag{4.7}$$

Where $b = l_{0,0}\Delta x$ and $b' = \mu_0 + \log(S_0^0) + x_{min}l_{0,0}$. Δx is the Gaussian discretization step while x_{min} denotes the lower Gaussian truncation value. l_{ik} is the i,k entry of the matrix $\mathbf{L} = \mathbf{C}\sqrt{(t-t_0)}$, where $\mathbf{C}$ is the lower triangular matrix given by using the Cholesky decomposition on the historical daily log-returns correlation matrix. Finally, $z = \frac{x}{2^P}$ where P is the amount of fraction places used to represent z.

The direct exponential amplitude loading approach loads:

$$\tilde{f}(x) = \begin{cases} e^{-a\hat{x}}, & \text{if } \frac{x}{2^P} \geq \frac{\log(K)-b'}{b} \\ Ke^{-(b'+ax_{max})}, & \text{if } \frac{x}{2^P} < \frac{\log(K)-b'}{b} \end{cases} \tag{4.8}$$

where $\hat{x}$ is simply the binary complement of x and x_{max} corresponds to the maximum value representable in the $|x\rangle$ register. In order to load $e^{-a\hat{x}}$, a register $|t\rangle$ is initialized to $|0\rangle$ and one controlled rotation is performed for each qubit. These rotations have angles $\theta_i = 2\arccos\left(\sqrt{e^{-a2^i}}\right)$. The probabilities of getting a $|0\rangle$ in the $|t\rangle$ register are then collected by a multi-controlled X gate and stored in the $|1\rangle$ state of an additional objective qubit. An illustration of the resulting circuit is depicted in Figure 4.2.

Demanding payoff infidelity of ϵ_f requires stricter ϵ_{RY} infidelity for each of the controlled-RY gates. Using k controlled-RY gates we incur in a payoff infidelity of $2k\epsilon_{RY}$, and when taking into account the the previous RY gate, the infidelity becomes $\epsilon_f = 2(k+1)\epsilon_{RY}$. Therefore, after these considerations, the depth of the exponential loading component and the controlled-RY part is as follows:

$$\begin{aligned} &\Delta^T_{exp}(\epsilon_f) + \Delta^T_{CRY}(\epsilon_f) \\ &= 2 \cdot \Delta^T_{CRY}(\epsilon_f) = 12\log_2\left(\frac{2(k+1)}{\epsilon_f}\right) \end{aligned} \tag{4.9}$$

The loading component is then followed by a multi-controlled X gate, that has $k+1$ controls. It is possible to obtain a reduced T-depth of the MCX using relative Toffoli gates [53]. In [15], the Toffoli gate depth optimization is obtained thanks to the Classiq platform synthesis engine [59], resulting in a T-depth of $14\log_3((k+1)/2) + 5$. Thus,

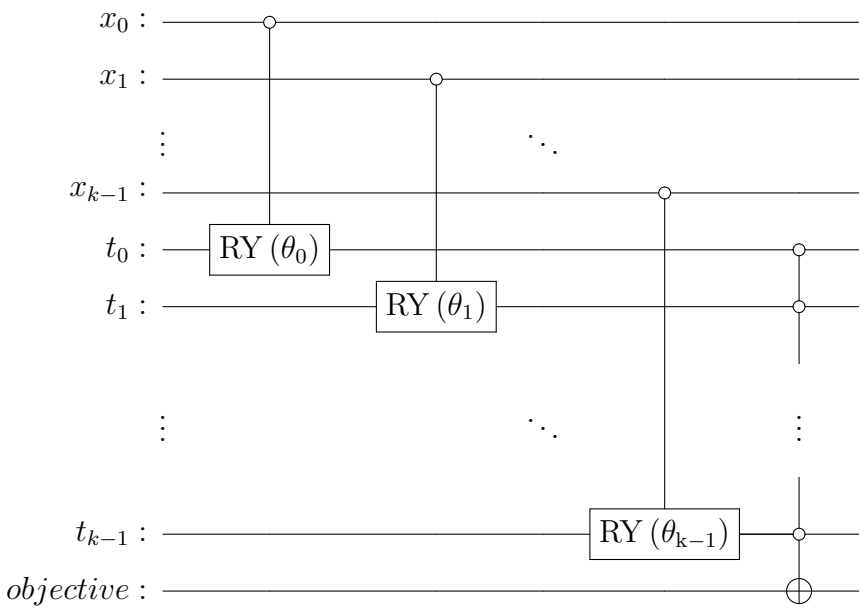


Figure 4.2: Circuit implementing the direct exponential amplitude loading.

$$\Delta^T_{MCX}(k) = 14 \log_3 \left(\frac{k+1}{2} \right) + 5. \tag{4.10}$$

All considered, for the direct method the final T-depth of the payoff function is as such:

$$\begin{aligned} \Delta^T_{al} &= \Delta^T_{comp} + \Delta^T_{exp} + \Delta^T_{MCX} + \Delta^T_{CRY} \\ &= 6 \log_2(k) + 14 \log_3 \left(\frac{k+1}{2} \right) + 12 \log_2 \left(\frac{2(k+1)}{\epsilon_f} \right) + 20. \end{aligned} \tag{4.11}$$

4.2 Integration Amplitude Loading

The authors in [15] introduce a second method, integration amplitude loading, inspired by the work in [46]. The idea relies on using the outcome of a comparator as an integrator to generate a modified version of a function h, $\tilde{h}$. The comparator is tasked with collecting the probabilities $g(r)$ of $|r\rangle$ state until $|r\rangle$ register is lower

than $|x\rangle$:

$$\begin{aligned}&\sum_{r=0}^{2^R-1}\sqrt{g(r)}|x\rangle|r\rangle|r\leq x\rangle\\&=|x\rangle\otimes\left[\sum_{r=0}^{x}\sqrt{g(r)}|r\rangle|1\rangle+\sum_{r=x}^{2^R-1}\sqrt{g(r)}|r\rangle|0\rangle\right]\end{aligned}\tag{4.12}$$

Restricting to the probability of having $r \leq x$ we can define the function:

$$\tilde{h}(x)=\sum_{r=0}^{x}g(r)\tag{4.13}$$

Evaluating the probability to get a $|1\rangle$ results in $\sum_{x=0}^{2^R-1}\tilde{h}(x)$. To obtain a given function $\tilde{h}$ a proper function $g(r)$ should be chosen. The $g(r)$ for $r=0$ value must therefore be $g(0)=\tilde{h}(0)$ and for all the other r:

$$g(r)=\tilde{h}(r)-\tilde{h}(r-1)$$

To load faster the exponential function $h(x)=e^{ax}$ we need to consider that

$$\tilde{h}(x)=\frac{h(x+1)-h(0)}{h(x_{max}+1)-h(0)}=\frac{e^{a(x+1)}-1}{e^{a(x_{max}+1)}-1}\tag{4.14}$$

and thus

$$g(r)=\frac{e^{ar}}{\sum_{j=0}^{x_{max}}e^{aj}}.\tag{4.15}$$

These values are loaded using an exponential state preparation circuit [17] of unitary depth that does not require controlled operations. The circuit is shown in Figure 4.3 where $\alpha_i=2\arctan\left(e^{\frac{a2^i}{2}}\right)$. We therefore consider the following:

$$\tilde{f}(x)=\begin{cases}\frac{e^{a(x+1)}-1}{e^{a(x_{max}+1)}-1}, & \text{if } \frac{x}{2^P}\geq\frac{\log(K)-b'}{b}\\ \frac{Ke^{-b'}}{c}-\frac{e^{-a}}{c}, & \text{if } \frac{x}{2^P}<\frac{\log(K)-b'}{b}\end{cases}\tag{4.16}$$

with $c=\frac{e^{a(x_{max}+1)}-1}{e^a}$. Note that it will be necessary to adjust the postprocessing accordingly [15].

