UNIVERSITY OF LJUBLJANA
SCHOOL OF ECONOMICS AND BUSINESS

MASTER'S THESIS

THE ROLE OF QUANTUM COMPUTING IN ENHANCING COMPUTATION SPEED AND EFFICIENCY IN FINANCE APPLICATIONS

Ljubljana, September 2024 GAŠPER TREBŠE

AUTHORSHIP STATEMENT

The undersigned Gašper Trebše, a student at the University of Ljubljana, School of Economics and Business, (hereafter: SEB LU), author of this written final work of studies with the title The role of quantum computing in enhancing computation speed and efficiency in finance applications, prepared under supervision of Professor Igor Masten PhD.

DECLARE

1. this written final work of studies to be based on the results of my own research;
2. the printed form of this written final work of studies to be identical to its electronic form;
3. the text of this written final work of studies to be language-edited and technically in adherence with the SEB LU's Technical Guidelines for Written Works, which means that I cited and/or quoted works and opinions of other authors in this written final work of studies in accordance with the SEB LU's Technical Guidelines for Written Works;
4. to be aware of the fact that plagiarism (in written or graphical form) is a criminal offence and can be prosecuted in accordance with the Criminal Code of the Republic of Slovenia;
5. to be aware of the consequences a proven plagiarism charge based on this written final work could have for my status at the SEB LU in accordance with the relevant SEB LU Rules;
6. to have obtained all the necessary permits to use the data and works of other authors which are (in written or graphical form) referred to in this written final work of studies and to have clearly marked them;
7. to have acted in accordance with ethical principles during the preparation of this written final work of studies and to have, where necessary, obtained permission of the Ethics Committee;
8. my consent to use the electronic form of this written final work of studies for the detection of content similarity with other written works, using similarity detection software that is connected with the SEB LU Study Information System;
9. to transfer to the University of Ljubljana free of charge, non-exclusively, geographically and time-wise unlimited the right of saving this written final work of studies in the electronic form, the right of its reproduction, as well as the right of making this written final work of studies available to the public on the World Wide Web via the Repository of the University of Ljubljana;
10. my consent to publication of my personal data that are included in this written final work of studies and in this declaration, when this written final work of studies is published.
11. that I have verified the authenticity of the information derived from the records using artificial intelligence tools.

Ljubljana, _____ Author's signature: _____

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF APPENDICES

LIST OF ABBREVIATIONS

API Application programming interface

CLOPS Circuit layer operations per second

CLT Central limit theorem

COBYLA Constrainted optimization by linear approximation

CPU Central processing unit

GPU Graphics processing unit

IQAE Iterative quantum ampliture estimation

LGD Loss given default

MAE Mean absolute error

NISQ Noisy Intermediate-Scale Quantum

PD Probability of default

QAE Quantum amplitude estimation

QAOA Quantum amplitude estimation

QEC Quantum error correction

QPE Quantum phase estimation

QUBO Quadratic unconstrained binary optimization

QV Quantum volume

RMSE Root mean square error

SDK Software development kit

VaR Value at risk

VQE Variational quantum eigensolver

1 INTRODUCTION

Quantum computing is a new type of computing that uses the principles of quantum mechanics to perform complex mathematical derivations and execute algorithms. Unlike traditional computers, which use 'bits' to process information, quantum computers use 'qubits'. In a traditional computer, bits can be either in a state of 0 or 1. However, qubits, thanks to quantum mechanics, can be in a state of 0, 1, or both at the same time. This property is known as 'superposition'. Another unique property of qubits is 'entanglement'. When qubits become entangled, the state of one qubit can affect the state of another, irrespective of the distance between them. These two properties - superposition and entanglement - make quantum computers potentially far more powerful than traditional computers.

Quantum computing is still a developing field, but it's advancing quickly. Major tech companies like IBM and Google are investing heavily in research and development of both quantum hardware and quantum algorithms that are used in conjunction with quantum computers. Similarly, many firms working in the financial services industry such as Goldman Sachs and Ernst and Young (EY, 2023), have already invested heavily in quantum computing. Given the massive interest and investments in quantum computing, rapid advancements in the technology are expected which to bring significant changes in the finance industry.

Financial services have always been at the forefront of using new technologies. From the earliest mechanical calculators to today's most advanced supercomputers, every new development in computing has found its way into finance. Quantum computing, with its potential for speed and efficiency, is likely no exception. Today, computing plays a crucial role in many areas of finance. From the stock market, where advanced computers track and store millions of transactions per second, to predicting future price movements, managing investment portfolios, and risk management, powerful computers are everywhere. But even the most powerful traditional computers often struggle with these tasks due to the overwhelming amount of data to process. Quantum computers, with their ability to perform many calculations at once could be a game changer for multiple sectors in finance. In this master's thesis we will explore the capabilities of quantum computers for solving problems in portfolio and risk management. Specifically, we will focus on the value at risk estimation and portfolio optimization problems.

This master's thesis is structured as follows. In the first section, we will explore the basic concepts of quantum computing, introduce the mathematical notation required to understand quantum algorithms and circuits and describe the algorithms that will be used in the thesis. In the second section, the computational approaches relevant for the finance industry and this master thesis will be described. In the third section, we will describe how we will apply quantum algorithms to estimate VaR and optimize portfolios. Additionally, we will describe how we will perform algorithm benchmarking. In the fourth section, we will present the results of the experiments and discuss the implications of the results. Finally, in the last section, we will summarize and discuss the results of the thesis.

2 INTRODUCTION TO QUANTUM COMPUTING

As should be clear at this point quantum computing differs from classical computing in many ways. One of the most important differences are the mathematical concept and notations that are associated with quantum computing, which rely primarily on vector operations, complex linear algebra and rudimentary probability theory. Additionally, there are several differences in how quantum information is processed and analyzed compared to classical data. While the main purpose of this thesis is not to explain the aforementioned differences, it is nonetheless crucial for a further understanding of the thesis to explain the basic building blocks of quantum computing, which will be used extensively throughout the thesis.

2.1 Mathematical foundations

2.1.1 Bra-ket notation

The Bra-ket notation, otherwise known as Dirac's notation is often used in quantum mechanics to denote quantum states. It was created and published by the famous theoretical physicist Paul Dirac in 1939 with the purpose of simplifying and unifying the notations used in quantum mechanics (Dirac, 1939).

First, let's define the *ket*, which denotes a vector and represents a state of a quantum system. For the purposes of quantum computing we want to describe the state of a qubit, which can be in a state of 0 or 1, or in a superposition of both. To describe this system we define the standard basis vectors given by:

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

You can think of these vectors as the base states in which a qubit can be in. Note that a quantum bit can also be in a superposition of these two states, which is denoted by:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \tag{2}$$

We can interpret the equation above by translating it to a classical setting. To make it more intuitive we can map the standard basis vectors to states *heads* and *tails* using the relation:

$$|0\rangle = |heads\rangle \text{ and } |1\rangle = |tails\rangle \tag{3}$$

For the fair coin example, the equation (2) transforms into the following:

$$|\psi\rangle = \frac{1}{2}|heads\rangle + \frac{1}{2}|tails\rangle \tag{4}$$

The equation should now be more intuitive. It describes a fair coin toss, which after a flip (in quantum computing we would call this a measurement) has a 50% chance of being in a state of heads or tails. Note that in a quantum setting the

interpretation of α and β in equation (2) is slightly different. This difference will be further explored in the subsequent sections.

Next, let's explore the first part of the notation, the *bra*, which is used to denote a row vector. In the example below we can observe the basis vectors defined in equation (1) written in bra notation:

$$\langle 0| = \begin{pmatrix} 1 & 0 \end{pmatrix} \text{ and } \langle 1| = \begin{pmatrix} 0 & 1 \end{pmatrix} \tag{5}$$

By combining the *bra* and *ket* we get the *bra-ket*. This operation represents the inner product of two vectors. For example, the inner product of the basis vectors defined in equation (1) is given by:

$$\langle x|y\rangle = \begin{pmatrix} x_1 & x_2 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = x_1 * y_1 + x_2 * y_2 \tag{6}$$

Using the basis vectors defined in equation (1) the formula above can be interpreted as:

$$\langle x|y\rangle = \begin{cases} 1, & x = y \\ 0, & x \neq y \end{cases} \tag{7}$$

2.1.2 Quantum state vectors

In this section the *bra-ket* notation outlined in Section 2.1.1 will be used to describe the state of a quantum system. We will also explore the concept of measurements, which are crucial for understanding of quantum computing. A system's quantum state is represented by a column vector, similar to probabilistic states, with two important differences:

- Elements of the vector are complex numbers;
- The sum of the squares of absolute values of the elements is equal to 1.

The first of the two facts above enables quantum state vectors to contain negative real numbers (if the imaginary part of the complex number is negative), in contrast to classical states, which are required to be positive. The second fact shows that the sum of all elements of the state vector does not have to be equal to one, as is the case with probabilistic states. These differences may seem trivial at first, but they are the key reason behind the speedup of quantum computers compared to classical computers.

The state vector of a qubit is a linear combination of the basis vectors defined in equation (1). Below is an example of qubit state vectors:

$$|\psi\rangle = \begin{pmatrix} \frac{2-i}{3} \\ \frac{2i}{3} \end{pmatrix} = \frac{2-i}{3}|0\rangle + \frac{2i}{3}|1\rangle \tag{8}$$

2.1.2.1 Measuring quantum states

When the state of a qubit is measured, the quantum state collapses to one of the basis states. The probability of measuring a given state is equal to the absolute value of its entry in the state vector squared. For the state described in equation (8) above.

$$Pr(outcome = 0) = |\langle\psi|0\rangle|^2 = \left|\frac{2i-1}{3}\right|^2 = \frac{5}{9} \tag{9}$$

$$Pr(outcome = 1) = |\langle\psi|1\rangle|^2 = \left|\frac{2i}{3}\right|^2 = \frac{4}{9} \tag{10}$$

As outlined in Section 2.1.2, the sum of the probabilities of all possible outcomes is equal to 1, similarly to probabilistic states, while the entries of the state vector do not have to sum to 1 and can be complex numbers (Rué and Xambó, 2011). This property was first formulated by Max Born, and is also known as the Born rule (Gleason, 1975).

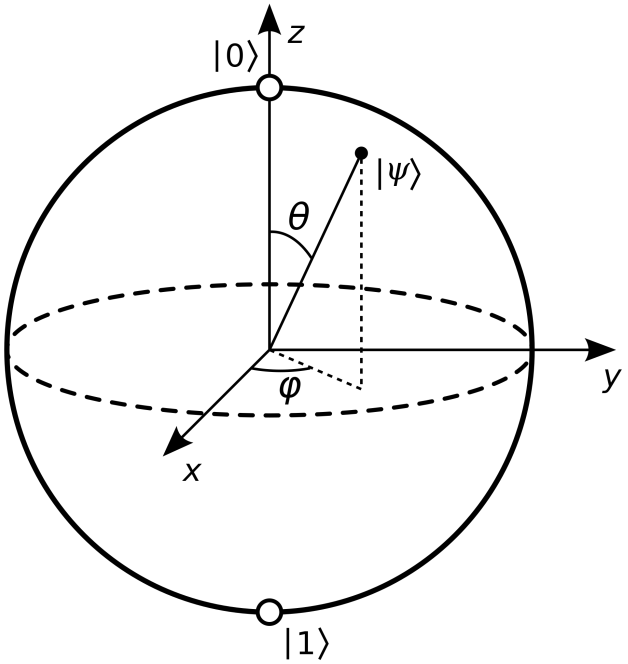
2.1.3 Bloch sphere

In addition to the *bra-ket* notation, which is used to intuitively describe quantum states in a text format, we can also use the Bloch sphere to graphically depict the state of a qubit. In Figure 1 below we can see an example of a Bloch sphere, where the North Pole of the sphere represents the $|0\rangle$ state and the South Pole represents the $|1\rangle$ state. The Bloch sphere below also shows a quantum state vector $\psi\rangle$ which is in a superposition of the two states. The state can be described by the following equation.

$$|\psi\rangle = \cos(\theta/2)|0\rangle + e^{i\varphi}\sin(\theta/2)|1\rangle = \cos(\theta/2)|0\rangle + (\cos\varphi + i\sin\varphi)\sin(\theta/2)|1\rangle \tag{11}$$

Similarly to the spherical coordinate system, θ represents the angle between the radial line and the z-axis, while φ represents the angle between the projection of the radial line on the x-y plane and the x-axis.

Figure 1: Bloch sphere



Source: Smite-Meister (2009).

The Bloch sphere will be used throughout the thesis to aid the understanding of the quantum state vectors and the effects of quantum operations on the state vectors.

2.1.4 Operations on quantum states

The final concept that we will explore in this section are operations on quantum states. Similarly, to the probabilistic setting, operations that we perform on the quantum state vectors are linear mappings. These are most often represented by matrices, however unlike the probabilistic setting, where we use stochastic matrices, in the quantum setting we use unitary matrices. A unitary matrix must satisfy the following conditions:

$$U^\dagger U = \mathbb{1} \text{ and } UU^\dagger = \mathbb{1} \tag{12}$$

where the $\mathbb{1}$ denotes the identity matrix and $U^\dagger$ denotes the conjugate transpose of the matrix U. The conjugate transpose of a matrix is obtained by taking the transpose of the matrix and then taking the complex conjugate of each entry. By ensuring that the matrices are unitary we ensure the euclidean norm of the state vector is preserved and is equal to 1. This is important as it ensures that a quantum state, multiplied by a unitary matrix remains a quantum state vector.

2.1.4.1 Basic operations

In this section we will describe the basic operations that can be performed on quantum states. These are the operations that are most often used in quantum computing and are the building blocks of more complex operations.

Pauli operations The Pauli matrices consist of three complex matrices, each of 2x2 dimension that are unitary, hermitian and involutory. The matrices are given by:

$$\sigma_x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \sigma_y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \sigma_z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \tag{13}$$

The Pauli-X matrix can be interpreted as a rotation of the qubit state vector around the x-axis of the Bloch sphere by π radians. The operation can also be seen as a classical not operation, which flips the state of the qubit from 0 to 1 or vice versa. The Pauli-Y and Pauli-Z matrices can be interpreted similarly, with the difference being the axis around which the rotation is performed.

Rotation operators The Pauli matrices, which were described in the previous section, are a variant of the general rotation operators where the rotation is fixed at π radians. The rotation operation can be generalized to any angle θ and axis, and is given by the following matrices.

$$R_x(\theta) = \begin{pmatrix} \cos(\theta/2) & -i\sin(\theta/2) \\ -i\sin(\theta/2) & \cos(\theta/2) \end{pmatrix} \tag{14}$$

$$R_y(\theta) = \begin{pmatrix} \cos(\theta/2) & -\sin(\theta/2) \\ \sin(\theta/2) & \cos(\theta/2) \end{pmatrix} \tag{15}$$

$$R_z(\theta) = \begin{pmatrix} e^{-i\theta/2} & 0 \\ 0 & e^{i\theta/2} \end{pmatrix} \tag{16}$$

Hadamard operation The Hadamard matrix, denoted by H is a 2x2 matrix and is given by:

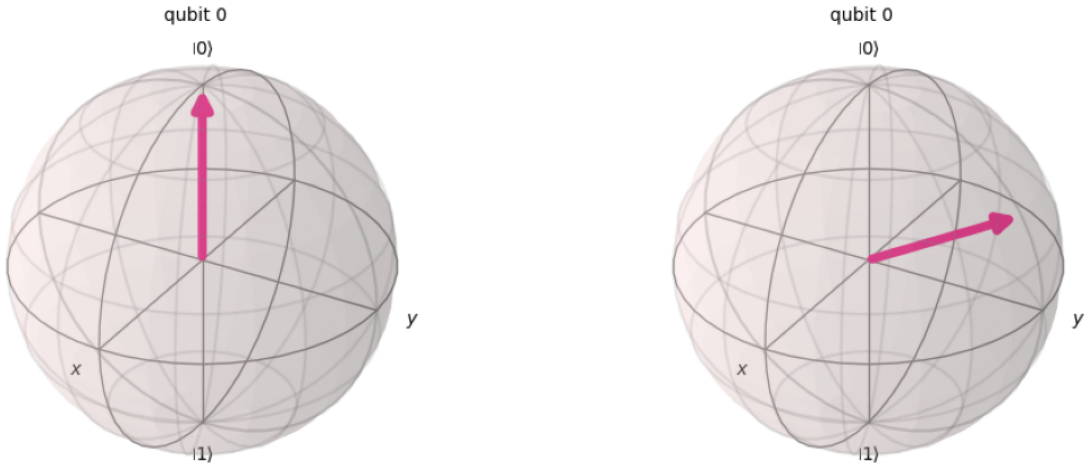
$$H = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix} \tag{17}$$

This matrix is most commonly used to create superpositions of qubits. For example, if we apply the Hadamard matrix to the basis vector $|0\rangle$ we get the following:

$$H|0\rangle = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix} = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle \tag{18}$$

The equation above describes a quantum state that has equal probabilities of being in a state of 0 or 1. This operation can also be depicted with a Bloch sphere. In Figure 2 below we can see the effect of the Hadamard operation on the state vector of a qubit. The state vector is rotated by π radians around the y-axis of the Bloch sphere, which results in a state vector that is in a superposition of 0 and 1.

Figure 2: Hadamard operation on the Bloch sphere



Source: Own work.

Phase operations The phase operation is given by the matrix:

$$P_\theta = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\theta} \end{pmatrix} \tag{19}$$

Applying the phase operation to a quantum state will change the phase of the state by the angle defined by θ.

2.2 NISQ quantum computers

To describe the current stage in quantum computing the term "Noisy Intermediate-Scale Quantum" is often used (Brooks, 2019). The term is derived from the fact that the current quantum computers are noisy, and typically only have up to a few hundred bits. The term "noisy" is used due to high error rates present in the system and "intermediate-scale" due to the low number of available qubits in such systems. Nonetheless, NISQ quantum computers represent a significant step in our understanding of the technology and give researchers a platform to test and develop new quantum algorithms. In this section we will explore the challenges related to NISQ quantum computers and how they can affect the performance of quantum algorithms.

2.2.1 Decoherence

Decoherence is a process in quantum mechanics where a quantum state loses its initial properties when it interacts with its environment. For example, when a qubit is in a superposition of 0 and 1 as defined in equation (18) and interacts with its environment, the superposition decoheres to a different state, and the probability

of measuring a 0 or 1 will not be equal anymore. This process limits the number of sequential operations that can be performed on a single quantum state, limiting the complexity of quantum algorithms that can be run on NISQ quantum computers. These types of errors can present a significant challenge and can limit the theoretical convergence rate of quantum algorithms (Preskill, 2018). For this reason, quantum computers are often kept at extremely low temperatures, close to absolute zero and are shielded from the surrounding environment.

