Dissertation

Quantum Computing in Option Pricing

Tom Frederik Ewen

Vom Fachbereich Mathematik der Rheinland-Pfälzischen Technischen Universität Kaiserslautern-Landau zur Verleihung des akademischen Grades Doktor der Naturwissenschaften (Doctor rerum naturalium, Dr. rer. nat.) genehmigte Dissertation.

1. Gutachter | Prof. Dr. Ralf Korn
2. Gutachter | Prof. Dr. Wolfgang Lechner

Datum der Disputation: 10.06.2025

DE-386

Acknowledgements

I would like to express my heartfelt gratitude to my supervisor, Prof. Ralf Korn from RPTU Kaiserslautern, for your unwavering support and openness in supervising my exploration of quantum computing in financial mathematics. Your insights and guidance on identifying relevant applications and noteworthy results have been invaluable to my research.

I extend my sincere thanks to Prof. Wolfgang Lechner from the University of Innsbruck for graciously allowing me to join your group and for sharing your extensive knowledge of quantum computing. Our discussions on suitable quantum algorithms, along with your constructive feedback on my work and insights into your group's research, have significantly enriched my understanding.

I am also grateful to the Financial Mathematics department at the Fraunhofer Institute for Industrial Mathematics ITWM, for facilitating this work. I would like to specifically acknowledge my colleagues, Ivica Turkalj, Mark-Oliver Wolf, and Roman Horsky, for their engaging discussions and constructive feedback.

A special thank you goes to my mother, Brita Schneider, for her valuable input on all design-related aspects of this work.

Lastly, I want to convey my deepest appreciation to Christine for your continuous support and encouragement throughout this journey.

Contents

Zusammenfassung

Optionen sind Finanzinstrumente, deren Auszahlungen durch den Wert eines zugrunde liegenden Vermögenswerts bestimmt werden. Diese zugrunde liegenden Vermögenswerte sind typischerweise Aktien, Zinssätze oder Wechselkurse, können aber auch Commodities wie Energie oder Agrarprodukte umfassen. Optionen dienen unterschiedlichen Zwecken, darunter Absicherung gegen Preisschwankungen und Spekulation auf Marktbewegungen. Darüber hinaus spielen sie eine entscheidende Rolle bei der Bewertung bestimmter Positionen in den Bilanzen von Versicherungsunternehmen, insbesondere im Kontext der Berechnung der Solvenzkapitalanforderungen. Da Optionen erst in der Zukunft ausgezahlt werden und diese Auszahlung, durch die Abhängigkeit von dem zukünftigen Preis des zugrunde liegenden Wertes, stochastisch ist, ist die Bestimmung des fairen Preises für heute eine nicht triviale Aufgabe. Der Fokus dieser Arbeit liegt auf der Suche nach Quantenalgorithmen zur Berechnung dieser Preise und dem Vergleich mit Algorithmen, die für klassische Computer entwickelt wurden. Diese Arbeit verfolgt zwei verschiedene Ansätze, um die klassischen Methoden der Optionsbewertung zu verbessern. Der erste Ansatz besteht darin, Monte-Carlo-Methoden zu verbessern, und der zweite darin, Fourier-basierte Bewertungsalgorithmen anzupassen.

Der Amplitudenschätzalgorithmus (AE) verspricht eine quadratische Beschleunigung im Vergleich zur klassischen Monte-Carlo. Das macht diesen Algorithmus zu einem interessanten Kandidaten für Verbesserungen durch Quantenalgorithmen. In der Literatur wurde gezeigt, dass einfache Optionen mit sehr grober Diskretisierung der zugrunde liegenden stochastischen Modelle auf einem Quantencomputer implementiert und mit Amplitudenschätzung ausgewertet werden können. Eine große Herausforderung ist die effiziente Implementierung des Auszahlungsprofils der Option. Um dieses Problem zu lösen, haben Kollegen und ich einen multikriteriellen genetischen Algorithmus vorgestellt, zur automatischen Suche von leistungsstarken und effizienten Quantenschaltkreise, die die Auszahlung pfadunabhängiger europäischer Optionen, mit einem oder mehreren zugrunde liegenden Vermögenswerten, implementieren.

Eine weitere Strategie zur Bewertung von Optionen basiert darauf, dass in vielen Modellen die charakteristische Funktion des zugrunde liegenden Vermögenswerts sich numerisch besser erfassen lässt als seine Verteilungsfunktion. Es gibt Methoden, die sich diese Tatsache zunutze machen und die Leistungsfähigkeit der schnellen Fourier Transformation nutzen, um die Preise für europäische Call-Optionen für viele Ausübungspreise auf einmal effizient zu berechnen. Die Quantenversion der schnellen Fourier Transformation, die Quanten Fourier Transformation (QFT), ist noch leistungsfähiger als ihr klassisches Pendant. Wir haben die bestehende klassische Methode angepasst, um die QFT zu nutzen und von ihrer Leistung zu profitieren.

Schließlich werden die verschiedenen Strategien getestet und hinsichtlich ihrer Genauigkeit, ihres Skalierungspotenzials und ihrer Anpassungsfähigkeit an unterschiedliche Optionen und Modelle verglichen. Das Ergebnis dieses Vergleichs sind Anforderungen an die Quantenhardware, damit die Methoden praktikabel sind, sowie Anwendungsfälle, in denen jede Methode ihre Vorteile gegenüber den anderen hat.

Abstract

Options are financial instruments whose payoffs are determined by the value of an underlying asset. These underlying assets are typically stocks, interest rates, or exchange rates, but can also include commodities such as energy or agricultural products. Options serve diverse purposes, including hedging against price volatility and speculating on market movements. Additionally, they play a critical role in assessing certain positions on the balance sheets of insurance companies, particularly in the context of solvency capital requirement calculations. As an option's payoff lies in the future and is stochastic in nature, finding a fair price today is a non-trivial task. The focus of this work is finding quantum algorithms for calculating these prices and compare them to algorithms designed for classical computers. This work follows two different directions on how to improve on classical methods for option pricing. The first direction is improving on Monte-Carlo (MC) methods and the second one is adapting Fourier based pricing algorithms.

The Amplitude Estimation (AE) algorithm promises a quadratic speedup compared to classical MC. This fact makes this algorithm an interesting candidate for improvements by quantum algorithms. It was shown in the literature, that simple options with very crude discretization of the underlying stochastic models, can be implemented on a quantum device and evaluated with AE. One big challenge is the efficient implementation of the option's payoff profile. To solve this problem colleagues and I have introduced a multi-objective genetic algorithm to automatically find well performing and efficient circuits to implement the payoff of non-path-dependent European options on one or more underlyings.

Another strategy for pricing options is based on the fact, that in many models the characteristic function of the underlying is numerically more tractable than its distribution function. There are methods that make use of this fact and the performance of the Fast Fourier Transform (FFT) to efficiently calculate prices for European call options for many strikes in one go. The quantum version of the FFT, the Quantum Fourier Transform (QFT), is in some sense, even faster than its classical counterpart. We have adapted the existing classical method to make use of QFT, to benefit from its performance.

Finally, the different strategies are benchmarked and compared regarding their accuracy, their potential for scaling and their adaptability to different styles of derivatives and models. The result of this comparison are requirements on the quantum hardware for the methods to be viable as well as use cases where each method has its benefits over the others.

Chapter 1

Contents

1 Introduction

Quantum computers were first talked about in the 1980s, motivated by the difficulty to describe quantum states with classical computers. Shortly after, first algorithms that utilize the unique properties of quantum computers were developed. In recent years, the concept of a quantum computer has been implemented in actual hardware, sparking extensive research into practical applications for these devices.

In the literature, there are a few quantum algorithms known, that are in some sense better in solving specific problems, than all known classical algorithms. One of them, Amplitude Estimation (AE), will be used in this work and explained in detail in section 2.2.3. Although, in theory, quantum computers using these algorithms outperform classical computers, they come with severe limitations as well. Some of them are technical in nature and will, hopefully, be solved in the future. Some examples are measurement errors, bit flips or unwanted decoherence of the quantum state. Others are inherent to quantum computers and need to be overcome by algorithms. Two prominent examples are the stochastic nature of a quantum algorithm's results and a quantum computer's difficulties with basic arithmetic.

The stochastic nature means, that a quantum algorithm yields a quantum state, that needs to be measured. This measurement process is in general stochastic, the quantum state only determines which results are observed with which probabilities.

The problems with simple arithmetic comes from the physical principal in quantum computing, that all operations need to be reversible. Which, simple addition, for example, is not. We know that 0 plus 1 is 1, but just from knowing that the sum of two numbers is 1 we do not know what these numbers were.

A central challenge of current research in quantum computing is finding use cases where classical computers are lacking today and quantum computers offer an improvement. The goal of this thesis is, motivated by the work of Stamatopoulos et al. [63], to explore the applicability of quantum computing to option pricing.

As this work is aimed at readers with a background in financial mathematics as well as in quantum computing, both parts will be introduced without requiring any previous knowledge of the respective topic.

We will start with option pricing in section 1.1 and continue with quantum computing in section 1.2. In section 1.3 we will introduce the ZX-calculus, which may be of interest for readers with a background in quantum computing as well.

1.1 Option Pricing

Options are a special kind of financial product that form a contract between the buyer, also called holder, and the seller. Depending on some underlying, which can be stocks, interest rates, commodities or basically anything that is traded at exchanges, the holder

of the option gets either a strictly positive payoff or nothing. Historically these contracts gave the owner the right, but not the obligation, to either buy or sell the underlying at a specified time in the future at a fixed price, the strike. When a buyer has the right to purchase, we call the contract a Call. When the buyer has the right to sell, we call it a Put. The execution of that right is optional, hence the name option.

Nowadays, this right is usually settled by paying the cash that would be gained by, in case of a Call, executing the option and directly selling the underlying at market price, or in the case of a Put, buying the underlying at the market and then executing the option.

More generally an option promises a positive payoff that is a functional of the price process of the underlying until its maturity.

Definition 1.1 Option

Given an underlying $(S_t)_{t\geq 0}$ where $S_t \in \mathbb{R}^d$ for some $d \in \mathbb{N}_{\geq 0}$ an option consists of a maturity $T > 0$ and a payoff function $\Pi : (\mathbb{R}^d)^{[0,T]} \to \mathbb{R}_{\geq 0}$. The payoff of the option is given as $\Pi\Big((S_t)_{0\leq t\leq T}\Big)$ and is always non-negative.

In some cases the payoff only depends on the final value of the underlying, in these cases the options are called European options.

Definition 1.2 European Option

An option where the payoff only depends on the underlying's value at the maturity is called a European option. The payoff is then written as $\Pi(S_T)$.

For the remainder of this work we will only consider European options.

Options guarantee a non-negative payoff in the future, this means they have a worth. At the maturity this worth is straight forward to calculate, it is just the payoff. The interesting question is determining the value of an option before its maturity, as this value can be considered the option's fair price. The problem here is, that we do not what the value of the underlying will be at maturity. For the scope of this paper it will suffice to take the discounted expected payoff under the so called risk-neutral measure as the fair price. A more detailed explanation of what exactly the risk-neutral measure is, and why it is important to use it, is for example, presented by Korn and Korn [43].

For this work we will simply define what we will consider the fair price of a European option.

Definition 1.3 Fair Price of a European Option

The fair price of a European Option under the measure $\mathbb{Q}$ is given as

$$C_T := \mathrm{e}^{-rT}\mathbb{E}_{\mathbb{Q}}[\Pi(S_T)],$$

where $r \in \mathbb{R}$ is the constant interest rate. For simplicity, we omit the measure $\mathbb{Q}$ for the remainder and just write

$$C_T = \mathrm{e}^{-rT}\mathbb{E}[\Pi(S_T)]. \tag{1.1}$$

If S_T has a density f_{S_T} we can also write

$$C_T = \mathrm{e}^{-rT} \int_{\mathbb{R}} \Pi(S) f_{S_T}(X) \mathrm{d}S.$$

Note that we have assumed a constant interest rate, which could be extended to stochastic interest rate in more complex models.

To calculate this expectation, we have to assume some model for the underlying asset. Often this is done by assuming the asset price behaves according to a Stochastic Differential Equation (SDE). The most famous example for such a model is the Black-Scholes model.

Example 1.4 Black-Scholes Model

As a benchmark example we will use the Black-Scholes model. In this model the asset price is modeled by a geometric Brownian motion

$$\mathrm{d}S_t = \mu S_t \mathrm{d}t + \sigma S_t \mathrm{d}W_t$$

or explicitly

$$S_t = S_0 \exp\left(\left(\mu - \frac{\sigma^2}{2}\right)t + \sigma W_t\right).$$

This is the dynamic of the stock price under the real world measure. For option pricing, we need the dynamic under the risk neutral measure instead. Thus, we replace the drift μ by the riskless interest rate r and get

$$\mathrm{d}S_t = r S_t \mathrm{d}t + \sigma S_t \mathrm{d}W_t,$$

or explicitly

$$S_t = S_0 \exp\left(\left(r - \frac{\sigma^2}{2}\right)t + \sigma W_t\right).$$

In later parts of this work we will need the characteristic function of the logarithm of the price. Whenever we look at the logarithm of a value, we replace capital letters by small letters, this leads to

$$s_t := \ln(S_t) = s_0 + \left(r - \frac{\sigma^2}{2}\right)t + \sigma W_t. \tag{1.2}$$

Since $(W_t)_{t\geq 0}$ is a Brownian motion, we know that s_t is normally distributed with mean $s_0 + \left(r - \frac{\sigma^2}{2}\right)t$ and variance $\sigma^2 t$, i.e.

$$s_t \sim \mathcal{N}\left(s_0 + \left(r - \frac{\sigma^2}{2}\right)t, \sigma^2 t\right).$$

Since the logarithm of S_t is normally distributed, S_t itself is log-normally distributed. Thus, we know the characteristic function is given as

$$\Phi_{s_t}(u) = \exp\left(\mathbf{i}u\left(s_0 + \left(r - \frac{\sigma^2}{2}\right)t\right) - \frac{\sigma^2 t u^2}{2}\right).$$

This model is well suited to benchmark numerical methods, as it allows for closed formulas for some options. For all later benchmarks we use the following parameters:

$$S_0 = 100, \qquad r = 0.05, \qquad \sigma = 0.3, \qquad T = 0.5.$$

1.2 Quantum Computing

This part of the introduction is intended to give a broad idea of the quantum computing concepts used in this work and to establish the notation we will use later on. Readers already familiar with quantum computing may safely skip ahead to section 1.3, where we will introduce the ZX-calculus, which may be novel even for readers from the quantum computing field. For a more detailed introduction into quantum computing we defer to the works by Kaye, Laflamme, and Mosca [40] or Nielsen and Chuang [53].

Quantum computers, as the name hints at, are using quantum systems for computation. This leads to them having notably different properties compared to classical computers. They were first thought about in the early 1980s, independently by Feynman [29] and Yuri Manin. The latter published his thoughts in Russian and the publication is not accessible by the author, but a more recent publication by the same author is available [48] and references to that first publication as *Computable and uncomputable*.

1.2.1 Qubits

From classical computers we know the bit, the basic unit of information. Bits can be either zero or one and everything a computer does can be broken down to manipulating bits.

The basic unit of information of a quantum computer is a qubit. Qubits also have two possible states, called $|0\rangle$ and $|1\rangle$. But instead of always being in one state or the other, qubits can also be somewhere in between. Only when observed, they will *decide* on one of the states. Of course a qubit does not actually decide anything, we will at a later point discuss the implications of observing, also called measuring, a qubit.

To describe and work with qubits, we use the Dirac, or also Bra-ket, notation [23].

Definition 1.5 Dirac or Bra-ket notation

Given a vector space V we write $|v\rangle$ for an element from that space and call it a ket. Usually we talk about the special case where $V = \mathbb{C}^N$, and we can represent $|v\rangle$ by a complex vector $\mathbf{v} \in \mathbb{C}^N$ using the standard basis.

The second part of the Bra-ket notation, the bra, is written as $\langle f|$ and is formally an element of the dual space of V, i.e., $\langle f| : V \to \mathbb{C}$. When we write $\langle f|v\rangle$, we call this a braket, and it represents the complex number we get by applying the bra $\langle f|$ to the ket $|v\rangle$.

For the special case of $V = \mathbb{C}^N$ with its inner product $(\cdot, \cdot)$ the mapping $\langle f|$ is given as $(f, \cdot)$ and, by the Riesz representation theorem, can be identified with another element $\mathbf{f} \in \mathbb{C}^N$ where

$$\begin{aligned} \langle f| &: \mathbb{C}^N \to \mathbb{C} \\ \mathbf{v} = |v\rangle &\mapsto \langle f|v\rangle = (\mathbf{f}, \mathbf{v}) = \mathbf{f}^\dagger \mathbf{v}. \end{aligned}$$

For the special case we are interested in, $V = \mathbb{C}^N$, the short version states, bras are

row vectors, kets are column vectors and the complex conjugation transforms a bra into a ket, i.e., $\langle v| = |v\rangle^\dagger$.

With this notation we can formalize the description of a qubit.

Definition 1.6 Qubit

A qubit is a two-state quantum system with the states $|0\rangle$ and $|1\rangle$. The quantum state $|\psi\rangle$ of a qubit is described by a linear combination of these two states

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle$$

where $\alpha, \beta \in \mathbb{C}$ and $|\alpha|^2 + |\beta|^2 = 1$. The two complex numbers α and β are called the amplitudes of the states $|0\rangle$ and $|1\rangle$ respectively.

If we decide to use the standard basis for $\mathbb{C}^2$, we can identify $|0\rangle$ with $\left(\begin{smallmatrix}1\\0\end{smallmatrix}\right)$ and $|1\rangle$ with $\left(\begin{smallmatrix}0\\1\end{smallmatrix}\right)$. With that we can identify $|\psi\rangle$ with $\left(\begin{smallmatrix}\alpha\\\beta\end{smallmatrix}\right)$. In this notation the assumption on the amplitudes can be expressed as

$$\left\|\begin{pmatrix}\alpha\\\beta\end{pmatrix}\right\|_2^2 = 1.$$

For the rest of this work we will always use the standard basis, when expressing quantum states as complex vectors. Another often used basis is the Hadamard basis

$$|+\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix}1\\1\end{pmatrix}, \tag{1.3}$$

$$|-\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix}1\\-1\end{pmatrix}. \tag{1.4}$$

Now let us specify what happens when observing, or measuring, a qubit.

Definition 1.7 Measurement

Given the quantum state of a qubit $|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle$, when measuring in the standard basis, the probability to observe $|0\rangle$ is $|\alpha|^2$ and the probability to observe $|1\rangle$ is $|\beta|^2$.

More formally, we call the random variable that measures a state $|\psi\rangle$

$$\mathcal{M}(|\psi\rangle) : \Omega \mapsto \{0, 1\}.$$

The probability to measure $j \in \{0, 1\}$ is then given as

$$\mathbb{P}[\mathcal{M}(|\psi\rangle) = j] = \langle\psi|\,|j\rangle\langle j|\,|\psi\rangle = |\langle j|\psi\rangle|^2. \tag{1.5}$$

The matrix $|j\rangle\langle j|$ is called the measurement operator. In general any self-adjoint operator $\mathbf{M}$ can be the measurement operator, also called observable. In that case

$$\langle\psi|\,\mathbf{M}\,|\psi\rangle \in \mathbb{C}$$

represents the expected value when measuring the observable $\mathbf{M}$.

From this definition we also see, why we require the squared absolute amplitudes to add up to one, the probabilities of all possible outcomes of a random variable need to add up to one.

Another property can be derived from measuring, global phases do not matter.

Lemma 1.8

Two quantum states $|\psi\rangle$ and $|\varphi\rangle$ that only differ by a phase, i.e.,

$$|\varphi\rangle = \mathrm{e}^{\theta\mathbf{i}} |\psi\rangle ,$$

are equivalent, i.e., they cannot be differentiated by measuring.

Proof. The only access we have to quantum states is by measuring, thus distinguishing two states can only be done by measuring. First we note that

$$\langle\varphi| = |\varphi\rangle^\dagger = \left(\mathrm{e}^{\theta\mathbf{i}} |\psi\rangle\right)^\dagger = \mathrm{e}^{-\theta\mathbf{i}} \langle\psi| .$$

With that we can show that the probabilities to measure each state are equal for $|\psi\rangle$ and $|\varphi\rangle$.

$$\begin{aligned}
\mathbb{P}[\mathcal{M}(|\varphi\rangle) = j] &\overset{(1.5)}{=} \langle\varphi|j\rangle \langle j|\varphi\rangle \\
&= \mathrm{e}^{-\theta\mathbf{i}} \langle\psi|j\rangle \langle j| \mathrm{e}^{\theta\mathbf{i}} |\psi\rangle \\
&= \langle\psi|j\rangle \langle j|\psi\rangle \\
&\overset{(1.5)}{=} \mathbb{P}[\mathcal{M}(|\psi\rangle) = j],
\end{aligned}$$

for $j \in \{0, 1\}$. □

With the fact, that global phase does not matter, we can visualize the four real numbers used to describe a quantum state in three dimensions. And since we also know, that the squared norm is always one, we can interpret quantum states as points on the unit sphere in $\mathbb{R}^3$. We call this sphere for visualizing quantum states the Bloch sphere.

To calculate the coordinates of a quantum state on the Bloch sphere we start by factoring out a global phase to make the coefficient of $|0\rangle$ real. We call the state without that global phase $|\psi\rangle = \alpha |0\rangle + \beta |1\rangle$. Now that $\alpha \in \mathbb{R}$ we can find $\theta \in [0, 2\pi]$ such that

$$\alpha = \cos\left(\frac{\theta}{2}\right).$$

Since $|\alpha|^2 + |\beta|^2 = 1$ we know that

$$|\beta| = \sqrt{1 - \alpha^2} = \sqrt{1 - \cos^2\left(\frac{\theta}{2}\right)} = \sin\frac{\theta}{2}.$$

Thus, we can find $\phi \in [0, 2\pi]$ such that

$$\beta = \mathrm{e}^{\mathbf{i}\phi} \sin\left(\tfrac{\theta}{2}\right).$$

The two real values θ and ϕ, interpreted as angles, uniquely define a point on the unit sphere visualized in figure 1.1.

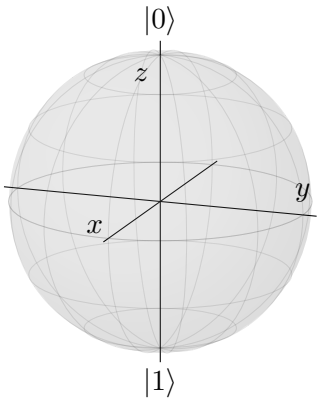


Figure 1.1: The Bloch sphere can be used to visualize the quantum state of a single qubit. All valid quantum states can be represented by a point on the surface of the sphere.

1.2.2 Quantum Gates

Now that we have the basic unit of information, the next step is manipulating this information. Since we already have the quantum states as complex vectors, it makes sense to represent manipulations of these states as mappings from $\mathbb{C}^2$ onto itself. We expect that the amplitudes of all states add up to one, also after we have applied such a mapping $\mathbf{U}$, thus for every $|\phi\rangle \in \mathbb{C}^2$ we want

$$\|\mathbf{U}\,|\phi\rangle\|_2 = 1 = \||\phi\rangle\|_2.$$

Linear algebra tells us that this norm preserving property is equivalent to $\mathbf{U}$ being a unitary operator.

Definition 1.9 Single-Qubit Gate

A single-qubit gate is a quantum operation performed on a single qubit. The operation is represented by a unitary matrix $\mathbf{U} \in \mathbb{C}^{2\times 2}$. Some gates depend on up to three parameters, in these cases we write $\mathbf{U}(\theta)$ for $\theta \in \mathbb{R}^d$ and $d \in \{1, 2, 3\}$.

All single-qubit quantum operations can be interpreted as rotations on the Bloch sphere. While basically every unitary mapping from $\mathbb{C}^2$ onto itself can be interpreted as a quantum gate, in practice only some specific operations are used. In the following example we introduce a popular selection.

Example 1.10 Single-Qubit Gates

In this example we introduce quantum gates that are commonly used, and also used later in this work. For every gate we state the unitary matrix by which they are represented and visualize its effect on the Bloch sphere. The **green** arrow will represent the state before the gate is applied and the **blue** one the state after.

- Although trivial, we will need the identity as a gate doing nothing

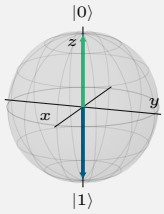
$$\mathbf{I} := \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

- The X gate, NOT gate or Pauli-X gate is the quantum version of a classical NOT gate, that simply flips a bit. It is represented by

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

Applying the X gate, for example, to the state $|0\rangle$ leads to

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

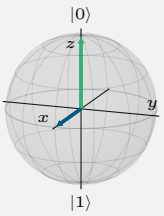


- The Hadamard gate introduces superposition. It transforms for example the state $|0\rangle$ into the equal superposition of $|0\rangle$ and $|1\rangle$. It is represented by

$$\mathbf{H} := \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}. \tag{1.6}$$

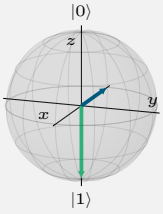
Applying this gate to the states $|0\rangle$ and $|1\rangle$ yields

$$\mathbf{H}|0\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}\begin{pmatrix} 1 \\ 0 \end{pmatrix} = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 \\ 1 \end{pmatrix} = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) =: |+\rangle,$$



respectively

$$\mathbf{H}|1\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}\begin{pmatrix} 0 \\ 1 \end{pmatrix} = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 \\ -1 \end{pmatrix} = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) =: |-\rangle.$$

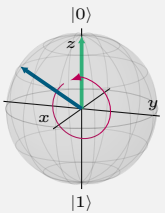


- The RX gate, or Pauli X gate, is a parameterized rotation around the x-axis of the Bloch sphere.

$$\mathbf{R}_X(\theta) := \begin{pmatrix} \cos\left(\frac{\theta}{2}\right) & -\mathbf{i}\sin\left(\frac{\theta}{2}\right) \\ -\mathbf{i}\sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right) \end{pmatrix}. \tag{1.7}$$

Applying this gate to base state $|0\rangle$ we get

$$\mathbf{R}_X(\theta)|0\rangle = \begin{pmatrix} \cos\left(\frac{\theta}{2}\right) & -\mathbf{i}\sin\left(\frac{\theta}{2}\right) \\ -\mathbf{i}\sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right) \end{pmatrix}\begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} \cos\left(\frac{\theta}{2}\right) \\ -\mathbf{i}\sin\left(\frac{\theta}{2}\right) \end{pmatrix}.$$

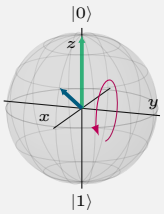


- The RY gate, or Pauli Y gate, is a parameterized rotation around the y-axis of the Bloch sphere.

$$\mathbf{R}_Y(\theta) := \begin{pmatrix} \cos\left(\frac{\theta}{2}\right) & -\sin\left(\frac{\theta}{2}\right) \\ \sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right) \end{pmatrix}.$$

  Applying this gate to base state $|0\rangle$ we get

$$\mathbf{R}_Y(\theta)\,|0\rangle = \begin{pmatrix} \cos\left(\frac{\theta}{2}\right) & -\sin\left(\frac{\theta}{2}\right) \\ \sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right) \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} \cos\left(\frac{\theta}{2}\right) \\ \sin\left(\frac{\theta}{2}\right) \end{pmatrix}.$$

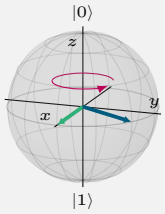


- The RZ gate, or Pauli Z gate, is a parameterized rotation around the z-axis of the Bloch sphere.

$$\mathbf{R}_Z(\theta) := \begin{pmatrix} e^{-\mathbf{i}\frac{\theta}{2}} & 0 \\ 0 & e^{\mathbf{i}\frac{\theta}{2}} \end{pmatrix}. \tag{1.8}$$

  Applying this gate to the state $|+\rangle$ we already saw as result of the Hadamard gate yields

$$\mathbf{R}_Z(\theta)\,|+\rangle = \begin{pmatrix} e^{-\mathbf{i}\frac{\theta}{2}} & 0 \\ 0 & e^{\mathbf{i}\frac{\theta}{2}} \end{pmatrix} \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} e^{-\mathbf{i}\frac{\theta}{2}} \\ e^{\mathbf{i}\frac{\theta}{2}} \end{pmatrix}.$$



The three simple rotation gates have a nice additive property which we will need in later chapters.

Proposition 1.11

For the three rotations, $\mathbf{R}_Z$, $\mathbf{R}_Y$ and $\mathbf{R}_X$, applying two of the same after each other is the same as applying one rotation with the sum of the angles. In formulas, for $\alpha, \beta \in [0, 2\pi] \in \mathbb{R}$

$$\begin{aligned} \mathbf{R}_X(\alpha)\mathbf{R}_X(\beta) &= \mathbf{R}_X(\alpha+\beta), \\ \mathbf{R}_Y(\alpha)\mathbf{R}_Y(\beta) &= \mathbf{R}_Y(\alpha+\beta), \\ \mathbf{R}_Z(\alpha)\mathbf{R}_Z(\beta) &= \mathbf{R}_Z(\alpha+\beta). \end{aligned}$$

Proof. This is easily shown by calculating the matrix product and using the trigonometric identities from equations (B.3) and (B.4). First for the X rotation

$$
\begin{aligned}
&\mathbf{R}_X(\alpha)\mathbf{R}_X(\beta) \\
&\quad= \begin{pmatrix} \cos\left(\frac{\alpha}{2}\right) & -\mathbf{i}\sin\left(\frac{\alpha}{2}\right) \\ -\mathbf{i}\sin\left(\frac{\alpha}{2}\right) & \cos\left(\frac{\alpha}{2}\right) \end{pmatrix} \begin{pmatrix} \cos\left(\frac{\beta}{2}\right) & -\mathbf{i}\sin\left(\frac{\beta}{2}\right) \\ -\mathbf{i}\sin\left(\frac{\beta}{2}\right) & \cos\left(\frac{\beta}{2}\right) \end{pmatrix} \\
&\quad= \begin{pmatrix} \cos\left(\frac{\alpha}{2}\right)\cos\left(\frac{\beta}{2}\right) - \sin\left(\frac{\alpha}{2}\right)\sin\left(\frac{\beta}{2}\right) & -\mathbf{i}\cos\left(\frac{\alpha}{2}\right)\sin\left(\frac{\beta}{2}\right) - \mathbf{i}\sin\left(\frac{\alpha}{2}\right)\cos\left(\frac{\beta}{2}\right) \\ -\mathbf{i}\sin\left(\frac{\alpha}{2}\right)\cos\left(\frac{\beta}{2}\right) - \mathbf{i}\cos\left(\frac{\alpha}{2}\right)\sin\left(\frac{\beta}{2}\right) & -\sin\left(\frac{\alpha}{2}\right)\sin\left(\frac{\beta}{2}\right) + \cos\left(\frac{\alpha}{2}\right)\cos\left(\frac{\beta}{2}\right) \end{pmatrix} \\
&\quad= \begin{pmatrix} \cos\left(\frac{\alpha+\beta}{2}\right) & -\mathbf{i}\sin\left(\frac{\alpha+\beta}{2}\right) \\ -\mathbf{i}\sin\left(\frac{\alpha+\beta}{2}\right) & \cos\left(\frac{\alpha+\beta}{2}\right) \end{pmatrix} \\
&\quad= \mathbf{R}_X(\alpha+\beta).
\end{aligned}
$$

Then for the Y rotation with the same trigonometric identities

$$
\begin{aligned}
&\mathbf{R}_Y(\alpha)\mathbf{R}_Y(\beta) \\
&\quad= \begin{pmatrix} \cos\left(\frac{\alpha}{2}\right) & -\sin\left(\frac{\alpha}{2}\right) \\ \sin\left(\frac{\alpha}{2}\right) & \cos\left(\frac{\alpha}{2}\right) \end{pmatrix} \begin{pmatrix} \cos\left(\frac{\beta}{2}\right) & -\sin\left(\frac{\beta}{2}\right) \\ \sin\left(\frac{\beta}{2}\right) & \cos\left(\frac{\beta}{2}\right) \end{pmatrix} \\
&\quad= \begin{pmatrix} \cos\left(\frac{\alpha}{2}\right)\cos\left(\frac{\beta}{2}\right) - \sin\left(\frac{\alpha}{2}\right)\sin\left(\frac{\beta}{2}\right) & -\cos\left(\frac{\alpha}{2}\right)\sin\left(\frac{\beta}{2}\right) - \sin\left(\frac{\alpha}{2}\right)\cos\left(\frac{\beta}{2}\right) \\ \sin\left(\frac{\alpha}{2}\right)\cos\left(\frac{\beta}{2}\right) + \cos\left(\frac{\alpha}{2}\right)\sin\left(\frac{\beta}{2}\right) & -\sin\left(\frac{\alpha}{2}\right)\sin\left(\frac{\beta}{2}\right) + \cos\left(\frac{\alpha}{2}\right)\cos\left(\frac{\beta}{2}\right) \end{pmatrix} \\
&\quad= \begin{pmatrix} \cos\left(\frac{\alpha+\beta}{2}\right) & -\sin\left(\frac{\alpha+\beta}{2}\right) \\ \sin\left(\frac{\alpha+\beta}{2}\right) & \cos\left(\frac{\alpha+\beta}{2}\right) \end{pmatrix} \\
&\quad= \mathbf{R}_Y(\alpha+\beta).
\end{aligned}
$$

And finally for the Z rotation

$$
\mathbf{R}_Z(\alpha)\mathbf{R}_Z(\beta) = \begin{pmatrix} \mathrm{e}^{-\mathbf{i}\frac{\alpha}{2}} & 0 \\ 0 & \mathrm{e}^{\mathbf{i}\frac{\alpha}{2}} \end{pmatrix} \begin{pmatrix} \mathrm{e}^{-\mathbf{i}\frac{\beta}{2}} & 0 \\ 0 & \mathrm{e}^{\mathbf{i}\frac{\beta}{2}} \end{pmatrix} = \begin{pmatrix} \mathrm{e}^{-\mathbf{i}\frac{\alpha+\beta}{2}} & 0 \\ 0 & \mathrm{e}^{\mathbf{i}\frac{\alpha+\beta}{2}} \end{pmatrix} = \mathbf{R}_Z(\alpha+\beta).
$$

□

1.2.3 Multi-Qubit Systems

Everything discussed until now was only regarding a single qubit. With single qubits we can already observe the first quantum property, the superposition. The second important property, entanglement, comes into play when looking at more than one qubit.