Here, an integrator follows after the comparator and the controlled rotation. This integrator is a controlled register-to-register comparator and therefore implemented by an uncontrolled subtractor, its uncomputation, and a controlled sign check (a CCX gate). Thus we obtain a T-depth:

$$\Delta^T_{integ}(k)=\left(\Delta^{\text{Toffoli}}_{comp}(k)+1\right)\cdot 3=6\log_2(k)+18.\tag{4.17}$$

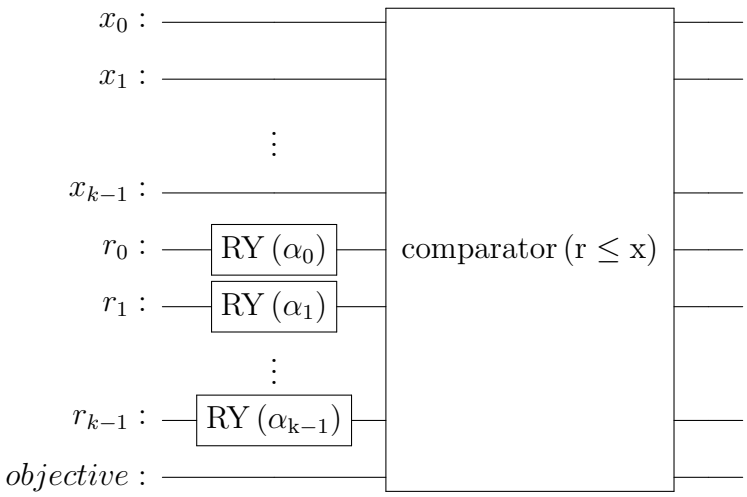


Figure 4.3: Circuit implementing the integration amplitude loading.

One can avoid counting the exponential state preparation since it is parallel to the state preparations of the assets and their *max* calculation. In conclusion, the T-depth would be

$$\begin{aligned} \Delta_{al}^{T} &= \Delta_{comp}^{T} + \Delta_{integ}^{T} + \Delta_{CRY}^{T} \\ &= 12\log_2(k) + 6\log_2\left(\frac{2}{\epsilon_f}\right) + 33. \end{aligned} \tag{4.18}$$

4.3 Experiments

To test the two model proposed in Section 4.1 and 4.2, a practical implementation of the rainbow option pricing algorithm was realised using the Classiq Platform [17, 35] and the Qmod language (See Section 1.5.2). The platform adds a layer of abstraction since it gives the possibility of describing the algorithm implemented by the user using high-level blocks. The various components of the full algorithm were designed through this high-level model. This process optimizes many of the low-level aspect of the quantum circuit design. As an example, it automatically chooses when to uncompute expressions to free auxiliary qubits that can then be re-used. This compilation approach aims at improving the scaling of the problem size with a given resource budget in terms of available qubit.

Table 4.1: Assets' initial price.

Asset	S_0
1	\$ 156.45
2	\$ 151.22

Table 4.2: Problem's parameters.

Asset	μ	Cov
1	$3.583 \cdot 10^{-4}$	$[4.75, 1.52] \cdot 10^{-4}$
2	$5.811 \cdot 10^{-4}$	$[1.52, 2.61] \cdot 10^{-4}$

4.3.1 Experimental Setup and Input Parameters

The simulator chosen for the experiments was again the `QasmSimulator` [65] that supports multiple simulation methods and configurable options. The chosen method was "*statevector*": a dense statevector simulation that can sample measurement outcomes from ideal circuits with all measurements at end of the circuit [65]. The simulator is available through the Qiskit library [45]. The simulations were performed using IQAE [37] with a precision $\epsilon = 0.01$ and (for the confidence interval) $\alpha = 0.05$.

For each experiment, the quantum circuit was executed 1000 times and measurements of the objective qubit were collected after the IQAE subroutine [37]. As a proof of concept, the algorithm was tested with a toy-model configuration of two assets whose value at the initial time $t = 0$ can be seen in Table 4.1 and a strike price $K = \$155$. The corresponding log-returns mean and covariance can be found in Table 4.2. Finally, we set a time to expiration of 250 days (maturity date - starting date) for the option contract.

In order to load each discretized Gaussian distribution, we used two qubits. In terms of precision, the arithmetic was rounded to one decimal place. An additional qubit was required for the comparator. Finally, one target qubit was used to encode the payoff for the Amplitude Estimation. The Classiq platform took care of assigning additional ancilla qubits, resulting in a quantum circuit requiring 24 qubits for our toy-model experiment.

4.3.2 Results

In order to have a validation of the proposed algorithm, we used the classically computed expected payoff value as a benchmark to make a comparison with the quantum method. To have a fair comparison, the classical estimation process uses the same discretization of the Gaussian distributions and the same arithmetic precision as the quantum process. Using this methodology and for the given example, the classically computed expected payoff is thus \$20.1970.

For what concerns the results of the quantum algorithm, the reader can refer to Figure 4.4. As can be observed, the expected result is in both cases well within the confidence interval of the simulation. This can be interpreted as a validation of the quantum approach that was able to successfully estimate the expected payoff for a rainbow option contract with the characteristics listed above.

Error Analysis and Limitations

In order to assess the accuracy and reliability of the proposed quantum pricing methods, we report a quantitative error analysis for both amplitude loading strategies discussed in this chapter. The estimated option prices and corresponding confidence intervals are summarized in Table 4.3.

Table 4.3: Estimated option price and confidence intervals for the two amplitude loading methods.

Method	**Estimate**	**Confidence Interval**	**Δ**
Direct Exponential Loading	19.96	[19.68, 20.24]	0.56
Integration Loading	20.13	[19.33, 20.93]	1.60

The direct exponential loading method yields an option value estimate of \$19.96 with a narrow confidence interval of 0.56, while the integration-based approach results in a slightly higher estimate (\$20.13) but with a broader uncertainty (interval width of 1.60). Both methods are consistent with the classically computed benchmark value of \$20.20, and the absolute errors fall within the respective confidence bounds.

These findings demonstrate that the quantum Monte Carlo pricing methods are capable of producing accurate results. The lower variance of the direct method suggests it may be more suitable for practical implementations. However, the integration method offers greater flexibility and can be generalized to more complex payoff structures.

Again, it is important to note that these results are obtained in an idealized, noise-free environment using a quantum simulator. In real-world applications, the fidelity of quantum circuits would be affected by hardware noise, qubit decoherence, and gate imperfections.