2.2.2 Error correction

Due to high error rates related to decoherence and other factors, quantum computers are often prone to errors. For this reason, error correction is a crucial technique in (NISQ) computing, which is applied to mitigate the effects of these errors and ensure the reliability of quantum computations. Quantum Error Correction (QEC) involves actively encoding quantum information into a state that is resistant to errors and can be corrected if an error is detected. Since its introduction in 1995 by Peter Shor (Shor, 1995), QEC has evolved significantly, with a variety of codes and procedures developed to protect against errors in quantum systems. These developments are essential for the practical realization of quantum computing, influencing all levels of the quantum computing stack, from the physical layout of qubits to software-level gate compilation strategies (Devitt et al., 2013, Roffe, 2019).

2.3 Quantum algorithms

In the previous section, we explored the differences in mathematics and properties of the qubits, measurements and operations performed on them. Now we will explore how these properties can be combined and used to build quantum algorithms that can in theory outperform classical computers in specific tasks. First, we will explore the Grover's algorithm, which is one of the most influential quantum algorithm and forms the basis for the Quantum Amplitude Estimation (QAE) algorithm, that will be used in further sections to perform VaR estimation. Additionally, we will describe the inner workings of the Quantum Approximate Optimization Algorithm (QAOA), used for optimization tasks such as portfolio optimization.

2.3.1 Grover's algorithm

Grover's algorithm, named after its developer Lov Grover, is a quantum computing algorithm that can be used for searching unsorted databases with a quadratic speedup over classical methods. Notably, it can find specific items in a database of N items in only $\sqrt{N}$ iterations, as opposed to approximately N/2 iterations required by the classical algorithm. The algorithm does not provide an exponential speedup over classical algorithms, like some other quantum algorithms, but it still provides quadratic speedup over classical methods (Grover, 1996).

Grover's algorithm works by employing the principle of superposition. It operates in the following manner:

1. **Initialization:** The algorithm starts by initializing a system of N qubits into a superposition of all possible states (see Equation (20) below), by applying the Hadamard operation (see Section 2.1.4.1) to each of the qubits. This operation creates a superposition, where each state of the quantum system has an equal probability of being measured.

$$|s\rangle = \frac{1}{\sqrt{|N_x|}} \sum_X |x\rangle \tag{20}$$

   Where X represents all classical states and N_x represents the number of all classical states.

2. **Oracle application:** In the next step, an oracle is applied to the system. The oracle can be viewed as a "black box" operation, that can recognize the state corresponding to the item we are searching for. The oracle will then flip the phase of the solution state, which will result in a negative sign in front of the state vector. The application of an oracle on a quantum state $|x\rangle$ is given by

$$U_\omega|x\rangle = \begin{cases} -|x\rangle & \text{for } x = \omega, \\ |x\rangle & \text{for } x \neq \omega. \end{cases} \quad or\ U_\omega|x\rangle = (-1)^{f(x)}|x\rangle \tag{21}$$

   Where U_ω is the oracle operator, ω is the solution state, and $f(x)$ is a function that returns 1 if x is the solution state and 0 otherwise. Alternatively we can define the oracle as:

$$U_\omega = \mathbb{1} - 2|\omega\rangle\langle\omega| \tag{22}$$

   The equality of equations (21) and (22) is shown in appendix 2.

3. **Grovers diffusion operator:** After applying the oracle operator, we apply the Grover diffusion operator, which is given by:

$$U_{\omega_\perp} = 2|\omega_\perp\rangle\langle\omega_\perp| - \mathbb{1} \tag{23}$$

   Where $\omega_\perp$ is the state orthogonal to the solution state ω. Applying the Grover diffusion operator will thus invert the amplitude of each state about the average amplitude.

   By applying the oracle operator and the Grover diffusion operator $\sqrt{N}$ times, the probability of measuring the solution state will be close to 1. This will be graphically shown in the next section.

4. **Measurement:** After applying the oracle and the Grover diffusion operator $\sqrt{N}$ times, the system is measured, revealing the solution state with high probability.

2.3.1.1 Proof

The mechanism of Grover's algorithm can be displayed through a graphical representation. In Figure 3 we have a two-dimensional subspace, determined by ω and $\omega_\perp$, where ω is the solution state and $\omega_\perp$ is the state orthogonal to ω. The initial state of the system equivalent to the one defined in equation (20) is given by $|s\rangle$, which can be expressed as a linear combination of ω and $\omega_\perp$. Similarly, it can be shown that the vector can be expressed using the angle $\theta/2$ that spans between $|s\rangle$ and $\omega_\perp$:

$$|s\rangle = \frac{1}{\sqrt{|N_x|}} \sum_X |x\rangle = \sqrt{\frac{N_x - 1}{N_x}}|\omega_\perp\rangle + \sqrt{\frac{1}{N_x}}|\omega\rangle \equiv \cos(\theta/2)|\omega_\perp\rangle + \sin(\theta/2)|\omega\rangle \quad (24)$$

Applying the oracle operator U_ω to the state $|s\rangle$ will result in the following state:

$$U_\omega|s\rangle = \cos(\theta/2)|\omega_\perp\rangle - \sin(\theta/2)|\omega\rangle \quad (25)$$

The application of the oracle operator is equivalent to reflection of the state vector $|s\rangle$ about the state $\omega_\perp$. Before applying the Grover diffusion operator, we express the quantum state described in equation (25), with vectors $|s\rangle$ and $|s_\perp\rangle$ [1]

$$\begin{aligned} |s\rangle &= \cos(\theta/2)|\omega_\perp\rangle - \sin(\theta/2)|\omega\rangle \\ &= (\cos^2(\theta/2) - \sin^2(\theta/2))|s\rangle - (2\sin(\theta/2)\cos(\theta/2))|s_\perp\rangle \\ &= \cos(\theta)|s\rangle - \sin(\theta)|s_\perp\rangle \end{aligned} \quad (26)$$

The next step is to apply the Grover diffusion operator U_s to the state $U_\omega|s\rangle$ and express the result in terms of $|s\rangle$ and $|s_\perp\rangle$:

$$\begin{aligned} U_s U_\omega|s\rangle &= \cos(\theta/2)|s\rangle + \sin(\theta/2)|s_\perp\rangle \\ &= \sin(3\theta/2)|\omega\rangle + \cos(3\theta/2)|\omega_\perp\rangle \end{aligned} \quad (27)$$

The equation above describing a single application of the Grover's iterate $U_s U_\omega$ can be generalized for k applications of the Grover's iterate:

$$G^k|s\rangle = (U_s U_\omega)^k|s\rangle = \sin((2k+1)\theta/2)|\omega\rangle + \cos((2k+1)\theta/2)|\omega_\perp\rangle \quad (28)$$

To find the solution state $|\omega\rangle$ the term $\sin((2k+1)\theta/2)$ must be approximately 1. To determine the number of applications of the Grover's iterate k we write:

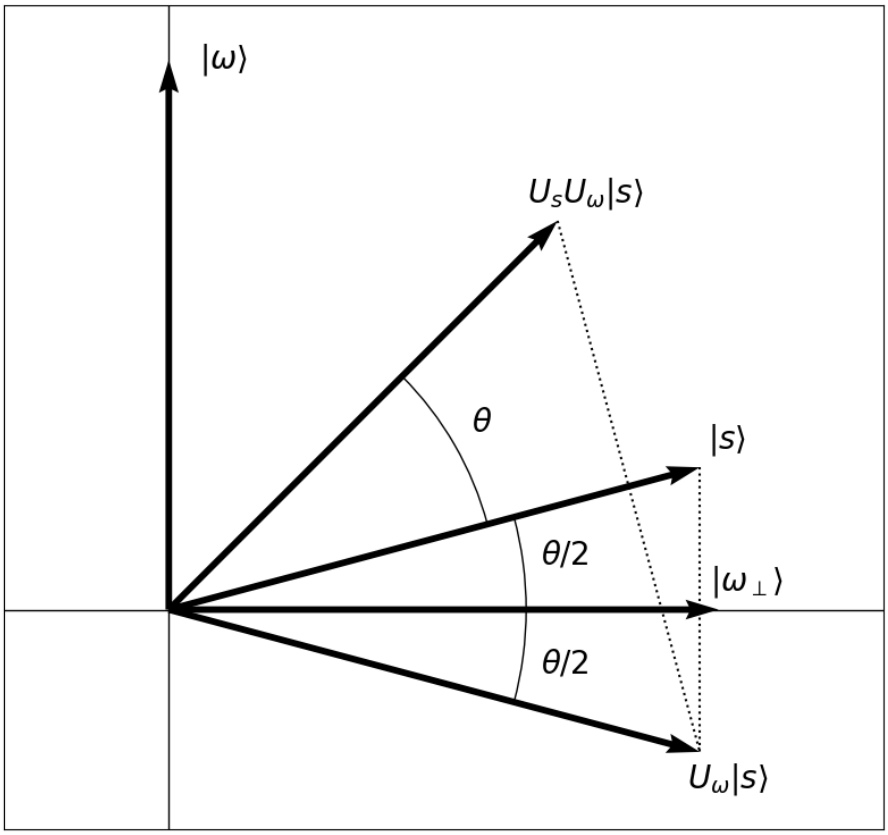
$$\begin{aligned} \sin((2k+1)\theta/2) &\sim 1 \\ (2k+1)\theta/2 &= \frac{\pi}{2} \\ 2k \arcsin\frac{1}{\sqrt{N_x}} &\sim \frac{\pi}{2} \\ k &\sim \frac{\pi}{4}\sqrt{N_x} \end{aligned} \quad (29)$$

This proves that the optimal number of applications of Grover's iterate scales with the square root of the number of states N_x, which is a quadratic speedup over the

[1] The state $|s_\perp\rangle$ is perpendicular to state $|s\rangle$ defined in equation (24) and can be expressed as $|s\rangle = \cos(\theta/2) - \sin(\theta/2)$.

classical algorithm. It is important to note that applying Grover's iterate more than k times will not result in a higher probability of measuring the solution state, and might even decrease the reliability of the answer.

Figure 3: Graphical representation of Grover's algorithm



Source: Own work.

2.3.2 Quantum amplitude estimation

Quantum Amplitude Estimation (QAE) algorithm is a quantum algorithm that can be used to estimate the amplitude or probability of a given quantum state. The algorithm is a generalization of Grover's algorithm and can be used to solve a variety of problems. In this master's thesis we will explore its application as a substitute for Monte Carlo simulation. The algorithm will be tested in two scenarios, first, we will use it to estimate the underlying probability p of a Bernoulli random (see Section 4.1), and second, we will use it to estimate the VaR of a portfolio of assets (see Section 4.2).

As described previously the algorithm takes a quantum state described by:

$$|\psi\rangle = X|0\rangle = \sqrt{(1-x)}|\psi_0\rangle + \sqrt{x}|\psi_1\rangle, \tag{30}$$

and estimates the amplitude or probability x of state $|\psi_1\rangle$:

$$x = |\langle\psi_1|\psi\rangle|^2 \tag{31}$$

Here we can also observe the difference compared to Grover's algorithm. The operator X used to generate the initial quantum state in equation (30) can be any unitary operator, while the state preparation operator in Grover's algorithm is restricted to generating an equal superposition, as described in equation (20). In our examples, X will describe the Bernoulli random variable and the cumulative loss distribution function used for VaR estimation. Additionally, Grover's algorithm is restricted to identification of correct states x such that $f(x) = 1$ if x is the correct state, while the QAE algorithm can be used to estimate the probability of any state x (Brassard et al., 2002). Similarly to Grover's algorithm, the QAE algorithm also uses the Grover operator, given by:

$$G = XS_0X^\dagger S_{\psi_1} \tag{32}$$

Where X is the state preparation operator, and S_0 and S_{ψ_1} are the reflections about the states $|0\rangle$ and $|\psi_1\rangle$.

The original QAE algorithm relies on Quantum Phase Estimation (QPE) (Dorner et al., 2009), which further increases the complexity of the algorithm. QPE is crucial for QAE because it allows for the extraction of the phase information that corresponds to the eigenvalue of the Grover operator. This phase information is directly related to the amplitude we seek to estimate.

The main drawback of the QAE algorithm proposed by Brassard (Brassard et al., 2002) is the complexity of the algorithm, in terms of the number of qubits and the number of operator applications required to execute the estimation (Suzuki et al., 2020). For this reason, several alternatives to the original QAE algorithm have been proposed, such as Iterative Amplitude Estimation (Grinko et al., 2021), Maximum Likelihood Amplitude Estimation (Suzuki et al., 2020), and Faster Amplitude Estimation (Nakaji, 2020).

2.3.3 Iterative Amplitude Estimation

As stated in the previous section, there has been significant interest in optimizing the QAE algorithm by removing its dependency on QPE. This interest arises from the potential to reduce resource requirements in terms of qubits and gates, thus making QAE more accessible in practical applications, especially on NISQ devices. One promising approach in this direction is Iterative Quantum Amplitude Estimation (IQAE). Unlike its predecessor, IQAE does not rely on QPE but rather combines various techniques in a novel way to achieve better results. IQAE has been proven to offer improved efficiency compared to the original QAE algorithm (Grinko et al., 2021), making it a compelling alternative for practical implementations.

In this thesis, we will utilize IQAE in the empirical sections to estimate the VaR of a portfolio loss because it is expected to decrease the required runtimes while not biasing the results.

2.3.4 Variational quantum eigensolver

The Variational Quantum Eigensolver (VQE) is a hybrid quantum algorithm that is most often used for solving problems in chemistry and physics. The algorithm can also be modified to solve optimization problems, such as portfolio optimization. Being a hybrid algorithm, the VQE uses both quantum and classical resources.

Initially the purpose of the algorithm was to find the lowest energy state (ground state) of a molecule. This problem can be written as the time-independent Schrödinger equation, which is given by:

$$\hat{H}|\psi\rangle = E|\psi\rangle \tag{33}$$

Where $\hat{H}$ is the Hamiltonian operator of the molecule, $|\psi\rangle$ is the wavefuncton of the molecule and E is the energy of the molecule.

Equation (33) is an eigenvalue problem, where the wave function ψ is the eigenfunction and the energy E is the eigenvalue. The objective is to find the lowest eigenvalue of the Hamiltonian operator, which will be the lowest energy state E_0 of a molecule. This can be performed by parametrizing the wave function $|\psi\rangle$ and using a classical optimizer to minimize the energy of the wave function by varying the parameter θ.

$$\min_{\theta} \langle\psi(\theta)|\hat{H}|\psi(\theta)\rangle \tag{34}$$

For problems in chemistry and physics the Hamiltonian $\hat{H}$, would describe the energy of the analyzed system, and it would be a sum of the kinetic and potential energy of the system. For optimization problems, the Hamiltonian can be expressed as the cost function of the quadratic unconstrained binary optimization (QUBO) problem, given by the equation:

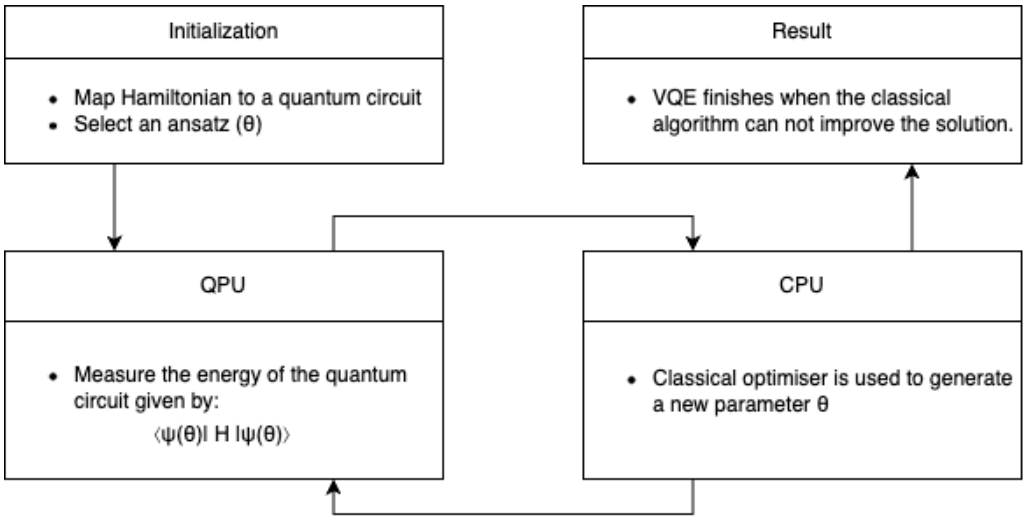
$$f(x) = x^T Q x + c^T x \tag{35}$$

Where Q is a matrix of size $n \times n$, c is a vector of size n, and x is a binary vector of size n representing the solution state, which can be identified using the VQE algorithm that minimizes the cost function above (Barkoutsos et al., 2020).

The operation of the VQE algorithm is summarized in Figure 4 below. The algorithm starts by mapping the QUBO problem to a Hamiltonian quantum circuit and selecting an ansatz[2] θ_0, that parametrizes the initial wave funcition. The quantum circuit is then executed on a quantum computer and the energy of the quantum state is measured. The energy is then used as an input to a classical optimizer, which updates the parameter θ. The process is repeated until the energy of the wave function converges to the lowest possible value. The value of the parameter θ that minimizes the energy of the wave function represents the solution of the optimization problem.

[2]An *ansatz* is a general term for a proposed form or assumption used to simplify or solve complex problems. In quantum computing, it refers to a trial wavefunction or state used as an initial point for approximations or optimizations.

Figure 4: VQE algorithm

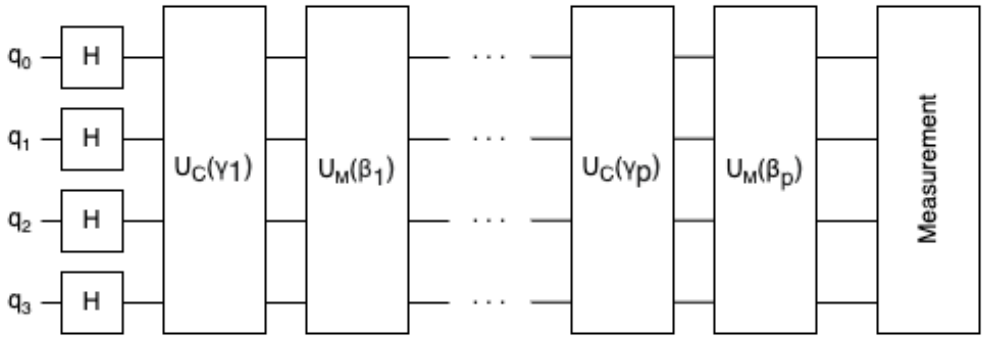


Source: Own work.

2.3.5 Quantum approximate optimization algorithm

Similarly to VQE, the Quantum Approximate Optimization Algorithm (QAOA) is a hybrid quantum algorithm that can be used to solve optimization problems. The algorithm was first described by Fahri et al in 2014 (Farhi et al., 2014) and works by encoding the cost function of an optimization problem into a Hamiltonian cost operator.