Definition 1.12 Multi-Qubit System

A system of $n \in \mathbb{N}_{\geq 0}$ qubits is called an n-qubit system. Its basis states are all possible combinations of the individual qubits basis states, i.e., the j-th basis state, for $j \in \{0, \dots, 2^n - 1\}$, is given as the Kronecker product of the individual states

$$
\left|e_j^n\right\rangle := \bigotimes_{k=n-1}^{0} \left|j_k\right\rangle .
$$

The $(j_k)_{k\in\{0,\dots,n-1\}}$ are the binary representation of j, i.e.,

$$\sum_{k=0}^{n-1} j_k 2^k = j.$$

See section A for more details on the Kronecker product. There are different notations for these basis states that are beneficial in different situations. We can either write out the string of zeros and ones inside the ket, or we just take the decimal representation of the corresponding integer

$$\left|e_j^n\right\rangle = \bigotimes_{k=n-1}^{0} |j_k\rangle = |j_{n-1}\cdots j_0\rangle = |j\rangle\,.$$

The quantum state $|\psi\rangle$ of the system can then be expressed as

$$|\psi\rangle = \sum_{j=0}^{2^n-1} \psi_j \left|e_j^n\right\rangle \tag{1.9}$$

and identified with the complex vector $(\psi_0 \quad \cdots \quad \psi_{2^n-1})^T \in \mathbb{C}^{2^n}$. Like in the one qubit case, the squares of the amplitudes need to add up to one, i.e.,

$$\sum_{j=0}^{2^n-1} \left|\psi_j\right|^2 \stackrel{!}{=} 1,$$

or equivalently

$$\left\|\psi_j\right\| \stackrel{!}{=} 1.$$

The different styles of notation can be ambitious at some times, for example $|0\rangle$ and $|1\rangle$ can reference either $|e_0^n\rangle$ and $|e_1^n\rangle$, or $\left|e_0^1\right\rangle$ and $\left|e_1^1\right\rangle$. Or $|11\rangle$ can stand in the string representation for $\left|e_3^2\right\rangle$, or in the integer representation for $|e_{11}^n\rangle$. For this reason we try to stick to using $|0\rangle$ and $|1\rangle$ only if we mean the two states of a single qubit and otherwise use the explicit notation of $\left|e_j^n\right\rangle$.

Without entanglement the quantum state of a multi-qubit system is the Kronecker product of the individual qubits, i.e., we can describe it by

$$|\psi\rangle = \bigotimes_{j=n-1}^{0} |\psi_j\rangle$$

where $\left|\psi_j\right\rangle$ is the quantum state of the j-th qubit.

Remark 1.13

Note that we decided here on two conventions regarding notation that are not done equally everywhere. We adapt the notation for example used by the python library qiskit [39].

The first convention is to start indexing everything at zero, the first qubit will be indexed by 0 and the last one by $n-1$. This also holds for the states.

The second convention is on how to order the qubits in a multi-qubit system, i.e., is the first qubit on the left or on the right. Or in formulas, do we write

$$|\psi\rangle = |\psi_0\rangle \otimes \cdots \otimes |\psi_{n-1}\rangle$$

or

$$|\psi\rangle = |\psi_{n-1}\rangle \otimes \cdots \otimes |\psi_0\rangle .$$

We will be using the second version, which has the benefit, that the binary representation of the index of a basis state is the same as the strings of zeros and ones in the Kronecker product.

The next step is to extend the concept of measurements introduced in definition 1.7 to the multi-qubit case.

Definition 1.14 Measurement

Given the quantum state of an n-qubit system $|\psi\rangle \in \mathbb{C}^{2^n}$ we call the random variable that describes measuring the system in the standard basis

$$\mathcal{M}(|\psi\rangle) \colon \Omega \mapsto \{0, \dots, 2^n - 1\}.$$

The probability to measure $j \in \{0, \dots, 2^n - 1\}$ is then given as

$$\mathbb{P}[\mathcal{M}(|\psi\rangle) = j] = \langle\psi| \left|e_j^n\right\rangle\!\left\langle e_j^n\right| |\psi\rangle = \left|\left\langle e_j^n|\psi\right\rangle\right|^2 = |\psi_j|^2. \tag{1.10}$$

The matrix $\left|e_j^n\right\rangle\!\left\langle e_j^n\right| \in \mathbb{C}^{2^n \times 2^n}$ is called the measurement operator.

Sometimes we are only interested in a subset $J \subset \{0, \dots, n-1\}$ of qubits, and only those are measured. For that random variable we write

$$\mathcal{M}_J(|\psi\rangle) \colon \Omega \mapsto \{0, \dots, 2^{|J|} - 1\}.$$

The probability to measure $j \in \{0, \dots, 2^{|J|} - 1\}$ is then given as

$$\mathbb{P}[\mathcal{M}_J(|\psi\rangle) = j] = \langle\psi| \mathbf{M} |\psi\rangle .$$

We build the measurement operator $\mathbf{M}$ as the following product

$$\mathbf{M} := \bigotimes_{k=n-1}^{0} \mathbf{M}_k,$$

where

$$\mathbf{M}_k := \begin{cases} \mathbf{I} & \text{for} \quad k \notin J \\ \left|j_{l(k)}\right\rangle\!\left\langle j_{l(k)}\right| & \text{for} \quad k \in J \end{cases}.$$

Here $(j_l)_{l \in \{0,\dots,|J|-1\}}$ is the binary representation of j and $l(k)$ stands for the k-th qubit of the selected subset J.

Let us have a simple example, to get a better understanding of the concepts introduced.

Example 1.15 Three Qubit System

We will have a look at a three qubit system. The $2^3 = 8$ basis states are given here first as integer representation, then as binary representation and finally by two representations as Kronecker product.

$$\begin{aligned}
\left|e_0^3\right\rangle &= |0\rangle = |000\rangle = |0\rangle \otimes |0\rangle \otimes |0\rangle = \begin{pmatrix}1\\0\end{pmatrix} \otimes \begin{pmatrix}1\\0\end{pmatrix} \otimes \begin{pmatrix}1\\0\end{pmatrix},\\
\left|e_1^3\right\rangle &= |1\rangle = |001\rangle = |0\rangle \otimes |0\rangle \otimes |1\rangle = \begin{pmatrix}1\\0\end{pmatrix} \otimes \begin{pmatrix}1\\0\end{pmatrix} \otimes \begin{pmatrix}0\\1\end{pmatrix},\\
\left|e_2^3\right\rangle &= |2\rangle = |010\rangle = |0\rangle \otimes |1\rangle \otimes |0\rangle = \begin{pmatrix}1\\0\end{pmatrix} \otimes \begin{pmatrix}0\\1\end{pmatrix} \otimes \begin{pmatrix}1\\0\end{pmatrix},\\
\left|e_3^3\right\rangle &= |3\rangle = |011\rangle = |0\rangle \otimes |1\rangle \otimes |1\rangle = \begin{pmatrix}1\\0\end{pmatrix} \otimes \begin{pmatrix}0\\1\end{pmatrix} \otimes \begin{pmatrix}0\\1\end{pmatrix},\\
\left|e_4^3\right\rangle &= |4\rangle = |100\rangle = |1\rangle \otimes |0\rangle \otimes |0\rangle = \begin{pmatrix}0\\1\end{pmatrix} \otimes \begin{pmatrix}1\\0\end{pmatrix} \otimes \begin{pmatrix}1\\0\end{pmatrix},\\
\left|e_5^3\right\rangle &= |5\rangle = |101\rangle = |1\rangle \otimes |0\rangle \otimes |1\rangle = \begin{pmatrix}0\\1\end{pmatrix} \otimes \begin{pmatrix}1\\0\end{pmatrix} \otimes \begin{pmatrix}0\\1\end{pmatrix},\\
\left|e_6^3\right\rangle &= |6\rangle = |110\rangle = |1\rangle \otimes |1\rangle \otimes |0\rangle = \begin{pmatrix}0\\1\end{pmatrix} \otimes \begin{pmatrix}0\\1\end{pmatrix} \otimes \begin{pmatrix}1\\0\end{pmatrix},\\
\left|e_7^3\right\rangle &= |7\rangle = |111\rangle = |1\rangle \otimes |1\rangle \otimes |1\rangle = \begin{pmatrix}0\\1\end{pmatrix} \otimes \begin{pmatrix}0\\1\end{pmatrix} \otimes \begin{pmatrix}0\\1\end{pmatrix}.
\end{aligned}$$

A quantum state then looks like this

$$|\psi\rangle = \sum_{j=0}^{7} \psi_j \left|e_j^3\right\rangle = \begin{pmatrix}\psi_0 & \psi_1 & \psi_2 & \psi_3 & \psi_4 & \psi_5 & \psi_6 & \psi_7\end{pmatrix}^T \in \mathbb{C}^8.$$

When measuring all qubits from a system in the state $|\psi\rangle$ the probabilities to measure each individual state are simply $\left|\psi_j\right|^2$. If we would only measure a subset of the qubits, say $J = \{1, 2\}$ the probability to observe a zero would be

$$\begin{aligned}
\mathbb{P}[\mathcal{M}_J(|\psi\rangle) = 0] &= \langle\psi| \left(|0\rangle\langle 0| \otimes |0\rangle\langle 0| \otimes \mathbf{I}\right) |\psi\rangle\\
&= \langle\psi| \left(\begin{pmatrix}1 & 0\\0 & 0\end{pmatrix} \otimes \begin{pmatrix}1 & 0\\0 & 0\end{pmatrix} \otimes \begin{pmatrix}1 & 0\\0 & 1\end{pmatrix} \right) |\psi\rangle .
\end{aligned}$$

Since

$$\begin{pmatrix}1 & 0\\0 & 0\end{pmatrix} \otimes \begin{pmatrix}1 & 0\\0 & 0\end{pmatrix} \otimes \begin{pmatrix}1 & 0\\0 & 1\end{pmatrix} = \begin{pmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{pmatrix}$$

we have

$$\mathbb{P}[\mathcal{M}_J(|\psi\rangle) = 0] = \begin{pmatrix}\psi_1^\dagger & \psi_2^\dagger & \psi_3^\dagger & \psi_4^\dagger & \psi_5^\dagger & \psi_6^\dagger & \psi_7^\dagger & \psi_8^\dagger\end{pmatrix} \begin{pmatrix}\psi_0\\ \psi_1\\ 0\\ 0\\ 0\\ 0\\ 0\\ 0\end{pmatrix}$$
$$= |\psi_0|^2 + |\psi_1|^2$$
$$= \mathbb{P}[\mathcal{M}(|\psi\rangle) = 0] + \mathbb{P}[\mathcal{M}(|\psi\rangle) = 1].$$

This result fits the intuition, that the probability to observe both qubits in zero while only measuring the second and third qubit should be the same as, while measuring all qubits, the sum of the probabilities of all results where the second and third qubits are zero.

To shorten some long expressions of Kronecker products, we introduce a shorthand notation for repeated terms. For some $\mathbf{U} \in \mathbb{C}^{k\times l}$ we write

$$\mathbf{U}^{\otimes n} := \bigotimes_{j=1}^{n} \mathbf{U}\,.$$

The next step is manipulating multi-qubit systems. We can apply single-qubit gates to one specific qubit in a multi-qubit system. The corresponding operator can, just like the combined states, be constructed by the Kronecker product. If we want to apply a quantum gate represented by the unitary $\mathbf{U} \in \mathbb{C}^{2\times 2}$ to the j-th qubit of an n-qubit system we call this operation $\mathbf{U}^{(\mathrm{j})}$, and it is represented by the matrix

$$\mathbf{U}^{(\mathrm{j})} := \bigotimes_{k=n-1}^{j+1} \mathbf{I} \otimes \mathbf{U} \otimes \bigotimes_{k=j-1}^{0} \mathbf{I}\,. \tag{1.11}$$

In this notation an empty Kronecker product, e.g., $\bigotimes_{k=-1}^{0}$ is simply a scalar 1.

If we want to simultaneously apply gates $\mathbf{U}_j$ to qubits $j \in \{0, \ldots, n-1\}$, we have $\prod_{j=0}^{n-1} \mathbf{U}^{(\mathrm{j})}{}_j = \bigotimes_{j=0}^{n-1} \mathbf{U}_j$.

Up until now, the complexity of quantum states is still manageable. The quantum states size does, in theory, increase exponentially in the number of qubits, but we can still express them as a Kronecker product of the individual states. This means we only need $2n$ instead of 2^n complex numbers to describe a state. This changes once we introduce entanglement.

Definition 1.16 Entanglement

We say that two qubits are entangled, if we can no longer express their combined state as Kronecker product of the individual qubits. Or in other words describing the state of each qubit individually is not enough to describe the system of the two qubits together.

In formulas, a state $|\psi\rangle \in \mathbb{C}^{2^n}$ is called an entangled state if

$$\nexists \left(|\psi_j\rangle\right)_{j=0,\dots,n-1} \subset \mathbb{C}^2 : |\psi\rangle = \bigotimes_{j=n-1}^{0} |\psi_j\rangle .$$

To produce entanglement between qubits, we need gates that manipulate more than one qubit at once.

Definition 1.17 Multi-Qubit Gate

Given an n-qubit system an m-qubit gate, for $1 \leq m \leq n$, is a quantum operation performed on m qubits. The operation is represented by a unitary matrix $\mathbf{U} \in \mathbb{C}^{2^m \times 2^m}$.

The operation can also depend on parameters, in these cases we write $\mathbf{U}(\boldsymbol{\theta})$, for $\boldsymbol{\theta} \in \mathbb{R}^P$ and $P \in \mathbb{N}_{\geq 0}$.

A common variant of multi-qubit gates are controlled operations. The idea is to perform an operation only if a control qubit is in state $|1\rangle$. This seems straight forward, but becomes less clear, once the control qubit is in a superposition state.

Example 1.18 Controlled Operations

A unitary that represents a controlled operation is, independently of the operation that should be controlled, constructed the same way. We start with the most simple example, we want to control a single-qubit gate $\mathbf{U}$ applied to the first qubit, controlled by the second one.

Remember that we decided on the convention to order the qubits from right to left, i.e., the first qubit will be the second term in a Kronecker product.

This operation will be called $\Lambda\left(1, \mathbf{U}^{(0)}\right)$, and it should ensure these relations

$$\Lambda\left(1, \mathbf{U}^{(0)}\right)(|0\rangle \otimes |\psi\rangle) \stackrel{!}{=} |0\rangle \otimes |\psi\rangle , \tag{1.12}$$

$$\Lambda\left(1, \mathbf{U}^{(0)}\right)(|1\rangle \otimes |\psi\rangle) \stackrel{!}{=} |1\rangle \otimes \mathbf{U}|\psi\rangle . \tag{1.13}$$

Meaning if the second qubit is in state $|0\rangle$ nothing happens to the first one and if the second one is in state $|1\rangle$ operation $\mathbf{U}$ should be applied to the first one. The control qubit should not be changed in both cases.

We claim that defining the controlled operation in the following way,

$$\Lambda\left(1, \mathbf{U}^{(0)}\right) := (|0\rangle\langle 0| \otimes \mathbf{I}) + (|1\rangle\langle 1| \otimes \mathbf{U}),$$

ensures that equations (1.12) and (1.13) hold.

First we validate equation (1.12)

$$\begin{aligned}
\Lambda\left(1, \mathbf{U}^{(0)}\right)(|0\rangle \otimes |\psi\rangle) &= ((|0\rangle\langle 0| \otimes \mathbf{I}) + (|1\rangle\langle 1| \otimes \mathbf{U}))(|0\rangle \otimes |\psi\rangle) \\
&= (|0\rangle\langle 0| \otimes \mathbf{I})(|0\rangle \otimes |\psi\rangle) + (|1\rangle\langle 1| \otimes \mathbf{U})(|0\rangle \otimes |\psi\rangle) \\
&= (|0\rangle\langle 0|\,|0\rangle \otimes \mathbf{I}|\psi\rangle) + (|1\rangle\langle 1|\,|0\rangle \otimes \mathbf{U}|\psi\rangle) \\
&= |0\rangle \otimes |\psi\rangle .
\end{aligned}$$

We make use of the fact, that the basis states are orthogonal to each other. This fact is also used to show that equation (1.13) holds.

$$\begin{aligned}\Lambda\left(1, \mathbf{U}^{(0)}\right)(|1\rangle \otimes |\psi\rangle) &= ((|0\rangle\langle 0| \otimes \mathbf{I}) + (|1\rangle\langle 1| \otimes \mathbf{U}))(|1\rangle \otimes |\psi\rangle) \\ &= (|0\rangle\langle 0| \otimes \mathbf{I})(|1\rangle \otimes |\psi\rangle) + (|1\rangle\langle 1| \otimes \mathbf{U})(|1\rangle \otimes |\psi\rangle) \\ &= (|0\rangle\langle 0|\,|1\rangle \otimes \mathbf{I}\,|\psi\rangle) + (|1\rangle\langle 1|\,|1\rangle \otimes \mathbf{U}\,|\psi\rangle) \\ &= |1\rangle \otimes \mathbf{U}\,|\psi\rangle\,.\end{aligned}$$

In the last step, we argue that the operator is actually unitary

$$\begin{aligned}\Lambda\left(1, \mathbf{U}^{(0)}\right)^\dagger \Lambda\left(1, \mathbf{U}^{(0)}\right) &= ((|0\rangle\langle 0| \otimes \mathbf{I}) + (|1\rangle\langle 1| \otimes \mathbf{U}))^\dagger ((|0\rangle\langle 0| \otimes \mathbf{I}) + (|1\rangle\langle 1| \otimes \mathbf{U})) \\ &= \left((|0\rangle\langle 0| \otimes \mathbf{I}) + \left(|1\rangle\langle 1| \otimes \mathbf{U}^\dagger\right)\right)((|0\rangle\langle 0| \otimes \mathbf{I}) + (|1\rangle\langle 1| \otimes \mathbf{U})) \\ &= (|0\rangle\langle 0| \otimes \mathbf{I})(|0\rangle\langle 0| \otimes \mathbf{I}) + (|0\rangle\langle 0| \otimes \mathbf{I})(|1\rangle\langle 1| \otimes \mathbf{U}) \\ &\quad + \left(|1\rangle\langle 1| \otimes \mathbf{U}^\dagger\right)(|0\rangle\langle 0| \otimes \mathbf{I}) + \left(|1\rangle\langle 1| \otimes \mathbf{U}^\dagger\right)(|1\rangle\langle 1| \otimes \mathbf{U}) \\ &= (|0\rangle\langle 0|\,|0\rangle\langle 0| \otimes \mathbf{I}) + (|0\rangle\langle 0|\,|1\rangle\langle 1| \otimes \mathbf{U}) \\ &\quad + \left(|1\rangle\langle 1|\,|0\rangle\langle 0| \otimes \mathbf{U}^\dagger\right) + \left(|1\rangle\langle 1|\,|1\rangle\langle 1| \otimes \mathbf{U}^\dagger\,\mathbf{U}\right) \\ &= (|0\rangle\langle 0| \otimes \mathbf{I}) + (|1\rangle\langle 1| \otimes \mathbf{I}) \\ &= (|0\rangle\langle 0| + |1\rangle\langle 1|) \otimes \mathbf{I} = \mathbf{I}\,.\end{aligned}$$

For the last step note that

$$|0\rangle\langle 0| + |1\rangle\langle 1| = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} = \mathbf{I}\,.$$

This can be generalized to $\Lambda\left(c, \mathbf{U}^{(t)}\right)$ representing the application of operation $\mathbf{U}$ to qubit t controlled by qubit c. For brevity, we will assume that $c < t$, if this does not hold, the corresponding terms have to change places in the constructed operator. We have

$$\begin{aligned}\Lambda\left(c, \mathbf{U}^{(t)}\right) := &\bigotimes_{j=n-1}^{t+1} \mathbf{I} \otimes \mathbf{I} \otimes \bigotimes_{j=t-1}^{c+1} \mathbf{I} \otimes |0\rangle\langle 0| \otimes \bigotimes_{j=c-1}^{0} \mathbf{I} \\ &+ \bigotimes_{j=n-1}^{t+1} \mathbf{I} \otimes \mathbf{U} \otimes \bigotimes_{j=t-1}^{c+1} \mathbf{I} \otimes |1\rangle\langle 1| \otimes \bigotimes_{j=c-1}^{0} \mathbf{I}\,.\end{aligned} \tag{1.14}$$

This can be even further generalized to the controlled application of multi-qubit gates. Let $\mathbf{U}' \in \mathbb{C}^{2^m \times 2^m}$ be unitary, representing an m-qubit operation and $J \subset \{0, \ldots, 2^n - 1\} \setminus \{c\}$ with $|J| = m$, then $\Lambda\left(c, \mathbf{U}'^{(J)}\right)$ represents the application of $\mathbf{U}'$ to the qubits in J controlled by qubit c.

In the following we will have an example that applies the above construction to the NOT gate, to construct the CNOT gate introduced in example 1.10.

Example 1.19 CNOT Gate

Assume a system of $n = 2$ qubits where the first one will be the target, $t = 0$, and the second the control qubit, $c = 1$. The controlled operation is the X gate from example 1.10. We use equation (1.14) to get

$$\begin{aligned}\Lambda\left(1, \mathbf{X}^{(0)}\right) &= |0\rangle\langle 0| \otimes \mathbf{I} + |1\rangle\langle 1| \otimes \mathbf{X} \\ &= \begin{pmatrix} \mathbf{I} & 0 \\ 0 & 0 \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ 0 & \mathbf{X} \end{pmatrix} \\ &= \begin{pmatrix} \mathbf{I} & 0 \\ 0 & \mathbf{X} \end{pmatrix} \\ &= \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}.\end{aligned}$$

Let's validate if this matrix behaves as expected. We write down all basis states of the two-qubit system and write next to it, what we would expect to observe after the controlled NOT gate. The control should never change, and the target if and only if the control is in state $|1\rangle$.

control	target	expected outcome
$\|0\rangle$	$\|0\rangle$	$\|0\rangle \otimes \|0\rangle$
$\|0\rangle$	$\|1\rangle$	$\|0\rangle \otimes \|1\rangle$
$\|1\rangle$	$\|0\rangle$	$\|1\rangle \otimes \|1\rangle$
$\|1\rangle$	$\|1\rangle$	$\|1\rangle \otimes \|0\rangle$

Let us now check that the matrix $\Lambda\left(1, \mathbf{X}^{(0)}\right)$ does as expected

$$\Lambda\left(1, \mathbf{X}^{(0)}\right)(|0\rangle \otimes |0\rangle) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} = |0\rangle \otimes |0\rangle\,,$$

$$\Lambda\left(1, \mathbf{X}^{(0)}\right)(|0\rangle \otimes |1\rangle) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} = |0\rangle \otimes |1\rangle\,,$$

$$\Lambda\left(1, \mathbf{X}^{(0)}\right)(|1\rangle \otimes |0\rangle) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} = |1\rangle \otimes |1\rangle\,,$$

$$\Lambda\left(1, \mathbf{X}^{(0)}\right)(|1\rangle \otimes |1\rangle) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} = |1\rangle \otimes |0\rangle .$$

Indeed, it performs as expected. If we switch control and target we get

$$\begin{aligned} \Lambda\left(0, \mathbf{X}^{(1)}\right) &= \mathbf{I} \otimes |0\rangle\langle 0| + \mathbf{X} \otimes |1\rangle\langle 1| \\ &= \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} + \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \\ &= \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}. \end{aligned}$$

A popular example for entangled states are the Bell states.

Example 1.20 Bell States

Given a two-qubit system the Bell states are the maximally entangled states

$$|\Phi^+\rangle = \frac{1}{\sqrt{2}}(|0\rangle \otimes |0\rangle + |1\rangle \otimes |1\rangle) = \frac{|00\rangle + |11\rangle}{\sqrt{2}} = \begin{pmatrix} \sqrt{2} \\ 0 \\ 0 \\ \sqrt{2} \end{pmatrix},$$

$$|\Phi^-\rangle = \frac{1}{\sqrt{2}}(|0\rangle \otimes |0\rangle - |1\rangle \otimes |1\rangle) = \frac{|00\rangle - |11\rangle}{\sqrt{2}} = \begin{pmatrix} \sqrt{2} \\ 0 \\ 0 \\ -\sqrt{2} \end{pmatrix},$$

$$|\Psi^+\rangle = \frac{1}{\sqrt{2}}(|1\rangle \otimes |0\rangle + |0\rangle \otimes |1\rangle) = \frac{|10\rangle + |01\rangle}{\sqrt{2}} = \begin{pmatrix} 0 \\ \sqrt{2} \\ \sqrt{2} \\ 0 \end{pmatrix},$$

$$|\Psi^-\rangle = \frac{1}{\sqrt{2}}(|1\rangle \otimes |0\rangle - |0\rangle \otimes |1\rangle) = \frac{|10\rangle - |01\rangle}{\sqrt{2}} = \begin{pmatrix} 0 \\ -\sqrt{2} \\ \sqrt{2} \\ 0 \end{pmatrix}.$$

For completeness, we convince ourselves, that these states cannot be represented by the Kronecker product of two single-qubit states. Let us assume we have two states $|a\rangle, |b\rangle \in \mathbb{C}^2$ such that

$$|\Phi^+\rangle = |a\rangle \otimes |b\rangle .$$

That then means that

$$|\Phi^+\rangle = \begin{pmatrix} \sqrt{2} \\ 0 \\ 0 \\ \sqrt{2} \end{pmatrix} = \begin{pmatrix} a_1 b_1 \\ a_1 b_2 \\ a_2 b_1 \\ a_2 b_2 \end{pmatrix} = |a\rangle \otimes |b\rangle \,.$$

But from $a_1 b_1 = \sqrt{2}$ and $a_1 b_2 = 0$ follows $b_2 = 0$, which contradicts with $a_2 b_2 = \sqrt{2}$. For the other Bell states the argumentation is the same.

If we measure only one qubit of such a bell state, we would equally likely measure 0 or 1, but if we measure both, they will always be in the same state, for $|\Phi^\pm\rangle$, or always in the opposite state, for $|\Psi^\pm\rangle$. This behavior can obviously not be explained, by only stating that both individual qubits are in equal superposition of $|0\rangle$ and $|1\rangle$. In this case we would expect to observe both qubits in different states equally often as in the same state.

1.2.4 Quantum Circuits

We have seen, that it is not always possible to write quantum states as Kronecker products of simpler states and the size of the matrices describing quantum operations increase exponentially in the number of qubits. This makes it desirable to have another way of describing quantum algorithms.

One common way to do this, is to describe a quantum algorithm by a quantum circuit.

Definition 1.21 Quantum Circuit

A quantum circuit is a visual representation of a quantum algorithm. It consists of qubits, represented by horizontal lines, and operations, also called quantum gates, applied to one or multiple qubits, represented by different symbols on the horizontal lines. Quantum circuits are read from left to right, in which order the operations are applied.

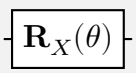
First we repeat the operations we introduced in example 1.10 extended by the CNOT gate from example 1.19, a general controlled operation and a measurement.

Example 1.22 Gates in Circuit Notation

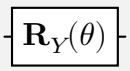
NOT

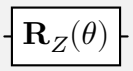
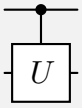
Hadamard

Pauli X/RX

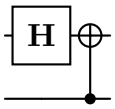


Pauli Y/RY

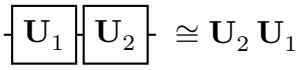


Pauli Z/RZ	
CNOT	
General Controlled Operation	
Measurement	

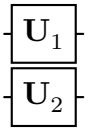
The first Bell state from example 1.20 can be generated by the following circuit.



Translating from a circuit back to a matrix one multiplies the matrices of consecutive gates from right to left and applies the Kronecker product for parallel operations, i.e.,



and



Note that here comes again the convention, which qubit is the first qubit, into play. In such a diagram the qubits are counted from top to bottom.

1.3 ZX-calculus

Another way of representing quantum algorithms is the ZX-calculus. The idea was introduced by Coecke and Duncan [19] in 2008 and the term was first used by the same authors in 2011 [20].

The ZX-calculus is a graphical representation of quantum algorithms, which we will call ZX-diagrams, together with a set of rewrite rules, that change the graph without changing the underlying algorithm. This section will only introduce concepts we will need later on, see for example, the work by van de Wetering [66] for a more detailed introduction.

1.3.1 ZX-diagrams

ZX-diagrams are a graph-like representation of linear mappings. The nodes of these graphs are called spiders and can be one of two types, that are visually differentiated by their color.

Definition 1.23 Z-spider

A Z-spider with $n \in \mathbb{N}$ inputs and $m \in \mathbb{N}$ outputs is the linear map from $\mathbb{C}^{2^n}$ to $\mathbb{C}^{2^m}$ that is given by

$$|0\rangle^{\otimes m} \langle 0|^{\otimes n} + \mathrm{e}^{\mathbf{i}\alpha} |1\rangle^{\otimes m} \langle 1|^{\otimes n}$$

and is visualized by

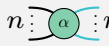
A Z-spider displayed without a phase, always has a phase of 0. Note, that both n and m could be zero.

Definition 1.24 X-spider

An X-spider with $n \in \mathbb{N}$ inputs and $m \in \mathbb{N}$ outputs is the linear map from $\mathbb{C}^{2^n}$ to $\mathbb{C}^{2^m}$ that is given by

$$|+\rangle^{\otimes m} \langle +|^{\otimes n} + \mathrm{e}^{\mathbf{i}\alpha} |-\rangle^{\otimes m} \langle -|^{\otimes n}$$

and is visualized by



An X-spider displayed without the phase, always has a phase of 0.Note, that both n and m could be zero.

Special cases of these two spiders relate to quantum gates in the following way.

Example 1.25

The X and Y-spider translate to the Pauli X and Pauli Y rotations respectively. We start with the X-spider

$$\begin{aligned}
&\equiv \quad |+\rangle\langle +| + \mathrm{e}^{\mathbf{i}\theta} |-\rangle\langle -| \\
&\overset{(1.3),(1.4)}{=} \frac{1}{2}\left(\begin{pmatrix}1\\1\end{pmatrix}(1 \quad 1) + \mathrm{e}^{\mathbf{i}\theta}\begin{pmatrix}1\\-1\end{pmatrix}(1 \quad -1)\right) \\
&= \frac{1}{2}\left(\begin{pmatrix}1 & 1\\1 & 1\end{pmatrix} + \mathrm{e}^{\mathbf{i}\theta}\begin{pmatrix}1 & -1\\-1 & 1\end{pmatrix}\right) \\
&= \frac{1}{2}\begin{pmatrix}1+\mathrm{e}^{\mathbf{i}\theta} & 1-\mathrm{e}^{\mathbf{i}\theta}\\1-\mathrm{e}^{\mathbf{i}\theta} & 1+\mathrm{e}^{\mathbf{i}\theta}\end{pmatrix} \\
&= \mathrm{e}^{-\mathbf{i}\frac{\theta}{2}}\begin{pmatrix}\frac{\mathrm{e}^{\mathbf{i}\frac{\theta}{2}}+\mathrm{e}^{-\mathbf{i}\frac{\theta}{2}}}{2} & \frac{\mathrm{e}^{\mathbf{i}\frac{\theta}{2}}-\mathrm{e}^{-\mathbf{i}\frac{\theta}{2}}}{2}\\ \frac{\mathrm{e}^{\mathbf{i}\frac{\theta}{2}}-\mathrm{e}^{-\mathbf{i}\frac{\theta}{2}}}{2} & \frac{\mathrm{e}^{\mathbf{i}\frac{\theta}{2}}+\mathrm{e}^{-\mathbf{i}\frac{\theta}{2}}}{2}\end{pmatrix} \\
&= \mathrm{e}^{-\mathbf{i}\frac{\theta}{2}}\begin{pmatrix}\cos\left(\frac{\theta}{2}\right) & -\mathbf{i}\sin\left(\frac{\theta}{2}\right)\\ -\mathbf{i}\sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right)\end{pmatrix} \\
&\overset{(1.7)}{=} \mathrm{e}^{-\mathbf{i}\frac{\theta}{2}}\mathbf{R}_X(\theta).
\end{aligned}$$

Next, we have a look at the Z-spider

$$\equiv |0\rangle\langle 0| + \mathrm{e}^{\mathbf{i}\theta}\,|1\rangle\langle 1| = \begin{pmatrix} 1 & 0 \\ 0 & \mathrm{e}^{\mathbf{i}\theta} \end{pmatrix} = \mathrm{e}^{-\mathbf{i}\frac{\theta}{2}} \begin{pmatrix} \mathrm{e}^{-\mathbf{i}\frac{\theta}{2}} & 0 \\ 0 & \mathrm{e}^{\mathbf{i}\frac{\theta}{2}} \end{pmatrix} \overset{(1.8)}{=} \mathrm{e}^{-\mathbf{i}\frac{\theta}{2}} \mathbf{R}_Z(\theta).$$

We see that the represented operations only differ by a phase, which by lemma 1.8 does not matter if we only look at the individual qubit.

A third possible vertex represents the Hadamard gate, it differs from the others, in that it can only have one input and one output edge. Because of this, it can also be viewed as a property of an edge.

Definition 1.26 Hadamard Edge

A Hadamard gate is represented by

and stands for the operation introduced in equation (1.6). To simplify notation, an edge with a Hadamard gate is often replaced by a blue edge, i.e.,

.

We have two possibilities to connect two ZX-diagrams, horizontally and vertically. Connecting them vertically, i.e., stacking them over each other, means the Kronecker product of the two linear maps. Composing two diagrams horizontally, by connecting the output wires of the first with the input wires of the second, means chaining the linear maps.

1.3.2 Rewrite Rules

In itself ZX-diagrams are not really useful. The benefits lie in the ZX-calculus, a set of rules how ZX-diagrams can be manipulated without changing the represented linear map. A comprehensive list of these rewrite rules is, for example, presented by van de Wetering [66] in chapter 4.

An important question is, what ZX-diagrams have to do with quantum circuits.

Definition 1.27 Circuit-like ZX-diagram

A ZX-diagram is called circuit-like, if it has the same number of input and output wires and is composed only of these diagrams:

, , .

First we convince ourselves that the first diagram represents a CNOT gate, $\Lambda\left(0, \mathbf{X}^{(1)}\right)$, from example 1.19. Note, that the ZX-calculus allows rewriting

=.