4.4 Extension to Path-Dependent Options

In the context of a path-dependent derivative product, for the j-th underlying asset of our contract, we can denote the price at time t' as

$$S_j^{t'} = S_j^0 \prod_{t=1}^{t'} e^{R_j^t}, \tag{4.19}$$

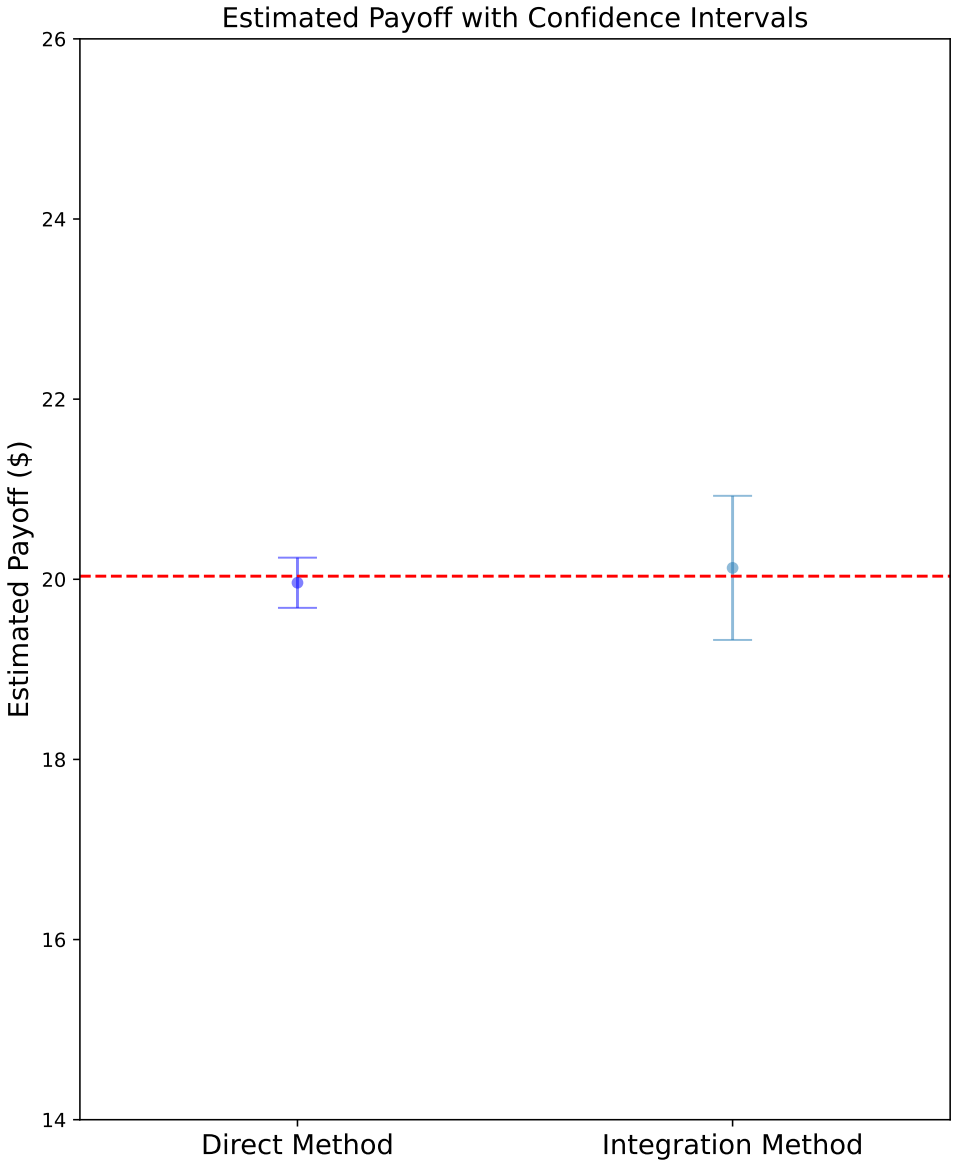


Figure 4.4: Estimated payoff (with relative confidence intervals) using the two proposed methodologies for amplitude loading and the IBM QASM simulator. The dotted line represents the classically computed expected value (20.20$).

where S_j^0 is the price of the j-th asset at time 0 and R_j^t is the return of the j-th asset at time t. As detailed in Section 2.3.2, when asset prices follow a log-normal distribution, it is advantageous to transition to return space since the log-returns are normally distributed [14].

The probability of a given sequence of returns will therefore be associated with a multivariate normal distribution and is given by:

$$P(\vec{R}^t) = \frac{\exp(-\frac{1}{2}(R^t - \vec{\mu})^T \Sigma^{-1}(R^t - \vec{\mu}))}{2\pi^{d/2}\det(\Sigma)^{1/2}}, \tag{4.20}$$

where d corresponds to the number of underlying assets, Σ is the covariance matrix of the d underlying assets, and the d elements of $\vec{\mu}$ are given by

$$\mu_j = (r - \frac{1}{2}\sigma_j^2)\Delta t,$$

where r is the risk-free rate and σ_j is the volatility of the j-th asset. Finally, Δt is the time duration between steps of the stochastic process.

In general, loading normal distributions is less complex, since the underlying stochastic evolution is independent of the price at previous time steps. This is important because, as pointed out in [14], the loading of general distributions is exponentially hard, although alternative methods for loading the path distribution have been proposed. A good example is using a qGAN [88], as originally proposed in [75]. However, it is not clear how big the overhead from the associated training process will be in practice.

Thus, transitioning to return space leaves us with several distributions that can be loaded by preparing, in parallel, many standard normals. Once this is done, the next step would be to apply affine transformations to obtain the right means and standard deviations.

This is precisely what is prescribed in [14] to obtain a method that avoids normalization issues and reduces the computational requirements for path-dependent options. The approach can be used to extend the results presented in this chapter regarding rainbow options to more complex classes of options such as autocallables.

Chapter 5

Conclusions and Outlook

The integration of quantum algorithms into financial applications represents a rapidly evolving research frontier, offering the potential to address computational challenges that remain intractable for classical methodologies. This thesis has explored the applicability of quantum computing techniques across fundamental domains of quantitative finance such as portfolio optimization, credit risk analysis, and derivative pricing. Each of these areas presents unique computational difficulties that stem from the complexity of their underlying mathematical models, the combinatorial nature of decision variables, and the need for efficient risk assessment under uncertainty.

The field of portfolio optimization, which traditionally relies on convex optimization and heuristic approaches, can benefit from quantum-inspired and quantum-native methods, particularly through QUBO formulations. Such representations allow the deployment of algorithms like the QAOA and VQE, which have demonstrated promising empirical results in solving combinatorial problems.

In credit risk analysis, the estimation of default probabilities and portfolio-level risk exposure entails the computation of high-dimensional integrals and complex correlation structures among financial entities. Traditional Monte Carlo simulations, though effective, suffer from high computational costs, particularly in stress-testing scenarios. Quantum Monte Carlo (QMC) methods, leveraging QAE and its quadratic quantum speedup, present a viable alternative that could outperform classical techniques, reducing the sample complexity from $O(1/\varepsilon^2)$ to $O(1/\varepsilon)$. Nevertheless, practical implementations remain limited by quantum hardware constraints and state preparation complexity, requiring further exploration and optimization of current approaches.

In the context of derivative pricing, the computational burden stems from the need to model stochastic processes governing asset price dynamics and to evaluate complex payoff structures under risk-neutral measures. The quantum computing paradigm offers avenues for improvement, although again, further improvements are needed to efficiently model complex derivatives and achieve practical usefulness.

In this document, we elucidated the fundamental components that make quantum computing a captivating instrument for the creation of innovative algorithms, thereby transitioning the paradigm from conventional computation to a framework grounded in the peculiar and intriguing principles of quantum mechanics. Subsequently, this thesis presented the theoretical and practical implications of quantum computing for financial applications, several challenges remain. We also briefly discussed how hardware limitations, noise, and algorithm scalability continue to be significant bottlenecks that hinder the widespread adoption of quantum algorithms in industry. In order to mitigate these limitations, the final chapters were dedicated to illustrating proposed methods to enhance the efficiency of the state preparation operator design, which is crucial for the successful large-scale implementation of QMC approaches. We focused on avoiding costly quantum arithmetic by employing tools such as QSVT and amplitude loading. We also presented experimental results validating and confirming the proposed approaches. Although not definitively resolutive, these improvements contribute to advancing towards the practical utility of the analyzed algorithms.