Figure 5: QAOA quantum circuit



Source: Own work

Figure 5 above depicts the QAOA quantum circuit. The circuit starts by preparing the initial state of qubits, whose count corresponds to the number of unknown variables in the optimization problem. The initial state is prepared using the Hadamard operation, which creates a superposition of all possible states:

$$|s\rangle = \sum \frac{1}{\sqrt{2^n}}|x\rangle \tag{36}$$

Next the quantum circuit applies the cost operator $U_c(\gamma)$, which is a parametrized operator that encodes the cost function of the optimization problem and is defined

by:

$$U_c(\gamma) = e^{-i\gamma H_c} \tag{37}$$

Where H_c is the cost Hamiltonian of the optimization problem. Next the quantum circuit applies the mixer operator $U_m(\beta)$ defined by:

$$U_m(\beta) = e^{-i\beta H_m} \tag{38}$$

Where H_b is the mixer Hamiltonian. The mixer and cost Hamiltonian circuits are applied alternately p times, where p is the number of layers of the quantum circuit. At the end of the circuit, the state of the qubits is measured, and the result is used as an input to a classical optimizer, which updates the parameters γ and β. The process is repeated until the energy of the measured result converges to the lowest possible value.

2.4 Software development tools for quantum hardware

Quantum computing is at the forefront of technological innovation, and the development of specialized software tools is crucial for harnessing the power of quantum hardware. These tools serve as the bridge between the complex quantum hardware and the practical computing applications.

To program quantum computers, several Software Development Kits (SDKs) have been created, each with unique features tailored to different quantum hardware platforms. Among the most prominent SDKs are Qiskit (Javadi-Abhari et al., 2024), developed by IBM in collaboration with the open source community; PyQuil (Smith et al., 2016), associated with Rigetti Computing; and Microsoft's Quantum Development Kit (Misrosoft, 2024). Qiskit in particular, is renowned for its comprehensive ecosystem and robust community support, which is why it was chosen as the primary SDK for this thesis.

2.4.1 Qiskit

Qiskit, an open-source quantum computing framework, allows users to work with quantum circuits and quantum algorithms across various domains. It is structured into multiple libraries, each targeting a specific area of quantum computing. For instance, Qiskit Nature provides the functional elements for solving quantum mechanical natural science problems, while Qiskit Aer offers powerful simulators for testing and validation. For applications in finance Qiskit Finance is particularly useful as it encapsulates tools and algorithms tailored for financial analysis and optimization. Qiskit Finance offers high-level APIs that abstract the complexity of quantum hardware, enabling users to focus on solving financial problems without delving into the intricacies of quantum algorithms. This abstraction is particularly beneficial for those with a background in finance rather than quantum physics. As mentioned above, Qiskit Aer includes local simulators that allow developers to run and debug quantum circuits on their local machines. These simulators are invaluable for prototyping and can accurately simulate the behavior of quantum hardware

and will be used in the empirical part of this thesis in applications where run-time is not a critical factor.

Complementing Qiskit is the IBM Quantum platform (IBM, 2024), which grants access to a fleet of IBM's quantum computers through the cloud interface. The platform serves as a gateway to not only IBM's quantum processors but also to simulators that can emulate the behavior of quantum hardware. The IBM Quantum platform caters to a wide range of users providing both free and paid access to quantum hardware. For the purposes of this Master's thesis the free access plan was used as it allows users to use both 32 and 127 qubit quantum processors, while paid users also gain access to systems with up to 133 qubits.

In summary, Qiskit and the IBM Quantum platform represent a comprehensive toolkit for anyone researching quantum computing. The choice to utilize Qiskit in this master's thesis is a reflection of its accessibility, versatility, and the substantial support it receives from the quantum computing community.

3 FRAMEWORK AND METHODOLOGY

This section of the thesis explores the role of computational approaches in finance, with a specific emphasis on optimization and Monte Carlo simulations and their applications in Value at Risk (VaR) estimation and portfolio optimization. It will first introduce the core concepts and techniques of these computational methods, and then detail how they are employed to estimate financial risk and optimize asset allocations. The effectiveness of these methods will be evaluated through performance measures such as Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE), providing a basis for comparing classical algorithms with emerging quantum computational approaches. Additionally, this section will outline the experimental setup for these comparisons, discussing the integration of quantum algorithms in solving complex financial models, thus offering insights into the potential advantages and current limitations of quantum computing in finance.

3.1 Computational approaches in finance

In finance and risk management, optimization and Monte Carlo methods are widely used to solve a variety of problems. In this section, we will explore the fundamental concepts behind these computational approaches and how they are used in finance.

3.1.1 Monte Carlo methods

Monte Carlo methods are a class of computational approaches that rely on repeated random sampling to estimate approximate numerical results. Monte Carlo methods are often used to solve both deterministic and probabilistic problems that are

not easily solved analytically. These methods are often used for drawing from an underlying probability distribution, integration and optimization.

Monte Carlo methods differ slightly based on their application, however, they all follow a similar algorithmic structure:

- Determine the underlying probability distribution of the problem;
- Generate the random samples from the probability distribution;
- Use the random samples to estimate the solution to the problem;
- Repeat the process multiple times to improve the accuracy of the solution.

Monte Carlo methods, despite their simple and straightforward design are computationally expensive. To achieve a level of accuracy required for risk modeling a prohibitively large number of samples is required (see Section 3.1.1.1 for information regarding the convergence rate of MC approaches), leading to long computation times and high computation costs. This drawback is mitigated by the fact that Monte Carlo methods fall into the *embarrassingly parallelizable* class of algorithms, as they are extremely easy to parallelize and execute on multiple CPU cores or even GPUs. The parallelization of MC methods also makes the benefits of quantum alternatives less pronounced. The benefits of parallelization of MC methods on both CPUs and GPUs will be explored in Section 4.1.2.

3.1.1.1 Convergence rate

Assume that we want to estimate the expected value of a random variable X denoted as $\mu = E[X]$. We estimate the mean μ by taking the sample mean of N samples from an independent and identically distributed (i.i.d.) random variable X:

$$\hat{\mu} = \frac{1}{N} \sum_{i=1}^{N} X_i \tag{39}$$

The central limit theorem (CLT) states that as the sample size N increases, the distribution of the sample mean $\hat{\mu}$ will approach a normal distribution with mean μ and variance σ^2/N, where σ^2 is the variance of the random variable X.

$$\sqrt{N}(\hat{\mu} - \mu) \to \mathcal{N}(0, \sigma^2) \tag{40}$$

This shows that the Monte Carlo estimator $\hat{\mu}$ is an unbiased estimator of μ and that the standard deviation of the estimator decreases with the sample size N. The standard deviation of a sample mean is also called the standard error of the mean and is given by:

$$SE(\hat{\mu}) = \frac{\sigma}{\sqrt{N}} \tag{41}$$

This shows that the error of the Monte Carlo estimator decreases with the square root of the sample size N. The convergence rate of MC methods is thus $\mathcal{O}(1/\sqrt{N})$.

This implies that to decrease the error of the MC estimator by a factor of 100, we need to increase the sample size by a factor of 10,000, which is a significant drawback of MC and might make it infeasible for some applications. The theoretical convergence rate of MC methods will be empirically tested in Section 4.1.3.

3.1.2 Optimization problems

In finance and risk management, optimization problems are abundant and are used to help minimize risks and costs, and maximize returns. Such problems include optimal allocation of assets in a portfolio, fitting a predictive model to a dataset, and many others. Formally, optimization problems can be defined as minimization or maximization of an objective function, with respect to a set of constraints. This can be written as follows:

$$\begin{aligned} \min \quad & f(x) \\ \text{s.t.} \quad & x \in S \end{aligned} \tag{42}$$

Where $f(x)$ is the objective function, x is the solution vector, and S is the set of the constrained allowable solutions. Optimization problems can be further classified into the following main categories (Cornuejols and Tütüncü, 2006):

- Linear programming problems
- Quadratic programming problems
- Conic optimization problems
- Integer programming problems
- Dynamic programming problems

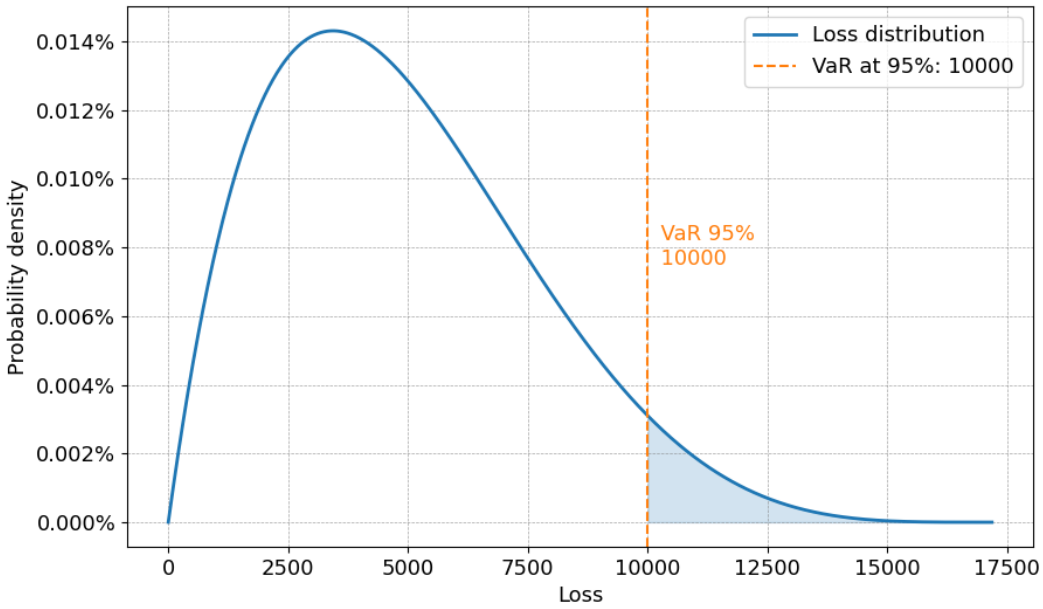
Modern portfolio theory and portfolio optimization (See Section 3.3 for further details), introduced by Harry Markowitz (H. Markowitz, 1952, H. M. Markowitz, 1991) is classified as quadratic programming problems. To solve this problem, Harrry Markowitz introduced a bespoke algorithm, named the "critical line method" (H. M. Markowitz, 1956), which can be used for all quadratic problems and can additionally handle linear constraints in addition to upper and lower bounds on the solution vector.

3.2 Value at risk estimation

Value at Risk (hereinafter VaR) is a statistical measure that quantifies the risk of loss on a specific portfolio of assets. Specifically, given a probability p VaR is defined as the loss that will not be exceeded with a probability of p, while the probability of a loss exceeding VaR is $1 - p$ (Jorion, 2007). It is important to note, that VaR is not a measure of the maximum loss or expected loss, but rather the loss that will not be exceeded with a probability of p. For example, if a given portfolio has a

VaR of 10,000 USD with a probability of 95%, this means that there is a 5% chance that the portfolio loss will be greater than 10,000 USD. See Figure 6, for a graphical representation of the given example. The interpretability and relative simplicity of VaR make it a popular risk measure metric in finance and risk measurement. In financial institutions it is used to determine the amount of capital required to cover for potential losses or to determine the risk that an institution can take on.

Figure 6: Graphical representation of VaR



Source: Own work.

There are multiple approaches to VaR computation. It can be estimated either parametrically in the case of variance-covariance VaR, or nonparametrically using historical simulation and methods like Monte Carlo. A McKinsley report from 2012 (Mehta et al., 2012) indicates that around 15% of surveyed banks use Monte Carlo methods as their primary method for VaR estimation. The report also noted that a typical runtime for VaR estimation using Monte Carlo methods ranges from 2 to 15 hours, and even more in stressed market scenarios. This shows that there is a real market demand for faster VaR estimation methods.

Even though VaR is industry-standard for risk measure, it has some drawbacks. Most notably, VaR tends to underestimate risk (Kuester et al., 2006), and it does not correctly reflect the risk in cases of VaR breach, which gives senior executives and decision makers in financial institutions false confidence and can lead to over-leveraging. Nassim Nicolas Taleb went as far as to propose a general ban on VaR in a congress hearing after the 2008 financial crisis, due to its inability to correctly estimate tail risk (Taleb and Investments, 2009). Nevertheless, VaR is still one of the most widely used risk measures in finance and risk management.

3.2.1 Mathematical model

Let's define a portfolio with n assets with a total loss L. VaR of this portfolio at confidence interval α is defined as the smallest value x (representing the loss), for which the probability of loss exceeding x is at least $(1-\alpha)$. This can be written as:

$$VaR_\alpha(L) = \inf\{x|\mathbb{P}(L \leq x) \geq 1-\alpha\} \tag{43}$$

Where α is the confidence interval, and L is the loss of the portfolio.

3.2.2 VaR estimation on quantum hardware

VaR estimation on a quantum computer is relatively similar to a classical Monte Carlo simulation where we sample an underlying probability distribution. In this master's thesis we will follow an approach described by J.Egger (Egger et al., 2020). In this approach, one must first define a default probability distribution, and encode it into a quantum circuit. The approach employs a variation of the asymptotic single factor (ASRF) model called the Gaussian conditional independence model to model default probability distributions, which is also used under Basel II and Internal Ratings-Based (IRB) frameworks for the regulatory capital estimation (Rutkowski and Tarca, 2015). In the model specification the conditional probability of default (PD) of individual assets is dependent on the realization of the underlying systematic risk factor Y and is given by:

$$PD_i(Y) = \Phi\left(\frac{\Phi^{-1}(PD_i) - \sqrt{\rho_i}Y}{\sqrt{1-\rho_i}}\right) \tag{44}$$

Where Φ is the cumulative distribution function of the standard normal distribution, Φ^{-1} is the inverse cumulative distribution function of the standard normal distribution, PD_i is the unconditional probability of default of the i-th asset, and ρ_i is the correlation between the i-th asset and the systematic risk factor Y.

Using this model we transform the unconditional PD of individual assets into a conditional PD, dependent on a single systematic risk factor. This approach is valuable for Value at Risk (VaR) estimation on quantum hardware because it allows the probability distribution of an entire portfolio to be encoded using a relatively small number of qubits to represent the systematic risk factor. To model the default probabilities of individual assets in a quantum system, we start by performing a single-qubit Y-rotation $R_Y(\theta_k^p)$, which maps each asset's unconditional PD to its corresponding qubit. The matrix representation of this rotation operation is provided in equation (15). The operator that encodes the default rates into a quantum state representing a portfolio of asset defaults is then defined as:

$$U = \bigotimes_{k=1}^{K} R_Y(\theta_k^p) = \bigotimes_{k=1}^{K} \left(\sqrt{1-p_k}|0\rangle + \sqrt{p_k}|1\rangle \right) \tag{45}$$

$$= \left(\sqrt{1-p_1}|0\rangle + \sqrt{p_1}|1\rangle \right) \otimes \left(\sqrt{1-p_2}|0\rangle + \sqrt{p_2}|1\rangle \right) \tag{46}$$

$$\otimes \cdots \otimes \left(\sqrt{1-p_K}|0\rangle + \sqrt{p_K}|1\rangle \right) \tag{47}$$

Where k is the index for assets, p_k is the unconditional probability of default for asset k, and $\otimes$ represents the tensor product.

Next, to account for correlations between default events, we introduce an additional quantum register for the systematic risk factor Z, which is assumed to follow a standard normal distribution. The state of Z is prepared as:

$$|Z\rangle = \sum_{i=0}^{2^{n_z}-1} \sqrt{p_z^i}|z_i\rangle \tag{48}$$

Here, p_Z^i represents the probability of the i-th state of Z, $|z_i\rangle$ denotes the corresponding basis state of the Z register, and n_z represents the number of qubits used to quantize the normal distribution.[3]

This quantum state $|Z\rangle$ is then used to control the rotation angles for the qubits representing individual asset defaults. Combining these components, the final quantum state that encodes both the default probabilities and the correlations due to the risk factor is given by:

$$|\Psi\rangle = \left(\sum_{i=0}^{2^{n_z}-1} \sqrt{p_z^i}|z_i\rangle \right) \otimes \left(\bigotimes_{k=1}^{K} \left(\sqrt{1-p_k(z_i)}|0\rangle + \sqrt{p_k(z_i)}|1\rangle \right) \right) \tag{49}$$

In this final state $|\Psi\rangle$, each qubit k representing an asset is in a state that reflects its conditional default probability $p_k(z_i)$ given the corresponding state $|z_i\rangle$ of the risk factor Z.

In the next step, a circuit is created that performs a weighted summation in order to estimate the total loss of the portfolio. The output of the circuit, representing the total loss is then mapped to a single qubit, on which we can perform the QAE algorithm to estimate the total losses of a portfolio. The circuit is described by the

[3]The number of qubits n_z used for quantizing the normal distribution is determined by the desired accuracy, where more qubits result in a lower quantization error. For this master's thesis, only 2 qubits were used, due to the limitations of the available qubits on the IBM quantum platform where the code was executed.

following equation:

$$|L\rangle_n|0\rangle \mapsto \begin{cases} |L\rangle_n|1\rangle & \text{if} \quad L \leq x \\ |L\rangle_n|0\rangle & \text{if} \quad L > x \end{cases} \tag{50}$$

Where x a threshold loss value. The quantum state after the application of the circuit described by equation (50) is given by:

$$|\psi\rangle = \sum_{L=0}^{x} \sqrt{p_L}|L\rangle_{n_s}|1\rangle + \sum_{L=x+1}^{2^{n_s}-1} \sqrt{p_L}|L\rangle_{n_s}|0\rangle, \tag{51}$$

Where p_L is the probability of the L-th state of the total loss of the portfolio, and n_s is the number of qubits used to encode the total loss of the portfolio.

Finally, the QAE algorithm can be used on the quantum state described by equation (51) to estimate the probability that the portfolio loss will exceed the value of x. To find the VaR of the portfolio a bisection search must be performed to find the value of x for which the probability of loss exceeding x is equal to the confidence interval α. The bisection search is an iterative method that starts with a search interval of $[a, b]$, where the probability of loss exceeding a is less than α, and the probability of loss exceeding b is greater than α. At each iteration, the midpoint m, calculated as the average of a and b, is evaluated using the QAE algorithm to estimate the probability of loss exceeding m. If this probability is less than α, the interval is updated to $[m, b]$; otherwise, it becomes $[a, m]$. This process is repeated until the interval converges to a value of x such that the estimated probability aligns with the confidence interval. The result of the VaR estimation using the aforementioned algorithm will be presented in Section 4.2.1.

3.2.3 VaR estimation using Monte Carlo methods

There exist a variety of approaches to VaR estimation using the MC methods. In this master's thesis we will use the MC method to generate random draws from a probability distribution of the total loss of a portfolio. Similar to in the quantum approach, we will use the Gaussian conditional independence model to model the conditional probability of default of individual assets. The classical approach will not require the approximations needed for implementation on a quantum computer. Additionally, the conditional probability of default will not be directly embedded into a quantum bit. To model a default we will use the following relation:

$$D_i = \mathcal{W}_i < \Phi^{-1}(PD_i) \tag{52}$$

Where D_i is the default event of the i-th firm, W_i is the random variable representing the variability of the i-th firms assets, PD_i is the unconditional probability of default of the i-th firm, and Φ^{-1} is the inverse standard normal distribution function.