This again can be split up into a phaseless Z-spider with one input and two outputs parallel to a wire without a spider, and an also phaseless X-spider with two inputs and

one output also parallel to a wire without spider. This allows us to write

$$\equiv \Big(\big(|+\rangle\langle+|^{\otimes 2} + |-\rangle\langle-|^{\otimes 2}\big)\otimes \mathbf{I}\Big)\Big(\mathbf{I}\otimes\big(|0\rangle^{\otimes 2}\langle 0| + |1\rangle^{\otimes 2}\langle 1|\big)\Big).$$

Then we look at the two parts individually and get

$$\begin{aligned}
\big(|+\rangle\langle+|^{\otimes 2} + |-\rangle\langle-|^{\otimes 2}\big)\otimes\mathbf{I} &= \frac{1}{2\sqrt{2}}\left(\begin{pmatrix}1\\1\end{pmatrix}\begin{pmatrix}1 & 1 & 1 & 1\end{pmatrix} + \begin{pmatrix}1\\-1\end{pmatrix}\begin{pmatrix}1 & -1 & -1 & 1\end{pmatrix}\right)\otimes\mathbf{I}\\
&= \frac{1}{2\sqrt{2}}\left(\begin{pmatrix}1&1&1&1\\1&1&1&1\end{pmatrix} + \begin{pmatrix}1&-1&-1&1\\-1&1&1&-1\end{pmatrix}\right)\otimes\mathbf{I}\\
&= \frac{1}{\sqrt{2}}\begin{pmatrix}1&0&0&1\\0&1&1&0\end{pmatrix}\otimes\begin{pmatrix}1&0\\0&1\end{pmatrix}\\
&= \frac{1}{\sqrt{2}}\begin{pmatrix}1&0&0&0&0&0&1&0\\0&1&0&0&0&0&0&1\\0&0&1&0&1&0&0&0\\0&0&0&1&0&1&0&0\end{pmatrix}
\end{aligned}$$

and

$$\begin{aligned}
\mathbf{I}\otimes\big(|0\rangle^{\otimes 2}\langle 0| + |1\rangle^{\otimes 2}\langle 1|\big) &= \mathbf{I}\otimes\left(\begin{pmatrix}1\\0\\0\\0\end{pmatrix}\begin{pmatrix}1 & 0\end{pmatrix} + \begin{pmatrix}0\\0\\0\\1\end{pmatrix}\begin{pmatrix}0 & 1\end{pmatrix}\right)\\
&= \mathbf{I}\otimes\left(\begin{pmatrix}1&0\\0&0\\0&0\\0&0\end{pmatrix} + \begin{pmatrix}0&0\\0&0\\0&0\\0&1\end{pmatrix}\right)\\
&= \begin{pmatrix}1&0\\0&1\end{pmatrix}\otimes\begin{pmatrix}1&0\\0&0\\0&0\\0&1\end{pmatrix} = \begin{pmatrix}1&0&0&0\\0&0&0&0\\0&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&0\\0&0&0&0\\0&0&0&1\end{pmatrix}.
\end{aligned}$$

Put together, this yields

$$\equiv \frac{1}{\sqrt{2}}\begin{pmatrix}1&0&0&0&0&0&1&0\\0&1&0&0&0&0&0&1\\0&0&1&0&1&0&0&0\\0&0&0&1&0&1&0&0\end{pmatrix}\begin{pmatrix}1&0&0&0\\0&0&0&0\\0&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&0\\0&0&0&0\\0&0&0&1\end{pmatrix} = \frac{1}{\sqrt{2}}\begin{pmatrix}1&0&0&0\\0&0&0&1\\0&0&1&0\\0&1&0&0\end{pmatrix}.$$

This is the same as $\Lambda\left(0, \mathbf{X}^{(1)}\right)$ up to the factor $1/\sqrt{2}$. Again remember the ordering of qubits, we count the qubits, also called wires in a ZX-diagram, from top to bottom and in a Kronecker product from right to left.

From the Z-spider and the Hadamard gate one can build the X-spider via the color change rule [66, equation (15)]. These two are equivalent to the $\mathbf{R}_{\mathrm{Z}}$ and the $\mathbf{R}_{\mathrm{X}}$ gates, which in turn can be used to build any single-qubit gate. Single-qubit gates, together with the CNOT gate, form a universal gate set [3]. This means every circuit can be represented by a circuit-like ZX-diagram.

This also gives a straight forward way of transforming a quantum circuit into a ZX-diagram, express the circuit in CNOT, $\mathbf{R}_{\mathrm{Z}}$ and Hadamard gates and then interpret the circuit as a circuit-like ZX-diagram.

The opposite direction is called circuit extraction and less straight forward, Backens et al. [2] give an in depth explanation of the topic. For this work we will only introduce some concepts and notation, that will be useful in section 3.3. The first concept was introduced by Duncan et al. [25].

Definition 1.28 Graph-like ZX-diagram

A ZX-diagram is called graph-like if:

- Every spider in the diagram is a Z-spider;
- Every wire between two Z-spiders is a Hadamard edge;
- There are no parallel Hadamard edges and no self-loops;
- Every input wire and every output wire is connected to a Z-spider;
- Every Z-spider is connected to at most one input wire or output wire.

In [25, lemma 3.2] it was proved, that every ZX-diagram is equal to a graph-like ZX-diagram.

The next concept is the general flow (gFlow). It is a graph theoretical concept with a very technical definition, that goes beyond the scope of this work. We refer to [25] for the full definition.

Definition 1.29 Open Graph [25, definition 3.3]

An open graph is a triple (G, I, O) consisting of an undirected graph $G = (V, E)$ and two subsets of vertices $I, O \subseteq V$ called inputs and outputs, respectively.

Given a graph-like ZX-diagram D the underlying open graph $G(D)$ is an open graph constructed from the ZX-diagram in the following way. The vertices V are the spiders, the edges E are the Hadamard edges between the spiders, the inputs I are the spiders that are connected to an input wire and the outputs O are the spiders connected to an output wire.

The neighborhood of a vertex $v \in V$ in the graph G is denoted by $N_G(v)$. Note, that a vertex itself is never part of its neighborhood, i.e., $v \notin N_G(v)$. With that definition we can state the following.

Theorem 1.30 [25, lemma 3.7 and section 7]

a) Let D be a circuit-like ZX-diagram and D' the corresponding graph-like diagram. Then $G(D')$ admits a gFlow.

b) Let D' be a graph-like diagram. If D' has the same number of input and output wires, and if the underlying open graph $G(D')$ has a gFlow, then the linear map associated to D' is unitary.

The second claim is proven by Duncan et al. [25] by presenting an algorithm, that does the extraction. The same is shown by Backens et al. [2] in more detail.

Theorem 1.30 motivates the usage of gFlow preserving operations when modifying ZX-diagrams, if one is interested in preserving extractability.

Chapter 2

Contents

2 Quantum Algorithms

In this chapter we will have a look at a selection of quantum algorithms that will be applied in later chapters. The term quantum algorithm will be used in a very broad way and not only refer to single quantum circuits, but also to hybrid algorithms, that combine one or more quantum circuits with classical computations.

2.1 Quantum Fourier Transform

One very fundamental quantum algorithm is the Quantum Fourier Transform (QFT) introduced by Coppersmith [21]. It plays an important role in many other quantum algorithms, for example Shor's algorithm [61] or the AE algorithm [13] discussed in section 2.2.3. It will also be an integral part of the option pricing method we will see in section 4.2.

The QFT can be interpreted as a quantum version of the Discrete Fourier Transform (DFT), thus, we begin with introducing the classical version.

Definition 2.1 Discrete Fourier Transform

For some $N \in \mathbb{N}$ the DFT transforms a sequence $\left(x_j\right)_{j=0,\dots,N-1} \subset \mathbb{C}$ into another series $\left(y_k\right)_{k=0,\dots,N-1} \subset \mathbb{C}$ defined by

$$y_k := \sum_{j=0}^{N-1} x_j \mathrm{e}^{-\mathbf{i}\frac{2\pi}{N}jk}. \tag{2.1}$$

The inverse DFT maps the sequence $\left(y_k\right)_{k=1,\dots,N} \subset \mathbb{C}$ back to $\left(x_j\right)_{j=1,\dots,N} \subset \mathbb{C}$ where now

$$x_j := \frac{1}{N} \sum_{k=0}^{N-1} y_k \mathrm{e}^{\mathbf{i}\frac{2\pi}{N}jk}.$$

With that done we define the quantum version next.

Definition 2.2 Quantum Fourier Transform

For $N = 2^n$ with $n \in \mathbb{N}$ we define the QFT by how it acts on the basis states, i.e.,

$$\mathbf{QFT}_{\mathrm{n}} \left|e_k^n\right\rangle := \left|\mathcal{S}_N\left(\tfrac{k}{N}\right)\right\rangle \tag{2.2}$$

where for $0 \leq w < 1$

$$\left|\mathcal{S}_N(w)\right\rangle := \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} \mathrm{e}^{\mathbf{i}2\pi wj} \left|e_j^n\right\rangle. \tag{2.3}$$

The inverse QFT is defined

$$\mathbf{QFT}_{\mathrm{n}}^{-1} \left|e_k^n\right\rangle := \left|\mathcal{S}_N^{-1}\left(\tfrac{k}{N}\right)\right\rangle$$

where for $0 \leq w < 1$

$$\left|\mathcal{S}_N^{-1}(w)\right\rangle := \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} \mathrm{e}^{-\mathbf{i}2\pi wj} \left|e_j^n\right\rangle .$$

This definition only specifies how the QFT performs on basis states, but from that we can derive the effect on more general quantum states. Given a quantum state $|x\rangle = \sum_{j=0}^{N-1} x_j \left|e_j^n\right\rangle$ we have

$$\begin{aligned}
|y\rangle = \mathbf{QFT}_\mathrm{n} |x\rangle &= \sum_{j=0}^{N-1} x_j \, \mathbf{QFT}_\mathrm{n} \left|e_j^n\right\rangle \\
&\overset{(2.2)}{=} \sum_{j=0}^{N-1} x_j \left|\mathcal{S}_N\left(\tfrac{j}{N}\right)\right\rangle \\
&\overset{(2.3)}{=} \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} x_j \sum_{k=0}^{N-1} \mathrm{e}^{\mathbf{i}\frac{2\pi}{N}kj} \left|e_k^n\right\rangle \\
&= \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} \sum_{j=0}^{N-1} x_j \mathrm{e}^{\mathbf{i}\frac{2\pi}{N}kj} \left|e_k^n\right\rangle .
\end{aligned}$$

With $y_k = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} x_j \mathrm{e}^{\mathbf{i}\frac{2\pi}{N}kj}$ this leads to $|y\rangle = \sum_{k=0}^{N-1} y_k \left|e_k^n\right\rangle$. Analogously, we see that the inverse QFT maps a state $|y\rangle = \sum_{k=0}^{N-1} y_k \left|e_k^n\right\rangle$ to $|x\rangle = \sum_{j=0}^{N-1} x_j \left|e_j^n\right\rangle$ where

$$x_j = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} y_k \mathrm{e}^{\mathbf{i}\frac{2\pi}{N}jk}.$$

The QFT performs a very similar transform as the DFT, but there are two differences. First the QFT distributes the scaling factor $1/N$ between both directions instead of having it completely in the inverse direction. The second difference is the convention, which direction of the transform is called the inverse and which the original transform. This manifests in the sign in the exponent in equations (2.1) and (2.3).

The reason why we have a special interest in the QFT is, that it can be implemented with only $\mathcal{O}(n^2)$ gates for n qubits [42]. This is remarkable as an n-qubit state consists of 2^n complex numbers. In comparison, the famous class of algorithms to calculate a DFT, summarized by the term Fast Fourier Transform (FFT), would need $\mathcal{O}(2^n n)$ operations [45] which scales a lot worse than $\mathcal{O}(n^2)$.

2.2 Amplitude Estimation

The next important algorithm, we look at in detail, will be Amplitude Estimation (AE). The goal of that algorithm is, given a quantum state, to efficiently get the amplitude of one state. As the amplitude of a state is strongly connected to the probability to measure that state, the straight forward strategy to get the amplitude is Monte-Carlo (MC). We repeatedly measure the quantum state and estimate the probability of the state of interest by calculating the proportion of measurements that result in that state.

The performance of this method has nothing to do with quantum computing and one can expect an approximation error of $\mathcal{O}(1/\sqrt{N})$ with N the number of measurements [37, section 2.7].

Using unique properties of quantum computers AE promises to increase the performance to an approximation error of $\mathcal{O}(1/N)$. In section 2.2.3 we will follow the ideas from Brassard et al. [13] to verify that claim. But before that, we look into Grover's Search Algorithm, itself a famous quantum algorithm, that is a fundamental part of AE.

2.2.1 Grover's Search Algorithm

As its name suggests this algorithm was introduced by Grover [31]. Its goal is to find one item fulfilling a given condition in an unsorted database with N elements. Without further knowledge or structure in the data, a classical algorithm would need to check one element after the other. This leads to an average of $N/2$ entries the algorithm has to check. Grover's Search algorithm makes use of properties of a quantum computer to improve on that. For simplicity, we assume $N = 2^n$ for some $n \in \mathbb{N}$.

Before we can state the algorithm we need some ingredients.

Definition 2.3 Phase Rotation

The phase of an individual state can be modified by a phase rotation gate. The gate is represented by a diagonal matrix where all entries are one, except the one corresponding to the state that should have its phase changed. The operator can be represented by

$$\mathbf{PR}_j(\theta) := \begin{pmatrix} 1 & 0 & \cdots & & \cdots & 0 & 0 \\ 0 & 1 & \cdots & & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & & & \vdots & \vdots \\ 0 & 0 & \cdots & e^{\mathbf{i}\theta} & \cdots & 0 & 0 \\ \vdots & \vdots & & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & \cdots & 1 & 0 \\ 0 & 0 & \cdots & 0 & \cdots & 0 & 1 \end{pmatrix} \begin{matrix} \\ \\ \\ j \\ \\ \\ \\ \end{matrix}.$$

This type of matrix will always be unitary as it only has entries on its diagonal and $e^{\mathbf{i}\theta}(e^{\mathbf{i}\theta})^\dagger = e^{\mathbf{i}\theta}e^{-\mathbf{i}\theta} = e^0 = 1$.

Definition 2.4 Diffusion Operator

The diffusion transform $\mathbf{D}$ is elementwise defined as

$$\mathbf{D}_{jk} := \begin{cases} \frac{2}{N} & \text{if} \quad j \neq k \\ -1 + \frac{2}{N} & \text{if} \quad j = k \end{cases} \qquad j,k \in \{0,\dots,N-1\}.$$

Introducing a projection matrix $\mathbf{P}$, defined as

$$\mathbf{P}_{jk} := \frac{1}{N}, \quad j,k \in \{1,\dots,N\}, \tag{2.4}$$

we can write $\mathbf{D}$ as

$$\mathbf{D} = -\mathbf{I} + 2\,\mathbf{P}\,. \tag{2.5}$$

With these two ingredients, we can now state Grover's search algorithm as it was introduced by Grover [31].

Algorithm 2.5 Grover's Search Algorithm

Input: Oracle χ that tells us if a state is the desired one or not, i.e., for a state $x \in \{0, \dots, 2^n - 1\}$

$$\chi(x) = \begin{cases} 1 & \text{if x is the desired state} \\ 0 & \text{else} \end{cases}$$

1: Bring all states in equal superposition by Hadamard gates.
2: **loop** ▷ $\mathcal{O}\left(\sqrt{N}\right)$ times
3: Rotate the phases of all states x that fulfill $\chi(x) = 1$ by π.
4: Apply the diffusion operation $\mathbf{D}$ to the system.
5: **end loop**
6: **return** Measurement of the system.

This algorithm leaves some open questions, for example how to implement the state dependent phase rotation or how often the loop should be executed exactly. Before we try to answer these questions let us verify that this algorithm produces the desired state.

Theorem 2.6 Follows from [31, Theorem 3]

If $N \geq 8$, i.e., $n \geq 3$, there exists a number $M < \sqrt{2N}$ such that after M iterations of algorithm 2.5 the probability to measure the desired state is above $1/2$.

With a probability of 50 % to observe the desired state, we will only need a few executions of that quantum algorithm, to make very sure that we actually observe it.

To prove theorem 2.6, we need some other claims beforehand. For brevity, we will use a simplified notation for the average of a vector

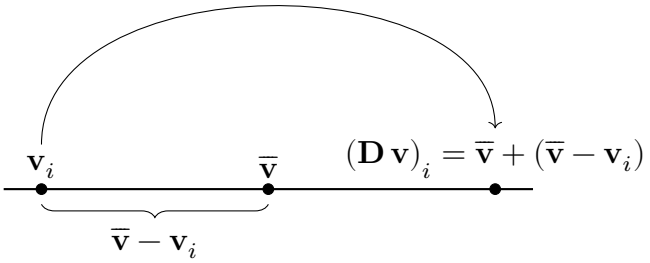
$$\overline{\mathbf{v}} := \frac{1}{N} \sum_{j=0}^{N-1} \mathbf{v}_j.$$

Lemma 2.7 [31, Section 4]

The diffusion operation $\mathbf{D}$, from equation (2.5) is a unitary operator and an inversion of the amplitudes about their average. In formulae

$$\mathbf{D}\,\mathbf{D}^\dagger = \mathbf{I},$$
$$\forall \mathbf{v} \in \mathbb{R}^N : \mathbf{D}\,\mathbf{v} = 2\overline{\mathbf{v}} - \mathbf{v} = \overline{\mathbf{v}} + (\overline{\mathbf{v}} - \mathbf{v}).$$

Before we look at the proof, let us look at a visualization for one element of $\mathbf{v}$ to see that this is indeed the inversion about the average.



Lemma 2.8 [31, Section 4]

The matrix $\mathbf{P}$ from equation (2.4) is a projection, i.e.,

$$\mathbf{P}^2 = \mathbf{P}, \tag{2.6}$$

and maps any vector to a constant vector where all entries are the average of the original vector

$$\forall \mathbf{v} \in \mathbb{R}^N, \quad 0 \leq j < N : \quad (\mathbf{P}\,\mathbf{v})_j = \overline{\mathbf{v}}. \tag{2.7}$$

Proof. We show the first claim by calculating $\mathbf{P}^2$ elementwise. We use $\mathbf{P}_{j\cdot}$ to denote the j-th row and $\mathbf{P}_{\cdot k}$ for the k-th column. With that we get

$$\left(\mathbf{P}^2\right)_{jk} = \mathbf{P}_{j\cdot}\,\mathbf{P}_{\cdot k} = \sum_{l=0}^{N-1} \mathbf{P}_{jl}\,\mathbf{P}_{lk} \overset{(2.4)}{=} \sum_{l=0}^{N-1} \frac{1}{N^2} = \frac{1}{N} = \mathbf{P}_{jk}\,.$$

For the second part let $\mathbf{v} \in \mathbb{R}^n$, then

$$(\mathbf{P}\,\mathbf{v})_i = \mathbf{P}_{i\cdot}\,\mathbf{v} = \sum_{j=0}^{N-1} \mathbf{P}_{ij}\,\mathbf{v}_j \overset{(2.4)}{=} \frac{1}{N} \sum_{j=0}^{N-1} \mathbf{v}_j = \overline{\mathbf{v}}.$$

□

Proof of lemma 2.7. First we show that $\mathbf{D}$ is unitary. Note that $\mathbf{D}$ is purely real and by construction symmetric, i.e., $\mathbf{D}^\dagger = \mathbf{D}^T = \mathbf{D}$. Thus,

$$\mathbf{D}^\dagger\,\mathbf{D} = \mathbf{D}^2 \overset{(2.5)}{=} (-\mathbf{I} + 2\,\mathbf{P})^2 = \mathbf{I}^2 - 4\,\mathbf{P} + 4\,\mathbf{P}^2 \overset{(2.6)}{=} \mathbf{I}\,.$$

Now we proof that $\mathbf{D}$ is the inversion about the average. For this let $\mathbf{v} \in \mathbb{R}^n$

$$\mathbf{D}\,\mathbf{v} \overset{(2.5)}{=} (-\mathbf{I} + 2\,\mathbf{P})\mathbf{v} = -\mathbf{v} + 2\,\mathbf{P}\,\mathbf{v} \overset{(2.7)}{=} 2\overline{\mathbf{v}} - \mathbf{v} = \overline{\mathbf{v}} + (\overline{\mathbf{v}} - \mathbf{v}).$$

□

Now we have everything we need to prove theorem 2.6, the proof is based on Grover [31].

Proof of theorem 2.6. We show the claim by proving a lower bound for the increase of the desired states amplitude in every iteration. In step j we will call the desired states amplitude k_j and the amplitude of all the other states l_j. This is valid, as we will see, that all states, that are not the desired one, have the same amplitude.

We want to show a lower bound for the increase in k_j in every iteration,

$$\Delta k_{j+1} := k_{j+1} - k_j \overset{!}{>} \frac{1}{\sqrt{N}}, \tag{2.8}$$

as long as

$$k_j > 0, \tag{2.9}$$
$$l_j > 0, \tag{2.10}$$
$$k_j < \frac{1}{\sqrt{2}}. \tag{2.11}$$

We will just assume that equation (2.11) holds, since the moment it does no longer hold, the algorithm is finished, as an amplitude above $\frac{1}{\sqrt{2}}$ translates to a success probability above 50 %. The other two assumptions, equations (2.9) and (2.10), we will show by induction. After step 2.5 of algorithm 2.5 all states are in equal superposition, and we have

$$k_0 = l_0 = \frac{1}{\sqrt{N}}.$$

Thus, both requirements are fulfilled for the start of the induction. For the induction step we put in formulas the effect of one iteration of step 2.5 of algorithm 2.5. After the phase flip we have

$$\hat{k}_j = \mathrm{e}^{\pi i} k_j = -k_j, \qquad \hat{l}_j = l_j.$$

For the inversion around the amplitudes average, we first calculate that average

$$A_j = \frac{1}{N}\left(\hat{k}_j + (N-1)\hat{l}_j\right) = l_j - \frac{l_j + k_j}{N},$$

and get by lemma 2.7

$$k_{j+1} = 2A_j - \hat{k}_j = 2\left(l_j - \frac{l_j + k_j}{N}\right) + k_j = l_j 2\left(1 - \frac{1}{N}\right) + k_j\left(1 - \frac{2}{N}\right), \tag{2.12}$$
$$l_{j+1} = 2A_j - \hat{l}_j = 2\left(l_j - \frac{l_j + k_j}{N}\right) - l_j = l_j\left(\frac{N-2}{N}\right) - k_j\frac{2}{N}. \tag{2.13}$$

We directly see, that $k_{j+1} > 0$ for $N > 2$. Next we show the same for l_{j+1}, i.e.,

$$l_{j+1} = l_j\left(\frac{N-2}{N}\right) - k_j\frac{2}{N} \overset{!}{>} 0.$$

First we bring the k_j term to the right

$$l_j\left(\frac{N-2}{N}\right) \overset{!}{>} k_j\frac{2}{N}$$

and divide by l_j, which is positive by equation (2.10), and $\frac{2}{N}$ to get to

$$\frac{N-2}{2} \overset{!}{>} \frac{k_j}{l_j}. \tag{2.14}$$

As we know that $\mathbf{D}$ is a unitary operation by lemma 2.7, we know it preserves the norm of the amplitudes, thus,

$$1 = k_j^2 + (N-1)l_j^2.$$

From that we get a lower bound for l_j^2

$$l_j^2 = \frac{1-k_j}{N-1} \overset{(2.11)}{>} \frac{1-\frac{1}{2}}{N-1} = \frac{1}{2N-2} > \frac{1}{2N},$$

and since we already know that l_j is positive, we can conclude using equation (2.11)

$$l_j > \frac{1}{\sqrt{2N}}. \tag{2.15}$$

Together with equation (2.11) this yields

$$\frac{k_j}{l_j} < \frac{\frac{1}{\sqrt{2}}}{\frac{1}{\sqrt{2N}}} = \sqrt{N} \tag{2.16}$$

and for $N > 7$ we have $\sqrt{N} < \frac{N-2}{2}$, i.e., equation (2.14) holds, thus $l_{j+1} > 0$.

Now it is only left to show, that from equations (2.9) to (2.11) the claim from equation (2.8) follows. Plugging equation (2.12) into equation (2.8) we get

$$\Delta k_{j+1} = l_j 2\left(1 - \frac{1}{N}\right) - k_j \frac{2}{N}.$$

From equation (2.15) we have a lower bound for l_j and from equation (2.11) we have an upper bound for k_j, together this gives a lower bound for Δk_{j+1}

$$\begin{aligned}\Delta k_{j+1} &> \frac{2}{\sqrt{2N}}\left(1 - \frac{1}{N}\right) - \frac{1}{\sqrt{2}}\frac{2}{N} \\ &= \frac{1}{\sqrt{2N}}\left(2 - \frac{2}{N} - \frac{2}{\sqrt{N}}\right) \\ &\overset{N>7}{>} \frac{1}{\sqrt{2N}}.\end{aligned}$$

With this lower bound for improvement in each step, which is constant for a fixed N, we can see that the algorithm needs at most $\sqrt{2N}$ steps to reach a success probability of at least $1/2$. □

Now that we have proven, that algorithm 2.5 performs as claimed, we will address the questions regarding the implementation, the algorithm left open.

Step 2.5, the equal superposition, is realized by applying Hadamard gates on all n qubits, note that

$$\mathbf{H}^{\otimes n}|0\rangle^{\otimes n} = (\mathbf{H}|0\rangle)^{\otimes n} = \left(\frac{1}{\sqrt{2}}\right)^n \left(\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}\begin{pmatrix}1\\0\end{pmatrix}\right)^{\otimes n} = \frac{1}{2^{\frac{n}{2}}}\begin{pmatrix}1\\1\end{pmatrix}^{\otimes n} = \frac{1}{\sqrt{N}}\begin{pmatrix}1\\ \vdots \\ 1\end{pmatrix}.$$

The next step will be more involved as we need to flip the phase of only one state, which we do not know in advance. If we know the desired state, say $w \in \{0, \dots, N-1\}$, the oracle could be represented by a phase rotation $\mathbf{PR}_w(\pi)$ from definition 2.3. If the index of the desired state is not known in advance, what is the more realistic case, one has to construct an operation that detects the correct state and flips its amplitude. This operation is often called an oracle. In practice, implementing such an oracle can be very complicated and depends on the search criterion.

Step 2.5 of algorithm 2.5 is again not very precise on how to actually implement that diffusion. We will show that we can decompose the diffusion operator into a rotation matrix $\mathbf{R}$ which entries are given as

$$\mathbf{R}_{jk} := \begin{cases} 0 & j \neq k \\ 1 & j = k = 0 \\ -1 & j = k \neq 0 \end{cases} \qquad j,k \in \{0, \dots, N-1\} \tag{2.17}$$

and a Hadamard, also called Walsh-Hadamard transform, $\mathbf{W}^{(n)}$

$$\mathbf{W}^{(n)}_{jk} = \frac{1}{\sqrt{N}}(-1)^{\bar{j}\odot\bar{k}} \qquad j,k \in \{0, \dots, N-1\}. \tag{2.18}$$

Here $\bar{j}\odot\bar{k}$ stands for the sum of the element-wise products of j and k's binary representation. For example $\bar{5}\odot\bar{7}$ would be calculated as

$$\bar{5}\odot\bar{7} = 101\odot 111 = 1\cdot 1 + 0\cdot 1 + 1\cdot 1 = 2.$$

Lemma 2.9 [31, Theorem 1]

With the matrices from equations (2.17) and (2.18) we have

$$\mathbf{D} = \mathbf{W}^{(n)}\mathbf{R}\mathbf{W}^{(n)}.$$

Proof. We start with showing $\mathbf{W}^{(n)}\mathbf{W}^{(n)} = \mathbf{I}$, by calculating the individual entries

$$\left(\mathbf{W}^{(n)}\mathbf{W}^{(n)}\right)_{jk} = \mathbf{W}^{(n)}_{j\cdot}\mathbf{W}^{(n)}_{\cdot k} = \sum_{l=0}^{N-1}\mathbf{W}^{(n)}_{jl}\mathbf{W}^{(n)}_{lk} = \frac{1}{N}\sum_{l=0}^{N-1}(-1)^{\bar{j}\odot\bar{l}}(-1)^{\bar{l}\odot\bar{k}}.$$

For a number $x \in \{0, \dots, N-1\}$ we will write $\overline{x}_j$ for the value of the j^{th} bit in the binary representation of x, i.e., $\overline{x}_j \in \{0, 1\}$. Then we have

$$\begin{aligned}
\left(\mathbf{W}^{(n)}\mathbf{W}^{(n)}\right)_{jk} &= \frac{1}{N}\sum_{l=0}^{N-1}(-1)^{\bar{j}\odot\bar{l}}(-1)^{\bar{l}\odot\bar{k}} \\
&= \frac{1}{N}\sum_{l=0}^{N-1}\prod_{m=0}^{n-1}(-1)^{\bar{j}_m\bar{l}_m}(-1)^{\bar{k}_m\bar{l}_m} \\
&= \frac{1}{N}\sum_{l=0}^{N-1}\prod_{m=0}^{n-1}(-1)^{\bar{l}_m\left(\bar{j}_m+\bar{k}_m\right)} \\
&= \frac{1}{N}\sum_{l=0}^{N-1}\prod_{\substack{m=0\\ \bar{j}_m\neq\bar{k}_m}}^{n-1}(-1)^{\bar{l}_m\left(\bar{j}_m+\bar{k}_m\right)}\prod_{\substack{m=0\\ \bar{j}_m=\bar{k}_m}}^{n-1}(-1)^{\bar{l}_m\left(\bar{j}_m+\bar{k}_m\right)}.
\end{aligned}$$

We directly see, that $\bar{j}_m + \bar{k}_m$ is even if $\bar{j}_m = \bar{k}_m$, thus, we can ignore the second term. For the first term, we know that $\bar{j}_m + \bar{k}_m$ is always one if $\bar{j}_m \neq \bar{k}_m$, thus,

$$\begin{aligned}
\left(\mathbf{W}^{(n)}\mathbf{W}^{(n)}\right)_{jk} &= \frac{1}{N}\sum_{l=0}^{N-1}(-1)^{\bar{j}\odot\bar{l}}(-1)^{\bar{l}\odot\bar{k}} \\
&= \frac{1}{N}\sum_{l=0}^{N-1}\prod_{\substack{m=0\\ \bar{j}_m\neq\bar{k}_m}}^{n-1}(-1)^{\bar{l}_m}.
\end{aligned}$$

If $j = k$ each product is empty and thus the expression equals one. For $j \neq k$ we only look at the positions in the binary representation of every l where the binary representations of j and k differ. If the number of ones in these positions is even, the product is 1 and if it is odd the product is -1. As we sum over all possible values that can be represented by n bits, we have exactly as many numbers with an odd number of ones at the relevant positions as with an even, thus the sum is 0. In total, we get

$$\left(\mathbf{W}^{(n)}\mathbf{W}^{(n)}\right)_{jk} = \begin{cases} 1 & j = k \\ 0 & j \neq k \end{cases},$$

i.e., $\mathbf{W}^{(n)}\mathbf{W}^{(n)} = \mathbf{I}$.

Next we split $\mathbf{R}$ into $\mathbf{R} = \mathbf{R}_1 + \mathbf{R}_2$ where $\mathbf{R}_1 = -\mathbf{I}$ and $\mathbf{R}_2$ is defined as

$$\left(\mathbf{R}_2\right)_{jk} = \begin{cases} 2 & j = k = 0 \\ 0 & \text{else} \end{cases}. \tag{2.19}$$

We apply $\mathbf{W}^{(n)}$ to both parts of $\mathbf{R}$, and call the two parts $\mathbf{D}_1$ and $\mathbf{D}_2$. The first term simplifies to

$$\mathbf{D}_1 = \mathbf{W}^{(n)}\mathbf{R}_1\mathbf{W}^{(n)} = \mathbf{W}^{(n)}\left(-\mathbf{I}\,\mathbf{W}^{(n)}\right) = (-1)\mathbf{W}^{(n)}\mathbf{W}^{(n)} = -\mathbf{I}.$$

For the second term we calculate the individual entries

$$\begin{aligned}(\mathbf{D}_2)_{jk} &= \left(\mathbf{W}^{(n)}\mathbf{R}_2\mathbf{W}^{(n)}\right)_{jk} \\ &= \sum_{l=0}^{N-1}\sum_{m=0}^{N-1}\mathbf{W}^{(n)}_{jl}(\mathbf{R}_2)_{lm}\mathbf{W}^{(n)}_{mk} \\ &\overset{(2.19)}{=} 2\mathbf{W}^{(n)}_{j0}\mathbf{W}^{(n)}_{0k} \\ &\overset{(2.18)}{=} \frac{2}{n}(-1)^{\bar{j}\odot\bar{0}}(-1)^{\bar{0}\odot\bar{k}} \\ &= \frac{2}{N}\end{aligned}$$

Thus, by equation (2.4)

$$\mathbf{D}_2 = 2\,\mathbf{P}\,.$$

Everything put together yields

$$\mathbf{W}^{(n)}\mathbf{R}\mathbf{W}^{(n)} = \mathbf{W}^{(n)}\mathbf{R}_1\mathbf{W}^{(n)} + \mathbf{W}^{(n)}\mathbf{R}_2\mathbf{W}^{(n)} = \mathbf{D}_1 + \mathbf{D}_2 = -\mathbf{I} + 2\,\mathbf{P} \overset{(2.5)}{=} \mathbf{D}\,.$$

□

Remark 2.10

The Walsh-Hadamard transform $\mathbf{W}^{(n)}$ can be achieved by applying a Hadamard gate to every qubit, i.e.,

$$\mathbf{W}^{(n)} = \mathbf{H}^{\otimes n}\,.$$

This can be shown by induction, for $n = 1$ we have from equation (2.18)

$$\mathbf{W}^{(1)} = \frac{1}{\sqrt{2}}\begin{pmatrix}(-1)^{\bar{0}\odot\bar{0}} & (-1)^{\bar{0}\odot\bar{1}} \\ (-1)^{\bar{1}\odot\bar{0}} & (-1)^{\bar{1}\odot\bar{1}}\end{pmatrix} = \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1 \\ 1 & -1\end{pmatrix}.$$

Then

$$\mathbf{W}^{(n)} = \mathbf{H}\otimes\mathbf{W}^{(n-1)} = \frac{1}{\sqrt{2}}\begin{pmatrix}\mathbf{W}^{(n-1)} & \mathbf{W}^{(n-1)} \\ \mathbf{W}^{(n-1)} & -\mathbf{W}^{(n-1)}\end{pmatrix}.$$

Now we have to look at the four blocks individually. First, let $j, k < 2^{n-1}$, i.e., we are in the upper left block, then

$$\mathbf{W}^{(n)}_{jk} = \frac{1}{\sqrt{2}}\mathbf{W}^{(n-1)}_{jk} = \frac{1}{\sqrt{2}}2^{-\frac{n-1}{2}}(-1)^{\bar{j}\odot\bar{k}} = \frac{1}{\sqrt{N}}(-1)^{\bar{j}\odot\bar{k}}.$$

Let $j < 2^{n-1}$ and $k \geq 2^{n-1}$, i.e., we are in the lower left block, then

$$\mathbf{W}^{(n)}_{jk} = \frac{1}{\sqrt{2}}\mathbf{W}^{(n-1)}_{j(k-2^{n-1})} = \frac{1}{\sqrt{2}}2^{-\frac{n-1}{2}}(-1)^{\bar{j}\odot\overline{k-2^{n-1}}} = \frac{1}{\sqrt{N}}(-1)^{\bar{j}\odot\bar{k}}.$$

The last equality holds, as j is smaller than 2^{n-1} and thus in its binary representation the first digit is zero and the difference in the binary representations of k and $k-2^{n-1}$ is only in the first digit.