It is important to emphasize that the advantage of quantum algorithms such as QAE is expressed in terms of *query complexity*, that is the number of times an oracle (or payoff evaluation circuit) must be called to estimate an expected value within a desired precision. This is fundamentally different from runtime comparisons on physical hardware, which depend heavily on implementation details, noise, and compiler optimizations. Therefore, a direct wall-clock or resource-based comparison with classical Monte Carlo is not meaningful at this stage. Instead, the quadratic speedup refers to the scaling behavior: whereas classical methods require $\mathcal{O}(1/\varepsilon^2)$ oracle queries to achieve an error ε, quantum algorithms require only $\mathcal{O}(1/\varepsilon)$. This asymptotic improvement highlights the potential algorithmic advantage intrinsic to the quantum approach.

Still, future developments of this work could investigate conducting tests on real hardware, assessing the potential of various implementation technologies, beginning with the one currently considered among the most advanced: superconducting qubits. In this regard, the recent installation of an IQM Spark 5-qubit machine at the Polytechnic University of Turin certainly presents an opportunity to conduct preliminary tests that could provide valuable insights into the behavior of the algorithms proposed here beyond the simulated context. A comparative analysis could then be performed by accessing other quantum infrastructures via the cloud.

Looking at the bigger picture, the outlook for quantum finance depends on continued advancements in both quantum hardware and algorithmic development. The emergence of error-corrected quantum computers and improvements in quantum annealing, variational algorithms, and fault-tolerant gate-based approaches could drive meaningful breakthroughs in financial applications. Additionally, the integration of hybrid quantum-classical models may offer practical pathways for

near-term adoption, leveraging quantum speedup while mitigating current hardware limitations.

Appendix A

Source Code and Scripts

A.1 QSVT for Credit Risk

A.1.1 MATLAB Code for Computing the Polynomial Approximation

```
function [coef_full] = cvx_poly_coef(func, deg, opts)
%% Find a polynomial approximation for a given function.
% ------------------------------------------------------
%
% Input:
%       func, deg, opts
% Output:
%       c --- Chebyshev coefficients of best approximation
   polynomial
%
% ------------------------------------------------------
% Author:           Lin Lin      update 06/2022
%                   Jiasu Wang   update 08/2022
%
% ------------------------------------------------------


if( ~isfield(opts, 'npts') )
  opts.npts = 200;
end
if( ~isfield(opts, 'epsil') )
  opts.epsil = 0.01;
end
if( ~isfield(opts, 'fscale') )
  opts.fscale = 1-opts.epsil;
end
if( ~isfield(opts, 'intervals') )
  opts.intervals = [0,1];
```

```
28 end
29 if( ~isfield(opts, 'isplot') )
30   opts.isplot = false;
31 end
32 if( ~isfield(opts, 'objnorm') )
33   opts.objnorm = inf;
34 end
35
36 % check variables and assign local variables
37 assert(mod(length(opts.intervals), 2) == 0);
38 parity = mod(deg, 2);
39 epsil = opts.epsil;
40 npts = opts.npts;
41
42 [xpts, ~] = chebpts(2*npts);
43 xpts=union(xpts,opts.intervals');
44 xpts = xpts(xpts>=0);
45 npts = length(xpts);
46
47 n_interval = length(opts.intervals) / 2;
48 ind_union = [];
49 for i = 1 : n_interval
50   ind_set{i} = find((xpts >= opts.intervals(2*i-1)) &...
51     (xpts<=opts.intervals(2*i)));
52
53   ind_union = union(ind_union, ind_set{i});
54 end
55
56
57 % evaluate the target function
58 fx = zeros(npts,1);
59 fx(ind_union) = opts.fscale * func(xpts(ind_union));
60
61
62 % prepare the Chebyshev polynomials
63 if( parity == 0 )
64   n_coef = deg / 2 + 1;
65 else
66   n_coef = (deg + 1) / 2;
67 end
68
69 Ax = zeros(npts, n_coef);
70 for k = 1 : n_coef
71   if( parity == 0 )
72     Tcheb = chebpoly(2*(k-1));
73   else
74     Tcheb = chebpoly(2*k-1);
75   end
76   Ax(:,k) = Tcheb(xpts);
77 end
```

```
78
79
80 %% Use CVX to optimize the Chebyshev coefficients
81 cvx_begin quiet
82 variable coef(n_coef)
83 variable y(npts)
84 minimize ( norm(y(ind_union)-fx(ind_union), opts.objnorm))
85
86 y==Ax*coef
87 y>=-(1-epsil)
88 y<=(1-epsil)
89 cvx_end
90
91 err_inf = norm(y(ind_union)-fx(ind_union), opts.objnorm);
92 fprintf('norm error = %g\n', err_inf);
93
94
95 % use chebfun to make sure the maximum is less than 1
96 coef_full = zeros(deg+1,1);
97 if( parity == 0 )
98   coef_full(1:2:end) = coef;
99 else
100   coef_full(2:2:end) = coef;
101 end
102 sol_cheb = chebfun(coef_full, 'coeffs');
103 max_sol = max(abs(sol_cheb));
104 fprintf('max of solution = %g\n', max_sol);
105 if( max_sol > 1.0-1e-10 )
106   error('Solution is not bounded by 1. Increase npts');
107 end
108
109 if( opts.isplot )
110   figure(1)
111   clf
112   hold on
113   plot(xpts, y, 'ro', 'LineWidth',1.5);
114   for i = 1 : n_interval
115     plot(xpts(ind_set{i}), y(ind_set{i}),'b-','LineWidth',2)
116   end
117   hold off
118   xlabel('$x$', 'Interpreter', 'latex', 'FontSize', 15);
119   ylabel('$f(x)$', 'Interpreter', 'latex', 'FontSize', 15);
120   legend({'polynomial','target'},'FontSize', 15);
121
122
123   figure(2)
124   clf
125   hold on
126   for i = 1 : n_interval
```

```
127     plot(xpts(ind_set{i}), abs(y(ind_set{i})-fx(ind_set{i})), 'k-'
        ,...
128         'LineWidth',1.5)
129   end
130   hold off
131   xlabel('$x$', 'Interpreter', 'latex','FontSize', 15);
132   ylabel('$|f_\mathrm{poly}(x)-f(x)|$', 'Interpreter', 'latex','
        FontSize', 15);
133 end
134
135
136 end
```

Listing A.1: MATLAB function that computes the polynomial approximation. Source: [24, 66].

A.1.2 MATLAB Code for Computing the Phases

```
1  function [phi_proc,out] = QSP_solver(coef,parity,opts)
2  %-------------------------------------------------------------------
3  % Given coefficients of a polynomial P, yield corresponding phase
       factors
4  %
5  %
6  % To simplify, a constant pi/4 is added to both sides of
7  % the phase factors when evaluating the objective and the gradient.
8  % In the output, the FULL phase factors with pi/4 are given.
9  %
10 % Input:
11 %   coef --- Coefficients of polynomial P under Chevyshev basis, P
12 %     should be even/odd, only provide non-zero coefficients.
13 %     Coefficients should be ranked from low order term to high
14 %     order term.
15 %   parity --- Parity of polynomial P (0 -- even, 1 -- odd)
16 %   opts --- Options structure with fields
17 %                criteria: stop criteria
18 %                useReal: use only real arithmetics if true
19 %                targetPre: want Pre to be target function if true
20 %                method: choose from 'LBFGS', 'FPI', or 'Newton'
21 %                typePhi: full or reduced phase factors
22 %
23 % Output:
24 %   phi_proc --- Solution of optimization problem, FULL phase
       factors
25 %   out --- Information of solving process
26 %
27 %-------------------------------------------------------------------%
28 % Reference: Yulong Dong, Xiang  Meng, K.Birgitta Whaley and Lin
       Lin
```

```
%            Efficient Phase Factor Evaluation in Quantum Signal Processing
%
% Author:          Xiang Meng, Yulong Dong    update 06/2020
%                  Jiasu Wang                update 07/2022
%
%-------------------------------------------------------------------
% setup options for L-BFGS solver
if ~isfield(opts,'maxiter');             opts.maxiter = 5e4; end
if ~isfield(opts,'criteria');            opts.criteria = 1e-12; end
if ~isfield(opts,'useReal');             opts.useReal = true; end
if ~isfield(opts,'targetPre');           opts.targetPre = true; end
if ~isfield(opts,'method');              opts.method = 'FPI'; end
if ~isfield(opts,'typePhi');             opts.typePhi = 'full'; end


if strcmp(opts.method,'LBFGS')

    %---------------------------------------------------------------
    % initial preparation

    tot_len = length(coef);
    delta = cos((1:2:(2*tot_len-1))*(pi/2/(2*tot_len)))';
    if (opts.targetPre == false)
        opts.target = @(x, opts) -ChebyCoef2Func(x, coef, parity, true);
    else
        opts.target = @(x, opts) ChebyCoef2Func(x, coef, parity, true);
    end
    opts.parity = parity;
    obj = @QSPObj_sym;
    if (opts.useReal == true)
        grad = @QSPGrad_sym_real;
    else
        grad = @QSPGrad_sym;
    end

    %---------------------------------------------------------------
    % solve by L-BFGS with selected initial point

    tic;
    [phi,err,iter] = QSP_LBFGS(obj,grad,delta,zeros(tot_len,1),opts);
    % convert phi to reduced phase factors
    if( parity == 0 )
        phi(1) = phi(1)/2;
    end
    runtime = toc;