The random variable W_i is represented by the following equation:

$$W_i = \sqrt{\rho}Y + \sqrt{1-\rho}Z_i \tag{53}$$

Where random variables Z_i and Y are standard normal and independent, and ρ is the market correlation coefficient. The variable Y represents the systematic risk that is common to all obligors, while Z_i represents the idiosyncratic risk of the i-th obligor. This definition of W is also employed in the IRB approach to capital requirement calculation and is taken from Vasicek (Vasicek, 2002). Finally, total loss of the portfolio is calculated as the sum of the losses of individual assets.

$$LGD = \sum_{i=1}^{K} LGD_i * f(Z_i < \Phi^{-1}(PD_i)) \tag{54}$$

Where L_i is the loss given default of the i-th asset, and K is the number of assets in the portfolio and $f(Z_i < \Phi^{-1}(PD_i))$ is the function representing default of individual asstes.

The simulation will then be repeated a large number of times to estimate the probability distribution of the total loss of the portfolio. To estimate the VaR, the loss distribution will be sorted, and the VaR will be estimated as the loss value that will not be exceeded with a probability of $1-\alpha$. The result of the VaR estimation using the MC method will be presented in Section 4.2.2. The selected approach to VaR estimation using MC methods is similar to the one used in the quantum approach, which will allow for a direct comparison of the convergence rates of the two methods. Additionally, it will allow for an analysis of the computational error, introduced in the quantum approach due to the approximation of a random distribution with a limited number of qubits and the linear approximation of the conditional probability of default.

3.3 Portfolio optimization

Portfolio optimization, also known as Mean-Variance analysis, is the process of selecting optimal the allocation of an investor's assets between all available investment vehicles. The objective is usually to maximize profits, minimizing the variance of returns and minimizing risk, while taking into account the individual investor's risk appetite. Using portfolio optimization an investor can thus choose between increasing the risk of their portfolio if they want to increase their expected returns, or decrease it in order to minimize risk. Even after more than 70 years since the introduction of the modern portfolio theory by Harry Markowitz (H. Markowitz, 1952, H. M. Markowitz, 1991), portfolio optimization is still one of the most widely used tools in finance and risk management. It is popular due to its simplicity, intuitive appeal, and the fact that it maximizes investors' expected utility in a wide variety of settings (Levy and Markowitz, 1979).

Modern portfolio theory (MPL) and portfolio optimization are often criticized. One of the earliest drawbacks pointed out by the critics is the assumption of normal distribution of the portfolio returns (Low et al., 2016). Even though returns often have fat tails and have varying volatility, the normal distribution is still widely used. Nonetheless, researchers have shown better results when using alternative assumptions for the distribution of returns (Doganoglu et al., 2007). Another drawback of

the MPL is the fact that both risk and return are statistical measures of the past, and are thus not guaranteed to be the same in the future. Usually, historical data is used to estimate both the expected returns and the covariance matrix, which can lead to inaccuracies, due to unforeseeable changes in the investment landscape (Low et al., 2016). Additionally, in portfolio optimization both downside and upside volatility are treated equally and are both considered as risk. This aspect is often criticized as investors do not have a linear loss aversion function and are usually more sensitive to downside volatility than upside potential. Finally, to solve the portfolio optimization problem one needs to minimize the expression given in equation (59). This is fundamentally a quadratic programming problem, which makes it computationally expensive to solve, especially for large portfolios. To aid with the computation Harry Markowitz, proposed the critical line algorithm (H. M. Markowitz, 1956), that can handle portfolio optimization problems and its constraints.

3.3.1 Mathematical model

As described in Section 3.3, portfolio optimization ensures the maximization of expected returns while minimizing risk. The expected returns are defined as the exposure-weighted returns of the chosen portfolio. The relation is defined as follows:

$$E(\mu) = \sum_{i=1}^{n} w_i E(\mu_i) \tag{55}$$

Where $E(\mu)$ is the expected return of the portfolio, w_i is the weight of the i-th asset in the portfolio, and $E(\mu_i)$ is the expected return of the i-th asset. Note that the weight of each asset must sum up to 1. The variance of portfolio returns is used as a measure of risk in portfolio optimization. The variance of a portfolio is given by:

$$\sigma_p^2 = \sum_{i=1}^{n} w_i^2 \sigma_i^2 + \sum_{i=1}^{n} \sum_{j \neq i}^{n} w_i w_j \sigma_i \sigma_j \rho_{ij} \tag{56}$$

Where σ_p^2 is the variance of the portfolio, w_i and w_j are the weights of the i-th and j-th asset in the portfolio, ρ_{ij} is the correlation coefficient of returns of the i-th and j-th asset, and σ_i is the standard deviation of the returns for asset i. The problem can also be written in matrix form, where the expected returns are given by:

$$E(\mu) = w^T E(\mu) \tag{57}$$

where w is the weight vector of the portfolio, and $E(\mu)$ is the vector of expected returns for each asset. The variance of the portfolio can be written as:

$$\sigma_p^2 = w^T \Sigma w \tag{58}$$

where Σ is the covariance matrix of the portfolio returns and w is the weight vector of the portfolio. To find the optimal portfolio allocation given the risk aversion q we can write the following optimization problem:

$$\begin{aligned} \min \quad & w^T \Sigma w - q \times w^T E(\mu) \\ \text{s.t.} \quad & \sum_{i=1}^{n} w_i = 1 \end{aligned} \tag{59}$$

Where q is the risk aversion parameter, Σ is the covariance matrix of the portfolio returns, $E(\mu)$ is the expected portfolio return, and w are the portfolio allocation weights. Alternatively we can write the portfolio optimization problem where the goal is to minimize risk for a target return μ_{target} as (Lai et al., 2011):

$$\begin{aligned} \min \quad & w^T \Sigma w \\ \text{s.t.} \quad & w^T E(\mu) = \mu_{target} \\ & \sum_{i=1}^{n} w_i = 1 \end{aligned} \tag{60}$$

3.3.2 QUBO formulation of the portfolio optimization problem

Due to the current limitations of quantum algorithms and quantum hardware (see Section 2.3.4), it is not feasible to solve the portfolio optimization problem as a standard quadratic programming problem. However, the portfolio optimization problem can be formulated as a QUBO problem (Glover et al., 2018), which is compatible with quantum hardware. The QUBO formulation of the portfolio optimization problem is similar to the standard formulation defined in equation (59), with the exception that the weight vector w is replaced with a binary vector x, where $x_i = 1$ if the i-th asset is included in the portfolio, and $x_i = 0$ if the i-th asset is not included in the portfolio. The QUBO formulation of the portfolio optimization problem is given by the following equation:

$$\begin{aligned} \min \quad & w^T \Sigma w - q \times w^T E(\mu) \\ \text{s.t.} \quad & x \in \{0,1\}^n \ \wedge \ \sum_{i=1}^{n} x_i = 1 \end{aligned} \tag{61}$$

3.4 Quantum algorithm benchmarking

Benchmarking quantum algorithms is an important task in the development and research of quantum computers. As a part of this master's thesis we will explore multiple dimensions of benchmarking, including time complexity, convergence and quantum hardware requirements such as the number of qubits required for operation. Additionally, we will explore how benchmarking of classical and quantum algorithms differs and the challenges of benchmarking quantum algorithms.

3.4.1 Convergence rate

The convergence rate is a measure of the rate at which a given algorithm converges to the optimal solution. It is a crucial factor in determining the performance of an algorithm, and it can be used to preemptively determine the number of iterations

required to reach a solution with a desired level of confidence. Usually the convergence rate of an algorithm is defined as the rate at which the error decreases as the number of iterations increases. This can be expressed as:

$$\lim_{k\to\infty} \frac{\epsilon_{k+1}}{\epsilon_k^q} < C, \tag{62}$$

Where ϵ_k is the error at the k-th iteration, q is the convergence rate, and C is a constant, which must be lower than 1 for the algorithm to converge. A q value of 1 represents linear convergence, while a value of 2 represents quadratic convergence (Nelson, 2020).

To evaluate the error rate convergence of the algorithms tested and to determine how close the solution is to the optimal solution we will use the following performance measures:

- Root mean squared error (RMSE)
- Mean absolute error (MAE)

In the context of these performance measures, a lower value corresponds to a higher accuracy in the solution.

Root mean squared error The Root Mean Squared Error (RMSE) is one of the most widely used measures of error of a model in predicting data. It is a measure of the difference between the predicted value and the actual value. It is given by the following equation:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{63}$$

Where n is the number of observations, y_i is the observed value for the i-th observation and $\hat{y}_i$ is the predicted value for the i-th observation. RMSE has the following properties:

- RMSE gives larger weights to large errors. This means that RMSE should be used in applications where large deviations should be penalized more than smaller deviations. This property also makes RMSE more sensitive to outliers (Pontius et al., 2008).
- The unit of RMSE is different from the unit of the original variable, which makes the interpretation of the RMSE value difficult.
- RMSE is scale dependent, which means that it is not comparable across datasets (Hyndman and Koehler, 2006).

Mean absolute error The Mean Absolute Error (MAE) is another common error measure. Unlike RMSE it measures the simple average of errors on a sequence of predictions (Willmott and Matsuura, 2005). It is given by the following equation:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{64}$$

Where n is the number of observations, y_i is the observed value for the i-th observation and $\hat{y}_i$ is the predicted value for the i-th observation. MAE has the following properties:

- It is easily interpreted as it is in the same unit as the original variable, and it directly represents the magnitude of error.
- It is less sensitive to outliers than RMSE as errors are not squared.

3.4.2 Convergence estimation

This master's thesis involves analysing the convergence rates of the quantum and classical algorithms applied. Empirical convergence rates will be estimated by running multiple simulations of the algorithm and observing the rate at which the RMSE and MAE of the solution decrease. The results of these analyses are presented in Sections 4.1.3, 4.2.3 and 4.3. To empirically estimate the convergence rate of quantum and classical algorithms and compare them to the theoretical convergence rates, we will use a simple regression analysis. The analysis will provide evidence on the relationship between the dependent variable, which in this case is the error rate, and the independent variable, which in this case is the number of iterations or oracle applications. The linear regression model form that will be used is defined below:

$$y = \beta_0 + \beta_1 x + \epsilon \tag{65}$$

Where y is the dependent variable, x is the independent variable, β_1 is the slope, β_0 is the intercept, and ϵ is the error term. The coefficient β_1 representing the slope of the regression line will be used in the convergence rate analysis as an estimator of the convergence rate of the algorithm.

3.4.3 Computation speed

The computation speed of any algorithm plays a significant role in determining its practicality and usability in real-world applications. Computation speed here refers to the time required by an algorithm to solve a given problem. Quantum algorithms promise quadratic, and sometimes even exponential speedups in terms of error rate convergence. However, the time it takes a quantum algorithm reaches the optimal solution can be significantly longer due to the overhead of the NISQ quantum hardware, which can cause a quantum algorithm to arrive to the optimal solution even slower than classical algorithms.

As a part of this master's thesis we will explore the real-world computation speed of quantum algorithms and compare them to classical algorithms. By doing this we will determine whether the speed-up of error rate convergence of the quantum algorithms is enough to offset the overhead of the quantum hardware and make them a viable option for solving real-world problems (see Sections 4.1, 4.2, and 4.3 for the analysis of algorithms used in VaR estimation, and portfolio optimization).

3.4.4 Quantum hardware requirements

The use of quantum hardware in finance is heavily dependent on the capability of quantum hardware, specifically on the number of available qubits and their fidelity. Solving real-world problems in finance requires a significant qubit count, which makes benchmarking these requirements crucial to gauge when quantum computers will be a viable option for solving real-world problems in finance.

Current quantum hardware offers qubit counts too low for industrial scale application of quantum algorithms such as Grover's algorithm and the QAOA algorithms that will be applied in the empirical part of this master's thesis. Benchmarking will thus involve analyzing the qubit count requirements of the quantum algorithms used in relation to the problem size. Additionally, we will look at the Moore's law, which predicts an exponential increase in hardware capabilities (Schaller, 1997), and the current trends of quantum hardware developments to predict a hardware development timeline for the future. By combining the hardware development timeline and the hardware requirements of quantum algorithms, we will try to predict a rough timeline of when quantum computers will be able to solve real-world problems in finance.

4 EMPIRICAL ANALYSIS AND RESULTS

In this section, we conduct a comprehensive empirical analysis to evaluate the performance of quantum algorithms in various financial applications. The section is structured as follows: first, we explore the performance of the Quantum Amplitude Estimation (QAE) algorithm in estimating the probability of a Bernoulli random variable. Next, we estimate Value at Risk (VaR) using both QAE and MC approaches. Additionally, we perform a comprehensive performance analysis of the Variational Quantum Eigensolver (VQE) and Quantum Approximate Optimization Algorithm (QAOA) for portfolio optimization. Finally, we examine the hardware requirements necessary for implementing these quantum algorithms and provide a future outlook on quantum computing in finance.

4.1 Bernoulli random variable probability estimation

In this section, we will evaluate the performance of the QAE algorithm in estimating the probability of a Bernoulli random variable. We will compare its performance to the Monte Carlo (MC) simulation in terms of convergence rates, computation speed, and quantum hardware requirements. The evaluation of QAE in estimating the Value at Risk (VaR) of a portfolio of assets will be addressed in the next section.

We begin by testing QAE and MC on the problem of estimating the probability p of a Bernoulli random variable, a discrete variable with two outcomes (0 or 1), where p is the probability of the variable taking the value 1. Below is a mathematical description of the Bernoulli random variable, with X denoting the random variable:

$$X(p) = \begin{cases} 1, & \text{with probability } p \\ 0, & \text{with probability } 1-p \end{cases} \tag{66}$$

To test the QAE algorithm we first encode the Bernoulli random variable into a quantum state. This is done by applying a rotation operator $R_Y(\theta)$ (see Section 2.1.4.1) to a $|0\rangle$ state. The θ parameter is computed as:

$$\theta = 2 \arcsin \sqrt{p} \tag{67}$$

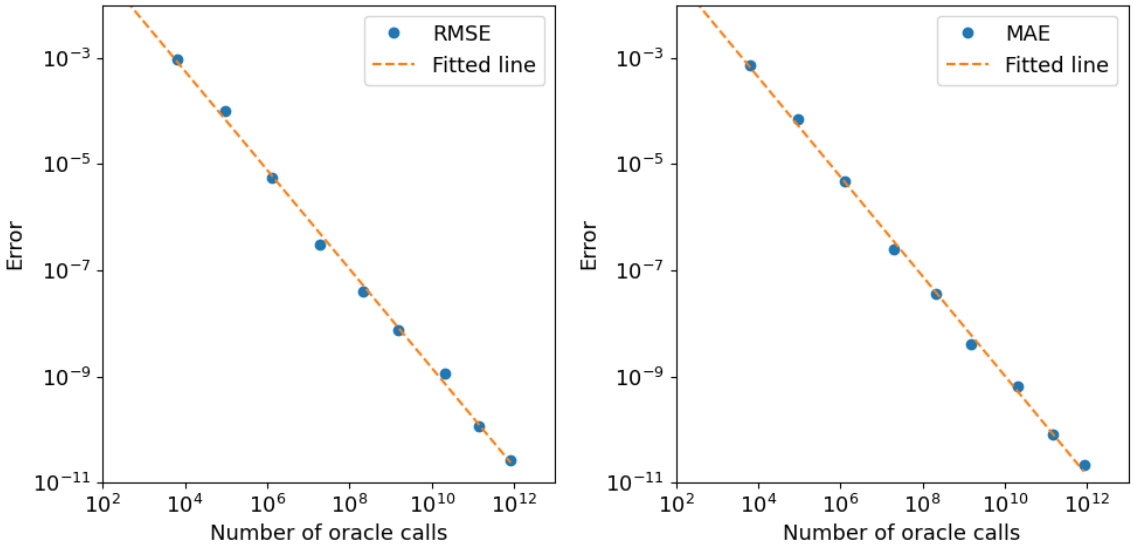
Where p is the probability of the Bernoulli random variable. Next, we apply the oracle operation U_ω N times to the state representing the Bernoulli random variable. When analyzing the results, we will show how the number of oracle applications N affects the accuracy of the QAE algorithm. A similar exercise will be performed for the MC simulation. The MC simulation will be run N times, where N is the number of simulation runs. In each simulation we will sample from a Bernoulli random variable with probability p. The p estimate will be computed as a simple average of the simulation outcomes.

To test the runtime and efficiency of the execution we will also time the execution of both algorithms. The execution time will be measured and plotted against the number of oracle applications N and the number of MC simulation runs N. This result will provide insight into the real-world benefits of quantum computers.

4.1.1 QAE estimation results

The Figure 7 below presents the results of the estimation of the underlying probability p of a Bernoulli variable using the QAE algorithm. For the analysis a p value of 0.5 was used. To balance the computation time required and the accuracy of MAE and RMSE, the estimation was performed 10 times for each number of oracle calls N. The y-axis of the figure represents the error rate, which is on the left plot represented by the RMSE and on the right plot by the MAE. On the x-axis in both plots is the number of oracle calls N. In both plots we can observe that the error decreases linearly and monotonically with the number of oracle calls N.

Figure 7: MAE and RMSE analysis



Source: Own work.

The results of the analysis presented in Figure 7 are further confirmed by the regression analysis of the results. The results are presented in the Table 1 below and show that the slope of the regression line is -0.93 for RMSE and -0.94 for MAE. This confirms that the convergence rate of the algorithm is close to $\mathcal{O}(1/N)$. Additionally, a good fit of the regression line confirms that the error rate decreases linearly with the number of oracle calls N.

Table 1: QAE regression analysis results

Variable	RMSE	MAE
Intercept	0.4763	0.3922
Slope	-0.9314	-0.9381
R-squared	0.9977	0.9984

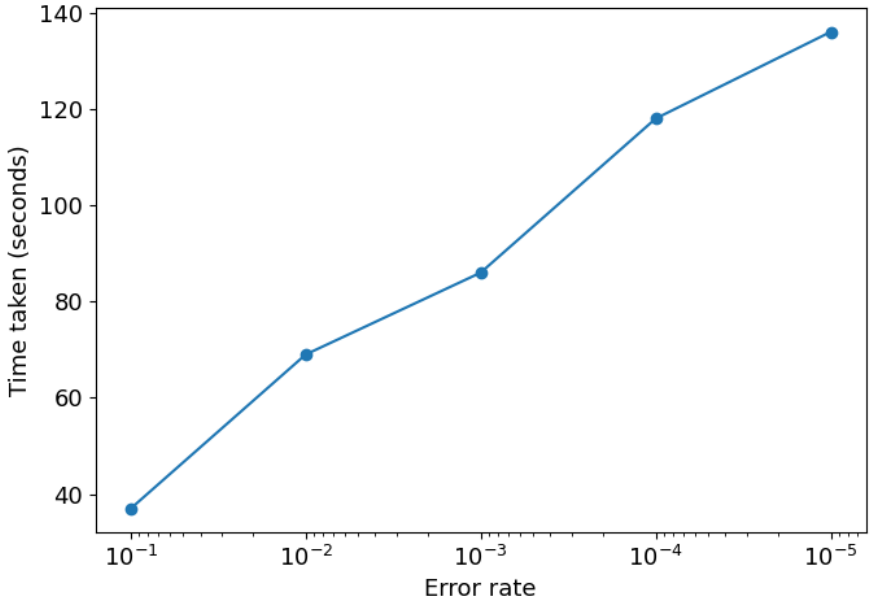
Source: Own work.