For the upper right block, i.e., $j \geq 2^{n-1}$ and $k < 2^{n-1}$, the argumentation is the same as for the lower left block.

Finally, let $j, k \geq 2^{n-1}$, i.e., we are in the lower right block, then

$$\mathbf{W}_{jk}^{(n)} = -\frac{1}{\sqrt{2}}\mathbf{W}_{(j-2^{n-1})(k-2^{n-1})}^{(n-1)} = -\frac{1}{\sqrt{2}}2^{-\frac{n-1}{2}}(-1)^{\overline{j-2^{n-1}}\odot\overline{k-2^{n-1}}} = \frac{1}{\sqrt{N}}(-1)^{\bar{j}\odot\bar{k}}.$$

The last equality holds as the difference between $\overline{j-2^{n-1}} \odot \overline{k-2^{n-1}}$ and $\bar{j}\odot\bar{k}$ is exactly one, coming from the first digits switching from zero to one.

The last open question is, how many iterations of the algorithm we should perform to achieve a high probability for the desired state. In figure 2.1 we can see the results of a numerical experiment with 10 qubits. Theory says, that after at most $\sqrt{2\cdot 2^{10}} \approx 45$ iterations we achieve a success probability of at least 50 %. From the experiment we see that this was already reached after 13 iterations. It is also important to note that the probability decreases again at some point. Unfortunately, we only have an upper bound for the number of repetitions and not the optimal number for maximizing the desired state's probability.

2.2.2 Amplitude Amplification

A generalization of Grover's Search algorithm was introduced by Brassard et al. [13]. The goal is now, given a quantum algorithm $\mathbf{A}$ that produces some state Ψ and an oracle χ that can decide if a measurement of Ψ is a good or a bad one, to find a good state. A naive idea is to repeatedly measure Ψ and check if the solution is good, until a good one is found. This would be comparable to classical MC.

Amplitude Amplification (AA) improves on this approach by, as the name suggests, amplifying the amplitude of good states. The amplification is done by applying the so-called Grover operator

$$\mathbf{Q} := -\mathbf{A}\,\mathbf{S}_0\,\mathbf{A}^{-1}\,\mathbf{S}_\chi. \tag{2.20}$$

The operation $\mathbf{S}_\chi$ flips the sign of the good states' phase, while doing nothing on the bad states. Similarly, $\mathbf{S}_0$ only flips the sign of the zero state's phase. This can also be written as

$$\mathbf{S}_\chi := \mathbf{I} - \frac{2}{a_\Psi}\left|\Psi_1\right\rangle\!\left\langle\Psi_1\right|, \tag{2.21}$$

$$\mathbf{S}_0 := \mathbf{I} - 2\left|0\right\rangle\!\left\langle 0\right|. \tag{2.22}$$

This amplification of the amplitude is formalized in the following theorem, in the remainder of this section we will follow the proof from Brassard et al. [13] with some more detail.

Theorem 2.11 Amplitude Amplification

After j applications of the Grover operator $\mathbf{Q}$ the probability to observe a good state is $\sin^2((2j+1)\theta_\Psi)$, i.e.,

$$\mathbb{P}\left[\chi\left(\mathcal{M}\left(\mathbf{Q}^j\left|\Psi\right\rangle\right)\right) = 1\right] = \sin^2((2j+1)\theta_\Psi). \tag{2.23}$$

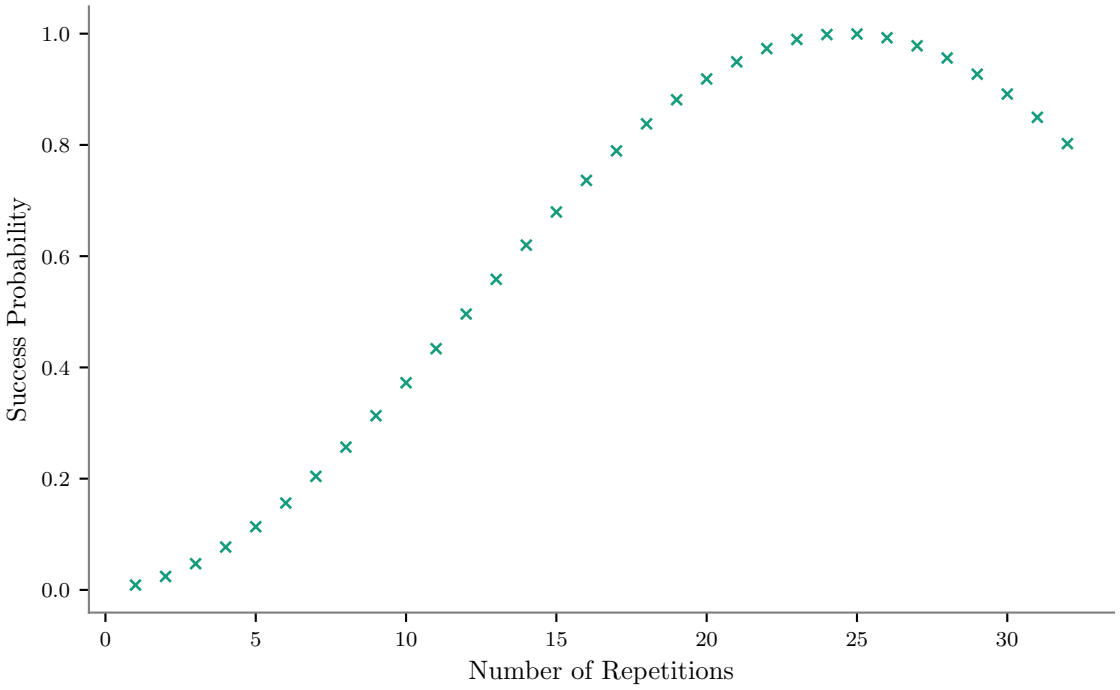


Figure 2.1: The success probabilities for increasing number of iterations of algorithm 2.5. In this example the circuit has 10 qubits, i.e., $N = 2^{10} = 1024$ and one random but fixed state is marked as the desired state. We see the success probability increase in every iteration, until it is almost 1, from where it decreases again.

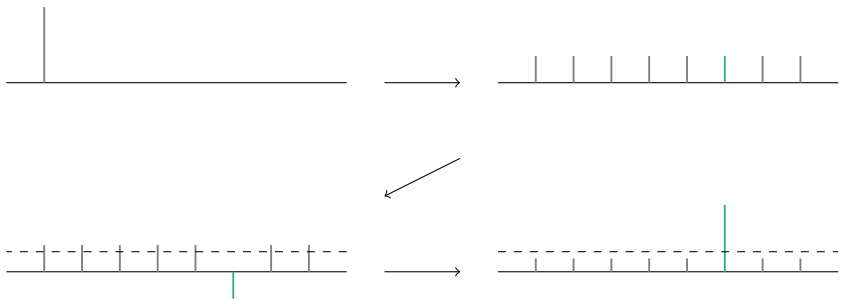


Figure 2.2: Visualization of the three steps of algorithm 2.5. The vertical bars represent the amplitudes of a system of three qubits and the dashed line the average of all amplitudes. First the system is brought into equal superposition of all states, then the target state is flipped. Next, all states are flipped around the average of all states. We see that the target state now has a much higher amplitude, than the other states.

To proof this we need some preparation. The oracle χ induces a partition of the n-qubit system's statespace, the good space

$$\mathcal{H}_1 := \operatorname{span}\{|e_j^n\rangle \mid \chi(j) = 1\}$$

and the bad space

$$\mathcal{H}_0 := \operatorname{span}\{|e_j^n\rangle \mid \chi(j) = 0\}.$$

Every element $|\Phi\rangle$ of the statespace can be written as $|\Phi\rangle = |\Phi_0\rangle + |\Phi_1\rangle$, where one part lives in the bad subspace and the other one in the good, i.e., if

$$|\Phi\rangle = \sum_{j=0}^{N-1} x_j |e_j^n\rangle$$

then

$$|\Phi_1\rangle = \sum_{\chi(j)=1} x_j |e_j^n\rangle \quad \text{and} \quad |\Phi_0\rangle = \sum_{\chi(j)=0} x_j |e_j^n\rangle .$$

From that definition we directly see, that

$$\langle\Phi_1|\Phi_0\rangle = \langle\Phi_0|\Phi_1\rangle = 0. \tag{2.24}$$

The probability to measure a good state is, following definition 1.14,

$$\begin{aligned} a_\Phi &= \mathbb{P}[\chi(\mathcal{M}(|\Phi\rangle)) = 1] \\ &= \sum_{\chi(j)=1} \mathbb{P}[\mathcal{M}(|\Phi\rangle) = j] \\ &= \sum_{\chi(j)=1} \langle\Phi|e_j^n\rangle \langle e_j^n|\Phi\rangle \\ &= \sum_{\chi(j)=1} |x_j|^2 \\ &= \sum_{\chi(j)=1} |x_j|^2 \langle e_j^n|e_j^n\rangle \\ &= \left(\sum_{\chi(j)=1} x_j^\dagger \langle e_j^n|\right)\left(\sum_{\chi(j)=1} x_j |e_j^n\rangle\right) \\ &= \langle\Phi_1|\Phi_1\rangle . \end{aligned} \tag{2.25}$$

Similarly, the probability to observe a bad state is given by

$$b_\Phi = \mathbb{P}[\chi(\mathcal{M}(|\Phi\rangle)) = 0] = \langle\Phi_0|\Phi_0\rangle . \tag{2.26}$$

These two probabilities should, and by their construction do, add up to one, i.e.,

$$a_\Phi + b_\Phi = 1. \tag{2.27}$$

We introduce

$$|\Psi\rangle := \mathbf{A}|0\rangle . \tag{2.28}$$

Lemma 2.12 Lemma 8 Brassard and Hoyer [12] or Lemma 1 Brassard et al. [13]

Given the notation above we have

$$\mathbf{Q}\,|\Psi_1\rangle = (1-2a_\Psi)\,|\Psi_1\rangle - 2a_\Psi\,|\Psi_0\rangle\,, \tag{2.29}$$
$$\mathbf{Q}\,|\Psi_0\rangle = 2(1-a_\Psi)\,|\Psi_1\rangle + (1-2a_\Psi)\,|\Psi_0\rangle\,. \tag{2.30}$$

Proof. We start with the effect of $\mathbf{Q}$ on the good part of $|\Psi\rangle$

$$\begin{aligned}
\mathbf{Q}\,|\Psi_1\rangle &\overset{(2.20)}{=} -\mathbf{A}\,\mathbf{S}_0\,\mathbf{A}^{-1}\,\mathbf{S}_\chi\,|\Psi_1\rangle \\
&\overset{(2.21)}{=} -\mathbf{A}\,\mathbf{S}_0\,\mathbf{A}^{-1}\left(\mathbf{I}-\frac{2}{a_\Psi}\,|\Psi_1\rangle\langle\Psi_1|\right)|\Psi_1\rangle \\
&= -\mathbf{A}\,\mathbf{S}_0\,\mathbf{A}^{-1}\,|\Psi_1\rangle + \frac{2}{a_\Psi}\,\mathbf{A}\,\mathbf{S}_0\,\mathbf{A}^{-1}\,|\Psi_1\rangle\,\langle\Psi_1|\Psi_1\rangle \\
&\overset{(2.25)}{=} \mathbf{A}\,\mathbf{S}_0\,\mathbf{A}^{-1}\,|\Psi_1\rangle \\
&\overset{(2.22)}{=} \mathbf{A}\,(\mathbf{I}-2\,|0\rangle\langle 0|)\,\mathbf{A}^{-1}\,|\Psi_1\rangle \\
&= \mathbf{A}\,\mathbf{A}^{-1}\,|\Psi_1\rangle - 2\,\mathbf{A}\,|0\rangle\,\langle 0|\mathbf{A}^{-1}|\Psi_1\rangle \\
&\overset{(2.28)}{=} |\Psi_1\rangle - 2\,|\Psi\rangle\,\langle 0|\mathbf{A}^{-1}|\Psi_1\rangle\,.
\end{aligned}$$

Looking at the complex conjugate of both sides of equation (2.28), we get

$$\begin{aligned}
&|\Psi\rangle = \mathbf{A}\,|0\rangle \\
\iff\ &\langle\Psi| = \langle 0|\,\mathbf{A}^\dagger = \langle 0|\,\mathbf{A}^{-1}\,.
\end{aligned}$$

Plugging this in we get

$$\begin{aligned}
\mathbf{Q}\,|\Psi_1\rangle &= |\Psi_1\rangle - 2\,|\Psi\rangle\,\langle 0|\mathbf{A}^{-1}|\Psi_1\rangle \\
&= |\Psi_1\rangle - 2\,|\Psi\rangle\,\langle\Psi|\Psi_1\rangle \\
&= |\Psi_1\rangle - 2\,|\Psi\rangle\,\langle\Psi_0+\Psi_1|\Psi_1\rangle \\
&= |\Psi_1\rangle - 2\,|\Psi\rangle\,(\langle\Psi_0|\Psi_1\rangle + \langle\Psi_1|\Psi_1\rangle) \\
&= |\Psi_1\rangle - 2\,|\Psi\rangle\,(0+a_\Psi) \\
&= |\Psi_1\rangle - 2a_\Psi(|\Psi_1\rangle + |\Psi_0\rangle) \\
&= (1-2a_\Psi)\,|\Psi_1\rangle - 2a_\Psi\,|\Psi_0\rangle\,.
\end{aligned}$$

With analogous calculations we get

$$\begin{aligned}
\mathbf{Q}\,|\Psi_0\rangle &= -\mathbf{A}\,\mathbf{S}_0\,\mathbf{A}^{-1}\,\mathbf{S}_\chi\,|\Psi_0\rangle \\
&= -\mathbf{A}\,\mathbf{S}_0\,\mathbf{A}^{-1}\,|\Psi_0\rangle \\
&= -\mathbf{A}\,(\mathbf{I}-2\,|0\rangle\langle 0|)\,\mathbf{A}^{-1}\,|\Psi_0\rangle \\
&= -\mathbf{A}\,\mathbf{A}^{-1}\,|\Psi_0\rangle + 2\,\mathbf{A}\,|0\rangle\,\langle 0|\mathbf{A}^{-1}|\Psi_0\rangle \\
&= -|\Psi_0\rangle + 2\,|\Psi\rangle\,\langle\Psi|\Psi_0\rangle \\
&= -|\Psi_0\rangle + 2(|\Psi_0\rangle + |\Psi_1\rangle)(1-a_\Psi) \\
&= 2(1-a_\Psi)\,|\Psi_1\rangle + (1-2a_\Psi)\,|\Psi_0\rangle\,.
\end{aligned}$$

□

We define the subspace spanned by the good and the bad state as

$$\mathcal{H}_\Psi := \operatorname{span}\{|\Psi_0\rangle, |\Psi_1\rangle\}.$$

Lemma 2.12 shows, that $\mathbf{Q}$ is a stable operation on the subspace $\mathcal{H}_\Psi$, i.e.,

$$\forall\, |\Phi\rangle \in \mathcal{H}_\Psi :\ \mathbf{Q}\,|\Phi\rangle \in \mathcal{H}_\Psi.$$

If $0 < a_\Psi < 1$ this space has dimension 2, otherwise one of the vectors is the zero vector and $\mathcal{H}_\Psi$ has dimension 1.

Lemma 2.13

On the subspace $\mathcal{H}_\Psi$

$$\mathbf{Q} = \mathbf{U}_\Psi\, \mathbf{U}_{\Psi_0}$$

where

$$\mathbf{U}_\Psi := \mathbf{I} - 2\,|\Psi\rangle\langle\Psi|,$$
$$\mathbf{U}_{\Psi_0} := \mathbf{I} - \frac{2}{1-a_\Psi}\,|\Psi_0\rangle\langle\Psi_0|.$$

Proof. We will show the effect on $|\Psi_0\rangle$

$$\begin{aligned}
\mathbf{U}_\Psi\, \mathbf{U}_{\Psi_0}\,|\Psi_0\rangle = &\quad (\mathbf{I} - 2\,|\Psi\rangle\langle\Psi|)\left(\mathbf{I} - \frac{2}{1-a_\Psi}\,|\Psi_0\rangle\langle\Psi_0|\right)|\Psi_0\rangle \\
\overset{(2.26)}{=} &\quad (\mathbf{I} - 2\,|\Psi\rangle\langle\Psi|)\left(|\Psi_0\rangle - 2\frac{b_\Psi}{1-a_\Psi}\,|\Psi_0\rangle\right) \\
\overset{(2.27)}{=} &\quad -(\mathbf{I} - 2\,|\Psi\rangle\langle\Psi|)\,|\Psi_0\rangle \\
= &\quad -|\Psi_0\rangle + 2\,|\Psi\rangle\langle\Psi|\,|\Psi_0\rangle \\
= &\quad -|\Psi_0\rangle + 2(|\Psi_0\rangle + |\Psi_1\rangle)(\langle\Psi_0| + \langle\Psi_1|)\,|\Psi_0\rangle \\
\overset{(2.24),(2.26)}{=} &\quad -|\Psi_0\rangle + 2(|\Psi_0\rangle + |\Psi_1\rangle)b_\Psi \\
= &\quad 2b_\Psi\,|\Psi_1\rangle + (2b_\Psi - 1)\,|\Psi_0\rangle \\
\overset{(2.27)}{=} &\quad 2(1-a_\Psi)\,|\Psi_1\rangle + (1-2a_\Psi)\,|\Psi_0\rangle \\
\overset{(2.30)}{=} &\quad \mathbf{Q}\,|\Psi_0\rangle,
\end{aligned}$$

and with similar arguments the effect on $|\Psi_1\rangle$

$$\begin{aligned}
\mathbf{U}_\Psi\, \mathbf{U}_{\Psi_0}\,|\Psi_1\rangle = &\ (\mathbf{I} - 2\,|\Psi\rangle\langle\Psi|)\left(\mathbf{I} - \frac{2}{1-a_\Psi}\,|\Psi_0\rangle\langle\Psi_0|\right)|\Psi_1\rangle \\
= &\ (\mathbf{I} - 2\,|\Psi\rangle\langle\Psi|)\,|\Psi_1\rangle \\
= &\ |\Psi_1\rangle - 2(|\Psi_1\rangle + |\Psi_0\rangle)a_\Psi \\
= &\ (1-2a_\Psi)\,|\Psi_1\rangle - 2a_\Psi\,|\Psi_0\rangle \\
\overset{(2.29)}{=} &\ \mathbf{Q}\,|\Psi_1\rangle.
\end{aligned}$$

□

Lemma 2.14

Let $0 < a_\Psi < 1$, the two vectors

$$|\Psi_\pm\rangle := \frac{1}{\sqrt{2}}\left(\frac{1}{\sqrt{a_\Psi}}|\Psi_1\rangle \pm \frac{\mathbf{i}}{\sqrt{1-a_\Psi}}|\Psi_0\rangle\right) \tag{2.31}$$

are eigenvectors of $\mathbf{Q}$ with corresponding eigenvalues

$$\lambda_\pm = \mathrm{e}^{\pm \mathbf{i} 2\theta_\Psi}, \tag{2.32}$$

where

$$\theta_\Psi := \arcsin(\sqrt{a_\Psi}). \tag{2.33}$$

Proof. We use lemma 2.12 to get

$$\begin{aligned}
\mathbf{Q}|\Psi_\pm\rangle = \quad & \mathbf{Q}\frac{1}{\sqrt{2}}\left(\frac{1}{\sqrt{a_\Psi}}|\Psi_1\rangle \pm \frac{\mathbf{i}}{\sqrt{1-a_\Psi}}|\Psi_0\rangle\right) \\
= \quad & \frac{1}{\sqrt{2}}\left(\frac{1}{\sqrt{a_\Psi}}\mathbf{Q}|\Psi_1\rangle \pm \frac{\mathbf{i}}{\sqrt{1-a_\Psi}}\mathbf{Q}|\Psi_0\rangle\right) \\
\overset{(2.29),(2.30)}{=} & \frac{1}{\sqrt{2}}\left(\frac{1}{\sqrt{a_\Psi}}((1-2a_\Psi)|\Psi_1\rangle - 2a_\Psi|\Psi_0\rangle)\right. \\
& \left. \pm\frac{\mathbf{i}}{\sqrt{1-a_\Psi}}(2(1-a_\Psi)|\Psi_1\rangle + (1-2a_\Psi)|\Psi_0\rangle)\right).
\end{aligned}$$

Reordering for $|\Psi_0\rangle$ and $|\Psi_1\rangle$ yields

$$\begin{aligned}
\mathbf{Q}|\Psi_\pm\rangle = \frac{1}{\sqrt{2}}\left(|\Psi_1\rangle\left(\frac{1}{\sqrt{a_\Psi}} - \frac{2a_\Psi}{\sqrt{a_\Psi}} \pm \frac{2\mathbf{i}}{\sqrt{1-a_\Psi}} \mp \frac{2a_\Psi\mathbf{i}}{\sqrt{1-a_\Psi}}\right)\right. \\
\left. + |\Psi_0\rangle\left(-\frac{2a_\Psi}{\sqrt{a_\Psi}} \pm \frac{\mathbf{i}}{\sqrt{1-a_\Psi}} \mp \frac{2a_\Psi\mathbf{i}}{\sqrt{1-a_\Psi}}\right)\right).
\end{aligned}$$

We look at the two parts individually and simplify the first part to

$$\begin{aligned}
\frac{1}{\sqrt{a_\Psi}} - \frac{2a_\Psi}{\sqrt{a_\Psi}} \pm \frac{2\mathbf{i}}{\sqrt{1-a_\Psi}} \mp \frac{2a_\Psi\mathbf{i}}{\sqrt{1-a_\Psi}} &= \frac{1}{\sqrt{a_\Psi}}\left(1 - 2a_\Psi \pm \frac{2\mathbf{i}\sqrt{a_\Psi}}{\sqrt{1-a_\Psi}} \mp \frac{2a_\Psi\mathbf{i}\sqrt{a_\Psi}}{\sqrt{1-a_\Psi}}\right) \\
&= \frac{1}{\sqrt{a_\Psi}}\left(1 - 2a_\Psi \pm \frac{2\mathbf{i}(1-a_\Psi)\sqrt{a_\Psi}}{\sqrt{1-a_\Psi}}\right) \\
&= \frac{1}{\sqrt{a_\Psi}}\left(1 - 2a_\Psi \pm 2\mathbf{i}\sqrt{a_\Psi}\sqrt{1-a_\Psi}\right).
\end{aligned}$$

Introducing

$$\hat{\lambda}_\pm := 1 - 2a_\Psi \pm 2\mathbf{i}\sqrt{a_\Psi}\sqrt{1-a_\Psi} \tag{2.34}$$

we get

$$\frac{1}{\sqrt{a_\Psi}} - \frac{2a_\Psi}{\sqrt{a_\Psi}} \pm \frac{2\mathbf{i}}{\sqrt{1-a_\Psi}} \mp \frac{2a_\Psi\mathbf{i}}{\sqrt{1-a_\Psi}} = \frac{1}{\sqrt{a_\Psi}}\hat{\lambda}_\pm.$$

The second part simplifies to

$$\begin{aligned}
-\frac{2a_\Psi}{\sqrt{a_\Psi}} \pm \frac{\mathbf{i}}{\sqrt{1-a_\Psi}} \mp \frac{2a_\Psi \mathbf{i}}{\sqrt{1-a_\Psi}} &= \frac{\mathbf{i}}{\sqrt{1-a_\Psi}} \left(\frac{-2\sqrt{a_\Psi}\sqrt{1-a_\Psi}}{\mathbf{i}} \pm 1 \mp 2a_\Psi \right) \\
&= \frac{\mathbf{i}}{\sqrt{1-a_\Psi}} \left(2\mathbf{i}\sqrt{a_\Psi}\sqrt{1-a_\Psi} \pm 1 \mp 2a_\Psi \right) \\
&= \pm \frac{\mathbf{i}}{\sqrt{1-a_\Psi}} \left(1 - 2a_\Psi \pm 2\mathbf{i}\sqrt{a_\Psi}\sqrt{1-a_\Psi} \right) \\
&= \pm \frac{\mathbf{i}}{\sqrt{1-a_\Psi}} \hat{\lambda}_\pm .
\end{aligned}$$

Putting both parts back together we get

$$\mathbf{Q} \left| \Psi_\pm \right\rangle = \hat{\lambda}_\pm \frac{1}{\sqrt{2}} \left(\frac{1}{\sqrt{a_\Psi}} \left| \Psi_1 \right\rangle \pm \frac{\mathbf{i}}{\sqrt{1-a_\Psi}} \left| \Psi_0 \right\rangle \right) = \hat{\lambda}_\pm \left| \Psi_\pm \right\rangle .$$

This shows, that $\left|\Psi_\pm\right\rangle$ are eigenvectors with eigenvalues $\hat{\lambda}_\pm$. Now we need to show that $\hat{\lambda}_\pm$ is the actually the same as $\lambda_\pm$,

$$\begin{aligned}
\hat{\lambda}_\pm &= 1 - 2a_\Psi \pm 2\mathbf{i}\sqrt{a_\Psi}\sqrt{1-a_\Psi} \\
&\overset{(2.33)}{=} 1 - 2\sin^2(\theta_\Psi) \pm 2\mathbf{i}\sin(\theta_\Psi)\sqrt{1-\sin^2(\theta_\Psi)} \\
&\overset{(B.5),(B.8)}{=} \cos(2\theta_\Psi) \pm 2\mathbf{i}\sin(\theta_\Psi)\cos(\theta_\Psi) \\
&\overset{(B.6)}{=} \cos(2\theta_\Psi) \pm \mathbf{i}\sin(2\theta_\Psi) \\
&\overset{(B.1),(B.2)}{=} \cos(\pm 2\theta_\Psi) + \mathbf{i}\sin(\pm 2\theta_\Psi) \\
&= \mathrm{e}^{\pm \mathbf{i} 2\theta_\Psi} \\
&\overset{(2.32)}{=} \lambda_\pm .
\end{aligned}$$

□

Lemma 2.15

Let $0 < a_\Psi < 1$, the two vectors $\left|\Psi_\pm\right\rangle$ from equation (2.31) form an orthonormal basis of $\mathcal{H}_\Psi$.

Proof. Both eigenvectors are linear combinations of $\left|\Psi_0\right\rangle$ and $\left|\Psi_1\right\rangle$ and therefore in $\mathcal{H}_\Psi$. For the next step we need the corresponding bras

$$\left\langle \Psi_\pm \right| = \left| \Psi_\pm \right\rangle^\dagger = \frac{1}{\sqrt{2}} \left(\frac{1}{\sqrt{a_\Psi}} \left\langle \Psi_1 \right| \mp \frac{\mathbf{i}}{\sqrt{1-a_\Psi}} \left\langle \Psi_0 \right| \right).$$

The important change here is the sign flip of the second part coming from the complex conjugation. We start with showing the normality, looking at

$$\left\langle \Psi_\pm | \Psi_\pm \right\rangle = \frac{1}{2} \left(\frac{1}{\sqrt{a_\Psi}} \left\langle \Psi_1 \right| \mp \frac{\mathbf{i}}{\sqrt{1-a_\Psi}} \left\langle \Psi_0 \right| \right) \left(\frac{1}{\sqrt{a_\Psi}} \left| \Psi_1 \right\rangle \pm \frac{\mathbf{i}}{\sqrt{1-a_\Psi}} \left| \Psi_0 \right\rangle \right).$$

Since $|\Psi_1\rangle$ and $|\Psi_0\rangle$ are orthogonal to each other we get

$$\begin{aligned}\langle\Psi_\pm|\Psi_\pm\rangle &= \frac{1}{2}\left(\frac{1}{a_\Psi}\langle\Psi_1|\Psi_1\rangle + \frac{1}{1-a_\Psi}\langle\Psi_0|\Psi_0\rangle\right)\\ &= \frac{1}{2}\left(\frac{1}{a_\Psi}a_\Psi + \frac{1}{1-a_\Psi}(1-a_\Psi)\right)\\ &= 1.\end{aligned}$$

For the orthogonality we have

$$\begin{aligned}\langle\Psi_\pm|\Psi_\mp\rangle &= \frac{1}{2}\left(\frac{1}{\sqrt{a_\Psi}}\langle\Psi_1| \mp \frac{\mathbf{i}}{\sqrt{1-a_\Psi}}\langle\Psi_0|\right)\left(\frac{1}{\sqrt{a_\Psi}}|\Psi_1\rangle \mp \frac{\mathbf{i}}{\sqrt{1-a_\Psi}}|\Psi_0\rangle\right)\\ &= \frac{1}{2}\left(\frac{1}{a_\Psi}\langle\Psi_1|\Psi_1\rangle - \frac{1}{1-a_\Psi}\langle\Psi_0|\Psi_0\rangle\right)\\ &= 0.\end{aligned}$$

Now it remains to argue, that span $\{|\Psi_+\rangle, |\Psi_-\rangle\} = \mathcal{H}_\Psi$. Since $0 < a_\Psi < 1$ we know from its definition, that $\mathcal{H}_\Psi$ has dimension two. Thus, the two element set $\{|\Psi_+\rangle, |\Psi_-\rangle\}$ of orthogonal, i.e., independent, elements is a basis and therefore spans the space. □

Lemma 2.16

For every $j \in \mathbb{N}$

$$\mathbf{Q}^j|\Psi\rangle = \frac{-\mathbf{i}}{\sqrt{2}}\left(\mathrm{e}^{(2j+1)\mathbf{i}\theta_\Psi}|\Psi_+\rangle - \mathrm{e}^{-(2j+1)\mathbf{i}\theta_\Psi}|\Psi_-\rangle\right) \tag{2.35}$$

and

$$\mathbf{Q}^j|\Psi\rangle = \frac{1}{\sqrt{a_\Psi}}\sin((2j+1)\theta_\Psi)|\Psi_1\rangle + \frac{1}{\sqrt{1-a_\Psi}}\cos((2j+1)\theta_\Psi)|\Psi_0\rangle. \tag{2.36}$$

Proof. We need the following identities derived from Euler's Formula

$$\begin{aligned}\mathrm{e}^{\mathbf{i}x} - \mathrm{e}^{-\mathbf{i}x} &= \cos(x) + \mathbf{i}\sin(x) - \cos(-x) - \mathbf{i}\sin(-x)\\ &= 2\mathbf{i}\sin(x)\end{aligned}$$

and

$$\begin{aligned}\mathrm{e}^{\mathbf{i}x} + \mathrm{e}^{-\mathbf{i}x} &= \cos(x) + \mathbf{i}\sin(x) + \cos(-x) + \mathbf{i}\sin(-x)\\ &= 2\cos(x).\end{aligned}$$

This allows us to show equality of the right sides of equations (2.35) and (2.36). Using equation (2.31) we see that

$$\begin{aligned}\left(\mathrm{e}^{(2j+1)\mathbf{i}\theta_\Psi}|\Psi_+\rangle - \mathrm{e}^{-(2j+1)\mathbf{i}\theta_\Psi}|\Psi_-\rangle\right) = \mathrm{e}^{(2j+1)\mathbf{i}\theta_\Psi}\frac{1}{\sqrt{2}}\left(\frac{1}{\sqrt{a_\Psi}}|\Psi_1\rangle + \frac{\mathbf{i}}{\sqrt{1-a_\Psi}}|\Psi_0\rangle\right)\\ - \mathrm{e}^{-(2j+1)\mathbf{i}\theta_\Psi}\frac{1}{\sqrt{2}}\left(\frac{1}{\sqrt{a_\Psi}}|\Psi_1\rangle - \frac{\mathbf{i}}{\sqrt{1-a_\Psi}}|\Psi_0\rangle\right),\end{aligned}$$

which can be reordered to

$$\left(\mathrm{e}^{(2j+1)\mathbf{i}\theta_\Psi}\left|\Psi_+\right\rangle - \mathrm{e}^{-(2j+1)\mathbf{i}\theta_\Psi}\left|\Psi_-\right\rangle\right) = \frac{1}{\sqrt{2}}\left(\left|\Psi_1\right\rangle \frac{1}{\sqrt{a_\Psi}}\left(\mathrm{e}^{(2j+1)\mathbf{i}\theta_\Psi} - \mathrm{e}^{-(2j+1)\mathbf{i}\theta_\Psi}\right) + \left|\Psi_0\right\rangle \frac{\mathbf{i}}{\sqrt{1-a_\Psi}}\left(\mathrm{e}^{(2j+1)\mathbf{i}\theta_\Psi} + \mathrm{e}^{-(2j+1)\mathbf{i}\theta_\Psi}\right)\right).$$

Using the identities from above this leads to

$$\begin{aligned}
&\frac{-\mathbf{i}}{\sqrt{2}}\left(\mathrm{e}^{(2j+1)\mathbf{i}\theta_\Psi}\left|\Psi_+\right\rangle - \mathrm{e}^{-(2j+1)\mathbf{i}\theta_\Psi}\left|\Psi_-\right\rangle\right) \\
&\quad = -\frac{\mathbf{i}}{2}\left(\left|\Psi_1\right\rangle \frac{1}{\sqrt{a_\Psi}} 2\mathbf{i}\sin((2j+1)\theta_\Psi) + \left|\Psi_0\right\rangle \frac{\mathbf{i}}{\sqrt{1-a_\Psi}} 2\cos\left((2j+1)\theta_\Psi\right)\right) \\
&\quad = \frac{1}{\sqrt{a_\Psi}}\sin((2j+1)\theta_\Psi)\left|\Psi_1\right\rangle + \frac{1}{\sqrt{1-a_\Psi}}\cos\left((2j+1)\theta_\Psi\right)\left|\Psi_0\right\rangle.
\end{aligned}$$

Now we will show, by induction, that they both are equal to $\mathbf{Q}^j\left|\Psi\right\rangle$. For $j=0$ we use the second representatin and see

$$\begin{aligned}
&\frac{1}{\sqrt{a_\Psi}}\sin((2j+1)\theta_\Psi)\left|\Psi_1\right\rangle + \frac{1}{\sqrt{1-a_\Psi}}\cos((2j+1)\theta_\Psi)\left|\Psi_0\right\rangle \\
&\overset{j=0}{=} \frac{1}{\sqrt{a_\Psi}}\sin(\theta_\Psi)\left|\Psi_1\right\rangle + \frac{1}{\sqrt{1-a_\Psi}}\cos\left(\theta_\Psi\right)\left|\Psi_0\right\rangle \\
&\overset{(2.33)}{=} \left|\Psi_1\right\rangle + \left|\Psi_0\right\rangle \\
&= \left|\Psi\right\rangle \\
&= \mathbf{Q}^0\left|\Psi\right\rangle.
\end{aligned}$$

Now let us check $j \to j+1$. We will use the first representation for that case and see that

$$\begin{aligned}
\mathbf{Q}^{j+1}\left|\Psi\right\rangle &= \mathbf{Q}\,\mathbf{Q}^j\left|\Psi\right\rangle \\
&= \mathbf{Q}\,\frac{-\mathbf{i}}{\sqrt{2}}\left(\mathrm{e}^{(2j+1)\mathbf{i}\theta_\Psi}\left|\Psi_+\right\rangle - \mathrm{e}^{-(2j+1)\mathbf{i}\theta_\Psi}\left|\Psi_-\right\rangle\right) \\
&= \frac{-\mathbf{i}}{\sqrt{2}}\left(\mathrm{e}^{(2j+1)\mathbf{i}\theta_\Psi}\,\mathbf{Q}\left|\Psi_+\right\rangle - \mathrm{e}^{-(2j+1)\mathbf{i}\theta_\Psi}\,\mathbf{Q}\left|\Psi_-\right\rangle\right).
\end{aligned}$$

Using lemma 2.14 we get

$$\begin{aligned}
\mathbf{Q}^{j+1}\left|\Psi\right\rangle &= \frac{-\mathbf{i}}{\sqrt{2}}\left(\mathrm{e}^{(2j+1)\mathbf{i}\theta_\Psi}\lambda_+\left|\Psi_+\right\rangle - \mathrm{e}^{-(2j+1)\mathbf{i}\theta_\Psi}\lambda_-\left|\Psi_-\right\rangle\right) \\
&\overset{(2.32)}{=} \frac{-\mathbf{i}}{\sqrt{2}}\left(\mathrm{e}^{(2j+1)\mathbf{i}\theta_\Psi}\mathrm{e}^{\mathbf{i}2\theta_\Psi}\left|\Psi_+\right\rangle - \mathrm{e}^{-(2j+1)\mathbf{i}\theta_\Psi}e^{-\mathbf{i}2\theta_\Psi}\left|\Psi_-\right\rangle\right) \\
&= \frac{-\mathbf{i}}{\sqrt{2}}\left(\mathrm{e}^{(2(j+1)+1)\mathbf{i}\theta_\Psi}\left|\Psi_+\right\rangle - \mathrm{e}^{-(2(j+1)+1)\mathbf{i}\theta_\Psi}\left|\Psi_-\right\rangle\right).
\end{aligned}$$

That concludes the induction and the proof. □

Now we have everything we need, to proof theorem 2.11.