```

```
elseif strcmp(opts.method,'FPI')
    [phi, err, iter, runtime] = QSP_CM(coef, parity, opts);

elseif strcmp(opts.method, 'Newton')
    [phi, err, iter, runtime] = QSP_Newton(coef, parity, opts);
else
    fprintf("Assigned method doesn't exist. Please choose method
   from 'LBFGS', 'FPI' or 'Newton'.\n");
end

%-------------------------------------------------------------------
% output information

out.iter = iter;
out.time = runtime;
out.value = err;
out.parity = parity;
out.targetPre = opts.targetPre;

if strcmp(opts.typePhi,'full')
    phi_proc = rdc_phase_factor_to_full(phi, parity, opts.targetPre
   );
    out.typePhi = 'full';
else
    phi_proc = phi;
    out.typePhi = 'reduced';
end

end
```

Listing A.2: MATLAB function that computes the phases for QSVT starting from Chebyshev coefficients. Source: [24, 66].

A.1.3 Python Code to Generate the State Preparation Circuit using QSVT

```
class AmplitudeLoadingVar(QuantumCircuit):
    """
    Create the circuit that given the state |j>|0> returns
        |j>(cos(alpha_j)|0> + sin(alpha_j)|1>)
    where alpha_j is an item of a given list (scaled_values).

    Args:
        num_state_qubits: number of qubits corresponding to the
   encoding of states j's
        scaled_values: trigonometric functions argument in target
   states (scaled_values=[alpha_0, alpha_1, ... ,alpha_{2^
   num_state_qubits -1}])
        name: given name of the circuit
```

```
    """

    def __init__(self, num_state_qubits: int, scaled_values: List[float], starting_offset:float = 0, name: str = "f(x)") -> None:
        """
        Args:
            num_state_qubits: the number of state qubits.
            scaled_values: the list of the values in the range [-pi/2, pi/2]. Its size must be 2^num_state_qubits.
            name: the name of the circuit
        """
        self.num_state_qubits = num_state_qubits
        self.scaled_values = scaled_values
        self.name = name

        if (min(scaled_values) < -np.pi) or (max(scaled_values) > np.pi):
            raise ValueError("The range must be contained in [-pi, pi].")

        if num_state_qubits != len(scaled_values):
            raise ValueError("The number of values in input does not match the power of 2 of the number of qubits that encode the state.")

        alpha_values = [s*2 for s in scaled_values]
        # print(f'alphas: {alpha_values}')

        ctrl_q = QuantumRegister(num_state_qubits, name="State")
        target_q = QuantumRegister(1, name="Target")

        super().__init__(ctrl_q, target_q, name=self.name)

        self.append(MultiCRGateVar(list(alpha_values), ctrl_q[:] + target_q[:], 'Y', 2*starting_offset), ctrl_q[:] + target_q[:])
        self.x(target_q)


def MultiCRGateVar(alpha_values: list, q_reg : QuantumRegister, axis='Y', starting_offset:float = 0) :
    circuit = QuantumCircuit(q_reg)
    q_target=q_reg[-1]
    if axis=='Y':
        rgate = RYGate
        crgate = CRYGate
    else:
        rgate = RXGate
        crgate = CRXGate

    if starting_offset != 0:
```

```
53          circuit.append(rgate(starting_offset, label="start_offset_"
    +axis), [q_target])
54      # TODO: controllare che il numero degli angoli e dei qubit
    siano coerenti
55      for i in range(0,len(q_reg)-1) :
56          c_gate = crgate(alpha_values[i], label="rot_"+axis+"_"+str(
    i))
57          circuit.append(c_gate, [q_reg[i],q_target])
58      return circuit.to_gate()
59
60
61  def get_expected_probability_circuit(K: int, uncertainity_model:
    QuantumCircuit, lgd: list, target_loss: float, phases=None,
    poly:list=None, threshold=0.5, enable_switch=True, epsilon=0.6)
     -> (QuantumCircuit, QuantumCircuit):
62
63      offsets = lgd
64
65      maximum = sum(offsets)
66      target_loss_scaled = target_loss/maximum
67      minimum = target_loss_scaled
68      arc_threshold = np.arcsin(threshold)
69      maximum_angle = np.pi/2
70      minimum_angle = 0
71      maximum_range = None
72      minimum_range = None
73
74      if target_loss_scaled > (arc_threshold - minimum_angle)/(
    maximum_angle - minimum_angle):
75          minimum_range = minimum_angle
76          unitary_gap = (arc_threshold - minimum_range) /
    target_loss_scaled
77      else:
78          maximum_range = maximum_angle
79          unitary_gap = (maximum_range - arc_threshold) / (1-
    target_loss_scaled)
80
81      transform_losses_to_angles = lambda loss: unitary_gap*(loss/
    maximum - target_loss_scaled) + arc_threshold
82      scaled_offsets = [transform_losses_to_angles(off) -
    transform_losses_to_angles(0) for off in offsets]
83
84
85      obj_e_loss = AmplitudeLoadingVar(K, scaled_offsets,
    starting_offset = transform_losses_to_angles(0))
86
87      qsp = QSP(
88          obj_e_loss,
89          [obj_e_loss.num_qubits - 1], [obj_e_loss.num_qubits - 1],
90          poly=poly,
```

```
        phases=phases,
        adjust_conventions=True,
        ctrl_zero_qubits1=[0],
        ctrl_zero_qubits2=[0]
    )

    state_preparation = QuantumCircuit(
        QuantumRegister(uncertainity_model.num_qubits - K, 'z'),
        *qsp.qregs,
        name="state_preparation"
    )
    state_preparation.append(uncertainity_model, state_preparation.
    qubits[0:len(state_preparation.qubits)-2])
    state_preparation.append(qsp, qsp.qubits[:])

    return state_preparation, obj_e_loss
```

Listing A.3: Python functions that create the circuit for the state preparation operator.