To analyze the real-world performance and time requirements of the QAE algorithm we will compute time required to achieve a given error rate. In order to get a real-world performance estimate, which takes into account all drawbacks of current quantum hardware, the IBM Eagle r3 quantum processor was used to execute the QAE algorithm. The algorithm was executed only 4 times with different error rate targets due to limited availability of the processor.

The results of the analysis are presented in the Figure 8 below. The figure presents the relationship between the error rate and the runtime of the algorithm. Note that the error rate is presented on a logarithmic scale while the runtime is presented on a linear scale. The results show that the error rate decreases exponentially with the runtime of the algorithm (i.e. to reduce the error by a factor of 10 the runtime must be increased by n seconds irrespective of the error rate). These results also indicate that the runtime of the algorithm is not linearly dependent on the number of oracle

calls, which is probably caused by the overhead required to perform computation on quantum hardware.

Figure 8: Runtime analysis



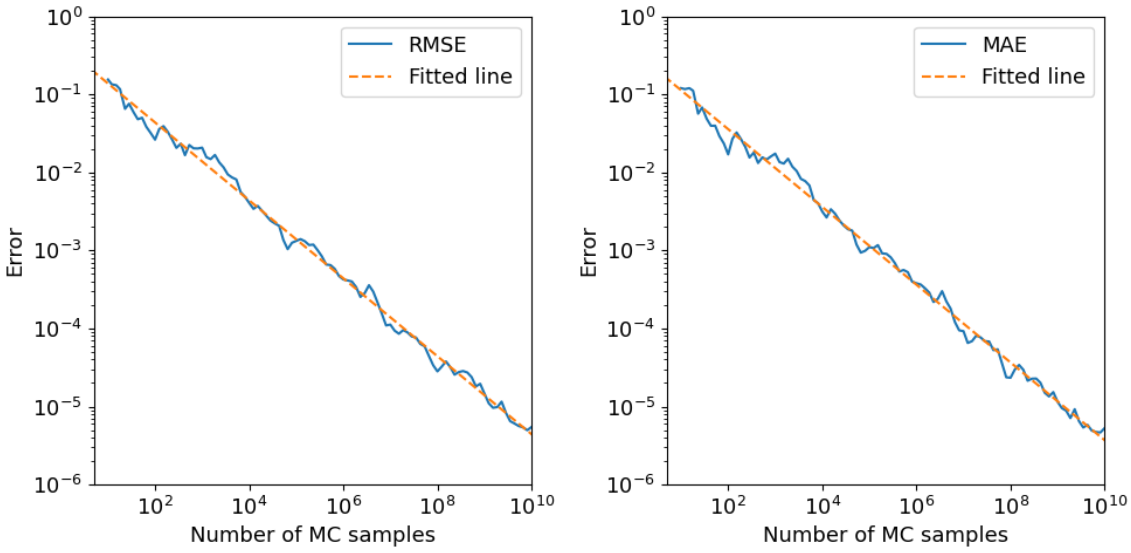
Source: Own work.

4.1.2 MC simulation results

To analyze the MC simulation, a similar setup was used as in the QAE analysis, where the underlying probability p of a Bernoulli random variable is being estimated. For the test a p value of 0.5 was used. The estimation of p was performed by simulating the Bernoulli random variable N times, where N is the number of MC simulation runs. The p estimate was computed as a simple average of the simulation outcomes. The whole process was executed 10 times in order to estimate RMSE and MAE with an adequate degree of accuracy, while limiting the execution time.

The results of the analysis are presented in Figure 9 below. Error rate is presented on the y-axis, while the number of simulation samples is presented on the x-axis. From the figure we can observe a linear relation between the error rate and the number of simulation samples.

Figure 9: MAE and RMSE analysis



Source: Own work.

Table 2 below presents the results of the regression analysis of the MC simulation error rate. The results confirm the linear relationship between the error and number of simulation iterations and show that the convergence rate closely matches the theoretical, as the slope of the regression line is approximately -0.50 for both RMSE and MAE.

Table 2: MC regression analysis results

Variable	RMSE	MAE
Intercept	-0.3615	-0.4499
Slope	-0.4993	-0.4981
R-squared	0.9952	0.9940

Source: Own work.

Next, we will perform a comprehensive performance analysis of four different implementations of the MC simulation to determine the underlying p vale of a Bernoulli random variable. The implementations include a single-threaded CPU implementation, a multithreaded CPU implementation, and two GPU accelerated implementations, which will be discussed further below. The analysis aims to investigate the performance of the different implementations and show scalability of such a problem on different hardware. To enable a fair comparison of the algorithm performance all analysis will be performed on an Apple MacBook Air with a M2 CPU with 8-core CPU, 10-core GPU, and 8GB of RAM. The programs running on the CPU will be implemented in C++ and will use the standard thread library to enable multithreading. The GPU accelerated implementations will be implemented using both C++ and Apple's Metal C++ interface (Apple, 2022), which allows you to execute code on the GPU.

The Figure 10 below shows a high-level overview of the MC simulation that the

aforementioned implementations will be running. The code executes a simple Monte Carlo simulation, where in each iteration a Bernoulli random variable is generated, and its value is added to a cumulative sum. To get the final estimate the cumulative sum is divided by the number of iterations. The algorithm is executed multiple times to generate accurate error rates and execution times.

Both the CPU and GPU implementations follow the same high-level approach there are however slight differences that are important to note. The single threaded and multithreaded implementation differ only in the number of threads used to execute the RunBatch procedure (see Figure 10). The GPU implementations differ from the CPU implementation in the way the Bernoulli random variables are generated. The CPU implementation uses the standard C++ library to generate the random variables, while the GPU implementation uses a custom implementation of the Mersenne Twister pseudo-random number generator[4] developed by Matsumo and Nishimura (Matsumoto and Nishimura, 1998). Lastly, the GPU implementations differ in the way the random variables are summed up. One implementation uses the GPU to both generate and sum up the random variables, while the other implementation uses the GPU only to generate the random variables, while the summation is performed on the CPU.

Figure 10: High-Level Monte Carlo Simulation

```
1: procedure RunBatch(simulationRepetitions)
2:     while simulationRepetitions > 0 do
3:         Generate N random Bernoulli variables
4:         cumulativeSum ← cumulativeSum + sum of N Bernoulli variables
5:         simulationRepetitions ← simulationRepetitions − N
6:     end while
7: end procedure
8: procedure MonteCarlo
9:     for MCIterations ∈ {1, 10, 100, ..., N} do
10:        Initialize simulation
11:        for simulationRepetitions ∈ {1, 2, 3, ..., M} do
12:            startTime ← Current time
13:            Launch threads to execute RunBatch(simulationRepetitions)
14:            Wait for all threads to complete
15:            stopTime ← Current time
16:            runTime ← stopTime − startTime
17:            Store runTime and other results
18:        end for
19:        Compute RMSE and MAE
20:    end for
21: end procedure
```

Source: Own work.

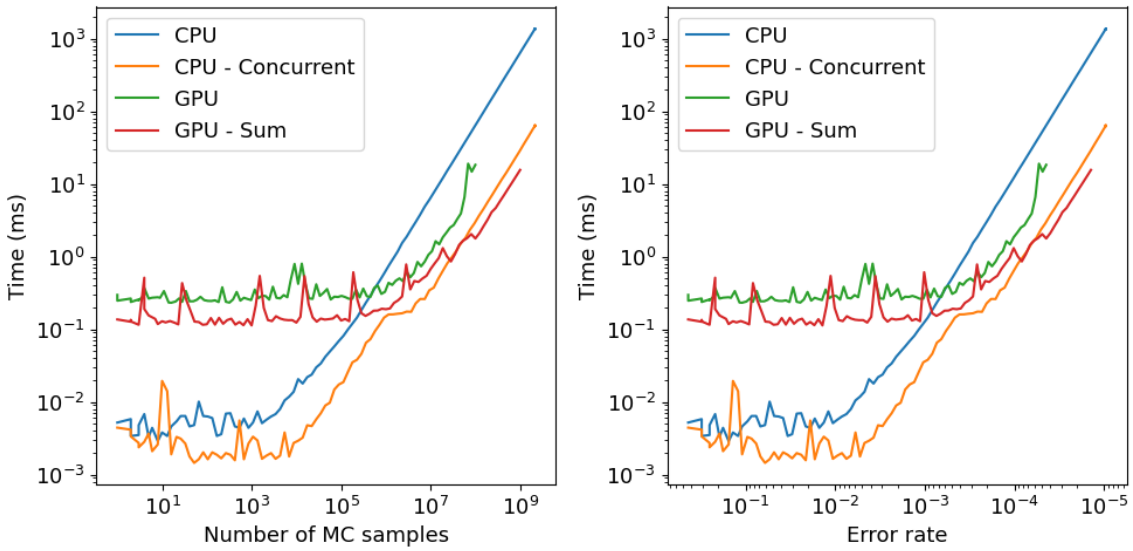
The results of the analysis are presented in Figure 11 below. The y-axis represents the execution time in ms, while the x-axis on the left graph represents the number of MC iterations, while on the right graph it represents the RMSE. First, we can

[4]The code can be found at https://github.com/gtrebse/Mersenne-Twister-metal

observe that for all implementations, the execution time remains relatively constant up to a threshold number of iterations. This behavior can be attributed to the overhead related to the initiation of computational resources, such as the launching of threads and GPU kernels, memory allocation, and the associated data reads and writes. Second, we can observe that the overhead for the GPU implementations is significantly higher when compared to the CPU versions. The disparity is most apparent up to approximately 10^5 iterations, where the execution time for the GPU implementations can be almost 100 times slower. This suggests that the setup costs for GPU computation are more substantial due to the process of initializing, synchronizing, and managing a large number of parallel threads.

With respect to the CPU-based implementations, the concurrent version running on the 8-core CPU outperformed the single-threaded variant by a significant margin. The observed speedup factor is approximately 6x. When comparing the two GPU implementations, we can observe that the version performing the summation directly on the GPU was significantly faster than its counterpart, which offloaded the summation to the CPU. The performance improvement can be largely ascribed to the reduction in memory reads and writes, which are known to be expensive operations on the GPU. By minimizing the data transferred between the GPU and CPU and allowing the GPU to handle both computation and summation, we achieved a more efficient use of GPU's capabilities. It is important to note that the more efficient GPU implementation shows a performance improvement only when the number of iterations exceeds approximately 10^8. This finding aligns with the findings of Gebraad (Gebraad and Fichtner, 2023), which indicate that GPUs tend to outperform CPUs only in scenarios where a large number of operations are required.

Figure 11: Runtime analysis



Source: Own work.

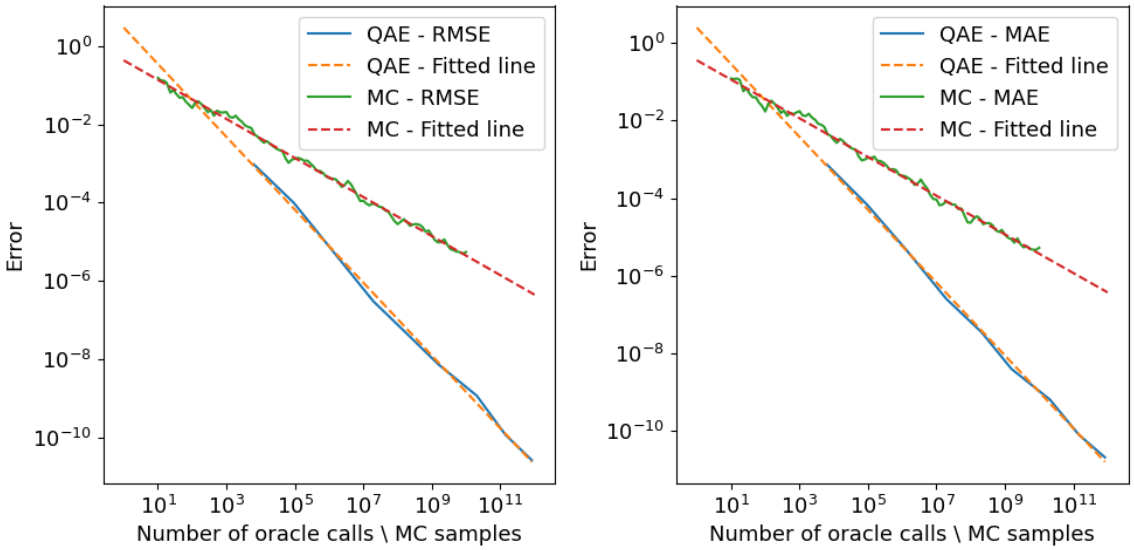
In summary, the performance analysis revealed that while both the CPU and GPU implementations have their respective overheads, the advantages of parallel processing become increasingly evident with larger workloads. The GPU, despite its higher initial overhead, provides a performance increase in computationally intensive tasks that require many iterations. However, it is important to note that the development

of GPU accelerated code is significantly more complex than the development of CPU code. Moreover, the GPU code is more difficult to debug and optimize, which makes it in many cases a less attractive option, compared to the CPU implementation.

4.1.3 Comparison of QAE and MC results

In this section we will compare the results of the QAE and classical MC simulations. The comparison will be based on convergence rate and computation speed. Finally, we will also give an estimate on the future use cases of quantum computers for such simulations. In the Figure 12 below the results of the convergence analysis are presented. The results indicate that the QAE algorithm converges significantly faster than the MC simulation with respect to the number of oracle calls and MC samples performed. This statement holds even for relatively small number of oracle calls and MC samples. Only below approximately 100 executions the MC simulation outperforms the QAE algorithm. These results show that quantum supremacy is possible, if we take in consideration all the drawbacks of current quantum hardware.

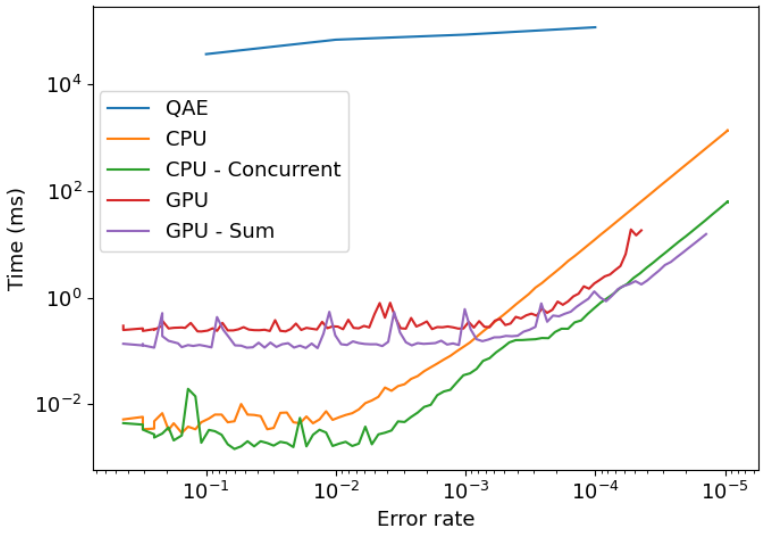
Figure 12: QAE and MC convergence comparison



Source: Own work.

As mentioned before, the convergence rate cannot give us a perfect insight into the real-world performance of an algorithm. For this reason we compare the actual execution times of the QAE and MC simulations. The comparison is shown in the Figure 13 below, where the x-axis represents the error rate, while the y-axis presents the execution time in ms. The results clearly show that the QAE algorithm performance is not yet comparable to the MC simulation. The execution time of the QAE algorithm is more than 1000 times slower for the same error rate for target error rates below 10^3. While this difference is lower for higher error rates, the execution time of the QAE algorithm is still significantly slower.

Figure 13: Execution time comparison



Source: Own work.

These results highlight the current state of quantum hardware and the significant challenges that must be addressed before quantum computers can be effectively used in real-world applications. To illustrate the substantial gap in performance, we can compare the metric used to measure execution speed. IBM uses Circuit Layer Operations Per Second (CLOPS) (IBM, 2022) to indicate how many operations can be performed per second on a quantum processor. For instance, the IBM Eagle r3 processor, which was used to execute the QAE algorithm, can perform 5000 CLOPS. In contrast, the Apple M2 processor operates at 3.49 GHz, which is significantly higher. This comparison, though rough and not accounting for factors like the number of qubits and instructions per second (IPS), underscores the need for substantial improvements in the execution speed of quantum hardware before it can match the performance of classical processors.

4.2 Value at risk estimation

In this section we will show the results of the VaR estimation using QAE and MC simulations, following the procedures described in Section 3.2. The results will then be compared in terms of convergence rate and accuracy of the VaR estimation (see Section 4.2.3).

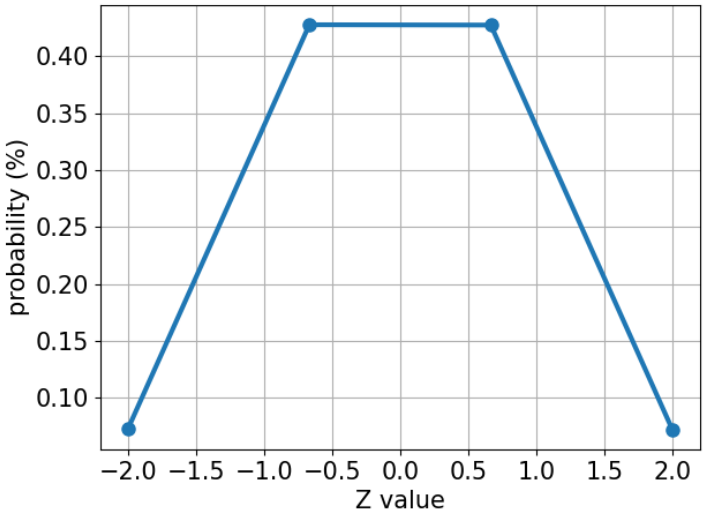
4.2.1 QAE VaR estimation

As discussed in Section 3.2.2, the first step in VaR estimation is the definition of a quantum circuit that models the default events of individual assets in the portfolio. In this example, we will use a sample portfolio of 10 assets. The selected default probabilities, correlation factors, and LGDs of the portfolio are presented in the appendix section 3. To ensure the accuracy of the circuit, we sampled its individual components 1 million times. These samples were then used to determine

the quantized normal distribution, asset default rates, portfolio losses and the VaR. The results are detailed in the subsequent paragraphs.