Proof of theorem 2.11. We want to calculate the probability to measure a good state after j applications of $\mathbf{Q}$. We start with restating the desired probability as

$$\begin{aligned}\mathbb{P}\left[\chi\left(\mathcal{M}\left(\mathbf{Q}^j\left|\Psi\right\rangle\right)\right)=1\right] &= \sum_{\chi(k)=1}\mathbb{P}\left[\mathcal{M}\left(\mathbf{Q}^j\left|\Psi\right\rangle\right)=k\right] \\ &= \sum_{\chi(k)=1}\left\langle\mathbf{Q}^j\,\Psi\middle|e_k^n\right\rangle\left\langle e_k^n\middle|\mathbf{Q}^j\,\Psi\right\rangle \\ &= \sum_{\chi(k)=1}\left|\left\langle e_k^n\middle|\mathbf{Q}^j\,\Psi\right\rangle\right|^2.\end{aligned} \tag{2.37}$$

Using equation (2.36) from lemma 2.16 we see that

$$\left\langle e_k^n\middle|\mathbf{Q}^j\,\Psi\right\rangle = \left\langle e_k^n\right|\left(\frac{1}{\sqrt{a_\Psi}}\sin((2j+1)\theta_\Psi)\left|\Psi_1\right\rangle + \frac{1}{\sqrt{1-a_\Psi}}\cos\left((2j+1)\theta_\Psi\right)\left|\Psi_0\right\rangle\right).$$

By definition, $\left|\Psi_0\right\rangle$ is orthogonal to all $\left|e_k^n\right\rangle$ with $\chi(k)=1$, thus,

$$\left\langle e_k^n\middle|\mathbf{Q}^j\,\Psi\right\rangle = \frac{1}{\sqrt{a_\Psi}}\sin((2j+1)\theta_\Psi)\left\langle e_k^n\middle|\Psi_1\right\rangle.$$

Plugging this into equation (2.37) leads to

$$\mathbb{P}\left[\chi\left(\mathcal{M}\left(\mathbf{Q}^j\left|\Psi\right\rangle\right)\right)=1\right] = \frac{1}{a_\Psi}\sin^2((2j+1)\theta_\Psi)\sum_{\chi(k)=1}\left\langle e_k^n\middle|\Psi_1\right\rangle.$$

And by the definitions of Ψ_1 and a_Ψ we get

$$\mathbb{P}\left[\chi\left(\mathcal{M}\left(\mathbf{Q}^j\left|\Psi\right\rangle\right)\right)=1\right] = \frac{1}{a_\Psi}\sin^2((2j+1)\theta_\Psi)\left\langle\Psi_1\middle|\Psi_1\right\rangle = \sin^2((2j+1)\theta_\Psi).$$

□

Just like for the Grover's search algorithm, it is not trivial how to efficiently implement AA. Especially the oracle and the reflections for only some states can be difficult. The reflection around zero can be implemented as a multi controlled Z rotation [60].

Remark 2.17

One can interpret AA as a generalization of Grover's search algorithm. If we set $\mathbf{A} = \bigotimes_{i=1}^n \mathbf{H}$ and select only one state as a good one we are back at Grover's search algorithm. One benefit of AA over Grover's search algorithm is, that we have more precise information about the number of repetitions we have to perform, not just an upper bound.

Let us conclude this section by looking at the implications of theorem 2.11. We start with a quantum algorithm $\mathbf{A}$ and a subset of states we are interested in, specified by the oracle χ. With AA we can change, ideally increase, the probability to observe a state we specified, by executing powers of the grover operator $\mathbf{Q}$ instead of $\mathbf{A}$. Theorem 2.11 states, that the probability to observe a good state after j applications of $\mathbf{Q}$ is given by $\sin^2((2j+1)\theta_\Psi)$. First note, that this is not globally increasing in j, as the sine is

periodic, i.e., after *too many* applications of $\mathbf{Q}$ the probability starts to decrease. It also means, if we want to maximize this probability, there are infinite many values for j that do so. But as the algorithm becomes more expensive to actually run the higher the power of $\mathbf{Q}$ becomes, we are only interested in the smallest optimal j. If we know the probability to observe a good state in advance, we can solve

$$\sin^2((2j+1)\theta_\Psi) \stackrel{!}{=} 1$$

to get

$$j^* = \frac{1}{2}\left(\frac{\pi}{2\theta_\Psi} - 1\right).$$

This is very likely not an integer, and we have to round it, arguing with the symmetry of the sine, the closest integer to j^* is the optimal power for $\mathbf{Q}$.

In the next section we will look at another way of using theorem 2.11 for the case where we do not know the original success probability, but want to know it.

2.2.3 Amplitude Estimation

The result from theorem 2.11 can also be used to estimate an unknown amplitude. The general idea is to apply different powers of the Grover operator $\mathbf{Q}$ and observe the relative frequency of the state, or states, of interest. From that we can infer an estimate for the amplitude.

In the literature there are different algorithms to do just that. But before going into the details, we give a short argument why AE is interesting.

In finance, we are often interested in the expected value of some complicated random variables, for example pricing derivatives or calculating losses for solvency capital requirements. Often there are no analytical representations for the expected values and MC methods are used. In short MC means approximating the expected value of a random variable by the average value of many independent realizations of this random variable. The translation to AE is designing a quantum circuit that produces a state that has as its amplitude the value we are interested in. We then specify this one state as the only good one, thus AE gives that states amplitude. Instead of applying AE, one could alos repeatedly measure the circuit, i.e., producing realizations, and estimate the amplitude by the relative frequency of the state of interest. This will yield the same convergence properties as classical MC, as it really is classical MC, and the quantum computer is only used as a very sophisticated random number generator. At that point AE promises a speedup: On a quantum computer the probability to observe a good state can be estimated quadratically faster by AE, than by naive MC.

This idea was introduced in section 4 of Brassard et al. [13], and to the authors' knowledge, it is also the first version of AE introduced. From lemma 2.14 we know the eigenvalues of $\mathbf{Q}$ depend on the probability a_Ψ to observe a good state. So we will estimate one of the eigenvalues and calculate the amplitude from that. First we introduce the following operator.

Definition 2.18

Let $M = 2^m$ for $m \in \mathbb{N}$, $N = 2^n$ for $n \in \mathbb{N}$ and $\mathbf{U} \in \mathbb{C}^{N\times N}$ unitary. We define

$$\begin{aligned} \Lambda_M(\mathbf{U}) : \quad & \mathbb{C}^N \to \mathbb{C}^N \\ & \left|e_j^n\right\rangle \otimes |y\rangle \mapsto \left|e_j^n\right\rangle \otimes \left(\mathbf{U}^j |y\rangle\right) \quad 0 \le j < M. \end{aligned} \tag{2.38}$$

Note that for $|\phi\rangle$ an eigenvector of $\mathbf{U}$ with eigenvalue $\mathrm{e}^{2\pi \mathbf{i} w}$ we have

$$\Lambda_M(\mathbf{U}) : \left|e_j^n\right\rangle \otimes |\phi\rangle \mapsto \mathrm{e}^{2\pi \mathbf{i} w j}\left(\left|e_j^n\right\rangle \otimes |\phi\rangle\right).$$

With that we can state the algorithm described under the name *Est Amp* by Brassard et al. [13].

Algorithm 2.19 *Est Amp* by Brassard et al. [13]

Input: Quantum Algorithm $\mathbf{A}$ operating on n qubits.

1: Initialize two registers of sizes m and n to the state $|0\rangle \otimes \mathbf{A}|0\rangle$.
2: Apply $\mathbf{QFT}_\mathrm{m}$ from definition 2.2 to the first register.
3: With $\mathbf{Q}$ given as in equation (2.20) apply $\Lambda_M(\mathbf{Q})$ to both registers.
4: Apply $\mathbf{QFT}_\mathrm{m}^{-1}$ to the first register.
5: Measure the first register, and let j be the measured state.
6: **return** $\overline{a_\Psi} = \sin^2\left(\pi \frac{j}{M}\right)$

The rest of this section will retrace the proof, that this is indeed a good estimation of the searched amplitude, given by Brassard et al. [13] in a bit more detail. The main result is the following theorem.

Theorem 2.20 Theorem 12 by Brassard et al. [13]

Let $M = 2^m$ for $m \in \mathbb{N}$ where $m \ge 2$, $k \in \mathbb{N}_{\ge 0}$ and $\overline{a_\Psi}$ the output of algorithm 2.19. Then

$$\mathbb{P}\left[|\overline{a_\Psi} - a_\Psi| \le 2\pi k \frac{\sqrt{a_\Psi(1-a_\Psi)}}{M} + k^2 \frac{\pi^2}{M^2}\right] \ge \begin{cases} \frac{8}{\pi^2} & k = 1 \\ 1 - \frac{1}{2k-2} & k > 1 \end{cases}.$$

This theorem says, very much simplified, that the result of algorithm 2.19 will, with a probability of at least $\frac{8}{\pi^2} \approx 0.81$, be as close to the true value as we want it to be, as long as M is large enough.

Before we start the proof we need some more theory.

Definition 2.21

For $w_0, w_1 \in [0, 1)$ we define

$$d(w_0, w_1) = \min_{z\in\mathbb{Z}} \{|w_1 - w_0 + z|\}.$$

If we think of w_0 and w_1 as points on a circle with circumference one, we can interpret $d(w_0, w_1)$ as the minimal distance between the two points along this circle. See figure 2.3 for a graphical explanation. This also gives a visual explanation, why $d(w_0, w_1)$ will always be between 0 and 1/2.

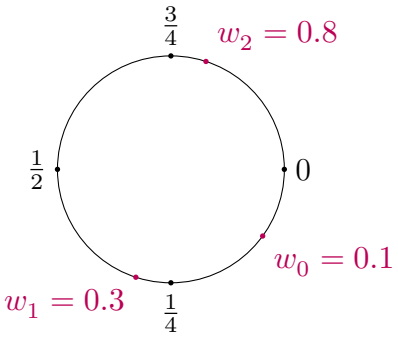


Figure 2.3: Graphical example for the distance from definition 2.21. Here we have $d(w_0, w_1) = |w_1 - w_0| = 0.2$ or $d(w_0, w_2) = |w_2 - w_0 - 1| = 0.3$. The distances can be interpreted as the shortest distance on the circle between the two points. We see that this value can never be above 0.5, since as it where, the other way around the circle would be shorter and below 0.5.

With the notation regarding the QFT from definition 2.2 we can state the following.

Lemma 2.22 Lemma 10 by Brassard et al. [13]

Let $w_0, w_1 \in [0, 1)$, if $d(w_0, w_1) = 0$ we have

$$|\langle \mathcal{S}_M(w_0) | \mathcal{S}_M(w_1) \rangle|^2 = 1,$$

otherwise

$$|\langle \mathcal{S}_M(w_0) | \mathcal{S}_M(w_1) \rangle|^2 = \frac{1}{M^2} \frac{\sin^2(M\pi d(w_0, w_1))}{\sin^2(\pi d(w_0, w_1))}.$$

Proof. First note that the ket of the QFT transform is given as

$$\langle \mathcal{S}_M(w_0)| = |\mathcal{S}_M(w_0)\rangle^\dagger \overset{(2.3)}{=} \frac{1}{\sqrt{M}} \sum_{j=0}^{M-1} \mathrm{e}^{-2\pi \mathbf{i} w_0 j} \left|e_j^n\right\rangle .$$

With that we can calculate

$$\begin{aligned}\langle \mathcal{S}_M(w_0) | \mathcal{S}_M(w_1) \rangle &= \frac{1}{M} \left(\sum_{j=0}^{M-1} \mathrm{e}^{-2\pi \mathbf{i} w_0 j} \left\langle e_j^n \right| \right) \left(\sum_{j=0}^{M-1} \mathrm{e}^{2\pi \mathbf{i} w_1 j} \left| e_j^n \right\rangle \right) \\ &= \frac{1}{M} \sum_{j=0}^{M-1} \sum_{k=0}^{M-1} \mathrm{e}^{-2\pi \mathbf{i} w_0 j} \mathrm{e}^{2\pi \mathbf{i} w_1 k} \left\langle e_j^n | e_k^n \right\rangle .\end{aligned}$$

Now we use that $\left\langle e_j^n | e_k^n \right\rangle$ is 1 if and only if $j = k$ and 0 else, to get

$$\langle \mathcal{S}_M(w_0) | \mathcal{S}_M(w_1) \rangle = \frac{1}{M} \sum_{j=0}^{M-1} \mathrm{e}^{2\pi \mathbf{i} w_1 j - 2\pi \mathbf{i} w_0 j} = \frac{1}{M} \sum_{j=0}^{M-1} \mathrm{e}^{2\pi \mathbf{i} j (w_1 - w_0)}.$$

We see that from $0 = d(w_0, w_1) = \min_{z \in \mathbb{Z}} \{|w_1 - w_0 + z|\}$ and $w_0, w_1 \in [0, 1)$ follows that $w_0 = w_1$ thus, every element of the sum is one, and we have shown the first part of the claim.

For the second part, we assume that $d(w_0, w_1) > 0$ and therefore $w_0 \neq w_1$. This guarantees that $\mathrm{e}^{2\pi\mathbf{i}(w_1-w_0)} \neq 1$, and we can use the closed formula for the partial sum of a geometric series

$$\sum_{j=0}^{n-1} ar^j = a\frac{1-r^n}{1-r},$$

if $r \neq 1$, to get

$$\begin{aligned}
\sum_{j=0}^{M-1} \mathrm{e}^{2\pi\mathbf{i}j(w_1-w_0)} &= \sum_{j=0}^{M-1} \left(\mathrm{e}^{2\pi\mathbf{i}(w_1-w_0)}\right)^j \\
&= \frac{1-\left(\mathrm{e}^{2\pi\mathbf{i}(w_1-w_0)}\right)^M}{1-\mathrm{e}^{2\pi\mathbf{i}(w_1-w_0)}} \\
&= \frac{1-\mathrm{e}^{2M\pi\mathbf{i}(w_1-w_0)}}{1-\mathrm{e}^{2\pi\mathbf{i}(w_1-w_0)}} \\
&= \frac{\mathrm{e}^{M\pi\mathbf{i}(w_1-w_0)}\left(\mathrm{e}^{-M\pi\mathbf{i}(w_1-w_0)} - \mathrm{e}^{M\pi\mathbf{i}(w_1-w_0)}\right)}{\mathrm{e}^{\pi\mathbf{i}(w_1-w_0)}\left(\mathrm{e}^{-\pi\mathbf{i}(w_1-w_0)} - \mathrm{e}^{\pi\mathbf{i}(w_1-w_0)}\right)} \\
&= \frac{\mathrm{e}^{M\pi\mathbf{i}(w_1-w_0)}\left(\mathrm{e}^{M\pi\mathbf{i}(w_1-w_0)} - \mathrm{e}^{-M\pi\mathbf{i}(w_1-w_0)}\right)2\mathbf{i}}{\mathrm{e}^{\pi\mathbf{i}(w_1-w_0)}\left(\mathrm{e}^{\pi\mathbf{i}(w_1-w_0)} - \mathrm{e}^{-\pi\mathbf{i}(w_1-w_0)}\right)2\mathbf{i}} \\
&= \frac{\mathrm{e}^{M\pi\mathbf{i}(w_1-w_0)}\sin(M\pi(w_1-w_0))}{\mathrm{e}^{\pi\mathbf{i}(w_1-w_0)}\sin(\pi(w_1-w_0))} \\
&= \mathrm{e}^{\pi\mathbf{i}(w_1-w_0)(M-1)}\frac{\sin(M\pi(w_1-w_0))}{\sin(\pi(w_1-w_0))}.
\end{aligned}$$

Finally, we get

$$\begin{aligned}
|\langle \mathcal{S}_M(w_0)|\mathcal{S}_M(w_1)\rangle|^2 &= \left|\frac{1}{M}\sum_{j=0}^{M-1} \mathrm{e}^{2\pi\mathbf{i}j(w_1-w_0)}\right|^2 \\
&= \frac{1}{M^2}\left|\mathrm{e}^{\pi\mathbf{i}(w_1-w_0)(M-1)}\frac{\sin(M\pi(w_1-w_0))}{\sin(\pi(w_1-w_0))}\right|^2 \\
&= \frac{1}{M^2}\left|\mathrm{e}^{\pi\mathbf{i}(w_1-w_0)(M-1)}\right|^2\left|\frac{\sin(M\pi(w_1-w_0))}{\sin(\pi(w_1-w_0))}\right|^2 \\
&= \frac{1}{M^2}\frac{\sin^2(M\pi(w_1-w_0))}{\sin^2(\pi(w_1-w_0))}.
\end{aligned}$$

Now we can use that $\sin^2(\pi x)$ has a periodicity of one and is symmetric around zero, to see that for $k \in \mathbb{N}$ and $z \in \mathbb{Z}$

$$\sin^2(k\pi(w_1-w_0)) = \sin^2(k\pi(w_1-w_0+z)) = \sin^2(k\pi|w_1-w_0+z|).$$

Thus,

$$|\langle \mathcal{S}_M(w_0)|\mathcal{S}_M(w_1)\rangle|^2 = \frac{1}{M^2}\frac{\sin^2(M\pi d(w_0,w_1))}{\sin^2(\pi d(w_0,w_1))}.$$

□

For brevity, we will use the following notation

$$|\zeta_w\rangle := \mathbf{QFT}_{\mathrm{m}}^{-1} |\mathcal{S}_M(w)\rangle ,$$

and

$$J_M := \{j \in \mathbb{N} \mid j < M\}.$$

Lemma 2.23 First part of theorem 11 from Brassard et al. [13]

Let $w \in [0,1)$, if $Mw \in \mathbb{N}$

$$\mathbb{P}[\mathcal{M}(|\zeta_w\rangle) = Mw] = 1. \tag{2.39}$$

If $Mw \notin \mathbb{N}$ and $j \in J_M$

$$\mathbb{P}[\mathcal{M}(|\zeta_w\rangle) = j] = \frac{\sin^2\left(M\pi d\left(\frac{j}{M}, w\right)\right)}{M^2 \sin^2\left(\pi d\left(\frac{j}{M}, w\right)\right)} \tag{2.40}$$

with an upper bound

$$\mathbb{P}[\mathcal{M}(|\zeta_w\rangle) = j] \leq \frac{1}{\left(2Md\left(\frac{j}{M}, w\right)\right)^2}. \tag{2.41}$$

Proof. For $j \in J_M$ we have

$$\mathbb{P}[\mathcal{M}(|\zeta_w\rangle) = j] \overset{(1.10)}{=} \left|\langle e_j^n| \mathbf{QFT}_{\mathrm{m}}^{-1} |\mathcal{S}_M(w)\rangle\right|^2.$$

Now we use the general property of all quantum operations, that the inverse equals the complex conjugated operation, and get

$$\begin{aligned}
\mathbb{P}[\mathcal{M}(|\zeta_w\rangle) = j] &= \left|\left(\mathbf{QFT}_{\mathrm{m}} |e_j^n\rangle\right)^\dagger |\mathcal{S}_M(w)\rangle\right|^2 \\
&\overset{(2.2)}{=} \left|\left(\left|\mathcal{S}_M\left(\tfrac{j}{M}\right)\right\rangle\right)^\dagger |\mathcal{S}_M(w)\rangle\right|^2 \\
&= \left|\left\langle \mathcal{S}_M\left(\tfrac{j}{M}\right) \middle| \mathcal{S}_M(w)\right\rangle\right|^2.
\end{aligned}$$

For $j = Mw \in \mathbb{N}$ we have

$$d\left(\frac{j}{M}, w\right) = d(w, w) = 0$$

and by the first case of lemma 2.22 shown equation (2.39). In general, we have, again by lemma 2.22,

$$\mathbb{P}[\mathcal{M}(|\zeta_w\rangle) = j] = \left|\left\langle \mathcal{S}_M\left(\tfrac{j}{M}\right) \middle| \mathcal{S}_M(w)\right\rangle\right|^2 = \frac{1}{M^2} \frac{\sin^2\left(M\pi d\left(\frac{j}{M}, w\right)\right)}{\sin^2\left(\pi d\left(\frac{j}{M}, w\right)\right)}.$$

Next, we show the upper bound given in equation (2.41). First note, that the sine is always bounded by one, thus

$$\frac{1}{M^2} \frac{\sin^2\left(M\pi d\left(\frac{j}{M}, w\right)\right)}{\sin^2\left(\pi d\left(\frac{j}{M}, w\right)\right)} \leq \frac{1}{M^2} \frac{1}{\sin^2\left(\pi d\left(\frac{j}{M}, w\right)\right)}.$$

Then, we use that by definition 2.21 we have $\pi d\left(\frac{j}{M}, w\right) \in [0, \frac{\pi}{2}]$, and therefore

$$\frac{1}{M^2}\frac{\sin^2\left(M\pi d\left(\frac{j}{M}, w\right)\right)}{\sin^2\left(\pi d\left(\frac{j}{M}, w\right)\right)} \leq \frac{1}{M^2}\frac{1}{\sin^2\left(\pi d\left(\frac{j}{M}, w\right)\right)} \overset{(B.11)}{\leq} \frac{1}{\left(2Md\left(\frac{j}{M}, w\right)\right)^2}.$$

□

Lemma 2.24 Second part of theorem 11 by Brassard et al. [13]

Let $w \in [0, 1)$, we have for $k > 1$

$$\mathbb{P}\left[d\left(\frac{\mathcal{M}(|\zeta_w\rangle)}{M}, w\right) \leq \frac{k}{M}\right] \geq 1 - \frac{1}{2k-2}$$

and for $M > 2$

$$\mathbb{P}\left[d\left(\frac{\mathcal{M}(|\zeta_w\rangle)}{M}, w\right) \leq \frac{1}{M}\right] \geq \frac{8}{\pi^2}.$$

Proof. For the first claim we look at the opposite event and decompose it to get

$$\begin{aligned}\mathbb{P}\left[d\left(\frac{\mathcal{M}(|\zeta_w\rangle)}{M}, w\right) \leq \frac{k}{M}\right] &= 1 - \mathbb{P}\left[d\left(\frac{\mathcal{M}(|\zeta_w\rangle)}{M}, w\right) > \frac{k}{M}\right] \\ &= 1 - \sum_{j=k}^{\infty} \mathbb{P}\left[\frac{j}{M} < d\left(\frac{\mathcal{M}(|\zeta_w\rangle)}{M}, w\right) \leq \frac{j+1}{M}\right].\end{aligned}$$

We look at each summand individually. As $\mathcal{M}(|\zeta_w\rangle)$ will always be an integer there are only two possible values it can take, that fulfill

$$\frac{j}{M} < d\left(\frac{\mathcal{M}(|\zeta_w\rangle)}{M}, w\right) \leq \frac{j+1}{M}, \tag{2.42}$$

we call them j_1 and j_2. Here j_1/M is smaller than w and j_2/M bigger by some values between j/M and $j+1/M$. With that we can rewrite

$$\begin{aligned}\mathbb{P}\left[\frac{j}{M} < d\left(\frac{\mathcal{M}(|\zeta_w\rangle)}{M}, w\right) \leq \frac{j+1}{M}\right] &= \mathbb{P}[\mathcal{M}(|\zeta_w\rangle) = j_1] + \mathbb{P}[\mathcal{M}(|\zeta_w\rangle) = j_2] \\ &\overset{(2.41)}{\leq} \frac{1}{\left(2Md\left(\frac{j_1}{M}, w\right)\right)^2} + \frac{1}{\left(2Md\left(\frac{j_2}{M}, w\right)\right)^2} \\ &\overset{(2.42)}{\leq} \frac{1}{2M^2\left(\frac{j}{M}\right)^2} \\ &= \frac{1}{2j^2}.\end{aligned}$$

Also note that since $0 \leq d(w_1, w_2) \leq \frac{1}{2}$ for all $w_1, w_2 \in [0, 1)$

$$\mathbb{P}\left[\frac{j}{M} < d\left(\frac{\mathcal{M}(|\zeta_w\rangle)}{M}, w\right) \leq \frac{j+1}{M}\right] \overset{j \geq \frac{M}{2}}{=} 0.$$

Now we know that

$$\mathbb{P}\left[d\left(\frac{\mathcal{M}(|\zeta_w\rangle)}{M}, w\right) \leq \frac{k}{M}\right] \geq 1 - \sum_{j=k}^{\frac{M}{2}} \frac{1}{2j^2}.$$

With noting that

$$\begin{aligned}
\sum_{j=k}^{\frac{M}{2}} \frac{1}{j^2} &\leq \sum_{j=k}^{\frac{M}{2}} \frac{1}{j}\frac{1}{j-1} \\
&= \sum_{j=k}^{\frac{M}{2}} \left(\frac{1}{j-1} - \frac{1}{j}\right) \\
&= \sum_{j=k}^{\frac{M}{2}} \frac{1}{j-1} - \sum_{j=k}^{\frac{M}{2}} \frac{1}{j} \\
&= \sum_{j=k-1}^{\frac{M}{2}-1} \frac{1}{j} - \sum_{j=k}^{\frac{M}{2}} \frac{1}{j} \\
&= \frac{1}{k-1} - \frac{2}{M}
\end{aligned}$$

we get

$$\begin{aligned}
\mathbb{P}\left[d\left(\frac{\mathcal{M}(|\zeta_w\rangle)}{M}, w\right) \leq \frac{k}{M}\right] &\geq 1 - \frac{1}{2}\sum_{j=k}^{\frac{M}{2}} \frac{1}{j^2} \\
&\geq 1 - \frac{1}{2}\left(\frac{1}{k-1} - \frac{2}{M}\right) \\
&= 1 - \frac{1}{2k-2} + \frac{1}{M} \\
&> 1 - \frac{1}{2k-2},
\end{aligned}$$

and we have shown the first claim.

To show the second claim we use again that $\mathcal{M}(|\zeta_w\rangle)$ will always be an integer and therefore only two values fulfill the inequality $d(\mathcal{M}(|\zeta_w\rangle)/M, w) \leq \frac{1}{M}$, therefore

$$\begin{aligned}
\mathbb{P}\left[d\left(\frac{\mathcal{M}(|\zeta_w\rangle)}{M}, w\right) \leq \frac{1}{M}\right] &= \mathbb{P}[\mathcal{M}(|\zeta_w\rangle) = \lceil Mw \rceil] + \mathbb{P}[\mathcal{M}(|\zeta_w\rangle) = \lfloor Mw \rfloor] \\
&\overset{(2.40)}{=} \frac{1}{M^2}\frac{\sin^2\left(M\pi d\left(\frac{\lceil Mw \rceil}{M}, w\right)\right)}{\sin^2\left(\pi d\left(\frac{\lceil Mw \rceil}{M}, w\right)\right)} + \frac{1}{M^2}\frac{\sin^2\left(M\pi d\left(\frac{\lfloor Mw \rfloor}{M}, w\right)\right)}{\sin^2\left(\pi d\left(\frac{\lfloor Mw \rfloor}{M}, w\right)\right)}.
\end{aligned}$$

We know that $Mw \notin \mathbb{N}$ thus $d\left(\frac{\lceil Mw \rceil}{M}, w\right), d\left(\frac{\lfloor Mw \rfloor}{M}, w\right) \neq 0$, so there is no problem with potentially dividing by zero. We use again the fact that $d\left(\frac{\lceil Mw \rceil}{M}, w\right), d\left(\frac{\lfloor Mw \rfloor}{M}, w\right) \in (0, \frac{1}{2}]$

to see that

$$\begin{aligned}
\mathbb{P}\left[d\left(\frac{\mathcal{M}(|\zeta_w\rangle)}{M}, w\right) \leq \frac{1}{M}\right] &= \frac{1}{M^2}\frac{\sin^2\left(M\pi d\left(\frac{\lceil Mw\rceil}{M}, w\right)\right)}{\sin^2\left(\pi d\left(\frac{\lceil Mw\rceil}{M}, w\right)\right)} + \frac{1}{M^2}\frac{\sin^2\left(M\pi d\left(\frac{\lfloor Mw\rfloor}{M}, w\right)\right)}{\sin^2\left(\pi d\left(\frac{\lfloor Mw\rfloor}{M}, w\right)\right)} \\
&\overset{(B.10)}{\geq} \frac{1}{M^2}\frac{\sin^2\left(M\pi d\left(\frac{\lceil Mw\rceil}{M}, w\right)\right)}{\left(\pi d\left(\frac{\lceil Mw\rceil}{M}, w\right)\right)^2} + \frac{1}{M^2}\frac{\sin^2\left(M\pi d\left(\frac{\lfloor Mw\rfloor}{M}, w\right)\right)}{\left(\pi d\left(\frac{\lfloor Mw\rfloor}{M}, w\right)\right)^2} \\
&\overset{(B.11)}{\geq} \frac{1}{M^2}\frac{\left(2Md\left(\frac{\lceil Mw\rceil}{M}, w\right)\right)^2}{\left(\pi d\left(\frac{\lceil Mw\rceil}{M}, w\right)\right)^2} + \frac{1}{M^2}\frac{\left(2Md\left(\frac{\lfloor Mw\rfloor}{M}, w\right)\right)^2}{\left(\pi d\left(\frac{\lfloor Mw\rfloor}{M}, w\right)\right)^2} \\
&= \frac{8}{\pi^2},
\end{aligned}$$

and conclude the proof. □

Lemma 2.25

Let $\theta, \bar{\theta} \in \mathbb{R}$ and $a = \sin^2(\theta)$, $\bar{a} = \sin^2\left(\bar{\theta}\right)$

$$\left|\theta - \bar{\theta}\right| \leq \epsilon \Rightarrow |a - \bar{a}| \leq 2\epsilon\sqrt{a(1-a)} + \epsilon^2.$$

Proof. First we introduce

$$\gamma := \bar{\theta} - \theta.$$

Now we show four additional identities we will need. First

$$2\sin(\theta)\cos(\gamma)\cos(\theta)\sin(\gamma) \overset{(B.6)}{=} \sin(2\theta)\sin(\gamma)\cos(\gamma), \tag{2.43}$$

second

$$\begin{aligned}
&\sin^2(\theta)\cos^2(\gamma) + \cos^2(\theta)\sin^2(\gamma) - \sin^2(\theta) \\
&\quad= \cos^2(\theta)\sin^2(\gamma) + \sin^2(\theta)\left(\cos^2(\gamma) - 1\right) \\
&\quad\overset{(B.5)}{=} \cos^2(\theta)\sin^2(\gamma) - \sin^2(\theta)\sin^2(\gamma) \\
&\quad= \sin^2(\gamma)\left(\cos^2(\theta) - \sin^2(\theta)\right) \\
&\quad\overset{(B.7)}{=} \sin^2(\gamma)\cos(2\theta),
\end{aligned} \tag{2.44}$$

third

$$\frac{\sin(2\theta)}{2} \overset{(B.6)}{=} \sin(\theta)\cos(\theta) = \sqrt{\sin^2(\theta)\cos^2(\theta)} \overset{(B.5)}{=} \sqrt{\sin^2(\theta)\left(1 - \sin^2(\theta)\right)} \tag{2.45}$$

and fourth

$$\sin^2(\gamma)\cos(2\theta) \overset{(B.9)}{=} \sin^2(\gamma)\left(1 - 2\sin^2(\theta)\right). \tag{2.46}$$

Putting everything together we get

$$\begin{aligned}
&\sin^2\left(\bar{\theta}\right) - \sin^2(\theta) \\
&\quad = \quad \sin^2(\theta+\gamma) - \sin^2(\theta) \\
&\quad \overset{(B.3)}{=} \quad (\sin(\theta)\cos(\gamma) + \cos(\theta)\sin(\gamma))^2 - \sin^2(\theta) \\
&\quad = \quad \sin^2(\theta)\cos^2(\gamma) + 2\sin(\theta)\cos(\gamma)\cos(\theta)\sin(\gamma) + \cos^2(\theta)\sin^2(\gamma) - \sin^2(\theta) \\
&\quad \overset{(2.43),(2.44)}{=} \quad \sin(2\theta)\sin(\gamma)\cos(\gamma) + \sin^2(\gamma)\cos(2\theta) \\
&\quad \overset{(B.6)}{=} \quad \frac{\sin(2\theta)\sin(2\gamma)}{2} + \sin^2(\gamma)\cos(2\theta) \\
&\quad \overset{(2.45),(2.46)}{=} \quad \sqrt{\sin^2(\theta)\left(1-\sin^2(\theta)\right)}\sin(2\gamma) + \left(1-2\sin^2(\theta)\right)\sin^2(\gamma) \\
&\quad = \quad \sqrt{a(1-a)}\sin(2\gamma) + (1-2a)\sin^2(\gamma).
\end{aligned} \tag{2.47}$$

To give a bound for the absolute value of $\sin^2\left(\bar{\theta}\right) - \sin^2(\theta)$ we give it for the expression itself and for its negative. To do so note that $\sin(x) \leq |x|$ and since $a \in [0,1]$ by properties of the sine, we have $1-2a \leq 1$. We further have by assumption that $|\gamma| \leq \epsilon$, thus,

$$\begin{aligned}
\sin^2\left(\bar{\theta}\right) - \sin^2(\theta) &\overset{(2.47)}{=} \sqrt{a(1-a)}\sin(2\gamma) + (1-2a)\sin^2(\gamma) \\
&\leq 2|\gamma|\sqrt{a(1-a)} + \gamma^2 \\
&\leq 2\epsilon\sqrt{a(1-a)} + \epsilon^2.
\end{aligned}$$

We also have $-\sin(x) \leq |x|$ and $2a-1 \leq 1$, therefore

$$\begin{aligned}
\sin^2(\theta) - \sin^2\left(\bar{\theta}\right) &\overset{(2.47)}{=} (2a-1)\sin^2(\gamma) - \sqrt{a(1-a)}\sin(2\gamma) \\
&\leq \gamma^2 + 2|\gamma|\sqrt{a(1-a)} \\
&\leq 2\epsilon\sqrt{a(1-a)} + \epsilon^2.
\end{aligned}$$

So in total we have

$$|a-\bar{a}| = \max\left\{\sin^2\left(\bar{\theta}\right) - \sin^2(\theta), \sin^2(\theta) - \sin^2\left(\bar{\theta}\right)\right\} \leq 2\epsilon\sqrt{a(1-a)} + \epsilon^2$$

and thus, shown the claim. □

We have now everything we need to proof the central theorem of this section.