Nomenclature

Acronyms / Abbreviations

CDF Cumulative Distribution Function

CNOT Controlled-Not

DSL Domain-Specific Language

EC Economic Capital

EDA Electronic Design Automation

EL Expected Loss

EPR Einstein, Podolsky e Rosen

ISA Instruction Set Architecture

LGD Loss Given Default

NISQ Noisy Intermediate Scale Quantum

PD Probability of Default

PDE Partial Differential Equation

QAA Quantum Amplitude Amplification

QAE Quantum Amplitude Estimation

QAOA Quantum Approximate Optimization Algorithm

QBLAS Quantum Basic Linear Algebra Subroutine

QEC Quantum Error Correction

QFT Quantum Fourier Transform

QMC Quantum Monte Carlo

QPE Quantum Phase Estimation

QSP Quantum Signal Processing

QSVT Quantum Singular Value Transformation

QUBO Quadratic Unconstrained Binary Optimization

SDE Stochastic Differential Equation

SVD Singular Value Decomposition

VaR Value at Risk

VQE Variational Quantum Eigensolver

Bibliography

[1] Israel F. Araujo et al. "A divide-and-conquer algorithm for quantum state preparation". In: *Scientific Reports* 11.1 (Mar. 2021). ISSN: 2045-2322. DOI: 10.1038/s41598-021-85474-1. URL: http://dx.doi.org/10.1038/s41598-021-85474-1.

[2] Alain Aspect, Jean Dalibard, and Gérard Roger. "Experimental Test of Bell's Inequalities Using Time- Varying Analyzers". In: *Physical Review Letters* 49.25 (Dec. 1982), pp. 1804–1807. ISSN: 0031-9007. DOI: 10.1103/physrevlett.49.1804. URL: http://dx.doi.org/10.1103/PhysRevLett.49.1804.

[3] Adriano Barenco et al. "Elementary gates for quantum computation". In: *Physical Review A* 52.5 (Nov. 1995), pp. 3457–3467. ISSN: 1094-1622. DOI: 10.1103/physreva.52.3457. URL: http://dx.doi.org/10.1103/PhysRevA.52.3457.

[4] Johannes Bausch. "Fast Black-Box Quantum State Preparation". In: *Quantum* 6 (Aug. 2022), p. 773. ISSN: 2521-327X. DOI: 10.22331/q-2022-08-04-773. URL: http://dx.doi.org/10.22331/q-2022-08-04-773.

[5] J. S. Bell. "On the Einstein Podolsky Rosen paradox". In: *Physics* 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.

[6] Ville Bergholm et al. *PennyLane: Automatic differentiation of hybrid quantum-classical computations*. 2018. DOI: 10.48550/ARXIV.1811.04968. URL: https://arxiv.org/abs/1811.04968.

[7] Daniel Bienstock. "Computational study of a family of mixed-integer quadratic programming problems". In: *Mathematical Programming* 74.2 (Aug. 1996), pp. 121–140. ISSN: 1436-4646. DOI: 10.1007/bf02592208. URL: http://dx.doi.org/10.1007/BF02592208.

[8] Fischer Black and Myron S. Scholes. "The Pricing of Options and Corporate Liabilities". In: *Journal of Political Economy* 81 (1973), pp. 637–654. URL: https://api.semanticscholar.org/CorpusID:154552078.

[9] Rainer Blatt and David Wineland. "Entangled states of trapped atomic ions". In: *Nature* 453.7198 (June 2008), pp. 1008–1015. ISSN: 1476-4687. DOI: 10.1038/nature07125. URL: http://dx.doi.org/10.1038/nature07125.

[10] Kostas Blekos et al. "A review on Quantum Approximate Optimization Algorithm and its variants". In: *Physics Reports* 1068 (June 2024), pp. 1–66. ISSN: 0370-1573. DOI: 10.1016/j.physrep.2024.03.002. URL: http://dx.doi.org/10.1016/j.physrep.2024.03.002.

[11] Max Born. "On the Quantum Mechanics of Collision Processes [English Translation]". In: *Terry's Archive Online* 2021.08 (Sept. 2021), p. 0828. ISSN: 2694-4596. DOI: 10.48034/20210828. URL: http://dx.doi.org/10.48034/20210828.

[12] Gilles Brassard et al. "Quantum amplitude amplification and estimation". In: *Quantum Computation and Information* (2002), pp. 53–74. ISSN: 0271-4132. DOI: 10.1090/conm/305/05215.

[13] M. Cerezo et al. "Variational quantum algorithms". In: *Nature Reviews Physics* 3.9 (Aug. 2021), pp. 625–644. ISSN: 2522-5820. DOI: 10.1038/s42254-021-00348-9. URL: http://dx.doi.org/10.1038/s42254-021-00348-9.

[14] Shouvanik Chakrabarti et al. "A Threshold for Quantum Advantage in Derivative Pricing". In: *Quantum* 5 (June 2021), p. 463. ISSN: 2521-327X. DOI: 10.22331/q-2021-06-01-463. URL: http://dx.doi.org/10.22331/q-2021-06-01-463.

[15] Francesca Cibrario et al. "Quantum Amplitude Loading for Rainbow Options Pricing". In: *2024 IEEE International Conference on Quantum Computing and Engineering (QCE)*. IEEE, Sept. 2024, pp. 211–220. DOI: 10.1109/qce60285.2024.00034. URL: http://dx.doi.org/10.1109/QCE60285.2024.00034.

[16] John Clarke and Frank K. Wilhelm. "Superconducting quantum bits". In: *Nature* 453.7198 (June 2008), pp. 1031–1042. ISSN: 1476-4687. DOI: 10.1038/nature07128. URL: http://dx.doi.org/10.1038/nature07128.

[17] Classiq Technologies Ltd. *Classiq Platform*. Version 0.35.1. Jan. 15, 2024. URL: https://platform.classiq.io.

[18] Andrew Cross et al. "OpenQASM 3: A Broader and Deeper Quantum Assembly Language". In: *ACM Transactions on Quantum Computing* 3.3 (Sept. 2022), pp. 1–50. ISSN: 2643-6817. DOI: 10.1145/3505636. URL: http://dx.doi.org/10.1145/3505636.

[19] Christopher M. Dawson and Michael A. Nielsen. "The Solovay-Kitaev algorithm". In: 6.1 (Jan. 2006), pp. 81–95. ISSN: 1533-7146.

[20] D. Deutsch. "Quantum theory, the Church-Turing principle and the universal quantum computer". In: *Proceedings of the Royal Society of London Series A* 400.1818 (July 1985), pp. 97–117. DOI: 10.1098/rspa.1985.0070.

[21] Cirq Developers. *Cirq*. Zenodo, Apr. 2025. DOI: 10.5281/ZENODO.4062499. URL: https://zenodo.org/doi/10.5281/zenodo.4062499.

[22] Simon J Devitt, William J Munro, and Kae Nemoto. "Quantum error correction for beginners". In: *Reports on Progress in Physics* 76.7 (June 2013), p. 076001. ISSN: 1361-6633. DOI: 10.1088/0034-4885/76/7/076001. URL: http://dx.doi.org/10.1088/0034-4885/76/7/076001.

[23] M. H. Devoret and R. J. Schoelkopf. "Superconducting Circuits for Quantum Information: An Outlook". In: *Science* 339.6124 (Mar. 2013), pp. 1169–1174. ISSN: 1095-9203. DOI: 10.1126/science.1231930. URL: http://dx.doi.org/10.1126/science.1231930.

[24] Yulong Dong et al. "Infinite quantum signal processing". In: *Quantum* 8 (Dec. 2024), p. 1558. ISSN: 2521-327X. DOI: 10.22331/q-2024-12-10-1558. URL: http://dx.doi.org/10.22331/q-2024-12-10-1558.

[25] Thomas G Draper et al. "A logarithmic-depth quantum carry-lookahead adder". In: *arXiv preprint quant-ph/0406142* (2004).

[26] Emanuele Dri et al. "A More General Quantum Credit Risk Analysis Framework". In: *Entropy* 25.4 (Mar. 2023), p. 593. ISSN: 1099-4300. DOI: 10.3390/e25040593. URL: http://dx.doi.org/10.3390/e25040593.