In Figure 14 below the first output of the circuit is presented. This is the quantized normal distribution, representing the systematic risk factor used to model the conditional probabilities of the default events. In the figure we can observe that the output values are correctly quantized to 4 values. The quantization error is relatively high due to the decision use only 4 quantization levels. Note that, this decision was necessary due to the limited availability of qubits.

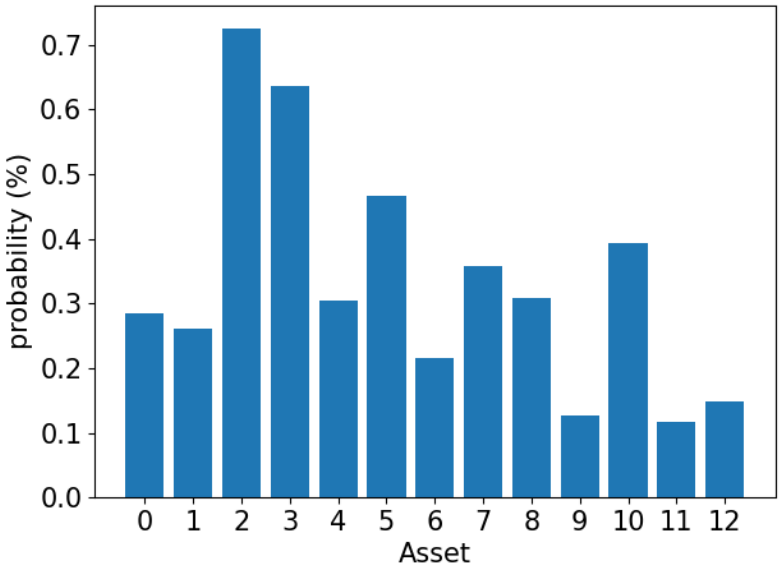
Figure 14: Sampled quantized normal distribution



Source: Own work.

The systematic risk factor is then utilized in the subsequent element of the circuit to model the conditional probabilities of assets within the portfolio. These conditional probabilities are embedded into quantum states, which were directly measured to determine individual default events. The results of the simulation are presented in Figure 15 below. The x-axis represents individual assets that comprise the portfolio, while the y-axis indicates the average default rates of the assets, as estimated using the sample measured in the simulation. Comparing the estimated default probabilities and the underlying unconditional probabilities that describe asset behaviour (see Appendix 3) we can observe that the estimated default probabilities are close to the underlying default probabilities. This indicates that the circuit correctly models the expected default probability.

Figure 15: Sampled default rates

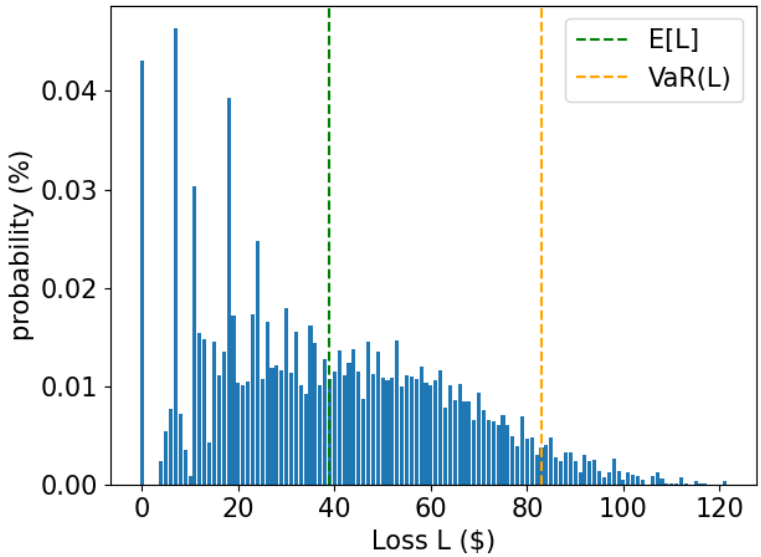


Source: Own work.

Using the default outcomes sampled from the quantum circuit and the LGDs of the assets, we generate the portfolio loss distribution, presented in Figure 16 below. The x-axis shows the portfolio loss values, while the y-axis presents the corresponding frequencies of losses. The expected loss and VaR at the 5th percentile are also marked on the graph. We can observe that the expected loss is slightly below 40 EUR, while the estimated VaR is 83 EUR.

Note that although the simulation of the quantum circuit provides a method for calculating VaR using the generated loss distribution, this approach does not offer any improvements to the convergence rate, as it is functionally similar to a MC simulation.

Figure 16: Sampled portfolio loss distribution

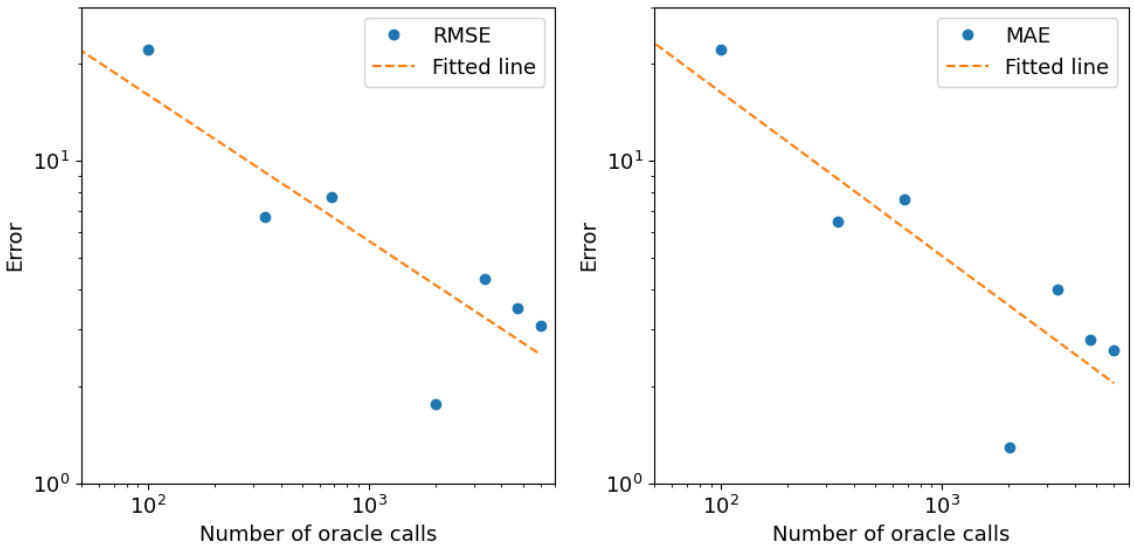


Source: Own work.

After confirming that the circuit is performing as specified, the next step is to use the circuit in combination with the QAE algorithm and the bisection search to find the VaR of the portfolio. VaR estimation was performed 10 times, where in each iteration the LGD, ρ and unconditional PD for all assets were randomly stressed by +/- 50%. The parameters were modified for a more robust estimation of the convergence rate of VaR estimation, since the bisection search always converges to the solution on the same path if the parameters are not changed. Finally, the RMSE and MAE were estimated at each step of the bisection search. The results are presented in Figure 17 below.

The results of the analysis are presented in the Figure 17 below. In the graph we can observe a clear downwards trend in the error rate with the increase in the number of oracle queries. We can also observe that the error rate decrease is not stable, which indicates that the to ensure that the VaR estimation is correct one would have to perform a large number of oracle queries, in order to ensure with high confidence that the VaR estimation is correct.

Figure 17: QAE VaR convergence rate



Source: Own work.

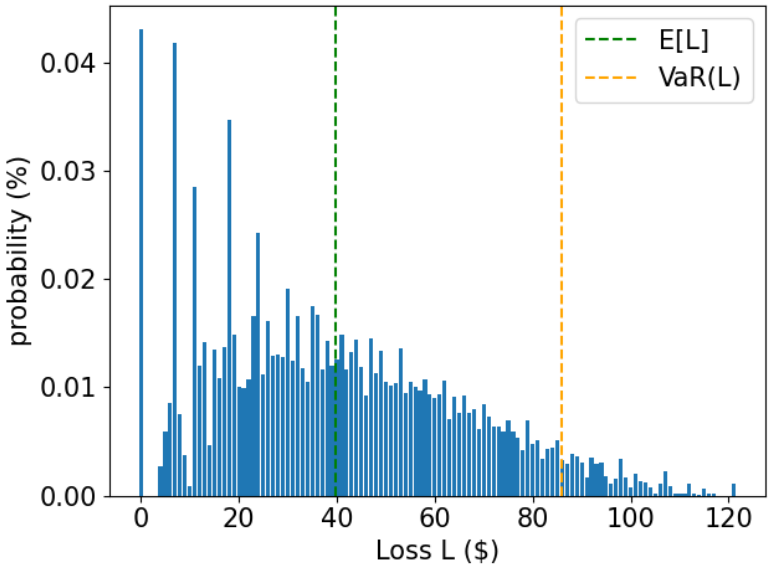
4.2.2 MC VaR estimation

In this section, we present the empirical results obtained from Monte Carlo simulation process described in Section 3.2.3. The primary focus of the analysis is to examine the simulated loss distribution function and to analyze the convergence behavior of the VaR estimation.

The loss distribution of the portfolio, produced by the MC simulation, is illustrated in Figure 18 below. The x-axis represents the possible loss values in EUR, while the y-axis represents the frequencies at which the losses occur. The shape of the loss distribution is consistent with what is expected, considering that the Gaussian conditional independence model is used to generate the default events. The estimated

expected loss and VaR are also presented in the figure. The expected loss of the portfolio is approximately 40EUR and the 5th percentile VaR is 86 EUR.

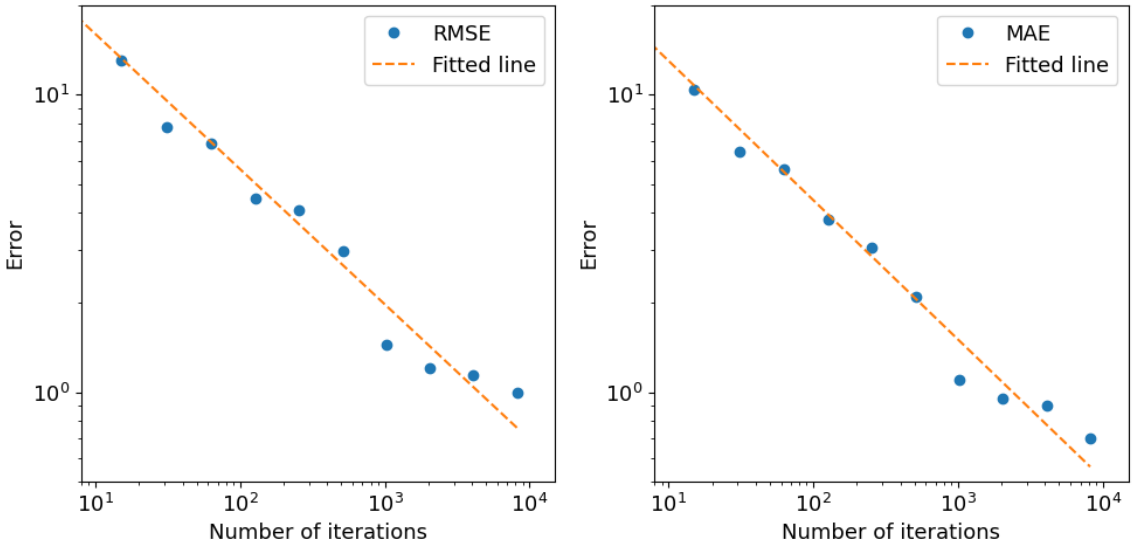
Figure 18: Loss distribution



Source: Own work.

Figure 19 below displays the convergence rate of the estimated VaR values as a function of the number of simulation iterations. The x-axis represents the number of simulation iterations, while the y-axis represents the RMSE and MAE, derived from VaR estimates and the exact value of VaR. In the figure, we can observe that the error decreases monotonically with the number of simulation iterations, indicating that the simulation is converging to the correct VaR estimate. Both error measures used decrease linearly with the number of simulation iterations, which is in line with our expectations and matches the results obtained in section 4.1.2. At approximately -0.46, the convergence rate for VaR estimation is however slightly lower than observed in the previous analysis, which can be attributed to a lower sample of simulation runs used in the convergence analysis and is not caused by the algorithm itself.

Figure 19: Convergence rate

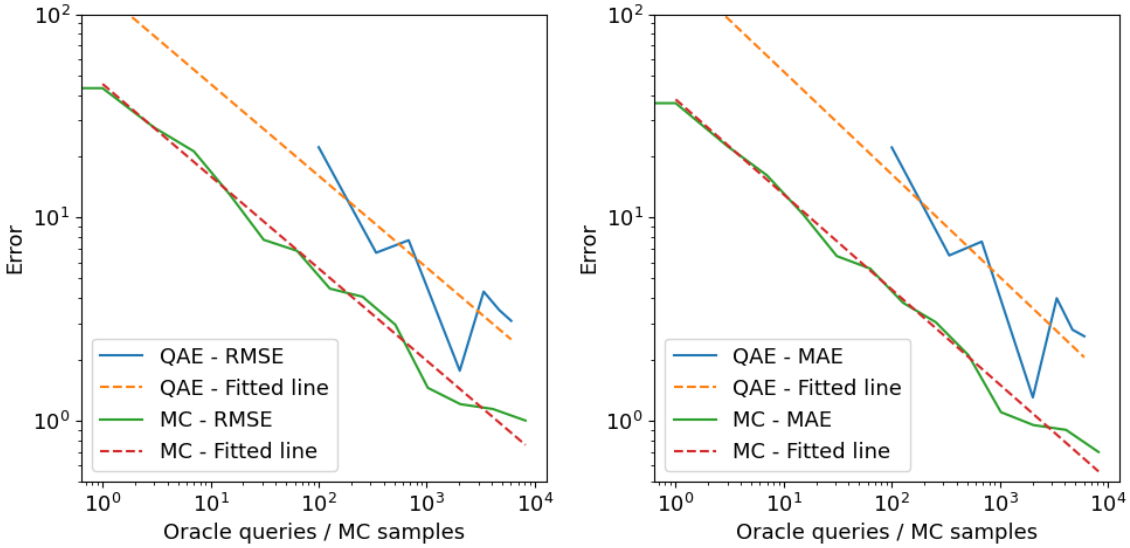


Source: Own work.

4.2.3 Comparison of VaR estimation results

Figure 20 below presents the convergence of the QAE and MC simulations. The graphs depict how the errors, measured by RMSE and MAE, decrease with the number of iterations for the MC simulation and the number of oracle queries for the QAE approach. Both algorithms show linear convergence, converging towards the correct VaR estimate. A closer look at the convergence rates presented in Table 3, reveals that, while both the QAE and MC simulations exhibit similar trends, the QAE approach demonstrates a slightly higher convergence rate. However, when evaluating the absolute error rates for the same number of iterations, the quantum algorithm performs worse than the MC simulation. This discrepancy is attributed to the bisection search method employed in the quantum approach, which begins in the middle of the possible VaR values range and iteratively narrows down the search. This method could be significantly enhanced with a more efficient ansatz for the bisection search, a topic that will be explored further in Section 4.2.4.1. Additionally, the optimization of the number of oracle queries required for VaR estimation will be discussed in Section 4.2.4.2.

Figure 20: Comparison of QAE and MC VaR estimation



Source: Own work.

Table 3: Comparison of convergence rates of QAE and MC VaR estimation

Algorith	RMSE	MAE
MC	-0.4540	-0.4680
QAE	-0.4522	-0.5056

Source: Own work.

The impact of the approximations used in the quantum approach to VaR estimation can be observed by comparing the VaR estimates and the loss distributions presented in Figures 16 and 18. The exact VaR value is 83 EUR, while the VaR estimate obtained using the quantum approach is 86 EUR. The difference is relatively small, at approximately 4%, and could be decreased with a better approximation of the normal distribution.

Overall, the results of the comparison show that both approaches can converge to an accurate VaR estimate. The quantum approach however does not appear to provide any meaningful advantages over the classical approach. Moreover, the use of bisection in the quantum approach hinders the possibility of this approach being used in real-world applications, even if the quantum hardware were to be improved, since the convergence rate of the approach is not fast enough compared to the MC simulation.

4.2.4 Improving the QAE VaR estimation

This section presents two enhancement proposals for quantum VaR estimation tested in the previous section. We will explore the improvement of the algorithm's search strategy, discuss the implementation of this improvement, and analyze the empirical

results. We will also propose an optimization of the number of oracle queries used in the VaR estimation process and discuss the potential benefits of this approach.

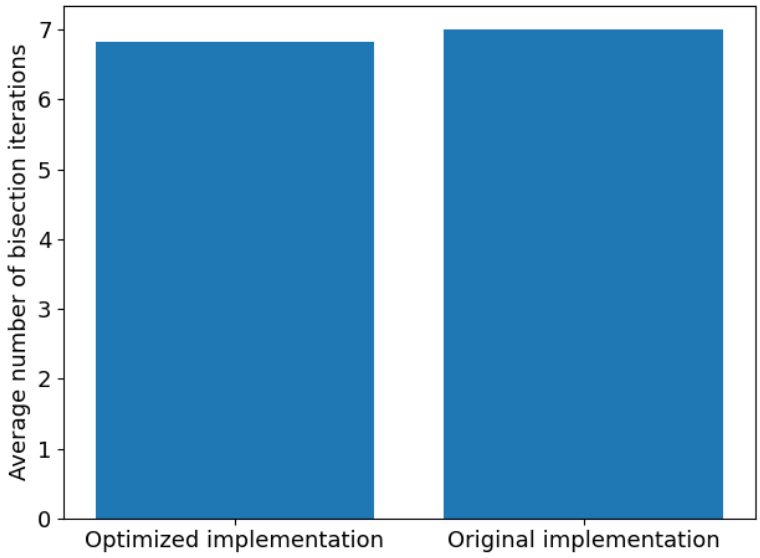
4.2.4.1 Bisection search improvement

The first improvement to the quantum VaR estimation algorithm is the optimization of the bisection search. The original algorithm utilized the midpoint between the minimum and maximum loss of the portfolio to generate the ansatz for the bisection search. While this approach guarantees the inclusion of the true VaR value within the search interval, it does not leverage any prior knowledge about the loss distribution that could be used to optimize the search.

To enhance the search process, we propose a better ansatz for the bisection search, that employs the expected loss of the portfolio as the initial bounds for the bisection search. This approach is based on the assumption that the loss distribution of the portfolio is not excessively skewed so that the expected loss is likely to be lower than the VaR value.

The proposed optimization was empirically tested in a setting analogous to the one used for the initial VaR estimation. Due to the considerable computational resources required for quantum simulation, the estimation was performed five times. The results, as depicted in Figure 21 below, show that the optimization of the bisection search strategy yields an approximately 2.5% improvement in the VaR convergence. While the improvement is modest, it must be noted that the portfolio in question is relatively small, meaning that the scope for optimization is inherently limited. In more complex portfolios with a broader loss distribution, the impact of the refined ansatz could be more pronounced, as the reduction of the search interval could translate to more significant convergence improvements.