Proof of theorem 2.20. We will go through the algorithm step by step. After the first step the quantum state will be

$$|\Upsilon_1\rangle := |0\rangle \otimes \mathbf{A}|0\rangle \overset{(2.28)}{=} |0\rangle \otimes |\Psi\rangle \overset{(2.35)}{=} |0\rangle \otimes \frac{-\mathbf{i}}{\sqrt{2}}(\mathrm{e}^{\mathbf{i}\theta_\Psi}|\Psi_+\rangle - \mathrm{e}^{-\mathbf{i}\theta_\Psi}|\Psi_-\rangle).$$

We can ignore a global phase of $-\mathbf{i}/2$ and write

$$|\Upsilon_1\rangle = \frac{1}{\sqrt{2}}|0\rangle \otimes (\mathrm{e}^{\mathbf{i}\theta_\Psi}|\Psi_+\rangle - \mathrm{e}^{-\mathbf{i}\theta_\Psi}|\Psi_-\rangle). \tag{2.48}$$

Applying the QFT to the first register yields

$$\begin{aligned}
|\Upsilon_2\rangle &:= (\mathbf{QFT}_m \otimes \mathbf{I})\,|\Upsilon_1\rangle \\
&\overset{(2.48)}{=} \frac{1}{\sqrt{2}}\,\mathbf{QFT}_m\,|0\rangle \otimes \left(e^{\mathbf{i}\theta_\Psi}\,|\Psi_+\rangle - e^{-\mathbf{i}\theta_\Psi}\,|\Psi_-\rangle\right) \\
&\overset{(2.2)}{=} \frac{1}{\sqrt{2}}\,|\mathcal{S}_M(0)\rangle \otimes \left(e^{\mathbf{i}\theta_\Psi}\,|\Psi_+\rangle - e^{-\mathbf{i}\theta_\Psi}\,|\Psi_-\rangle\right) \\
&\overset{(2.3)}{=} \frac{1}{\sqrt{2}}\left(\frac{1}{\sqrt{M}}\sum_{j=0}^{M-1} e^{0}\,|e_j^n\rangle\right) \otimes \left(e^{\mathbf{i}\theta_\Psi}\,|\Psi_+\rangle - e^{-\mathbf{i}\theta_\Psi}\,|\Psi_-\rangle\right) \\
&= \frac{1}{\sqrt{2M}}\sum_{j=0}^{M-1}\left(|e_j^n\rangle \otimes \left(e^{\mathbf{i}\theta_\Psi}\,|\Psi_+\rangle - e^{-\mathbf{i}\theta_\Psi}\,|\Psi_-\rangle\right)\right).
\end{aligned}$$

Now we apply $\Lambda_M(\mathbf{Q})$ and get

$$\begin{aligned}
|\Upsilon_3\rangle &:= \Lambda_M(\mathbf{Q})\,|\Upsilon_2\rangle \\
&= \frac{1}{\sqrt{2M}}\Lambda_M(\mathbf{Q})\sum_{j=0}^{M-1}\left(|e_j^n\rangle \otimes \left(e^{\mathbf{i}\theta_\Psi}\,|\Psi_+\rangle - e^{-\mathbf{i}\theta_\Psi}\,|\Psi_-\rangle\right)\right) \\
&\overset{(2.38)}{=} \frac{1}{\sqrt{2M}}\sum_{j=0}^{M-1}\left(|e_j^n\rangle \otimes \mathbf{Q}^j\left(e^{\mathbf{i}\theta_\Psi}\,|\Psi_+\rangle - e^{-\mathbf{i}\theta_\Psi}\,|\Psi_-\rangle\right)\right).
\end{aligned}$$

By lemma 2.14 we can replace the applications of $\mathbf{Q}$ by its eigenvalues

$$\begin{aligned}
|\Upsilon_3\rangle &= \frac{1}{\sqrt{2M}}\sum_{j=0}^{M-1}\left(|e_j^n\rangle \otimes \mathbf{Q}^j\left(e^{\mathbf{i}\theta_\Psi}\,|\Psi_+\rangle - e^{-\mathbf{i}\theta_\Psi}\,|\Psi_-\rangle\right)\right) \\
&= \frac{1}{\sqrt{2M}}\sum_{j=0}^{M-1}\left(|e_j^n\rangle \otimes \left(e^{\mathbf{i}\theta_\Psi}e^{2\mathbf{i}j\theta_\Psi}\,|\Psi_+\rangle - e^{-\mathbf{i}\theta_\Psi}e^{-2\mathbf{i}j\theta_\Psi}\,|\Psi_-\rangle\right)\right).
\end{aligned}$$

Now we see, that we can express $|\Upsilon_3\rangle$ in terms of $\mathcal{S}_M$

$$\begin{aligned}
|\Upsilon_3\rangle &= \frac{e^{\mathbf{i}\theta_\Psi}}{\sqrt{2M}}\sum_{j=0}^{M-1}\left(e^{2\mathbf{i}j\theta_\Psi}\,|e_j^n\rangle \otimes |\Psi_+\rangle\right) - \frac{e^{-\mathbf{i}\theta_\Psi}}{\sqrt{2M}}\sum_{j=0}^{M-1}\left(e^{-2\mathbf{i}j\theta_\Psi}\,|e_j^n\rangle \otimes |\Psi_-\rangle\right) \\
&= \frac{e^{\mathbf{i}\theta_\Psi}}{\sqrt{2M}}\left(\sum_{j=0}^{M-1} e^{2\mathbf{i}j\theta_\Psi}\,|e_j^n\rangle\right) \otimes |\Psi_+\rangle - \frac{e^{-\mathbf{i}\theta_\Psi}}{\sqrt{2M}}\left(\sum_{j=0}^{M-1} e^{-2\mathbf{i}j\theta_\Psi}\,|e_j^n\rangle\right) \otimes |\Psi_-\rangle \\
&= \frac{e^{\mathbf{i}\theta_\Psi}}{\sqrt{2M}}\left(\sum_{j=0}^{M-1} e^{2\mathbf{i}j\theta_\Psi}\,|e_j^n\rangle\right) \otimes |\Psi_+\rangle - \frac{e^{-\mathbf{i}\theta_\Psi}}{\sqrt{2M}}\left(\sum_{j=0}^{M-1} e^{2\mathbf{i}j\pi-2\mathbf{i}j\theta_\Psi}\,|e_j^n\rangle\right) \otimes |\Psi_-\rangle \\
&= \frac{e^{\mathbf{i}\theta_\Psi}}{\sqrt{2}}\left|\mathcal{S}_M\left(\tfrac{\theta_\Psi}{\pi}\right)\right\rangle \otimes |\Psi_+\rangle - \frac{e^{-\mathbf{i}\theta_\Psi}}{\sqrt{2}}\left|\mathcal{S}_M\left(1-\tfrac{\theta_\Psi}{\pi}\right)\right\rangle \otimes |\Psi_-\rangle\,.
\end{aligned}$$

In the last step we apply the inverse QFT to the first register and get the final state

$$\begin{aligned}
|\Upsilon_4\rangle &:= (\mathbf{QFT}_m^{-1} \otimes \mathbf{I})\,|\Upsilon_3\rangle \\
&= (\mathbf{QFT}_m^{-1} \otimes \mathbf{I})\left(\frac{e^{\mathbf{i}\theta_\Psi}}{\sqrt{2}}\left|\mathcal{S}_M\left(\tfrac{\theta_\Psi}{\pi}\right)\right\rangle \otimes |\Psi_+\rangle - \frac{e^{-\mathbf{i}\theta_\Psi}}{\sqrt{2}}\left|\mathcal{S}_M\left(1-\tfrac{\theta_\Psi}{\pi}\right)\right\rangle \otimes |\Psi_-\rangle\right) \\
&= \frac{e^{\mathbf{i}\theta_\Psi}}{\sqrt{2}}\left(\mathbf{QFT}_m^{-1}\left|\mathcal{S}_M\left(\tfrac{\theta_\Psi}{\pi}\right)\right\rangle \otimes |\Psi_+\rangle\right) - \frac{e^{-\mathbf{i}\theta_\Psi}}{\sqrt{2}}\left(\mathbf{QFT}_m^{-1}\left|\mathcal{S}_M\left(1-\tfrac{\theta_\Psi}{\pi}\right)\right\rangle \otimes |\Psi_-\rangle\right).
\end{aligned}$$

We see that when measuring the first register we will, with equal probabilities, measure from state $\mathbf{QFT}_{\mathrm{m}}^{-1}\left|\mathcal{S}_M\left(\frac{\theta_\Psi}{\pi}\right)\right\rangle$ or from state $\mathbf{QFT}_{\mathrm{m}}^{-1}\left|\mathcal{S}_M\left(1-\frac{\theta_\Psi}{\pi}\right)\right\rangle$. We introduce

$$\mathcal{M}_1 := \mathcal{M}\left(\mathbf{QFT}_{\mathrm{m}}^{-1}\left|\mathcal{S}_M\left(\tfrac{\theta_\Psi}{\pi}\right)\right\rangle\right), \qquad \mathcal{M}_2 := \mathcal{M}\left(\mathbf{QFT}_{\mathrm{m}}^{-1}\left|\mathcal{S}_M\left(1-\tfrac{\theta_\Psi}{\pi}\right)\right\rangle\right).$$

Since we have $M > 2$ we know from lemma 2.23 that

$$\mathbb{P}\left[d\left(\frac{\mathcal{M}_1}{M}, \frac{\theta_\Psi}{\pi}\right) \leq \frac{k}{M}\right] \geq \begin{cases} \frac{8}{\pi^2} & k = 1 \\ 1 - \frac{1}{2k-1} & k > 1 \end{cases} \tag{2.49}$$

and

$$\mathbb{P}\left[d\left(\frac{\mathcal{M}_2}{M}, 1-\frac{\theta_\Psi}{\pi}\right) \leq \frac{k}{M}\right] \geq \begin{cases} \frac{8}{\pi^2} & k = 1 \\ 1 - \frac{1}{2k-1} & k > 1 \end{cases}. \tag{2.50}$$

Now we will express the second equation in terms of $\mathcal{M}_1$ instead of $\mathcal{M}_2$. First we note that

$$d\left(\frac{j}{M}, \frac{\theta_\Psi}{\pi}\right) = \min_{z\in\mathbb{Z}}\left|\frac{\theta_\Psi}{\pi} - \frac{j}{M} + z\right| = \min_{z\in\mathbb{Z}}\left|1 - \frac{\theta_\Psi}{\pi} - 1 + \frac{j}{M} + z\right| = d\left(\frac{M-j}{M}, 1-\frac{\theta_\Psi}{\pi}\right).$$

If we plug this into equation (2.40), we get

$$\mathbb{P}[\mathcal{M}_1 = j] = \frac{\sin^2\left(M\pi d\left(\frac{j}{M}, \frac{\theta_\Psi}{\pi}\right)\right)}{M^2\sin^2\left(\pi d\left(\frac{j}{M}, \frac{\theta_\Psi}{\pi}\right)\right)} = \frac{\sin^2\left(M\pi d\left(\frac{M-j}{M}, 1-\frac{\theta_\Psi}{\pi}\right)\right)}{M^2\sin^2\left(\pi d\left(\frac{M-j}{M}, 1-\frac{\theta_\Psi}{\pi}\right)\right)} = \mathbb{P}[\mathcal{M}_2 = M-j].$$

With that we get

$$\mathbb{P}\left[d\left(\frac{\mathcal{M}_2}{M}, 1-\frac{\theta_\Psi}{\pi}\right) \leq \frac{k}{M}\right] = \mathbb{P}\left[d\left(\frac{M-\mathcal{M}_1}{M}, 1-\frac{\theta_\Psi}{\pi}\right) \leq \frac{k}{M}\right].$$

Now we translate the bounds on θ_Ψ to bounds for a_Ψ, first for equation (2.49)

$$\begin{aligned} & d\left(\frac{\mathcal{M}_1}{M}, \frac{\theta_\Psi}{\pi}\right) \leq \frac{k}{M} \\ \Leftrightarrow \quad & \min_{z\in\mathbb{Z}}\left|\frac{\theta_\Psi}{\pi} - \frac{\mathcal{M}_1}{M} + z\right| \leq \frac{k}{M} \\ \Leftrightarrow \quad & \min_{z\in\mathbb{Z}}\left|\theta_\Psi - \left(\frac{\pi\mathcal{M}_1}{M} + \pi z\right)\right| \leq \frac{\pi k}{M}, \end{aligned}$$

then for equation (2.50)

$$\begin{aligned} & d\left(\frac{M-\mathcal{M}_1}{M}, 1-\frac{\theta_\Psi}{\pi}\right) \leq \frac{k}{M} \\ \Leftrightarrow \quad & \min_{z\in\mathbb{Z}}\left|1 - \frac{\theta_\Psi}{\pi} - \frac{M-\mathcal{M}_1}{M} + z\right| \leq \frac{k}{M} \\ \Leftrightarrow \quad & \min_{z\in\mathbb{Z}}\left|\frac{\theta_\Psi}{\pi} - \frac{\mathcal{M}_1}{M} + z\right| \leq \frac{k}{M} \\ \Leftrightarrow \quad & \min_{z\in\mathbb{Z}}\left|\theta_\Psi - \left(\frac{\pi\mathcal{M}_1}{M} + \pi z\right)\right| \leq \frac{\pi k}{M}. \end{aligned}$$

From its definition in equation (2.33) and since $a_\Psi \in [0,1]$ we know that $\theta_\Psi \in [0, \frac{\pi}{2}]$. We call $\overline{\theta_\Psi} := \frac{\pi \mathcal{M}_1}{M} + \pi z^*$ where z^* is chosen to minimize the above expression. By the periodicity of the sine, the z does not matter once we look again at the squared sine

$$\sin^2\left(\frac{\pi \mathcal{M}_1}{M} + \pi z\right) = \sin^2\left(\frac{\pi \mathcal{M}_1}{M}\right).$$

Now lemma 2.25 gives for $\overline{a_\Psi} := \sin^2\left(\frac{\pi \mathcal{M}_1}{M} + \pi z^*\right) = \sin^2\left(\frac{\pi \mathcal{M}_1}{M}\right)$

$$|a_\Psi - \overline{a_\Psi}| \leq 2\pi k \frac{\sqrt{a_\Psi(1-a_\Psi)}}{M} + k^2 \frac{\pi^2}{M^2}.$$

Since the last step is no equivalence we only have

$$\begin{aligned}\left\{j \in J_M \,\middle|\, d\left(\frac{j}{M}, \frac{\theta_\Psi}{\pi}\right) \leq \frac{k}{M}\right\} &= \left\{j \in J_M \,\middle|\, d\left(\frac{M-j}{M}, 1 - \frac{\theta_\Psi}{\pi}\right) \leq \frac{k}{M}\right\} \\ &\subseteq \left\{j \in J_M \,\middle|\, |a_\Psi - \overline{a_\Psi}| \leq 2\pi k \frac{\sqrt{a_\Psi(1-a_\Psi)}}{M} + k^2 \frac{\pi^2}{M^2}\right\}.\end{aligned}$$

Thus, by lemma 2.23

$$\begin{aligned}\mathbb{P}\left[|a_\Psi - \overline{a_\Psi}| \leq 2\pi k \frac{\sqrt{a_\Psi(1-a_\Psi)}}{M} + k^2 \frac{\pi^2}{M^2}\right] &\geq \mathbb{P}\left[d\left(\frac{\mathcal{M}_1}{M}, \frac{\theta_\Psi}{\pi}\right) \leq \frac{k}{M}\right] \\ &\geq \begin{cases} \frac{8}{\pi^2} & k = 1 \\ 1 - \frac{1}{2k-2} & k > 1 \end{cases},\end{aligned}$$

we have shown the claim. □

In the literature there are other versions of AE available, that perform better on Noisy Intermediate-Scale Quantum (NISQ) devices. One makes use of a maximum likelihood approach [65] and another employs an iterative scheme [30].

2.3 Parameterized Quantum Circuits

As we have seen in section 1.2.2 some quantum gates have parameters. This fact leads to the idea of using these parameters to mimic machine learning approaches. Under the term Variational Quantum Algorithm (VQA) this concept has been extensively analyzed and used for different problems, Cerezo et al. [16] provide a comprehensive overview.

The basic idea of VQA is to construct a quantum circuit containing parameterized quantum gates, a Parameterized Quantum Circuit (PQC), and then tweak the parameters until the circuit behaves in a desired way. This tweaking is done in a hybrid loop between a quantum device and a classical computer. The quantum device evaluates the circuit and based on the outcome the classical computer modifies the parameters, with which the quantum device generates a new result and the loop repeats.

In this section we will formalize the concept of a PQC. But before we go into that, let us clarify what we intend to do with PQCs in this work, as this influences how we will describe them. We want to solve regression problems, i.e., we have training data, consisting of pairs of data and target values and want to learn the relation between those two.

Definition 2.26 Regression Problem

A regression problem consists of a data set

$$\chi = \begin{pmatrix} \mathbf{x}_0^T \\ \vdots \\ \mathbf{x}_{K-1}^T \end{pmatrix} \in \mathbb{R}^{K \times D},$$

where each row is called a data point

$$\mathbf{x}_j \in \mathbb{R}^D \quad \text{for} \quad 0 \leq j < K,$$

and the target set

$$\Upsilon = \begin{pmatrix} y_0 \\ \vdots \\ y_{K-1} \end{pmatrix} \in \mathbb{R}^K,$$

where each row is called a target point

$$y_j \in \mathbb{R} \quad \text{for} \quad 0 \leq j < K.$$

With $K \in \mathbb{N}_{\geq 0}$ being the number of data points and $D \in \mathbb{N}_{\geq 0}$ the dimension of the input data.

To define a regression problem we additionally need a class of functions

$$f \colon \mathbb{R}^D \times \mathbb{R}^P \to \mathbb{R},$$

that maps a data point and a parameter set $\boldsymbol{\theta} \in \mathbb{R}^P$ to a prediction, and a loss measure

$$\mathcal{L} \colon \mathbb{R}^K \times \mathbb{R}^K \to \mathbb{R}_{\geq 0},$$

that maps a prediction and the true values to a positive real number.

Then the regression problem is finding the optimal parameter set

$$\boldsymbol{\theta}^* := \underset{\boldsymbol{\theta} \in \mathbb{R}^P}{\operatorname{argmin}} \, \mathcal{L} \left(\begin{pmatrix} f(\mathbf{x}_0, \boldsymbol{\theta})^T \\ \vdots \\ f(\mathbf{x}_{K-1}, \boldsymbol{\theta})^T \end{pmatrix}, \Upsilon \right).$$

We will use PQCs as the function class, meaning we have to formulate them in a way that they take a parameter vector and a data point as input and produce a single real number as output.

A PQC consists of three types of gates, first the non-parameterized ones and then two kinds of parameterized gates. They differ in what the parameters represent, the first kind,

simply called parameterized gates, are the part of the PQC that will be tweaked to solve the regression problem. The second kind, we will call them encoding gates, bring the dependence on the data points into the PQC.

Every parameterized gate is associated either to one index of the parameter vector or to one index of the data points. Every parameter can only occur once, while data indices can occur arbitrarily often. The method of repeating data points is known as data re-uploading [54, 59]. Finally, the circuit needs to be measured, and the measurement needs to be translated into a numerical value of the appropriate dimension.

In the literature different suggestions can be found, how such a PQC could be structured. Often these structures consist of layers that fulfill the different roles, one layer for entangling the qubits, one layer for encoding the data and one layer for the parameters. These layers then will be applied repeatedly. It can be shown, that these PQCs represent truncated Fourier series whose order depends on the number of encoding gates [33, 59, 69].

Definition 2.27 Parameterized Quantum Circuit

A PQC is a circuit that consists of $L \in \mathbb{N}_{\geq 0}$ layers of data encoding blocks $\mathbf{S}_j(\mathbf{x})$ and parameter blocks $\mathbf{W}_j\left(\boldsymbol{\theta}^{(j)}\right)$. All of these blocks are unitaries that consist of fixed single- and two-qubit gates, and parameterized single-qubit gates. The PQC is given as

$$\mathbf{U}\left(\mathbf{x}, \boldsymbol{\theta}\right) = \prod_{j=0}^{L-1} \mathbf{S}_j(\mathbf{x}) \mathbf{W}_j\left(\boldsymbol{\theta}^{(j)}\right).$$

Here $\mathbf{x} \in \mathbb{R}^D$ is a data point and $\boldsymbol{\theta}^{(j)} \in \mathbb{R}^{P_j}$ is the parameter vector controlling the j-th parameter block. The parameter vectors are combined into $\boldsymbol{\theta} = \left(\boldsymbol{\theta}^{(0)} \quad \cdots \quad \boldsymbol{\theta}^{(L-1)}\right) \in \mathbb{R}^P$ where $P = \sum_{j=0}^{L-1} P_j$.

A PQC $\mathbf{U}$ together with an observable $\mathbf{M}$ can be used as the function class for regression problems

$$\begin{aligned} f_{\mathbf{U},\mathbf{M}} \colon \mathbb{R}^D \times \mathbb{R}^P &\to \mathbb{R} \\ (\mathbf{x}, \boldsymbol{\theta}) &\mapsto \left\langle e_0^n \right| \mathbf{U}\left(\mathbf{x}, \boldsymbol{\theta}\right)^\dagger \mathbf{M}\, \mathbf{U}\left(\mathbf{x}, \boldsymbol{\theta}\right) \left| e_0^n \right\rangle . \end{aligned} \tag{2.51}$$

Input Scaling

An interesting question is the scaling of the input. We only use rotational gates, which make use of trigonometric functions. This means all PQCs will be periodic in the input with a period of 2π, which could lead us to scale the input to $[0, 2\pi]$, or $[-\pi, \pi]$ if preferred. But then it is established in the quantum computing community, that the angle, that is put into a rotation, is divided by two, before applying the trigonometric functions, thus an interval of $[0, 4\pi]$ would be in order. Then one notices, that these rotations only influences the quantum state, which we cannot directly observe. What we can observe are amplitudes, which are the squared absolute values of the state. This again halves the period of the input encoding, and we are back at $[0, 2\pi]$. The final observation is, that trigonometric functions are periodic, while the functions we want to approximate are in general not. It has been shown, that specific structures of PQCs can approximate square

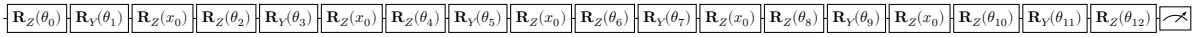
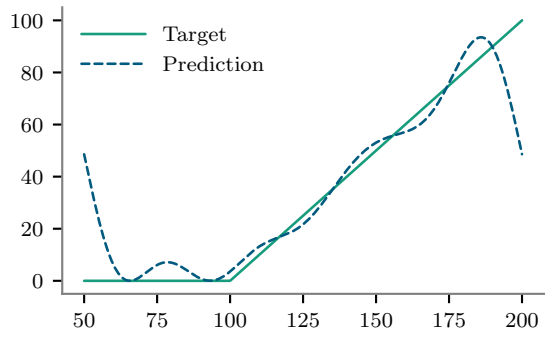
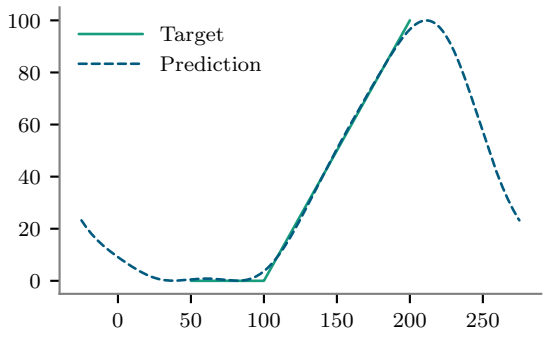


Figure 2.4: The circuit representing $U^{WZW}_{\theta,\phi,5}$ from Yu et al. [69].



(a) Input data scaled to $[0, 2\pi]$.



(b) Input data scaled to $[0, \pi]$. To visualize the periodicity the input range was extended back to $[0, 2\pi]$.

Figure 2.5: Approximation of the function from equation (2.52) by the circuit seen in figure 2.4.

integrable functions arbitrarily well in respect to the L^2 norm [69], but the periodicity makes point wise convergence impossible, if the target function is not periodic itself.

Thus, we limit the used range for the input data to $[0, \pi]$ and use the rest of the interval as a buffer to let the function be periodic.

While not extending on the theory presented by Yu et al. [69] we put the theory to a test in a simple example.

Example 2.28

We use a circuit that strictly follows the recipe given by Yu et al. [69] as $U^{WZW}_{\theta,\phi,L}$ with $L = 5$ layers. The circuit can be seen in figure 2.4. For the function that we try to approximate we choose

$$\begin{aligned} g\colon [50, 200] &\to \mathbb{R} \\ x &\mapsto \max\{x - 100, 0\}. \end{aligned} \tag{2.52}$$

To formulate this as a regression problem in the sense of definition 2.26 we have to choose a range of x-values, $[x_{\min}, x_{\max}]$, we are interested in and a number of discretization steps, K, to build up the data set

$$\chi = \begin{pmatrix} x_{\min} \\ \vdots \\ x_{\min} + j\frac{x_{\max} - x_{\min}}{K-1} \\ \vdots \\ x_{\max} \end{pmatrix} \in \mathbb{R}^K.$$

The target set is then given as

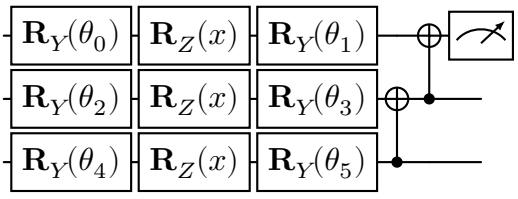
$$\Upsilon = \begin{pmatrix} g(x_{\min}) \\ \vdots \\ g(x_{\min} + j\frac{x_{\max}-x_{\min}}{K-1}) \\ \vdots \\ g(x_{\max}) \end{pmatrix} \in \mathbb{R}^K.$$

This function in equation (2.52) is obviously square integrable, thus the theory by Yu et al. [69] applies. The first step, and what we want to analyze in this example, is how to scale this function in order to achieve a good approximation.

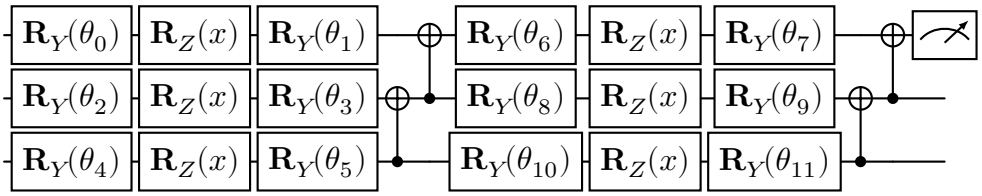
First we scale the input to $[0, 2\pi]$. The resulting approximation, after optimizing the free parameters, can be seen in figure 2.5a. We can see that the approximation is ok in the center of the functions range, but deteriorates closer to the borders. The reason is also directly visible, the approximation result is periodic, while the target function is not.

To remedy this problem we perform the same procedure, but now scale the input to $[0, \pi]$, i.e., only use half of the available range for the input. The resulting approximation is shown in figure 2.5b. To emphasize the effect of limiting the range, in this visualization the evaluated range for the function is extended back to the full possible range, and we can see again the periodicity, which now is not a problem as we are only using the center of the interval.

In a second and final example for this section we have a look at the influence of the number of parameters on the approximation quality of PQCs.



(a)



(b)

Figure 2.6: One (2.6a) and two (2.6b) layered PQCs used to learn functions.

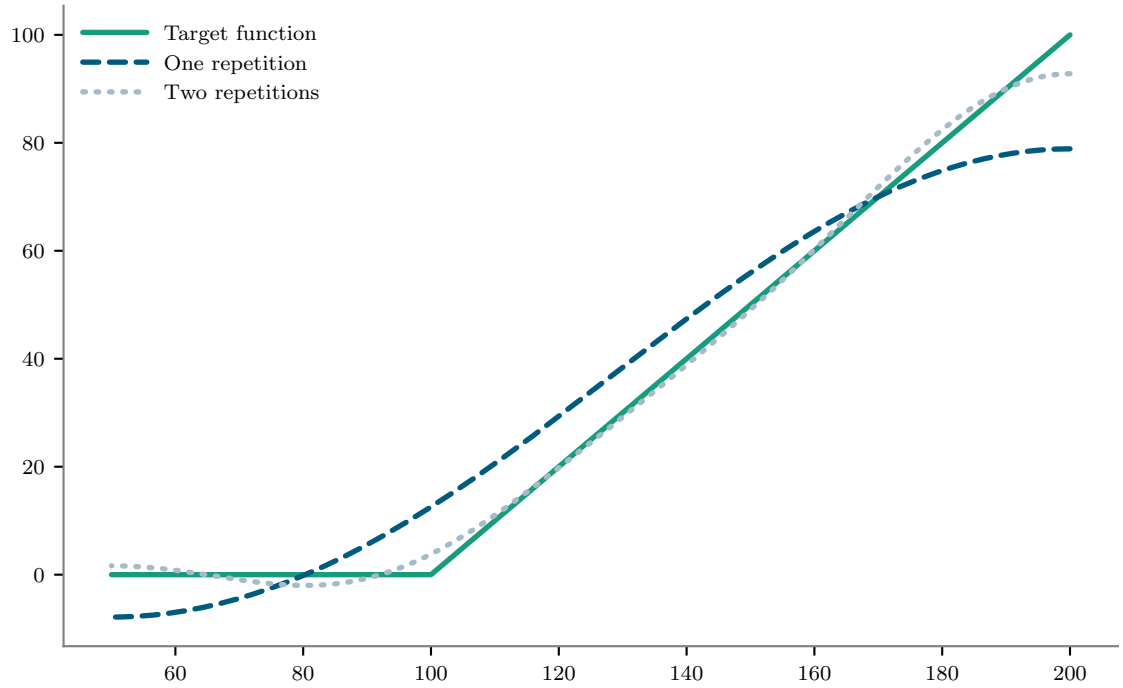


Figure 2.7: Results of training the PQCs from figure 2.6 on equation (2.52).

Example 2.29

We reuse the regression problem formulated in example 2.28. This time we will train two different circuits, that follow the same structure, first a simple one, seen in figure 2.6a and then one with two repetitions of the first one, seen in figure 2.6b. As output, we measure in both cases the final state with measurement operator $\mathbf{M} = \mathbf{I} \otimes \mathbf{I} \otimes |1\rangle\langle 1|$ and scale the result accordingly. This measurement operator means, we only measure the first qubit, and use the proportion of observed ones. The results are shown in figure 2.7, we can clearly see, that with more parameters, the approximation quality increases.