[27] Emanuele Dri et al. "Towards practical Quantum Credit Risk Analysis". In: *Journal of Physics: Conference Series* 2416.1 (Dec. 2022), p. 012002. ISSN: 1742-6596. DOI: 10.1088/1742-6596/2416/1/012002. URL: http://dx.doi.org/10.1088/1742-6596/2416/1/012002.

[28] Daniel J. Egger et al. "Credit Risk Analysis Using Quantum Computers". In: *IEEE Transactions on Computers* 70.12 (Dec. 2021), pp. 2136–2145. ISSN: 2326-3814. DOI: 10.1109/tc.2020.3038063. URL: http://dx.doi.org/10.1109/TC.2020.3038063.

[29] Daniel J. Egger et al. "Quantum Computing for Finance: State-of-the-Art and Future Prospects". In: *IEEE Transactions on Quantum Engineering* 1 (2020), pp. 1–24. ISSN: 2689-1808. DOI: 10.1109/tqe.2020.3030314. URL: http://dx.doi.org/10.1109/TQE.2020.3030314.

[30] 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.

[31] Russ Fein. *Quantum Computing Modalities*. Substack newsletter. Dec. 2024. URL: https://russfein.substack.com/p/quantum-computing-modalities (visited on 03/30/2025).

[32] 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.

[33] András Gilyén et al. "Quantum singular value transformation and beyond: exponential improvements for quantum matrix arithmetics". In: *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*. STOC 2019. Phoenix, AZ, USA: Association for Computing Machinery, 2019, pp. 193–204. ISBN: 9781450367059. DOI: 10.1145/3313276.3316366. URL: https://doi.org/10.1145/3313276.3316366.

[34] Paul Glasserman. *Monte Carlo Methods in Financial Engineering*. Springer New York, 2003. ISBN: 9780387216171. DOI: 10.1007/978-0-387-21617-1. URL: http://dx.doi.org/10.1007/978-0-387-21617-1.

[35] Tomer Goldfriend et al. *Design and synthesis of scalable quantum programs*. 2024. DOI: 10.48550/ARXIV.2412.07372. URL: https://arxiv.org/abs/2412.07372.

[36] William B. Gragg. "Matrix interpretations and applications of the continued fraction algorithm". In: *Rocky Mountain Journal of Mathematics* 4.2 (June 1974). ISSN: 0035-7596. DOI: 10.1216/rmj-1974-4-2-213. URL: http://dx.doi.org/10.1216/RMJ-1974-4-2-213.

[37] Dmitry Grinko et al. "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.

[38] Lov K. Grover. "A fast quantum mechanical algorithm for database search". In: *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing - STOC '96*. STOC '96. ACM Press, 1996, pp. 212–219. DOI: 10.1145/237814.237866. URL: http://dx.doi.org/10.1145/237814.237866.

[39] Lov K. Grover. "Synthesis of Quantum Superpositions by Quantum Computation". In: *Physical Review Letters* 85.6 (Aug. 2000), pp. 1334–1337. ISSN: 1079-7114. DOI: 10.1103/physrevlett.85.1334. URL: http://dx.doi.org/10.1103/PhysRevLett.85.1334.

[40] Martina Gschwendtner, Nicole Morgan, and Henning Soller. *Quantum technology use cases as fuel for value in finance*. URL: https://www.mckinsey.com/capabilities/mckinsey-digital/our-insights/tech-forward/quantum-technology-use-cases-as-fuel-for-value-in-finance (visited on 02/10/2025).

[41] *Hello world*. en. URL: https://docs.quantum.ibm.com/guides/hello-world (visited on 04/04/2025).

[42] Dylan Herman et al. "Quantum computing for finance". In: *Nature Reviews Physics* 5.8 (July 2023), pp. 450–465. ISSN: 2522-5820. DOI: 10.1038/s42254-023-00603-1. URL: http://dx.doi.org/10.1038/s42254-023-00603-1.

[43] *IntegerComparator (latest version)*. en. URL: https://quantum.cloud.ibm.com/docs/en/api/qiskit/qiskit.circuit.library.IntegerComparator (visited on 07/04/2025).

[44] *Ising model*. https://encyclopediaofmath.org/index.php?title=Ising_model. Accessed: 2025-04-13.

[45] Ali Javadi-Abhari et al. *Quantum computing with Qiskit*. 2024. DOI: 10.48550/arXiv.2405.08810. arXiv: 2405.08810 [quant-ph].

[46] Rajiv Krishnakumar and William Zeng. *Quantum Rounding*. 2021. arXiv: 2108.05949 [quant-ph]. URL: https://arxiv.org/abs/2108.05949.

[47] E. L. Lawler and D. E. Wood. "Branch-And-Bound Methods: A Survey". In: *Operations Research* 14.4 (1966), pp. 699–719. ISSN: 0030364X, 15265463. URL: http://www.jstor.org/stable/168733.

[48] Jongheon Lee et al. "T-depth reduction method for efficient SHA-256 quantum circuit construction". In: *IET Information Security* 17.1 (2023), pp. 46–65.

[49] Daniel Loss and David P. DiVincenzo. "Quantum computation with quantum dots". In: *Physical Review A* 57.1 (Jan. 1998), pp. 120–126. ISSN: 1094-1622. DOI: 10.1103/physreva.57.120. URL: http://dx.doi.org/10.1103/PhysRevA.57.120.

[50] Harry Markowitz. "Portfolio Selection". In: *The Journal of Finance* 7.1 (Mar. 1952), p. 77. ISSN: 0022-1082. DOI: 10.2307/2975974. URL: http://dx.doi.org/10.2307/2975974.

[51] Ana Martin et al. "Toward pricing financial derivatives with an IBM quantum computer". In: *Physical Review Research* 3.1 (Feb. 2021). ISSN: 2643-1564. DOI: 10.1103/physrevresearch.3.013167. URL: http://dx.doi.org/10.1103/PhysRevResearch.3.013167.

[52] John M. Martyn et al. "Grand Unification of Quantum Algorithms". In: *PRX Quantum* 2.4 (Dec. 2021). ISSN: 2691-3399. DOI: 10.1103/prxquantum.2.040203. URL: http://dx.doi.org/10.1103/PRXQuantum.2.040203.

[53] Dmitri Maslov. "Advantages of using relative-phase Toffoli gates with an application to multiple control Toffoli optimization". In: *Physical Review A* 93.2 (2016), p. 022311.

[54] Sam McArdle, András Gilyén, and Mario Berta. *Quantum state preparation without coherent arithmetic*. 2022. arXiv: 2210.14892 [quant-ph]. URL: https://arxiv.org/abs/2210.14892.

[55] Robert C. Merton. "On the Pricing of Corporate Debt: The Risk Structure of Interest Rates". In: *The Journal of Finance* 29.2 (May 1974), p. 449. ISSN: 0022-1082. DOI: 10.2307/2978814. URL: http://dx.doi.org/10.2307/2978814.

[56] Microsoft. *Q# Language Specification*. 2020. URL: https://github.com/microsoft/qsharp-language/tree/main/Specifications/Language#q-language.

[57] 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.

[58] Mikko Möttönen et al. "Transformation of quantum states using uniformly controlled rotations". In: *Quantum Info. Comput.* 5.6 (Sept. 2005), pp. 467–473. ISSN: 1533-7146.

[59] *Multi-Control-X Classiq*. https://docs.classiq.io/latest/user-guide/function-library/builtin-functions/mcx/. Accessed: 2024-01-02.