Figure 21: Bisection search improvement



Source: Own work.

4.2.4.2 Optimization of oracle queries

The QAE algorithm's performance in estimating VaR can be significantly influenced by the number of oracle queries. To optimize the bisection search process in combination with QAE we propose a strategy that varies the number of oracle queries depending on the stage of the search. Initially, the algorithm can operate with fewer oracle queries to provide a coarse estimation of the loss percentiles. This approach is based on the understanding that high precision is less critical in the early phase of bisection search, where the goal is to quickly eliminate the unlikely ranges for VaR. As the search interval narrows, the algorithm can increase the number of oracle queries to refine the accuracy of the percentile estimate, ensuring that the final VaR value is precise.

While the proposed algorithm offers a structured method for optimizing the number of oracle queries, its development and testing were beyond the scope of this master's thesis. Instead, a simplified approach was employed during the empirical testing of the QAE for VaR estimation. To expedite the computation, the number of oracle queries was manually decreased for the first three iterations of the bisection search. This pragmatic solution was implemented to save computational resources, while maintaining a focus on the primary research objective.

4.3 Portfolio optimization

In this section, we will conduct a comprehensive performance analysis of the VQE and QAOA, as previously introduced in Sections 2.3.4 and 2.3.5. This analysis will be based on their application in solving the QUBO formulation of the portfolio optimization problem, which is details in Section 3.3.2. To facilitate this analysis, a hypothetical portfolio comprising 15 distinct assets was constructed. The expected returns and the covariance matrix characterizing the risk and return of these assets are presented in the appendix section 4.

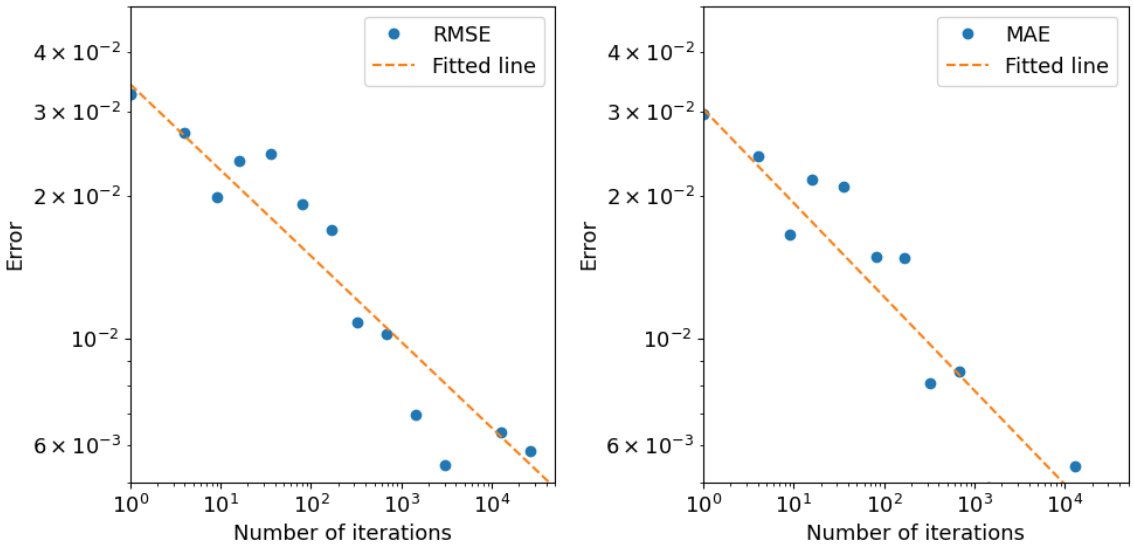
As described in the earlier sections, both VQE and QAOA are hybrid algorithms that require a classical optimization algorithm to determine the optimal parameters for the quantum circuits. For this analysis, the Constrained Optimization By Linear Approximation (COBYLA) method, as formulated by Powell in 1944 (Powell, 1994), was used as the classical optimization strategy to refine the parameters for both QAOA and VQE algorithms. The following subsections will detail the results of the computational experiments, which were set up to evaluate the efficacy of the quantum algorithms in identifying optimal asset allocations that maximize the expected return given a level of risk. The performance metrics used to assess the algorithms' effectiveness, along with a comparison of the algorithms, will also be provided.

In order to evaluate the convergence property of the VQE and QAOA a test was devised aimed at identifying the rate at which these algorithms approach the optimal solution. This was accomplished by systematically varying the maximum number

of iterations parameter of the COBYLA algorithm. The parameter was escalated from a minimum of 1 to a maximum of 1000, with exponential steps to capture the convergence behavior over a broad range of iterations. Throughout this process, the portfolio allocation weights derived at each iteration was recorded. The weights were then used to estimate the achieved cost function value, defined in equation (61). The RMSE and MAE were then computed using the calculated cost function values and the minimum cost function value, estimated using the optimal portfolio weights. It is important to recognize that the convergence rate is inherently influenced by both the classical and quantum components of the algorithms. Consequently, the total number of iterations, which serves as the basis for computing the convergence rate of the algorithms, is calculated as the product of the number of iterations of the classical optimization algorithm and the number of applications of the quantum circuit.

The results of the analysis of the convergence rate of the VQE algorithm are presented in Figure 22 below. The x-axis represents the number of iterations, while the y-axis represents the RMSE and MAE. The results show that the error decreases with the number of iterations, indicating a good convergence towards the optimal solution.

Figure 22: VQE convergence rate



Source: Own work.

The regression analysis conducted on the error metrics provides additional insight into the convergence rate of the VQE algorithm. The results presented in Table 4 below. The negative slopes of -0.18 and -0.2 for RMSE and MAE, respectively, indicate a steady decrease in error as the number of iterations increases. Moreover, the high R - sqaured values demonstrate a strong linear relationship between the number of iterations and the error metrics. This implies that the majority of the variation in the error metrics can be explained by the number of iterations of the VQE algorithn.

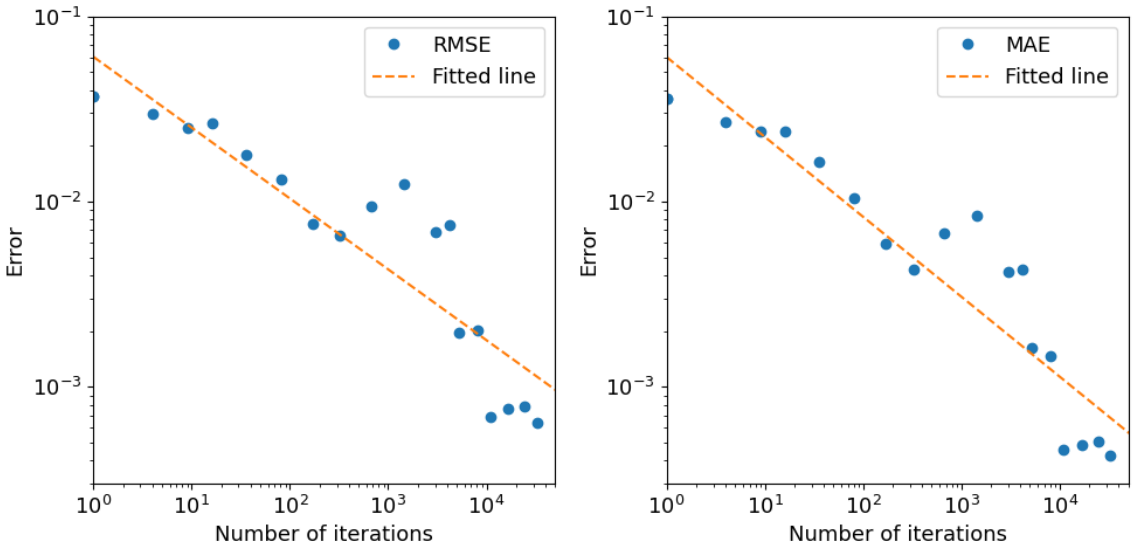
Table 4: VQE convergence rate

Variable	RMSE	MAE
Intercept	-1.4652	-1.5175
Slope	-0.1802	-0.1968
R-squared	0.8853	0.8778

Source: Own work.

Figure 23 shows the results of the convergence analysis of the QAOA algorithm. The results show a clear down-trend in the error rates as the number of iterations increases. The error rates however, are not monotonically decreasing and experience small deviations, most notably around 1000 iterations, where the error rates increase slightly. This behaviour is likely due to the number of iterations used to compute the RMSE and MAE and should not be a feature of the approach.

Figure 23: QAOA convergence rate



Source: Own work.

The regression analysis for the QAOA yielded the following results, presented in Table 5. The steep negative slopes of -0.3822 for RMSE and -0.43 for MAE indicate a rapid decrease in error as the algorithm advances through iterations, indicating an effective error minimization towards the optimal solution. The robustness of this linear relationship is further confirmed by the high R - squared values.

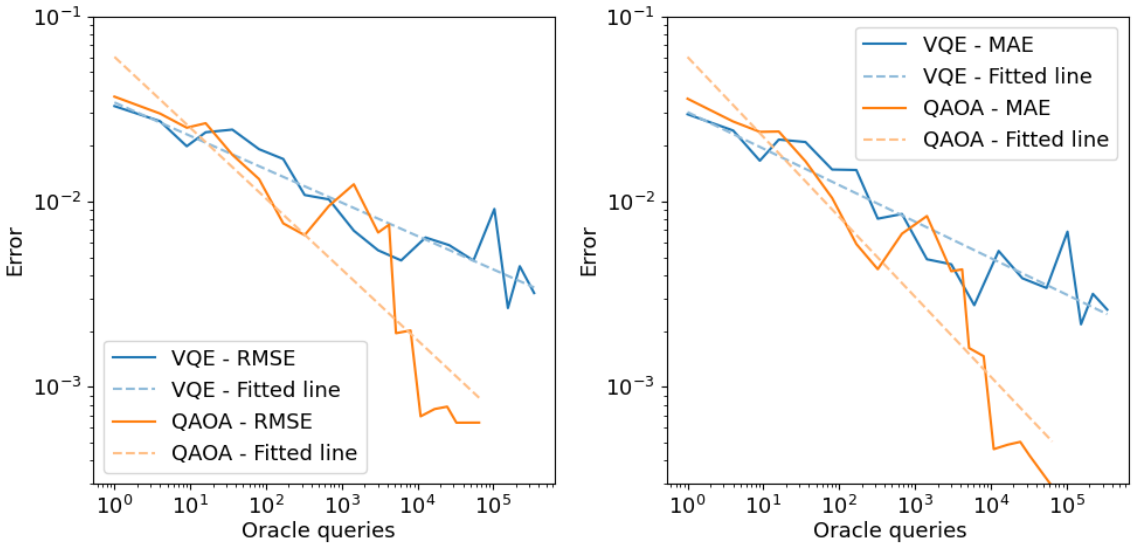
Table 5: QAOA convergence rate

Variable	RMSE	MAE
Intercept	-1.2193	-1.2201
Slope	-0.3822	-0.4311
R-squared	0.8456	0.8832

Source: Own work.

In a comparative analysis of the VQE and QAOA algorithms, presented in Figure 24, we observe distinct convergence characteristics unique to each algorithm that show their performance in the context of portfolio optimization. While both algorithms exhibit a strong linear relationship between the number of iterations and the error metrics, the QAOA demonstrated a steeper slope, indicating a more rapid convergence. The VQE, on the other hand, shows slightly higher R-squared values, suggesting a more predictable convergence trajectory.

Figure 24: QAOA and VQE convergence rate comparison



Source: Own work.

The differences in convergence rates and stability between the VQE and QAOA underscore the importance of algorithmic development in the field of quantum computing. As the results demonstrate, the choice and refinement of quantum algorithms can significantly influence performance outcomes. This insight is crucial for the advancement of quantum computing applications in finance, where the efficiency and accuracy of algorithms have direct implications for their practical deployment in complex optimization tasks.

4.4 Hardware requirements

As quantum computing technology continues to evolve, assesing the hardware requirements for finance applications becomes increasingly important. In this section, we will focus on the hardware needs for implementing the algorithms used in this thesis. We will also examine the growth trend in qubit count and discuss predictions for when quantum hardware might be sufficient for real-world financial applications.

4.4.1 Hardware requirements VaR estimation

The VaR estimation on a quantum computer presents a unique set of hardware requirements that are based on multiple factors. The first factor relates to the approximation of the normal distribution, which requires a sufficient number of qubits to adequately approximate the distribution function. The granularity of the approximation required is determined by the application, however a relatively low number of qubits will already provide a good representation of the distribution for financial models. For example, a 16 qubits allow for 65536 quantization levels.

Considering the assets in a portfolio, the number of qubits required to represent asset defaults, maps directly to the number of asset themselves. Therefore, a portfolio with 1000 assets would require a quantum computer with a minimum of 1000 qubits to represent the default events of each asset.

Finally, the process of loss estimation, involving the calculation of the total losses across all defaulting assets, directly impacts the qubit requirement. The number of qubits needed for this task scales logarithmically with the LGD, as described by the following equation:

$$n_q = \log_2(\sum_{i=1}^{n} LGD_i) + 1 \tag{68}$$

Where n_q is the number of qubits required, n is the number of assets in the portfolio, and LGD_i is the loss given default of asset i.

In summary, the hardware requirements for VaR estiamation on a quantum computer are determined by the number of assets in the portfolio, the granularity of the normal distribution approximation, and the maximum loss of the portfolio. To put things into perspective, for a portfolio with 1000 assets and 1 billion EUR in maximum loss, a quantum computer with more than 1050 qubits would be required to perform the VaR estimation.

4.4.2 Hardware requirements portfolio optimization

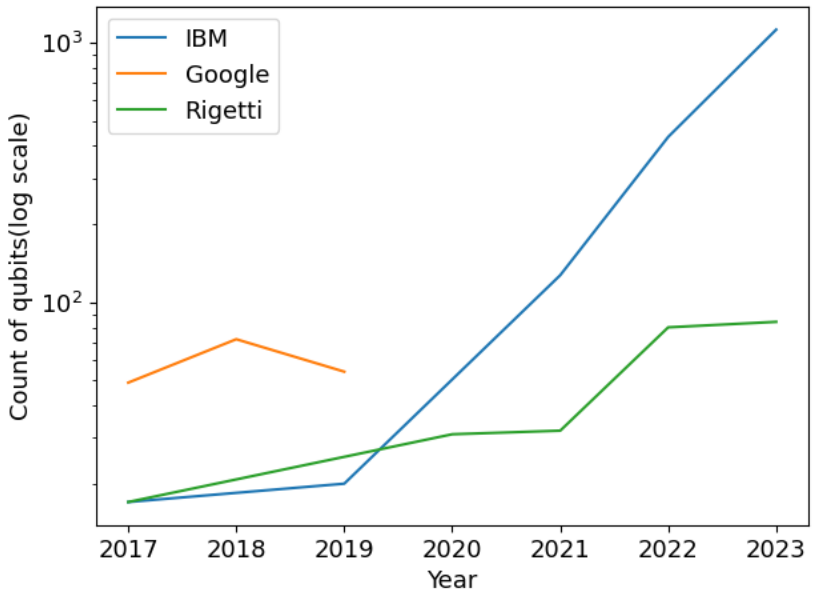
Portfolio optimization on a quantum computer translates each asset into a quantum bit, where the binary state of the qubit represents the inclusion or exclusion of the asset in the optimal portfolio. This one-to-one mapping between assets and qubits indicates a linear relationship between the number of assets and the number of qubits required. While this mapping is simple, it is also a limitation of the quantum approach to portfolio optimization as it restricts the weights of the assets to binary values and does not allow fractional asset allocations. Addressing this limitation might require the use of additional qubits to represent fractional weights, which would exponentially increase the requirement for qubits.

4.4.3 Future outlook

Moore's law, formulated by Gordon Moore in 1965 (Schaller, 1997), predicts that the number of transistors on an integraded circutit or microchip will double approximately every two years. Since its inception, the law has performed remarkably well in predicting the growth of the classical computing powers. In the realm of quantum computing, we can observe a similar trend in the growth of qubit count, although the field is still in very early stages.

An analysis of the historical data on qubit counts was conducted to assess the growth trend of qubit count. The results are presented in Figure 25 below. The graph shows a clear exponential growth in terms of qubit count, with IBM leading in terms of qubit availability. The data indicates that quantum technology is advancing rapidly and does not show any signs of the progress slowing down in the near future.

Figure 25: Number of available qubits over time



Source: Own work.

Based on the current trajectory, we can estimate that the number of qubits available could reach a value as high as 100000 by the end of the decade. Considering the hardware requirements for portfolio optimization discussed in Section 4.4.2, this would allow for the optimization of portfolios with a very large number of assets and even allow for fractional asset allocations. Similarly, considering VaR estimation, the increased qubit count would allow for a more precise approximation of the normal distribution and allow for VaR estimation on large portfolios, without a restriction on the number of assets.

However, it is crucial to note that the qubit count is only one of the aspects of quantum computer's performance. As noted by Wang et al. and Mckay et. al. (Wang et al., 2022, McKay et al., 2023) the performance of quantum computers is heavily influenced by the error rates, decoherence, gate fidelity, and other factors. Therefore, while the qubit count is an important metric, it is not the only one

that determines the performance of a quantum computer. Similarly, IBM developed their own metrics that such as Quantum Volume (QV) and CLOPS that are used to assess the performance of their quantum processors (IBM, 2022, Qiskit, 2020). The additional measures take into account factors such as error rates speed at which the quantum processor can perform operations in addition to the qubit count.

5 CONCLUSION

This master's thesis rigorously explores the impact of quantum computing in the finance sector, focusing on VaR estimation and portfolio optimization through VQE and QAOA. The study assesses the practical capabilities and limitations of current quantum computing as compared to traditional computational methods, emphasizing theoretical and empirical performance metrics.

Initially, the study examines the real-world performance of quantum algorithms, specifically the convergence rate of the QAE algorithm in VaR estimation, compared to theoretical expectations. While quantum algorithms theoretically promise quadratic speedups indicated by a convergence rate of $\mathcal{O}(M^{1/2})$, empirical findings suggest that the convergence rates achieved with current NISQ devices fall short of these predictions. This shortfall is primarily due to limitations in the search algorithms used, rather than the capabilities of QAE itself. This is demonstrated by QAE's superiod performance in estimating the underlying probability p of a uniformly distributed random variable, compared to classical Monte Carlo simulation.

Furthermore, the execution times for VaR estimation on quantum computers were evaluated against classical computing approaches utilizing both serial and parallel processing techniques. Although quantum computing theoretically offers substantial reductions in execution times, especially over serial processing, the advantages are less pronounced when compared to highly optimized classical parallel processing systems. Currently, quantum computers perform fewer operations per second than classical computers, which, even with faster convergence, results in quantum approaches lagging significantly behind in execution time.