This result is rather obvious, more parameters usually allow a model to fit a dataset better. But just as in classical machine learning, increasing the number of parameters has drawbacks. The first and most obvious one is that more parameters lead to higher computational costs when training. For quantum computers in the NISQ era even more important is the depth of the circuit, the number of used qubits, and the number of two-qubit gates, which all increase with more parameters.

This mediation between complexity and learning quality will be addressed in more detail in chapter 3.

2.4 Conditional Parameterized Quantum Circuits

Up until this point one evaluation of a PQC relates to evaluating the model for one data point only. This means that if we want to evaluate a function we have learned by a PQC

on 400 grid points, we have to execute the PQC 400 times. This is similar to classical regression models and perfectly fine in many situations.

But we have a specific use case in mind, we want to price options. Recall definition 1.3, the option's price is the discounted expected value of the payoff function applied to the random variable representing the underlying asset.

We can calculate this expectation by discretizing the value of the underlying, evaluating the payoff function at these discrete values and average the results weighted by a discretized version of the distribution of the underlying. This is a special case of MC integration and the quantum version is referred to in the literature as Quantum Monte-Carlo Integration (QMCI) [36].

Formally the goal of MC integration is for a function $f\colon \mathbb{R} \to \mathbb{R}$ and a random variable X to calculate $\mathbb{E}[f(X)]$. In the option pricing setting f is the option's payoff and X the price of the underlying asset at the option's maturity. As in general $\mathbb{E}[f(X)]$ is not the same as $f(\mathbb{E}[X])$ this can be difficult to calculate, even if the distribution of X is well known.

The idea of MC integration is to repeatedly sample from the random variable, $x_j \sim X$ for $0 \leq j < N-1$, and then approximate the expectation by the average $1/N \sum_{j=0}^{N-1} f(x_j)$. By the strong law of large numbers the arithmetic mean converges to the desired expectation. Under additional requirements on the moments of f, the central limit theorem ensures, that the rate of convergence, expressed as length of confidence intervals of a given confidence level, is $\mathcal{O}(1/N)$.

For QMCI we want to apply AE, discussed in section 2.2.3, to calculate this expectation with a quantum computer and benefit from the quadratic speedup of AE. To be able to apply AE we need to encode the expectation into the amplitude of one qubit.

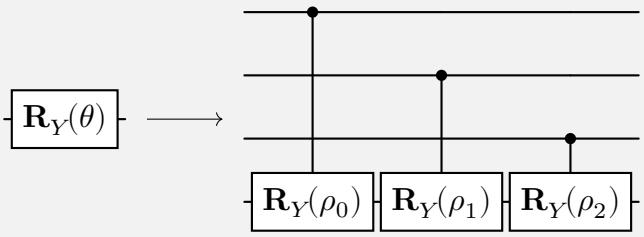
In the joint article with Wolf and Turkalj [68] we presented a novel approach how a pre-trained PQC can be used to do just that. The remainder of this section will restate the results of that publication.

A Conditional Parameterized Quantum Circuit (CPQC) is built by combining an ordinary PQC with a control register. The PQC takes the role of the function f while the control register represents the random variable X.

The basic idea, is to replace each data encoding gate in the PQC by fixed, controlled operations, one for every qubit in the control register. The data is discretized and instead of being applied as parameters to the gates they will be encoded into the control register. In a very simple example this can look the following:

Example 2.30

We start with a single data encoding gate and introduce three control qubits.



Now, let us check how the parameters ρ_j need to be chosen in this example. We assume that $\theta \in [0, \pi]$ and we have three qubits, i.e., eight states, for discretizing this interval. This leads to

$$\theta_j = j\frac{\pi}{7} \quad \text{for} \quad 0 \leq j \leq 7.$$

Next we identify each state of the control register with one of the discretization points, naturally we identify state $|j\rangle$ with θ_j. The following table summarizes which rotation will be applied for every discretized value of θ.

angle	discretized	state	binary	controlled angle
θ_0	0	$\|0\rangle$	000	0
θ_1	$\frac{1}{7}\pi$	$\|1\rangle$	001	ρ_0
θ_2	$\frac{2}{7}\pi$	$\|2\rangle$	010	ρ_1
θ_3	$\frac{3}{7}\pi$	$\|3\rangle$	011	$\rho_0 + \rho_1$
θ_4	$\frac{4}{7}\pi$	$\|4\rangle$	100	ρ_2
θ_5	$\frac{5}{7}\pi$	$\|5\rangle$	101	$\rho_0 + \rho_2$
θ_6	$\frac{6}{7}\pi$	$\|6\rangle$	110	$\rho_1 + \rho_2$
θ_7	π	$\|7\rangle$	111	$\rho_0 + \rho_1 + \rho_2$

Now we have to choose the parameters ρ_j such that for all discretization points, the sum of the angles performed by the controlled operations match the original rotation angle. Let for $0 \leq j \leq 7$ be the binary representation given by $j_2 j_1 j_0$, i.e.,

$$j = j_0 2^0 + j_1 2^1 + j_2 2^2.$$

Then we have to ensure that

$$\mathbf{R}_Y(\theta_j) \stackrel{!}{=} \mathbf{R}_Y(\rho_2)^{j_2} \mathbf{R}_Y(\rho_1)^{j_1} \mathbf{R}_Y(\rho_0)^{j_0}.$$

By proposition 1.11 this boils down to

$$\theta_j \stackrel{!}{=} \rho_2^{j_2} + \rho_1^{j_1} + \rho_0^{j_0}.$$

A trivial solution to that is given by

$$\rho_j = 2^j \frac{\pi}{7}.$$

We formalize the ideas and observations of the above example. First we restrict the data encoding blocks we allow for the underlying PQC.

Definition 2.31 Rotational Encoding Block

We call a data encoding block, that only consists of a single X, Z or Y rotation, a rotational encoding block and write it as

$$\mathbf{SR}_j(\mathbf{x}) := \mathbf{R}_j^{(l_j)}\left(x_{k_j}\right),$$

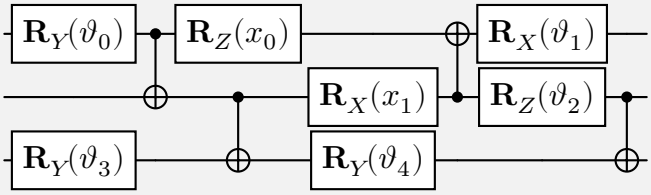
where $\mathbf{R}_j(\alpha) \in \{\mathbf{I}, \mathbf{R}_X(\alpha), \mathbf{R}_Y(\alpha), \mathbf{R}_Z(\alpha)\}$ is either the identity or a rotation and $k_j \in \{0, \dots, D-1\}$ and $l_j \in \{0, \dots, n-1\}$ assign each rotational encoding block one input feature and one qubit respectively.

Note that with the usage of rotational encoding blocks, the occurrence of entanglement is limited to the parameterized blocks.

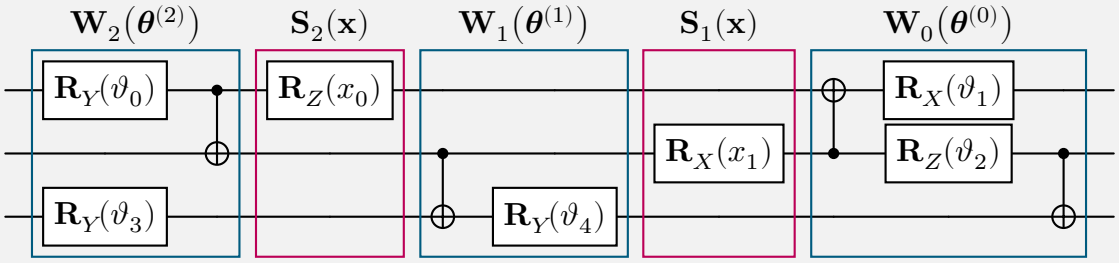
Before we go into the definition of a CPQC, let us have an example of how the notation introduced to describe PQCs exactly looks like.

Example 2.32

Let us examine how this circuit



would be expressed in the way introduced in definition 2.27. To better see the blocks, we stretch out some parallel gates.



Where $\boldsymbol{\theta}^{(0)} = (\vartheta_1 \quad \vartheta_2)$, $\boldsymbol{\theta}^{(1)} = (\vartheta_4)$ and $\boldsymbol{\theta}^{(2)} = (\vartheta_0 \quad \vartheta_3)$. We can also see in this example, that the representation in layers is not unique. For example, the block $\mathbf{W}_1(\boldsymbol{\theta}^{(1)})$ could be moved in front of $\mathbf{S}_1(\mathbf{x})$ and fused with $\mathbf{W}_0(\boldsymbol{\theta}^{(0)})$.

This way of expressing a PQC is in many cases not the most efficient way of expressing it, but it is a simple formula, that covers all forms of circuits we are interested in, and allows us to work more easily with them.

Now we have everything we need to introduce the concept of a CPQC.

Definition 2.33 Conditional Parameterized Quantum Circuit

Given an n qubit PQC

$$\mathbf{UR}(\mathbf{x}, \boldsymbol{\theta}) = \prod_{j=0}^{L-1} \mathbf{SR}_j(\mathbf{x}) \mathbf{W}_j\big(\boldsymbol{\theta}^{(j)}\big),$$

where the encoding blocks are all rotational ones, the corresponding CPQC is constructed by adding one extra control register for every data feature and replacing all encoding blocks by controlled operations.

For every dimension $j \in \{0, \dots, D-1\}$ of the input data we call the size of the corresponding register $d_j \in \mathbb{N}_{\geq 0}$ and the vector of all the discretization sizes we call

$$\mathcal{D} := (d_0 \quad \cdots \quad d_{D-1}). \tag{2.53}$$

The total number of discretization qubits are denoted by

$$|\mathcal{D}| := \sum_{j=0}^{D-1} d_j.$$

We identify the individual control registers by their qubit indices

$$I_j^C := \left(\textstyle\sum_{k=0}^{j-1} d_k \quad \sum_{k=0}^{j-1} d_k + 1 \quad \cdots \quad \sum_{k=0}^{j} d_k - 1\right)$$

and the l-th qubit of register j is written as

$$I_j^C(l) := \sum_{k=0}^{j-1} d_k + l.$$

We call all control registers together

$$I^C := \left(I_0^C \quad \cdots \quad I_{D-1}^C\right) = (0 \quad \cdots \quad |\mathcal{D}| - 1).$$

The sizes of the control registers determine the angles of the rotations each qubit of that register controls

$$\rho_k^{(j)} := \frac{2^k}{2^{d_j} - 1}\pi. \tag{2.54}$$

The original register will be the target of all control registers and will be denoted as

$$I^T := (|\mathcal{D}| \quad \cdots \quad |\mathcal{D}| + n - 1),$$

where n is the original number of qubits. Again we refer to the l-th qubit of that register as

$$I^T(l) := |\mathcal{D}| + l.$$

Next, we replace every input encoding by a static, controlled version. Given an encoding block $\mathbf{SR}_\mathrm{j}$ we write for the controlled version, using the notation for controlled operations introduced in equation (1.14),

$$\mathbf{CSR}_\mathrm{j} := \prod_{m=0}^{d_{k_j}-1} \Lambda\left(I_{k_j}^C(m), \mathbf{R}_j\left(\rho_m^{(k_j)}\right)^{(I^T(l_j))}\right). \tag{2.55}$$

With that we can write

$$\mathbf{C}_{\mathcal{D}}^{\mathbf{UR}}(\boldsymbol{\theta}) := \prod_{j=0}^{L-1} \mathbf{CSR}_{\mathrm{j}} \left(\mathbf{W}_j(\boldsymbol{\theta}^{(j)}) \otimes \mathbf{I}^{\otimes |\mathcal{D}|} \right).$$

Note the reverse order of the Kronecker products, which comes again from the convention on where to start counting qubits. The CPQC no longer depends on the input data, which now comes into play by the state of the control registers.

A CPQC can also be used as the function class for a regression problem, but the way the data gets into the circuit changes. We introduce some notation for discretizing the input. For a data point $\mathbf{x} \in \mathbb{R}^D$ and a discretization $\mathcal{D} = \begin{pmatrix} d_0 & \cdots & d_{D-1} \end{pmatrix}$ we denote the closest point on our discretization by indices for each input dimension

$$J_k(\mathbf{x}) := \underset{j \in \{0,\dots,2^{d_k}-1\}}{\operatorname{argmin}} \left| x_k - j \frac{\pi}{2^{d_k}-1} \right|.$$

Together these indices are denoted by $J(\mathbf{x}) = \begin{pmatrix} J_0(\mathbf{x}) & \cdots & J_{D-1}(\mathbf{x}) \end{pmatrix}$ and the space they live in we will call

$$\mathbb{N}_{\mathcal{D}} := \bigtimes_{k=0}^{D-1} \{0, \dots, 2^{d_k} - 1\}.$$

For any multi index $J = \begin{pmatrix} J_0 & \cdots & J_{D-1} \end{pmatrix} \in \mathbb{N}_{\mathcal{D}}$ we write

$$\mathbf{x}^{(J)} := \begin{pmatrix} x_0^{(J_0)} \\ \vdots \\ x_{D-1}^{(J_{D-1})} \end{pmatrix},$$

where

$$x_k^{(J_k)} := J_k \frac{\pi}{2^{d_k}-1}. \tag{2.56}$$

With that we can write for the state encoding a point on the discretized grid

$$\left| e_J^{\mathcal{D}} \right\rangle := \bigotimes_{k=D-1}^{0} \left| e_{J_k}^{d_k} \right\rangle.$$

The reversed order comes again from the convention in which way to order the qubits. For an observable that only cares for the qubits that are not used for data encoding we write

$$\mathbf{M}_{\mathcal{D}} := \mathbf{M} \otimes \mathbf{I}^{\otimes |\mathcal{D}|}. \tag{2.57}$$

If we describe the input data as a discrete random variable $\mathbf{X}$ we call the distribution of that random variable $\mathcal{P}$ where for every multi index $J \in \mathbb{N}_{\mathcal{D}}$ we have

$$\mathcal{P}(J) = \mathbb{P}\left[\mathbf{X} = \mathbf{x}^{(J)}\right].$$

Obviously we have $\sum_{J \in \mathbb{N}_{\mathcal{D}}} \mathcal{P}(J) = 1$. Given such a distribution we write

$$\begin{aligned} f_{\mathbf{C}_{\mathcal{D}}^{\mathbf{UR}}, \mathbf{M}, \mathcal{P}} \colon \mathbb{R}^P &\to \mathbb{R} \\ \boldsymbol{\theta} &\mapsto (\langle e_0^n| \otimes \langle \mathcal{P}|) \mathbf{C}_{\mathcal{D}}^{\mathbf{UR}}(\boldsymbol{\theta})^\dagger \, \mathbf{M}_{\mathcal{D}} \, \mathbf{C}_{\mathcal{D}}^{\mathbf{UR}}(\boldsymbol{\theta}) (|e_0^n\rangle \otimes |\mathcal{P}\rangle), \end{aligned} \tag{2.58}$$

where

$$|\mathcal{P}\rangle := \sum_{J\in\mathbb{N}_{\mathcal{D}}} \sqrt{\mathcal{P}(J)}\,\left|e_J^{\mathcal{D}}\right\rangle \tag{2.59}$$

is a superposition, where the probability to observe a state corresponding to a discretization point follows the distribution of the random variable $\mathbf{X}$.

If we want to use a CPQC for a regression problem we can use

$$\begin{aligned} f_{\mathbf{C}_{\mathcal{D}}^{\mathbf{UR}},\mathbf{M}} &\colon \mathbb{R}^D \times \mathbb{R}^P \to \mathbb{R} \\ &(\mathbf{x},\boldsymbol{\theta}) \mapsto \left(\langle e_0^n| \otimes \left\langle e_{J(\mathbf{x})}^{\mathcal{D}}\right|\right)\mathbf{C}_{\mathcal{D}}^{\mathbf{UR}}(\boldsymbol{\theta})^\dagger\, \mathbf{M}_{\mathcal{D}}\, \mathbf{C}_{\mathcal{D}}^{\mathbf{UR}}(\boldsymbol{\theta})\left(|e_0^n\rangle \otimes \left|e_{J(\mathbf{x})}^{\mathcal{D}}\right\rangle\right) \end{aligned} \tag{2.60}$$

as a function class, note that this does not use superposition in the input registers but only basis states.

Now that we have introduced quite a lot of notation, we will show that CPQCs are actually useful for QMCI.

First we show, that a CPQC applied to a basis state does the same as the corresponding PQC independently of the state of the target register.

Proposition 2.34

Given a rotational PQC $\mathbf{UR}$, a discretization $\mathcal{D} = (d_0 \quad \cdots \quad d_{D-1})$ and a parameter set $\boldsymbol{\theta} \in \mathbb{R}^P$ we have for any $J \in \mathbb{N}_{\mathcal{D}}$ and state $|\psi\rangle \in \mathbb{C}^{2^n}$

$$\mathbf{C}_{\mathcal{D}}^{\mathbf{UR}}(\boldsymbol{\theta})\left(|\psi\rangle \otimes \left|e_J^{\mathcal{D}}\right\rangle\right) = \mathbf{UR}\left(\mathbf{x}^{(J)},\boldsymbol{\theta}\right)|\psi\rangle \otimes \left|e_J^{\mathcal{D}}\right\rangle .$$

Proof. We will show that given a state $|\phi\rangle \in \mathbb{C}^{2^n}$ a rotation by $x_k^{(J_k)}$ has the same effect as the controlled version where the control register is in the state $\left|e_{J_k}^{d_k}\right\rangle$. First we need the binary representation of J_k, therefore let $J_{k,l} \in \{0,1\}$ for $l \in \{0,\dots,d_k-1\}$ such that

$$J_k = \sum_{l=0}^{d_k-1} J_{k,l} 2^l. \tag{2.61}$$

We look at a generic rotation $\mathbf{R}$ of an angle $x_k^{(J_k)}$ applied to only the target register of the state $|\phi\rangle \otimes \left|e_{J_k}^{d_k}\right\rangle$. The resulting state is

$$\begin{aligned} \mathbf{R}\left(x_k^{(J_k)}\right)|\phi\rangle \otimes \left|e_{J_k}^{d_k}\right\rangle &\overset{(2.56)}{=} \mathbf{R}\left(\frac{\pi}{2^{d_k}} J_k\right)|\phi\rangle \otimes \left|e_{J_k}^{d_k}\right\rangle \\ &\overset{(2.61)}{=} \mathbf{R}\left(\frac{\pi}{2^{d_k}} \sum_{l=0}^{d_k-1} J_{k,l} 2^l\right)|\phi\rangle \otimes \left|e_{J_k}^{d_k}\right\rangle \\ &\overset{(2.54)}{=} \mathbf{R}\left(\sum_{l=0}^{d_k-1} J_{k,l}\rho_l^{(k)}\right)|\phi\rangle \otimes \left|e_{J_k}^{d_k}\right\rangle \\ &\overset{\text{prop. 1.11}}{=} \left(\left(\prod_{l=0}^{d_k-1} \mathbf{R}\left(\rho_l^{(k)}\right)^{J_{k,l}}\right)|\phi\rangle\right) \otimes \left|e_{J_k}^{d_k}\right\rangle \\ &= \left(\prod_{l=0}^{d_k-1} \Lambda\left(l, \mathbf{R}\left(\rho_l^{(k)}\right)^{(I^T(l))}\right)\right)\left(|\phi\rangle \otimes \left|e_{J_k}^{d_k}\right\rangle\right). \end{aligned}$$

The last equality holds since the state $\left|e_{J_k}^{d_k}\right\rangle$ is defined as the state that hat every qubit in state $|1\rangle$ that corresponds to a one in the binary representation of J_k, thus, exactly these rotations are applied.

We see that the rotation $\mathbf{R}\left(x_k^{(J_k)}\right)$ applied to the target register is equivalent to this controlled rotation

$$\left(\prod_{l=0}^{d_k-1}\Lambda\left(l,\mathbf{R}\left(\rho_l^{(k)}\right)^{(I^T(l))}\right)\right)$$

applied to the full state. This controlled rotation is exactly how a controlled encoding block looks like, if we ignore the other control registers, which are not involved.

That proves, that a single parameterized rotation can be replaced by controlled rotations independently on the state the target register is currently in and without changing the control register. This, in turn, means that by replacing all input encoding rotations in a PQC, while leaving everything else as it is, the controlled circuit yields the same state in the target register and the claim is proven. □

In the next step, we show that the equivalence of a PQC and the corresponding CPQC also holds if we include the final measurement.

Theorem 2.35

Given a rotational PQC $\mathbf{UR}$, a discretization $\mathcal{D} = (d_0 \quad \cdots \quad d_{D-1})$, an observable $\mathbf{M}$ and a parameter set $\boldsymbol{\theta} \in \mathbb{R}^P$ we have for any $J \in \mathbb{N}_{\mathcal{D}}$

$$f_{\mathbf{UR},\mathbf{M}}\left(\mathbf{x}^{(J)},\boldsymbol{\theta}\right) = f_{\mathbf{C}_{\mathcal{D}}^{\mathbf{UR}},\mathbf{M}}\left(\mathbf{x}^{(J)},\boldsymbol{\theta}\right).$$

This guarantees, that, if we construct a CPQC according to definition 2.33, the original circuit and the conditional one deliver the same results on the grid points defined by the used discretization.

Proof. We know from proposition 2.34, that both the PQC and the CPQC yield the same state in the target register. Together with the construction of the measurement operator for the CPQC this gives

$$\begin{aligned}
&f_{\mathbf{C}_{\mathcal{D}}^{\mathbf{UR}},\mathbf{M}}\left(\mathbf{x}^{(J)},\boldsymbol{\theta}\right)\\
&\overset{(2.60)}{=} \left(\langle e_0^n|\otimes\langle e_J^{\mathcal{D}}|\right)\mathbf{C}_{\mathcal{D}}^{\mathbf{UR}}(\boldsymbol{\theta})^\dagger\,\mathbf{M}_{\mathcal{D}}\,\mathbf{C}_{\mathcal{D}}^{\mathbf{UR}}(\boldsymbol{\theta})\left(|e_0^n\rangle\otimes|e_J^{\mathcal{D}}\rangle\right)\\
&\overset{\text{prop. 2.34}}{=} \left(\langle e_0^n|\,\mathbf{UR}\left(\mathbf{x}^{(J)},\boldsymbol{\theta}\right)^\dagger\otimes\langle e_J^{\mathcal{D}}|\right)\mathbf{M}_{\mathcal{D}}\left(\mathbf{UR}\left(\mathbf{x}^{(J)},\boldsymbol{\theta}\right)|e_0^n\rangle\otimes|e_J^{\mathcal{D}}\rangle\right)\\
&\overset{(2.57)}{=} \left(\langle e_0^n|\,\mathbf{UR}\left(\mathbf{x}^{(J)},\boldsymbol{\theta}\right)^\dagger\otimes\langle e_J^{\mathcal{D}}|\right)\left(\mathbf{M}\otimes\mathbf{I}^{\otimes|\mathcal{D}|}\right)\left(\mathbf{UR}\left(\mathbf{x}^{(J)},\boldsymbol{\theta}\right)|e_0^n\rangle\otimes|e_J^{\mathcal{D}}\rangle\right)\\
&= \langle e_0^n|\,\mathbf{UR}\left(\mathbf{x}^{(J)},\boldsymbol{\theta}\right)^\dagger\mathbf{M}\,\mathbf{UR}\left(\mathbf{x}^{(J)},\boldsymbol{\theta}\right)|e_0^n\rangle\otimes\langle e_J^{\mathcal{D}}|e_J^{\mathcal{D}}\rangle\\
&= \langle e_0^n|\,\mathbf{UR}\left(\mathbf{x}^{(J)},\boldsymbol{\theta}\right)^\dagger\mathbf{M}\,\mathbf{UR}\left(\mathbf{x}^{(J)},\boldsymbol{\theta}\right)|e_0^n\rangle\\
&\overset{(2.51)}{=} f_{\mathbf{UR},\mathbf{M}}\left(\mathbf{x}^{(J)},\boldsymbol{\theta}\right).
\end{aligned}$$

□

Now that we have shown, that we can construct a conditional circuit, that replicates the original parameterized circuit on the specified grid points, we come to the main result of this section.

Theorem 2.36

Given a rotational PQC $\mathbf{UR}$, a discretization $\mathcal{D} = (d_0 \quad \cdots \quad d_{D-1})$, an observable $\mathbf{M}$ and a parameter set $\boldsymbol{\theta} \in \mathbb{R}^P$, we have for any discrete distribution $\mathcal{P} \in \bigtimes_{j=0}^{D-1} \mathbb{R}_{\geq 0}^{2^{d_j}}$

$$\sum_{J \in \mathbb{N}_{\mathcal{D}}} \mathcal{P}(J) f_{\mathbf{UR},\mathbf{M}}(\mathbf{x}^{(J)}, \boldsymbol{\theta}) = \sum_{J \in \mathbb{N}_{\mathcal{D}}} \mathcal{P}(J) f_{\mathbf{C}_{\mathcal{D}}^{\mathbf{UR}},\mathbf{M}}(\mathbf{x}^{(J)}, \boldsymbol{\theta}) = f_{\mathbf{C}_{\mathcal{D}}^{\mathbf{UR}},\mathbf{M},\mathcal{P}}(\boldsymbol{\theta}).$$

This theorem states that if we can encode a distribution for the input data into the control registers, we can, with only one circuit, get the expected value of the functional represented by the PQC, applied to a random variable, following that distribution. In section 4.1 we will see how this can be used to calculate option prices.

The proof is based on the proof of proposition 3 by Wolf, Ewen, and Turkalj [68].

Proof. We start with

$$\begin{aligned}
& f_{\mathbf{C}_{\mathcal{D}}^{\mathbf{UR}},\mathbf{M},\mathcal{P}}(\boldsymbol{\theta}) \\
\overset{(2.58)}{=} & (\langle e_0^n| \otimes \langle \mathcal{P}|) \mathbf{C}_{\mathcal{D}}^{\mathbf{UR}}(\boldsymbol{\theta})^\dagger \, \mathbf{M}_{\mathcal{D}} \, \mathbf{C}_{\mathcal{D}}^{\mathbf{UR}}(\boldsymbol{\theta}) (|e_0^n\rangle \otimes |\mathcal{P}\rangle) \\
\overset{(2.59)}{=} & \left(\sum_{J \in \mathbb{N}_{\mathcal{D}}} \langle e_0^n| \otimes \sqrt{\mathcal{P}(J)} \langle e_J^{\mathcal{D}}| \right) \mathbf{C}_{\mathcal{D}}^{\mathbf{UR}}(\boldsymbol{\theta})^\dagger \, \mathbf{M}_{\mathcal{D}} \, \mathbf{C}_{\mathcal{D}}^{\mathbf{UR}}(\boldsymbol{\theta}) \left(\sum_{J \in \mathbb{N}_{\mathcal{D}}} |e_0^n\rangle \otimes \sqrt{\mathcal{P}(J)} |e_J^{\mathcal{D}}\rangle \right) \\
= & \sum_{J \in \mathbb{N}_{\mathcal{D}}} \sum_{J' \in \mathbb{N}_{\mathcal{D}}} \sqrt{\mathcal{P}(J)} \sqrt{\mathcal{P}(J')} (\langle e_0^n| \otimes \langle e_J^{\mathcal{D}}|) \mathbf{C}_{\mathcal{D}}^{\mathbf{UR}}(\boldsymbol{\theta})^\dagger \, \mathbf{M}_{\mathcal{D}} \, \mathbf{C}_{\mathcal{D}}^{\mathbf{UR}}(\boldsymbol{\theta}) (|e_0^n\rangle \otimes |e_{J'}^{\mathcal{D}}\rangle).
\end{aligned}$$

We apply proposition 2.34 to every summand to get

$$\begin{aligned}
& (\langle e_0^n| \otimes \langle e_J^{\mathcal{D}}|) \mathbf{C}_{\mathcal{D}}^{\mathbf{UR}}(\boldsymbol{\theta})^\dagger \, \mathbf{M}_{\mathcal{D}} \, \mathbf{C}_{\mathcal{D}}^{\mathbf{UR}}(\boldsymbol{\theta}) (|e_0^n\rangle \otimes |e_{J'}^{\mathcal{D}}\rangle) \\
= & \left(\langle e_0^n| \otimes \langle e_J^{\mathcal{D}}|\right) \left(\mathbf{UR}(\mathbf{x}^{(J)}, \boldsymbol{\theta}) \otimes \mathbf{I}^{\otimes |\mathcal{D}|}\right)^\dagger \mathbf{M}_{\mathcal{D}} \left(\mathbf{UR}(\mathbf{x}^{(J')}, \boldsymbol{\theta}) \otimes \mathbf{I}^{\otimes |\mathcal{D}|}\right) \left(|e_0^n\rangle \otimes |e_{J'}^{\mathcal{D}}\rangle\right) \\
\overset{(2.57)}{=} & \left(\langle e_0^n| \, \mathbf{UR}(\mathbf{x}^{(J)}, \boldsymbol{\theta})^\dagger \, \mathbf{M} \, \mathbf{UR}(\mathbf{x}^{(J)}, \boldsymbol{\theta}) \, |e_0^n\rangle \right) \otimes \langle e_J^{\mathcal{D}} | e_{J'}^{\mathcal{D}} \rangle .
\end{aligned}$$

Note that the first part is zero unless $J = J'$, where it is one, thus

$$\begin{aligned}
f_{\mathbf{C}_{\mathcal{D}}^{\mathbf{UR}},\mathbf{M},\mathcal{P}}(\boldsymbol{\theta}) = & \sum_{J \in \mathbb{N}_{\mathcal{D}}} \mathcal{P}(J) \, \langle e_0^n| \, \mathbf{UR}(\mathbf{x}^{(J)}, \boldsymbol{\theta})^\dagger \, \mathbf{M} \, \mathbf{UR}(\mathbf{x}^{(J)}, \boldsymbol{\theta}) \, |e_0^n\rangle \\
\overset{(2.51)}{=} & \sum_{J \in \mathbb{N}_{\mathcal{D}}} \mathcal{P}(J) f_{\mathbf{UR},\mathbf{M}}(\mathbf{x}^{(J)}, \boldsymbol{\theta}) \\
= & \sum_{J \in \mathbb{N}_{\mathcal{D}}} \mathcal{P}(J) f_{\mathbf{C}_{\mathcal{D}}^{\mathbf{UR}},\mathbf{M}}(\mathbf{x}^{(J)}, \boldsymbol{\theta})
\end{aligned}$$

and the claim is shown. □

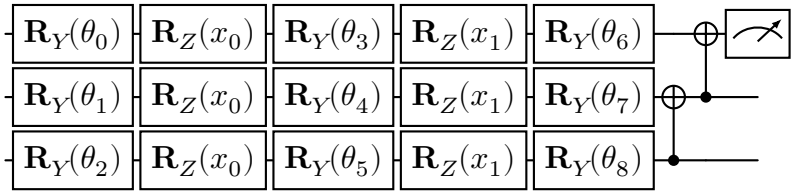


Figure 2.8: A simple circuit with three qubits and one repetition of the parameterized block and the entanglement. This circuit takes two-dimensional input data.

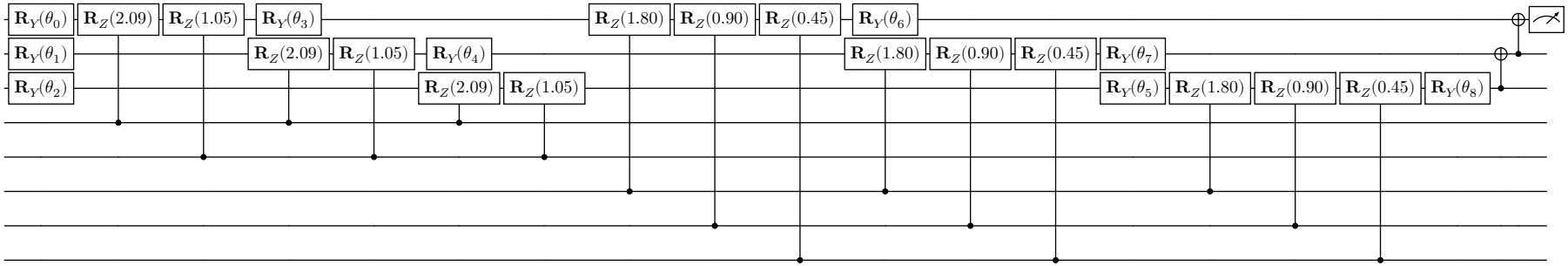


Figure 2.9: Controlled version of the circuit seen in figure 2.8. The first input is discretized using two qubits and the second using three qubits. For implementation reasons, the target register is moved to the top of the circuit.

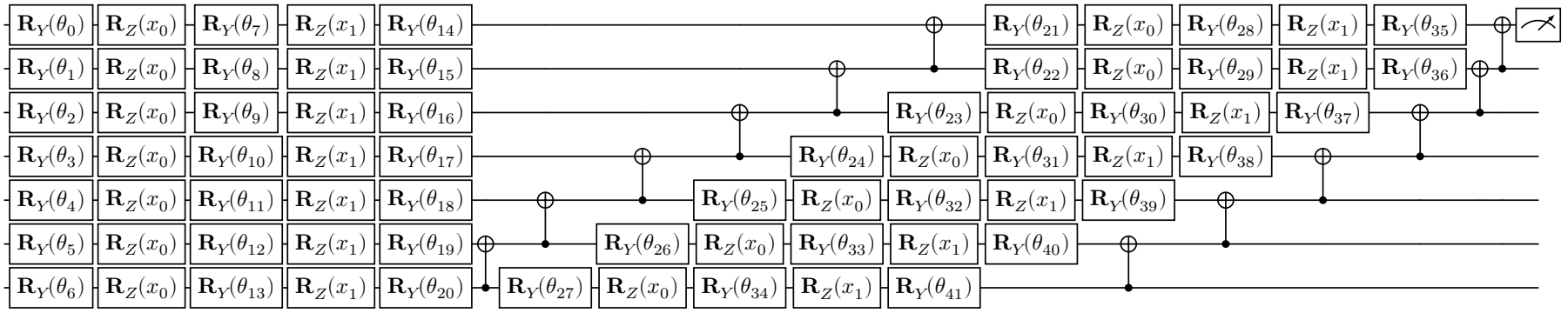


Figure 2.10: A simple circuit with three qubits and two repetitions of the parameterized block and the entanglement. This circuit takes two-dimensional input data.