[60] Kouhei Nakaji et al. "Approximate amplitude encoding in shallow parameterized quantum circuits and its application to financial market indicators". In: *Physical Review Research* 4.2 (May 2022). ISSN: 2643-1564. DOI: 10.1103/physrevresearch.4.023136. URL: http://dx.doi.org/10.1103/PhysRevResearch.4.023136.

[61] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, June 2012. ISBN: 9780511976667. DOI: 10.1017/cbo9780511976667. URL: http://dx.doi.org/10.1017/CBO9780511976667.

[62] 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.

[63] Dessislava A. Pachamanova and Frank J. Fabozzi. *Portfolio Construction and Analytics*. Wiley, Feb. 2016. ISBN: 9781118656747. DOI: 10.1002/9781118656747. URL: http://dx.doi.org/10.1002/9781118656747.

[64] John Preskill. "Quantum Computing in the NISQ era and beyond". In: *Quantum* 2 (Aug. 2018), p. 79. ISSN: 2521-327X. DOI: 10.22331/q-2018-08-06-79. URL: http://dx.doi.org/10.22331/q-2018-08-06-79.

[65] *QasmSimulator - Qiskit Aer 0.16.1*. URL: https://qiskit.github.io/qiskit-aer/stubs/qiskit_aer.QasmSimulator.html (visited on 04/11/2025).

[66] qsppack. *qsppack/QSPPACK*. original-date: 2020-02-24T08:11:30Z. Apr. 2025. URL: https://github.com/qsppack/QSPPACK (visited on 04/23/2025).

[67] Atanu Rajak et al. "Quantum annealing: an overview". In: *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 381.2241 (Dec. 2022). ISSN: 1471-2962. DOI: 10.1098/rsta.2021.0417. URL: http://dx.doi.org/10.1098/rsta.2021.0417.

[68] Sergi Ramos-Calderer et al. "Quantum unary approach to option pricing". In: *Physical Review A* 103.3 (Mar. 2021). ISSN: 2469-9934. DOI: 10.1103/physreva.103.032414. URL: http://dx.doi.org/10.1103/PhysRevA.103.032414.

[69] Arthur G. Rattew et al. "The Efficient Preparation of Normal Distributions in Quantum Registers". In: *Quantum* 5 (Dec. 2021), p. 609. ISSN: 2521-327X. DOI: 10.22331/q-2021-12-23-609. URL: http://dx.doi.org/10.22331/q-2021-12-23-609.

[70] Patrick Rebentrost, Brajesh Gupt, and Thomas R. Bromley. "Quantum computational finance: Monte Carlo pricing of financial derivatives". In: *Physical Review A* 98.2 (Aug. 2018). ISSN: 2469-9934. DOI: 10.1103/physreva.98.022321. URL: http://dx.doi.org/10.1103/PhysRevA.98.022321.

[71] Neil J Ross and Peter Selinger. "Optimal ancilla-free Clifford+ T approximation of z-rotations". In: *arXiv preprint arXiv:1403.2975* (2014).

[72] Yuval R. Sanders et al. "Black-Box Quantum State Preparation without Arithmetic". In: *Physical Review Letters* 122.2 (Jan. 2019). ISSN: 1079-7114. DOI: 10.1103/physrevlett.122.020502. URL: http://dx.doi.org/10.1103/PhysRevLett.122.020502.

[73] Peter W. Shor. "Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer". In: *SIAM Journal on Computing* 26.5 (Oct. 1997), pp. 1484–1509. ISSN: 1095-7111. DOI: 10.1137/s0097539795293172. URL: http://dx.doi.org/10.1137/S0097539795293172.

[74] Rafael Sotelo et al. "Quantum Computing in Finance: The Intesa Sanpaolo Experience". In: *IEEE Engineering Management Review* 52.2 (Apr. 2024), pp. 9–15. ISSN: 1937-4178. DOI: 10.1109/emr.2024.3373796. URL: http://dx.doi.org/10.1109/EMR.2024.3373796.

[75] Nikitas Stamatopoulos et al. "Option Pricing using Quantum Computers". In: *Quantum* 4 (July 2020), p. 291. ISSN: 2521-327X. DOI: 10.22331/q-2020-07-06-291. URL: http://dx.doi.org/10.22331/q-2020-07-06-291.

[76] Nikitas Stamatopoulos et al. *Quantum Risk Analysis of Financial Derivatives*. 2024. arXiv: 2404.10088 [quant-ph]. URL: https://arxiv.org/abs/2404.10088.

[77] Yohichi Suzuki et al. "Amplitude estimation without phase estimation". In: (2019). DOI: 10.48550/ARXIV.1904.10246. URL: https://arxiv.org/abs/1904.10246.

[78] Kristan Temme, Sergey Bravyi, and Jay M. Gambetta. "Error Mitigation for Short-Depth Quantum Circuits". In: *Physical Review Letters* 119.18 (Nov. 2017). ISSN: 1079-7114. DOI: 10.1103/physrevlett.119.180509. URL: http://dx.doi.org/10.1103/PhysRevLett.119.180509.

[79] Jules Tilly et al. "The Variational Quantum Eigensolver: A review of methods and best practices". In: *Physics Reports* 986 (Nov. 2022), pp. 1–128. ISSN: 0370-1573. DOI: 10.1016/j.physrep.2022.08.003. URL: http://dx.doi.org/10.1016/j.physrep.2022.08.003.

[80] Charles Van Loan. *Computational frameworks for the fast Fourier transform*. USA: Society for Industrial and Applied Mathematics, 1992. ISBN: 0898712858.

[81] Matan Vax et al. *Qmod: Expressive High-Level Quantum Modeling*. 2025. DOI: 10.48550/ARXIV.2502.19368. URL: https://arxiv.org/abs/2502.19368.

[82] Shengbin Wang et al. "Fast black-box quantum state preparation based on linear combination of unitaries". In: *Quantum Information Processing* 20.8 (Aug. 2021). ISSN: 1573-1332. DOI: 10.1007/s11128-021-03203-z. URL: http://dx.doi.org/10.1007/s11128-021-03203-z.

[83] Shengbin Wang et al. "Inverse-coefficient black-box quantum state preparation". In: *New Journal of Physics* 24.10 (Oct. 2022), p. 103004. ISSN: 1367-2630. DOI: 10.1088/1367-2630/ac93a8. URL: http://dx.doi.org/10.1088/1367-2630/ac93a8.

[84] *WeightedAdder (latest version)*. en. URL: https://docs.quantum.ibm.com/api/qiskit/qiskit.circuit.library.WeightedAdder (visited on 04/13/2025).

[85] Hung-Pin Wen, Yao Wen Chang, and Kwang Ting Cheng. *Electronic Design Automation: Synthesis, Verification, and Test*. English. 1st ed. Morgan Kaufmann Publishers, Inc., Feb. 2009. ISBN: 9780123743640.

[86] 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.

[87] Xiao-Ming Zhang, Man-Hong Yung, and Xiao Yuan. "Low-depth quantum state preparation". In: *Physical Review Research* 3.4 (Dec. 2021). ISSN: 2643-1564. DOI: 10.1103/physrevresearch.3.043200. URL: http://dx.doi.org/10.1103/physrevresearch.3.043200.

[88] Christa Zoufal, Aurélien Lucchi, and Stefan Woerner. "Quantum Generative Adversarial Networks for learning and loading random distributions". en. In: *npj Quantum Information* 5.1 (Nov. 2019), p. 103. ISSN: 2056-6387. DOI: 10.1038/s41534-019-0223-2. URL: https://www.nature.com/articles/s41534-019-0223-2 (visited on 04/07/2025).

This Ph.D. thesis has been typeset by means of the TEX-system facilities. The typesetting engine was pdfLATEX. The document class was `toptesi`, by Claudio Beccari, with option `tipotesi=scudo`. This class is available in every up-to-date and complete TEX-system installation.