In exploring whether VaR estimation algorithms on quantum computers can be enhanced by more sophisticated optimization techniques, the thesis identifies the bisection search as a primary bottleneck. It suggests that performance improvements can be achieved by dynamically varying the number of oracle queries in the bisection search process. This approach could mitigate some computational challenges posed by the bisection search algorithm, potentially enhancing the competitiveness of quantum algorithms against classical methods.

When comparing quantum and classical approaches to portfolio optimization, the QAOA shows promise in efficiently navigating complex optimization problems that classical algorithms struggle with, especially at high dimensions. However, the definitive advantage of quantum computing is not yet realized due to substantial hardware limitations, particularly the availability of qubits that restricts the prac-

tical implementation of quantum algorithms.

In conclusion, while quantum computing holds significant transformative potential for finance, particularly in VaR estimation and portfolio optimization, its superiority over classical methods in real-world applications is not yet consistent. The advancement of quantum hardware, along with innovations in algorithm design and the strategic use of hybrid models, is crucial for the practical adoption of quantum computing in financial computations. The potential of quantum computing to revolutionize financial methodologies cannot be understated, however realizing this potential heavily depends on overcoming the considerable technological challenges currently plaguing the field.

REFERENCES

1. Apple. (2022). *Getting started with metal-cpp*. https://developer.apple.com/metal/cpp/

2. Barkoutsos, P. K., Nannicini, G., Robert, A., Tavernelli, I., & Woerner, S. (2020). Improving variational quantum optimization using cvar. *Quantum*, *4*(1), 256–272.

3. Brassard, G., Hoyer, P., Mosca, M., & Tapp, A. (2002). Quantum amplitude amplification and estimation. *Contemporary Mathematics*, *305*, 53–74.

4. Brooks, M. (2019). Beyond quantum supremacy: The hunt for useful quantum computers. *Nature*, *574*(7776), 19–22.

5. Cornuejols, G., & Tütüncü, R. (2006). *Optimization methods in finance* (Vol. 5). Cambridge University Press.

6. Devitt, S. J., Munro, W. J., & Nemoto, K. (2013). Quantum error correction for beginners. *Reports on Progress in Physics*, *76*(7), 076001.

7. Dirac, P. A. M. (1939). A new notation for quantum mechanics. *Mathematical Proceedings of the Cambridge Philosophical Society*, *35*(3), 416–418.

8. Doganoglu, T., Hartz, C., & Mittnik, S. (2007). Portfolio optimization when risk factors are conditionally varying and heavy tailed. *Computational Economics*, *29*, 333–354.

9. Dorner, U., Demkowicz-Dobrzanski, R., Smith, B. J., Lundeen, J. S., Wasilewski, W., Banaszek, K., & Walmsley, I. A. (2009). Optimal quantum phase estimation. *Physical review letters*, *102*(4), 040403.

10. Egger, D. J., Gutiérrez, R. G., Mestre, J. C., & Woerner, S. (2020). Credit risk analysis using quantum computers. *IEEE transactions on computers*, *70*(12), 2136–2145.

11. EY. (2023). *Ey and ibm expand strategic alliance into quantum computing.* https://www.ey.com/en_gl/news/2023/04/ey-and-ibm-expand-strategic-alliance-into-quantum-computing

12. Farhi, E., Goldstone, J., & Gutmann, S. (2014). A quantum approximate optimization algorithm. *arXiv preprint arXiv:1411.4028*.

13. Gebraad, L., & Fichtner, A. (2023). Seamless gpu acceleration for c++-based physics with the metal shading language on apple's m series unified chips. *Seismological Society of America*, *94*(3), 1670–1675.

14. Gleason, A. M. (1975). *Measures on the closed subspaces of a hilbert space* (Vol. 1). Springer.

15. Glover, F., Kochenberger, G., & Du, Y. (2018). A tutorial on formulating and using qubo models. *arXiv preprint arXiv:1811.11538*.

16. Grinko, D., Gacon, J., Zoufal, C., & Woerner, S. (2021). Iterative quantum amplitude estimation. *npj Quantum Information*, *7*(1), 52.

17. Grover, L. K. (1996). A fast quantum mechanical algorithm for database search. *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, 212–219.

18. Hyndman, R. J., & Koehler, A. B. (2006). Another look at measures of forecast accuracy. *International journal of forecasting*, *22*(4), 679–688.

19. IBM. (2022). *Updating how we measure quantum quality and speed.* https://research.ibm.com/blog/quantum-metric-layer-fidelity

20. IBM. (2024). *Ibm quantum platform.* https://quantum.ibm.com/

21. Javadi-Abhari, A., Treinish, M., Krsulich, K., Wood, C. J., Lishman, J., Gacon, J., Martiel, S., Nation, P. D., Bishop, L. S., Cross, A. W., Johnson, B. R., & Gambetta, J. M. (2024). Quantum computing with Qiskit.

22. Jorion, P. (2007). *Value at risk: The new benchmark for managing financial risk.* McGraw-Hill.

23. Kuester, K., Mittnik, S., & Paolella, M. S. (2006). Value-at-risk prediction: A comparison of alternative strategies. *Journal of Financial Econometrics*, *4*(1), 53–89.

24. Lai, T. L., Xing, H., & Chen, Z. (2011). Mean–variance portfolio optimization when means and covariances are unknown. *The Annals of Applied Statistics*, *5*(2A). https://doi.org/10.1214/10-aoas422

25. Levy, H., & Markowitz, H. M. (1979). Approximating expected utility by a function of mean and variance. *The American Economic Review*, *69*(3), 308–317.

26. Low, R. K. Y., Faff, R., & Aas, K. (2016). Enhancing mean–variance portfolio selection by modeling distributional asymmetries. *Journal of Economics and Business*, *85*, 49–72.

27. Markowitz, H. M. (1991). *Efficient diversification of investments*. Wiley; Sons.

28. Markowitz, H. (1952). Portfolio selection. *The Journal of Finance*, *8*(1), 77–91.

29. Markowitz, H. M. (1956). The optimization of a quadratic function subject to linear constraints. *Naval research logistics Quarterly*, *3*(1-2), 111–133.

30. Matsumoto, M., & Nishimura, T. (1998). Mersenne twister: A 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, *8*(1), 3–30.

31. McKay, D. C., Hincks, I., Pritchett, E. J., Carroll, M., Govia, L. C., & Merkel, S. T. (2023). Benchmarking quantum processor performance at scale. *arXiv preprint arXiv:2311.05933*.

32. Mehta, A., Neukirchen, M., Pfetsch, S., & Poppensieker, T. (2012). Managing market risk: Today and tomorrow. *McKinsey & Company McKinsey Working Papers on Risk*, *32*(1), 24–36.

33. Misrosoft. (2024). *Azure quantum documentation*. https://learn.microsoft.com/en-us/azure/quantum/

34. Nakaji, K. (2020). Faster amplitude estimation. *arXiv preprint arXiv:2003.02417*.

35. Nelson, B. (2020). *Scientific computing with python*. https://caam37830.github.io/book/introduction.html

36. Pontius, R. G., Thontteh, O., & Chen, H. (2008). Components of information for multiple resolution comparison between maps that share a real variable. *Environmental and ecological statistics*, *15*, 111–142.

37. Powell, M. J. (1994). *A direct search optimization method that models the objective and constraint functions by linear interpolation*. Springer.

38. Preskill, J. (2018). Quantum Computing in the NISQ era and beyond. *Quantum*, *2*, 79. https://doi.org/10.22331/q-2018-08-06-79

39. Qiskit. (2020). *What is quantum volume, anyway?* https://medium.com/qiskit/what-is-quantum-volume-anyway-a4dff801c36f

40. Roffe, J. (2019). Quantum error correction: An introductory guide. *Contemporary Physics*, *60*(3), 226–245.

41. Rué, J., & Xambó, S. (2011). Mathematical essentials of quantum computing. *Mathematical essentials of quantum computing*. https://api.semanticscholar.org/CorpusID:124255513

42. Rutkowski, M., & Tarca, S. (2015). Regulatory capital modeling for credit risk. *International Journal of Theoretical and Applied Finance*, *18*(05), 1550034.

43. Schaller, R. R. (1997). Moore's law: Past, present and future. *IEEE spectrum*, *34*(6), 52–59.

44. Shor, P. W. (1995). Scheme for reducing decoherence in quantum computer memory. *Physical review A*, *52*(4), R2493.

45. Smite-Meister. (2009). *Bloch sphere*. https://t.ly/LO7qL

46. Smith, R. S., Curtis, M. J., & Zeng, W. J. (2016). A practical quantum instruction set architecture. https://arxiv.org/abs/1608.03355

47. Suzuki, Y., Uno, S., Raymond, R., Tanaka, T., Onodera, T., & Yamamoto, N. (2020). Amplitude estimation without phase estimation. *Quantum Information Processing*, *19*, 1–17.

48. Taleb, N. N., & Investments, L. (2009). Report on the risks of financial modeling, var and the economic breakdown. *United States Congress, Testimony (Subcommittee)*.

49. Vasicek, O. (2002). The distribution of loan portfolio value. *Risk*, *15*(12), 160–162.

50. Wang, J., Guo, G., & Shan, Z. (2022). Sok: Benchmarking the performance of a quantum computer. *Entropy*, *24*(10), 1467.

51. Willmott, C. J., & Matsuura, K. (2005). Advantages of the mean absolute error (mae) over the root mean square error (rmse) in assessing average model performance. *Climate research*, *30*(1), 79–82.

APPENDICES

APPENDIX 1: POVZETEK V SLOVENSKEM JEZIKU

Kvantni računalniki predstavljajo naslednjo generacijo računalniške tehnologije, ki temelji na osnovi kvantnih mehanizmov. Za razliko od klasičnih računalnikov, ki uporabljajo binarne bite za shranjevanje in procesiranje podatkov, kvantni računalniki uporabljajo kubite, ki se lahko se lahko simultano obstajajo v več stanjih, kar omgoča izvajanje kompleksnih operacij ter pospešitev reševanja določenih matematičnih problemov. Ta tehnologija ima potencial za izboljšanje obstoječih računalniških metodologij, ki se uporabljajo v farmaciji, razvoju materialov, in financah, kjer so potrebe po hitrem procesiranju podatkov še posebej izrazite. V kontekstu finančnih aplikacij, magistrska naloga raziskuje dve področji uporabe kvantnih računalnokov: oceno tvegane vrednosti (VaR) in optimizacijo portfeljev.

Ocena tvegane vrednosti je ključna za ocenjevanje potencialnih izgub v portfelju, kar je izjemno pomembno za upravljanje tveganj. Trenutno se VaR za kompleksne portfelje ocenjuje s pomočjo Monte Carlo simulacij, ki zahtevajo obsežne izračune in za doseganje statistične zanesljivosti, kar je časovno in stroškovno potratno. To kažejo tudi rezultati McKensly analize (Mehta et al., 2012), ki kažejo, da lahko ocenjevanje VaR na nekaterih bankah traja tudi do 15 ur. V magistrski nalogi je bil uporabljen QAE algoritem, ki omogoča kvadratno pospešitev ocenjevanja VaR, kar bi lahko pripomoglo k veliko hitrejšemu ocenjevanju VaR. Empirična analiza algoritma je pokazala, da je algoritem zmožen izboljšave konvergenčne hitrosti v primerjavi z Monte Karlo simulacijo, vendar zgolj na preprostem modelu kjer je bil algoritem uporabljen za oceno verjetnosti p Bernoullijeve spremenljivke. V primeru ocenjevanja VaR, je bil algorithm uporabljen v kombinaciji z bisekcijskim iskanjem, je bila konvergenca algoritma počasnejša od Monte Karlo simulaticje. Primerjava časa izvajanja obeh pristopov je pokazala, da je klasični pristop veliko hitrejši od kvantnega v obeh primerih, kar kaže na potrebo po izboljšavi kvantnih algoritmov in kvantne strojne opreme, če želimo, da bi kvantni pristop postal konkurenčen klasičnemu.

Pri optimizaciji portfeljev sta bila za reševanje QUBO optimizacijskega problema uporabljena VQE in QAOA algorithma. Oba algoritma uporabljata hibridni sistem, ki združuje kvantni in klasični računalnik za iskanje minimalne vrednosti QUBO fukncije, kar v financah prestavlja minimizacijo tveganja in maksimizacijo donosa portfelja. Empirična analiza konvergenčne hitrosti obeh alogiritmov je pokazala predvsem to, da je nadaljni razvoj kvantnih algoritmov ključnega pomena, saj je razlika v konvergenci med VQE in QAOA algoritma zelo izrazita.

Poleg rezultatov analiz konvergence in časov izvajanja kvantnih algoritmov, je magistrska naloga tudi raziskala strojne zahteve za izvajanje zahtevnejših operacij na kvantnih računalnikih. Na podlagi te analize je bilo ugotovljeno, da je pri ocenjevanju VaR in optimizaciji portfeljev število qubitov eden izmed ključnih omejitvenih faktorjev, ki vpliva na uporabnost kvantnih algoritmov v financah. Na podlagi trenutnih trendov rasti števila qubitov, je bilo ugotovljeno, da bi kvantni računalniki lahko postali konkurenčni klasičnim računalnikom še znotraj naslednjega desetletja.

Čeprav ima kvantno računalništvo izjemen potencial za spremeniti obstoječe računalniške pristope v financah, še posebej pri oceni VaR, njegova prednost pred klasičnimi metodami v realnih aplikacijah še ni dosledna. Napredek kvantne strojne opreme skupaj z inovacijami v kvantnih alogitmih in strateška uporaba hibridnih modelov so ključni za prktično sprejetje kvantnega računalništva v mnogih industrijah. Potencial kvantnega računalništva za revolucioniranje finančnih metodologij je nedvomo obsta, vendar je uresničitev tega potenciala močno odvisna od premagovanja obsežnih tehnoloških izzivov, ki trenutno pestijo to področje.

APPENDIX 2: GROVER OPERATOR EQUALITY

Grover operator, also known as the phase inversion operator, is defined as:

$$U_\omega = (-1)^{f(x)} \tag{69}$$

The operator can also be written as:

$$U_\omega = \mathbb{1} - 2|\omega\rangle\langle\omega| \tag{70}$$

To show the equivalency of the two forms, we first apply the latter operator to a quantum state $|x\rangle$:

$$U_\omega|x\rangle = (\mathbb{1} - 2|\omega\rangle\langle\omega|)|x\rangle = |x\rangle - 2|\omega\rangle\langle\omega|x\rangle \tag{71}$$

The expression $\langle\omega|x\rangle$ is an inner product that returns 1 if $|x\rangle = |\omega\rangle$ and 0 otherwise. Therefore, the expression $\langle\omega|x\rangle$ can be written as the Kronecker delta function:

$$\delta_{x\omega} = \begin{cases} 1, & x = \omega \\ 0, & x \neq \omega \end{cases} \tag{72}$$

Substituting (72) into the equation (71) above we show that the two forms of the Grover operator are equivalent:

$$U_\omega|x\rangle = |x\rangle - 2\delta_{x\omega}|\omega\rangle = \begin{cases} -|x\rangle, & x = \omega \\ |x\rangle, & x \neq \omega \end{cases} \tag{73}$$

APPENDIX 3: VAR - PORTFOLIO PARAMETERS

The table 6 contains the asset risk parameters used in the VaR estimation.

Table 6: Asset risk parameters

Asset	Unconditional PD (%)	ρ	LGD (EUR)
1	0.30	0.33	4
2	0.27	0.26	6
3	0.72	0.19	7
4	0.63	0.25	11
5	0.31	0.33	17
6	0.47	0.64	12
7	0.23	0.24	9
8	0.36	0.25	5
9	0.32	0.20	6
10	0.14	0.33	9

Source: Own work.

APPENDIX 4: PORTFOLIO OPTIMIZATION - PORTFOLIO PARAMETERS

The tables 7 and 8 below, contain the asset returns and risk parameters used in the portfolio optimization.

Table 7: Asset returns

Asset	Expected Return
Asset 1	0.0153
Asset 2	-0.0008
Asset 3	0.0005
Asset 4	0.0009
Asset 5	0.0103
Asset 6	-0.0017
Asset 7	0.0272
Asset 8	0.0003
Asset 9	-0.0011
Asset 10	-0.0124
Asset 11	0.0016
Asset 12	0.0038
Asset 13	-0.0059
Asset 14	-0.0000
Asset 15	-0.0021

Source: Own work.

Table 8: Asset risk parameters (multiplied by 1000)

Asset No.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	2.5	0.1	0.1	-0.1	-0.3	0.2	2.6	0.3	-0.0	0.9	-0.0	0.1	-0.1	-0.1	0.1
2	0.1	0.3	0.1	0.0	-0.0	-0.1	-0.3	-0.0	0.1	0.1	0.0	-0.1	0.0	0.0	-0.0
3	0.1	0.1	0.8	-0.1	-0.0	-0.2	-0.8	0.1	-0.1	0.1	0.1	-0.1	-0.0	0.1	0.1
4	-0.1	0.0	-0.1	0.2	0.1	-0.0	0.8	-0.0	0.0	0.1	0.0	-0.0	0.1	-0.0	0.0
5	-0.3	-0.0	-0.0	0.1	0.5	0.1	0.1	0.1	-0.0	0.1	0.1	0.1	0.2	0.0	0.1
6	0.2	-0.1	-0.2	-0.0	0.1	0.5	0.4	0.1	0.0	-0.0	0.1	0.0	0.1	-0.0	-0.1
7	2.6	-0.3	-0.8	0.8	0.1	0.4	43.3	0.5	0.6	2.0	1.0	0.9	1.1	-0.7	1.6
8	0.3	-0.0	0.1	-0.0	0.1	0.1	0.5	0.7	-0.1	0.1	0.1	0.0	0.0	-0.0	0.0
9	-0.0	0.1	-0.1	0.0	-0.0	0.0	0.6	-0.1	0.2	-0.1	-0.0	0.0	0.0	-0.0	0.0
10	0.9	0.1	0.1	0.1	0.1	-0.0	2.0	0.1	-0.1	1.8	-0.4	0.0	0.0	-0.1	0.1
11	-0.0	0.0	0.1	0.0	0.1	0.1	1.0	0.1	-0.0	-0.4	0.7	0.0	0.1	-0.0	0.1
12	0.1	-0.1	-0.1	-0.0	0.1	0.0	0.9	0.0	0.0	0.0	0.0	0.2	0.1	-0.1	0.1
13	-0.1	0.0	-0.0	0.1	0.2	0.1	1.1	0.0	0.0	0.0	0.1	0.1	0.5	-0.0	0.1
14	-0.1	0.0	0.1	-0.0	0.0	-0.0	-0.7	-0.0	-0.0	-0.1	-0.0	-0.1	-0.0	0.2	-0.1
15	0.1	-0.0	0.1	0.0	0.1	-0.1	1.6	0.0	0.0	0.1	0.1	0.1	0.1	-0.1	0.3

Source: Own work.