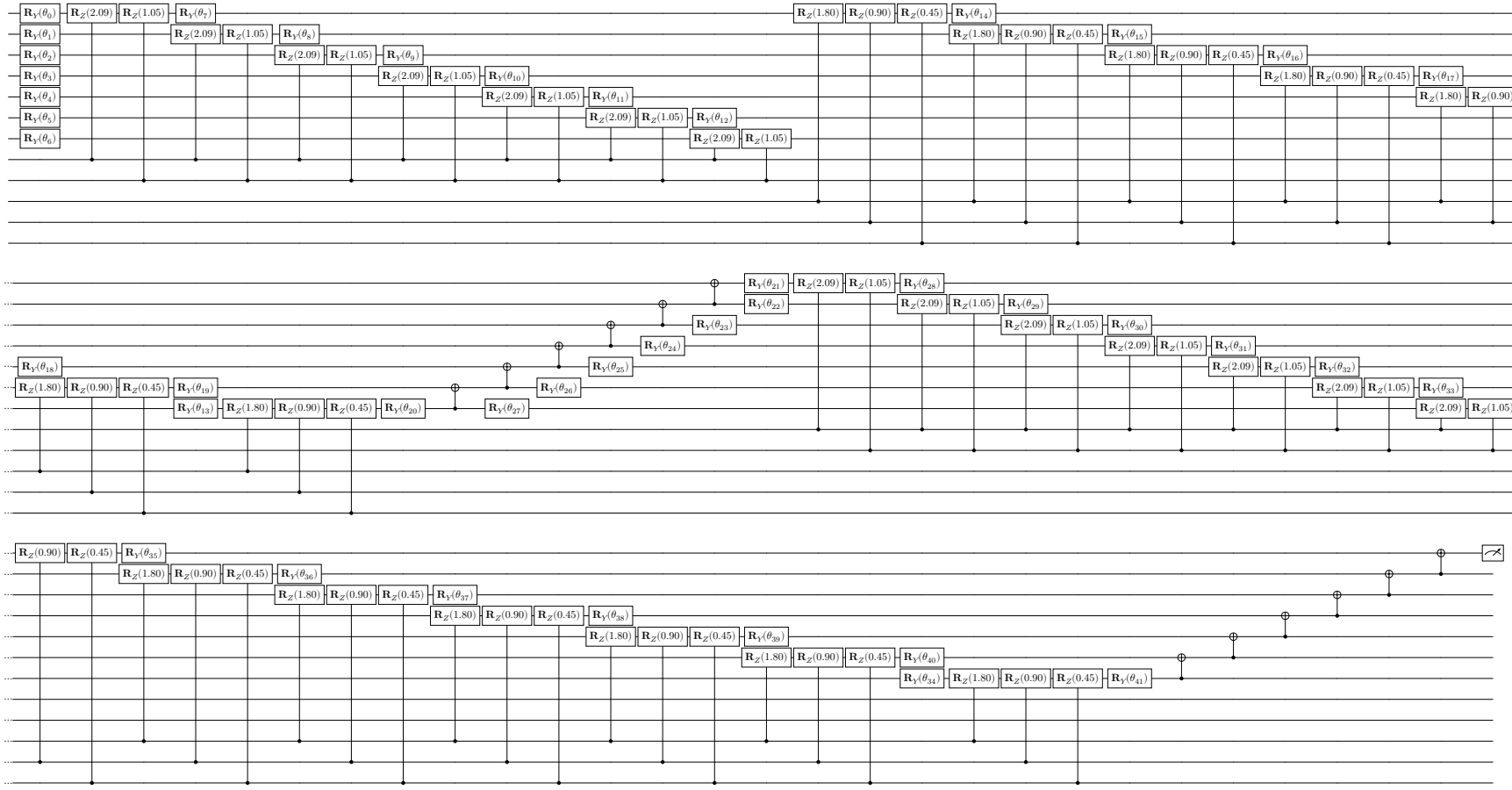


Figure 2.11: Controlled version of the circuit seen in figure 2.10. The first input is discretized using two qubits and the second using three qubits. For implementation reasons, the target register is moved to the top of the circuit.

We conclude this chapter with another example.

Example 2.37

In this example we will construct a circuit to learn the function

$$g(x, y) = \min\{0, \min\{x, y\} - 100\}, \tag{2.62}$$

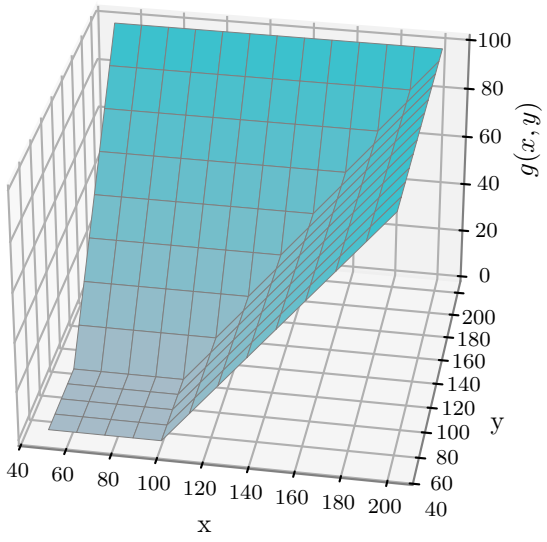
the payoff of a rainbow option.

We will use two different circuits, one simple and one more complex, to approximate this function and afterwards transform these PQCs into CPQCs. Both circuits follow the same structure, on every qubit a parameterized y-rotation is performed, followed by a z-rotation around the first input, a parameterized y-rotation, a z-rotation around the second input and a final parameterized y-rotation. This is then followed by CNOT gates to entangle all qubits. The first version uses three qubits and has the above structure only once, while the second one operates on seven qubits and has a second repetition of the structure. The circuits are visualized in figures 2.8 and 2.10.

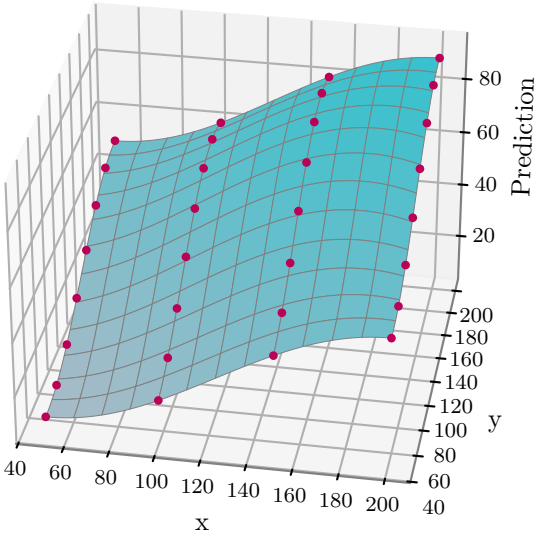
In the next step, we transform the circuits into their controlled versions. The resulting circuits are shown in figures 2.9 and 2.11. We have used two discretization qubits for the first input variable, x, and three for the second input, y. This leads to a discretization grid of $2^2 \cdot 2^3 = 4 \cdot 8$ points.

In figure 2.12 the results of using these circuits to learn the function are shown. The surfaces are the results of the original circuits, while the points are the results of the controlled circuits. Each point is generated by loading a basis state in both input registers.

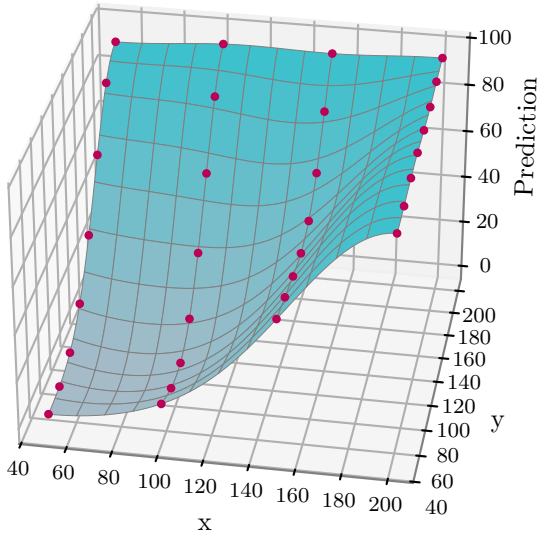
From this example, we can clearly see, that the more complex circuit performs better in approximating the function. In chapter 3 we will have a closer look at the process of finding a good balance between complexity and performance.



(a) Target Function from equation (2.62).



(b) Prediction of the simple circuit seen in figure 2.8.



(c) Prediction of the complex circuit seen in figure 2.10.

Figure 2.12: Comparison of the approximation quality of a simple (2.12b), and a complex (2.12c) PQC. The surface is the prediction of the PQC and the points are the predictions of the corresponding CPQC. As expected, the points are exactly on the surface. Both circuits are trained to approximate the target function seen in figure 2.12a.

Chapter 3

Contents

3 Quantum Architecture Search

In the NISQ era many approaches rely on variational methods, i.e., the usage of PQCs in hybrid loops, where a classical computer performs an iterative parameter optimization employing the quantum computer for evaluating the PQC for given parameter sets. These approaches have, for example, been used for optimization [9, 28], classification [54, 58] as well as the closely related function learning and regression [52, 59, 69]. The performance of all these approaches heavily depends on the structure that is chosen for the PQC.

Especially for function learning it is crucial, that the PQC is able to represent large classes of functions. It has been shown, that PQCs can be expressed as truncated Fourier series, where the number of available summands depends on the number of data encodings [59, 69]. This essentially states that, with arbitrarily complex circuits, we can approximate all functions that can be approximated by Fourier series. While this is a very strong result, arbitrary complex circuits bring their own problems. NISQ era devices, the only one currently available, have very restrictive limits on how complex circuits they can execute at all or with an acceptable noise level. The circuit width is limited by the number of qubits, the circuit depth by the coherence time of the devices and the number of gates is limited by the error rates of the individual gates. Using simulators can solve the noise problem, but they are also limited in the size and complexity of circuits they can execute, and all performance gains are lost on simulators.

Even if one would be able to perfectly execute suitably complex circuits, there is still another problem, in the literature known as barren plateaus [49]. Very much simplified it says, that while optimizing the parameters the gradients vanish exponentially fast for increasing circuit sizes [32]. A similar problem arises in classical machine learning when training deep neural networks [5, 35]. Searching for circuit structures in this setting of contradictory requirements is known in the literature as Quantum Architecture Search (QAS) [24, 46, 70]. The name is inspired by architecture search in classical machine learning.

In the joint article with Wolf and Turkalj we suggested a Genetic Programming (GP) approach on how to solve the multi criteria optimization problem, that is finding suitable PQCs [68]. Together with Turkalj, Holzer and Wolf we extended these ideas to utilize the ZX-calculus [27]. This chapter collects the ideas and results from these two articles regarding QAS.

3.1 Genetic Quantum Architecture Search

We will start this section with a very brief summary of GP, followed by our adaptation to the problem of QAS. In this part the algorithm will be explained without going into implementation details, this will follow in sections 3.2 and 3.3.

The term GP stands for an optimization technique, that iteratively transforms a population of possible solutions to solve a specific problem [44]. The iterative transformations try to emulate biological evolution, i.e., the fittest models propagate and mix together.

At the beginning of every GP algorithm stands an initial population that needs to be generated. From there every iteration, or in the context of GP, generation consists of three steps, selection, crossover and mutation. In the selection step members of the population are selected by some fitness criteria, often this step is not done deterministically, and the fitness rather influences the probability a certain individual is selected. From these selected individuals the next generation is formed by the other two steps. Crossover combines two individuals into a new one and mutation slightly changes a single individual.

Fitness Criteria

The first question that needs answering, and can be answered without going into implementation details, is what the criteria should be with which we evaluate the quantum algorithms. We want to solve regression problems with these algorithms, so an obvious choice is the regression error. In this work we always use the Mean Squared Error (MSE) for this part. Using only this criterion as fitness measure will encourage very complex quantum circuits, thus, we need some measure for the complexity of the circuit to counteract this tendency. Possible choices are the depth of the circuit, or the number of two-qubit gates, as these introduce more noise than other gates. Another idea is using the number of input encodings as a fitness measure, this becomes interesting in combination with CPQCs, as every input encoding introduces a number of two-qubit gates that scales linearly with the number of discretization qubits.

Selection

The pseudocode version of this step can be found at the end of this section as algorithm 3.1, all line references in this section refer to this algorithm.

Over the iterations we keep track of all models that are non-dominated. This means no other model is better in one fitness criterion without being worse in another. The set of non-dominated models is called the Pareto front.

We specify a number of individuals that we train in every iteration, as we have no control over the size of the Pareto front we cannot expect its size to match the number we specified. So we have to do something if there are either more individuals on the Pareto front than we need, or less.

In the former case we firstly select the individuals that are optimal in respect to each individual fitness criterion and then choose the remaining individuals by sampling uniformly from the remaining candidates. Here we do not allow duplicates, as there are more than enough candidates to choose from. This is done in algorithm 3.1.

In the latter case, we select all candidates and then fill the missing number of models by uniformly sampling from the candidates. Here we allow for duplicates, as we want to train

the same number of models in every iteration to maximally benefit from parallelization. This happens in algorithm 3.1.

After the mutation step, a second selection step occurs and all dominated models are discarded. Here not only the mutated models are considered but the previous Pareto front as well. This selection happens in algorithm 3.1. Additionally, we apply the optional hard limits, here only individuals get selected, that fulfill all hard limits. Unfortunately, it cannot be guaranteed, that after this filtering, any models are left. To solve this, we iteratively double all limits, until we find at least one individual, that fulfills the requirement. The implementation can be found in algorithm 3.1.

This is a very crude solution to the multiobjective optimization problem we are facing in QAS. Further work on this topic should look into existing solutions from multiobjective optimization and investigate the applicability to QAS.

Crossover and Mutation

Crossover is not part of our approach, as we could not identify a beneficial way of combining two quantum circuits, into a better one. Finding a good strategy for that can be the topic of further research as well. The mutation step depends on the implementation details and will be discussed in the subsequent sections.

Algorithm 3.1 Genetic QAS

Input:
Number of generations G
Number of mutations per generation M
Initial Population P_0
Number of trained models per generation N
Fitness Criteria F
Hard Limits for fitness criteria L

$P \leftarrow \texttt{non\_dominated}(P_0)$
repeat G **times**
  if $N \leq |P|$ **then**
    $P^\star \leftarrow P \cup \texttt{sample}(N - |P^\star|, P)$
  else
    $P^\star \leftarrow \{\}$
    for $f \in F$ **do**
      $P^\star \leftarrow P^\star \cup \left\{\text{argmin}_{p \in P}\{f(p)\}\right\}$
    end for
    $P^\star \leftarrow P^\star \cup \texttt{choices}(N - |P^\star|, P)$
  end if
  $P^m \leftarrow \{\}$
  for $p \in P^\star$ **do**
    $p^m \leftarrow p$
    repeat M **times**

```
            p^m ← mutate(p^m)
        end repeat
        P^m ← P^m ∪ {p^m}
    end for
    P' ← non_dominated(P* ∪ P)
    P ← {}
    α ← 1
    repeat
        for p ∈ P' do
            if all(f(p) ≤ α · L(f) for f ∈ F) then
                P ← P ∪ {p}
            end if
            α ← α · 2
        end for
    until |P| > 0
end repeat
return P
```

3.2 Quantum Architecture Search based on Quantum Circuits

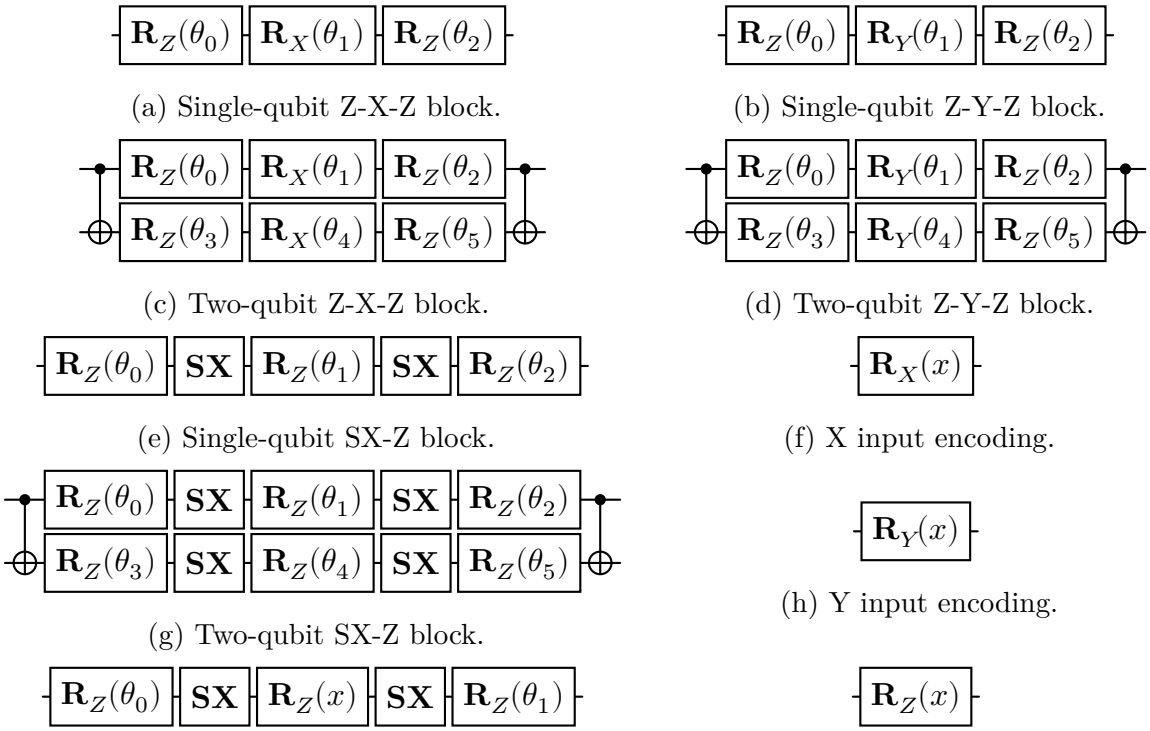
In this section we will have a look at the straight forward way of how to think about a quantum algorithm while iteratively mutating it. A quantum algorithm will be represented by the qubits it operates on and the quantum gates that act on these qubits.

The idea of iteratively altering the structure of a quantum circuit in this manner was, for example, done by Bilkis et al. [8]. We extend their approach by also considering the input of data, adding more possible mutations and, most notably, apply it in a multi-objective setting.

Algorithm 3.2 Circuit Mutation

Input:
Circuit C
Loss Function l

```
C' ← add_gates(C)
C' ← simplify_gates(C')
C' ← remove_gates(C')
p ← e^{-10 (l(C') - l(C)) / l(C)}
if rand() > p then
    C' ← C
end if
return C'
```



(a) Single-qubit Z-X-Z block.

(b) Single-qubit Z-Y-Z block.

(c) Two-qubit Z-X-Z block.

(d) Two-qubit Z-Y-Z block.

(e) Single-qubit SX-Z block.

(f) X input encoding.

(g) Two-qubit SX-Z block.

(h) Y input encoding.

(i) SX-Z input encoding.

(j) Z input encoding.

Figure 3.1: Blocks of quantum gates, that can be added to a quantum algorithm represented by a circuit.

The general idea of the circuit based mutation is given as pseudocode in algorithm 3.2. It consists of three steps, first gates are added, then gates are simplified and finally gates are removed. After that a decision is made, based on the old and new loss, if the new circuit is accepted, or not. At this point we deviate from the multi criteria approach of the outer, genetic optimization and select a single fitness criterion as the loss. Here we should choose a criterion that measures the approximation quality and not the complexity of the circuit.

The adding of gates works straight forward, we specify a list of possible blocks and randomly select one to add. We use blocks that act on one or two qubits. After selecting a block, we also randomly select the qubit, or the two qubits, it should act on, and the position in the circuit. If the selected block is one that encodes input data, we also randomly choose which input feature is used. See figure 3.1 for a list of blocks that we have used. One important fact, stated by Bilkis et al. [8], is that all of these blocks, as long as they do not encode input data, can be given a parameter set, where they represent the identity. This makes sure, that an addition of such a block can only improve on the previous performance. They are added with the parameters, that lead to the identity, and from there the parameters are optimized. As long as we use gradient descent algorithms for that optimization, we expect only improvements or no change at all.

After adding new gates a simplification step is done. This step resolves redundancies. For example two CNOT gates on the same qubits, directly after one another, can be left out, or two rotations around the same axis can be collected into a single one. We choose

to only use very simple rules, but in theory one could also use permutation rules and basically perform a circuit optimization, like qiskit's transpilation [39], in every step.

In the last modification step we remove gates that are not needed. To decide if a single gate is needed, we evaluate the loss function without that gate and check if the loss increases significantly. We do this for every gate in the circuit individually. Here are two possibilities to tweak the algorithm, on the one hand how much increase in the loss is considered significant and on the other hand, do we do a new parameter optimization run, after each gate is removed. Especially the latter has a huge impact on the computational cost of the algorithm.

Finally, we have to decide weather the new circuit is accepted or not. While the modified circuit is certainly accepted if the loss decreases, for an increased loss it is accepted with a probability that decreases exponentially in the amount the loss increases. This is similar to the idea of simulated annealing, and we could extend this procedure by adapting more concepts thereof and decrease the probability to accept a worse circuit over the optimization process [67].

3.3 Quantum Architecture Search based on ZX-Diagrams

As an alternative to directly manipulating quantum circuits, we will study, in this section, weather manipulating ZX-diagrams, introduced in section 1.3, instead brings any benefits. The section is based on the joint work with Turkalj, Holzer and Wolf [27].

Working with ZX-diagrams instead of quantum circuits brings one significant challenge, as mentioned in section 1.3 every circuit can be transformed into a ZX-diagram, the other direction is not given. If one starts with a valid circuit, transforms it into a ZX-diagram and only applies the rewrite rules, that do not change the underlying mapping, the problem is only finding an equivalent circuit. For the task of QAS we need to change linear mapping, thus, we have to deal with ensuring that the ZX-diagram keeps representing an actual quantum circuit.

We follow two approaches, on the one hand, we look into mutations that guarantee that the ZX-diagram stays extractable. On the other hand, we accept that sometimes invalid ZX-diagrams get generated and discard those. We always work with graph-like diagrams, i.e., all spiders are Z-spiders and all inner edges are Hadamard edges.

3.3.1 Mutations

We will start with describing all mutations that we have studied. Some of them were already used by Barnes [4], these we extended to work with parameters, for training as well as data encoding.

Local Complementation

The first mutation we describe is based on the following concept from graph theory.

Definition 3.3 Local Complementation

Let $G = (V, E)$ be a graph and $u \in V$ a vertex. The local complementation of G at u, denoted by $G \star u = (V', E')$, is the graph formed by complementing the subgraph $N_G(u)$. Thus, G is the same graph as $G \star u$, except on $N_G(u)$, where we have:

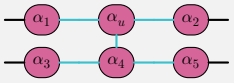
$$\forall v, w \in N_G(u) : (v, w) \in E' \iff (v, w) \notin E.$$

This operation has the benefit, that, as stated by Duncan et al. [25], it preserves the gFlow property.

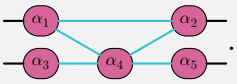
Now, let D be a graph-like diagram and $G = G(D)$ its underlying open graph. Since D is graph-like, there is a one to one correspondence, both between vertices in G and spiders in D and between edges in G and Hadamard edges in D. To perform the mutation, we pick a vertex $u \notin I \cup O$ from G at random and consider the local complementation $G \star u$. We apply the same changes to the ZX-diagram D while not touching the phases and finally remove the spider corresponding to u with all edges that are connected to it. That operation changes the underlying linear map, which is a good thing for QAS, as a mutation that does not change the underlying linear map does not help. As the gFlow property is not related to the phases, and the extractability is coupled to having a gFlow, see theorem 1.30, this mutation allows to change the linear map, while ensuring a ZX-diagram that can be transformed into a circuit again.

Example 3.4

Starting from



we perform the local complementation on the spider with phase α_u and get



.

The spider with phase α_u has been removed, together with all edges that connected to it. Since there were no edges between the spiders with phases α_1, α_2 and α_4, they have been added.

Inverse Local Complementation

The next mutation is the reversal of the previous mutation. This transformation introduces a new spider into the diagram.

From the set of inner spiders, we sample candidates which will make up the neighborhood of the newly added spider. All existing edges between these spiders will be removed, while all spiders will be connected to the newly added one, and will be connected to

spiders they have previously not been connected to. The number of spiders used for this neighborhood is also chosen randomly. The phase of the new spider is either a trainable parameter or a data encoding, again chosen at random.

This mutation is generally not gFlow preserving.

Pivoting

The next mutation is a combination of three local complementations and called pivoting.

Definition 3.5 Pivoting

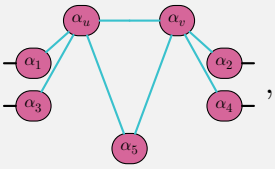
Let $G = (V, E)$ be a graph and $u, v \in V$ two vertices with $(u, v) \in E$. The pivot of G at (u, v), denoted $G \wedge (u, v)$, is defined as $((G \star u) \star v) \star u$.

The transformation can be simplified in the following way. Let $U := N_G(u) \setminus N_G(v)$, $V := N_G(v) \setminus N_G(u)$ and $W := N_G(u) \cap N_G(v)$. The new graph $G \wedge (u, v)$ is constructed by complementing the edges between the three subgraphs U, V and W.

For the mutation we randomly select two connected spiders and perform a pivot at the edge between those two. After that we also remove the two original spiders together with all their edges from the ZX-diagram.

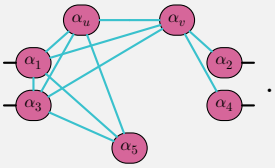
Example 3.6

Let us go step by step through such a mutation. We start from the following ZX-diagram



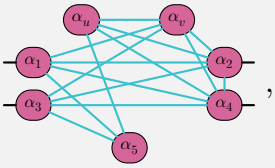
,

and perform a pivoting on the vertices with phases α_u and α_v. We apply a local complementation on the spider with phase α_u leading to



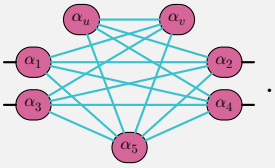
.

Next we apply a local complementation on the spider with phase α_v to get



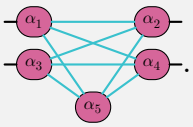
,

and another local complementation on the spider with phase α_u



.

Finally, we remove the two spiders the pivot was done along and get



.

From [25, theorem 4.5] we know that this operation is gFlow preserving.

Phase Gadget Addition

This mutation adds a so-called phase gadget to the ZX-diagram.

Definition 3.7 Phase Gadget

A phase gadget is a ZX-diagram of the form

,

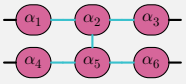
i.e., a Z-spider with a phase that has exactly one Hadamard edge to a phaseless Z-spider that is itself connected to arbitrary many other spiders.

Phase gadgets are defined a little different in the literature [22], but as we want to use graph-like ZX-diagrams, we use this modified version.

When we want to apply this mutation, we first sample the number of spiders we will attach the gadget to, then we sample the spiders, and finally we randomly decide if the new phase will be a trainable parameter or an input encoding.

Example 3.8

Adding a phase gadget to this diagram



leads, for example, to this

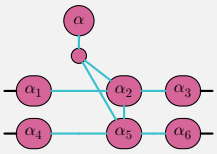


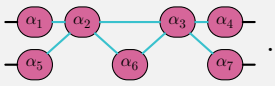
diagram.

Edge Flipping

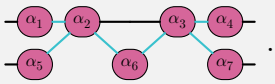
Edge flipping, as the name suggests, flips a random edge from Hadamard to a regular one or vice versa. Two things to note, we allow input and output wires to be selected for flipping, and as we convert to graph like diagrams after each mutation, such a flip can have non-obvious consequences.

Example 3.9

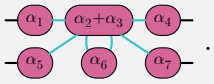
Consider the following ZX-diagram



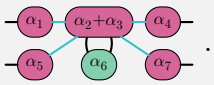
We perform an edge flip to the edge between vertices α_2 and α_3, which leads to



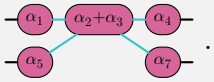
There are now two Z-spiders connected by a standard edge, which means we can fuse them to get



We now switch the color of vertex α_6 to get to



By the Hopf rule, one of the rewrite rules from ZX-calculus, we can disconnect α_6 from the rest of the diagram, thus, up to a global phase, removing it and finally get



Edge Addition

For this mutation two random spiders are selected, that are not already connected by an edge, and connected by a new edge, randomly chosen weather Hadamard or not.

Edge Removal

A randomly selected wire between two spiders is removed. During selection, input and output wires are excluded.

Edge Swap

This mutation combines an edge addition with a removal. First a random edge is removed, here input and output wires are again excluded. Then two previously unconnected spiders get connected.

Edge Split

This mutation introduces a new Z-spider by splitting an existing edge into two parts and inserting the new spider in between. The edge to be split is selected at random from the set of all edges. It is randomly decided weather the new phase will be a trainable parameter or an input encoding, the two edges will be of the same type as the one that was split.

With the mutations introduced in the previous section we can state the mutation algorithm for ZX-diagrams.

Algorithm 3.10 ZX-diagram Mutation

Input:
ZX-diagram D
Loss Function l
List of Mutations and application probabilities $\left[(\mathrm{M_j}, p_j)\right]_{j\in\{1,\dots,N\}}$
Maximal mutation tries m

for $j \in \{1, \dots, N\}$ **do**
  if `rand()` $\leq p_j$ **then**
    repeat m **times**
      $D' \leftarrow \mathrm{M_j}(D)$
      $D' \leftarrow$ `to_graph_like`(D')
      if `test_extractability`(D') **then**
        $D \leftarrow D'$
        break
      end if
    end repeat
  end if
end for
return D

The basic idea of algorithm 3.10 is, that every mutation that should be considered, is coupled with a probability. In every step each mutation is applied with that probability. If a mutation is applied, this application is tried, up to specified number of times, until an application is successful. Successful means, that from the resulting diagram a valid circuit can be extracted.

3.3.2 Impact of the mutations

Having introduced the set of possible mutations two questions arise, how often are the respective mutations successful and, if they are successful, how much of a difference do they actually make.

To answer these questions we performed numerical experiments. For the first question the strategy is simple, we apply the mutations to different ZX-diagrams and check if the result allows the extraction of a circuit. For the second question, we need a measure for

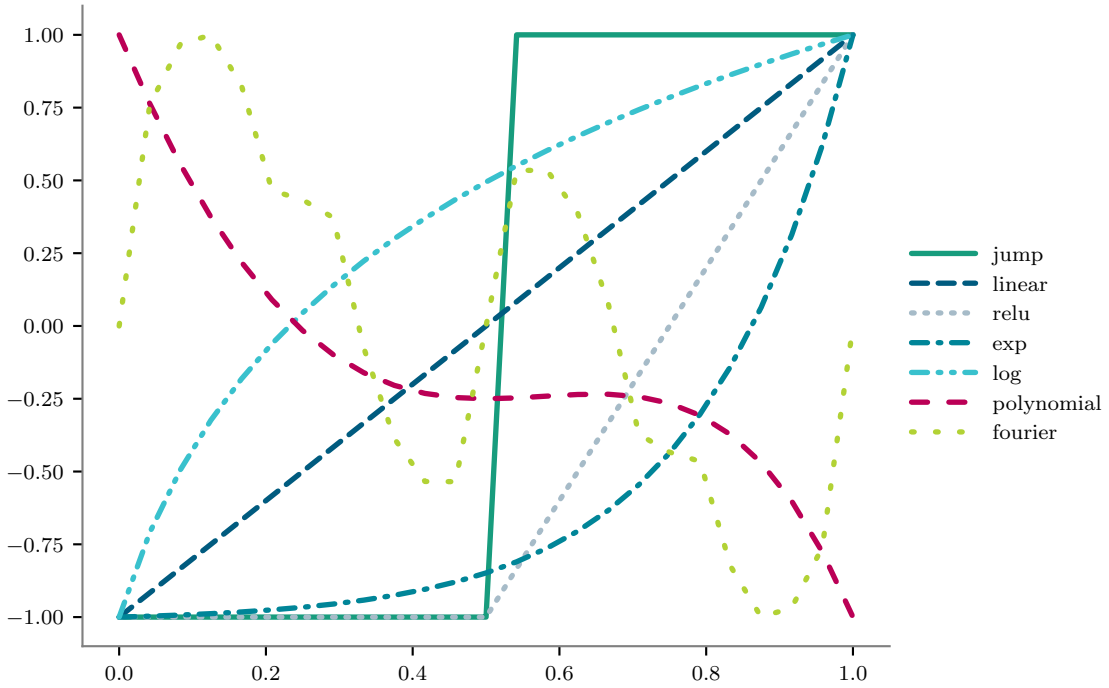


Figure 3.2: Test functions to evaluate ZX-diagram mutations. We use a jump function, a simple linear function, a ReLU function, an exponential function a logarithmic function, a polynomial function of order four and a truncated Fourier series with four summands. All functions are scaled to map $[0, 1]$ to $[-1, 1]$.

the change in the ZX-diagram. One could use metrics for linear maps, but the diagrams are parameterized, and the resulting expressions would be very hard to calculate and to interpret.| Thus, we used a heuristic approach closer to what we are actually interested in. We use the parameterized ZX-diagrams to approximate functions, therefore we are mainly interested in how much a mutation changes the ability of the diagram to learn functions.

To this end we introduce a set of test functions, depicted in figure 3.2. We then generated 3309 random ZX-diagrams with different numbers of qubits, vertices and number of edges. For the random generation we used generator functions from the python library *pyzx*, namely `CNOT_HAD_PHASE_circuit`, `cliffords` and `cnots`, with a randomly chosen number of qubits between 2 and 10 and depths between 10 and 30. We recorded for every ZX-diagram the number of qubits, the number of vertices and the connectivity, which is the proportion of actual edges to the maximal possible number of edges. This allows us to analyze, if these properties have an influence on the performance of the mutations.

For every generated ZX-diagram we train the parameters once for every test function and record the approximation error. Then we apply the mutations, for every ZX-diagram we apply each mutation, if the mutation yields an extractable ZX-diagram we train the parameters for the test functions again and record the approximation error after the mutation. Since all mutations involve randomness, we repeat this ten times for every combination of test function, ZX-diagram and mutation. Note that every single mutation is looked at individually, i.e., each mutation is applied to the original ZX-diagram and before retraining the parameters only a single mutation is applied.

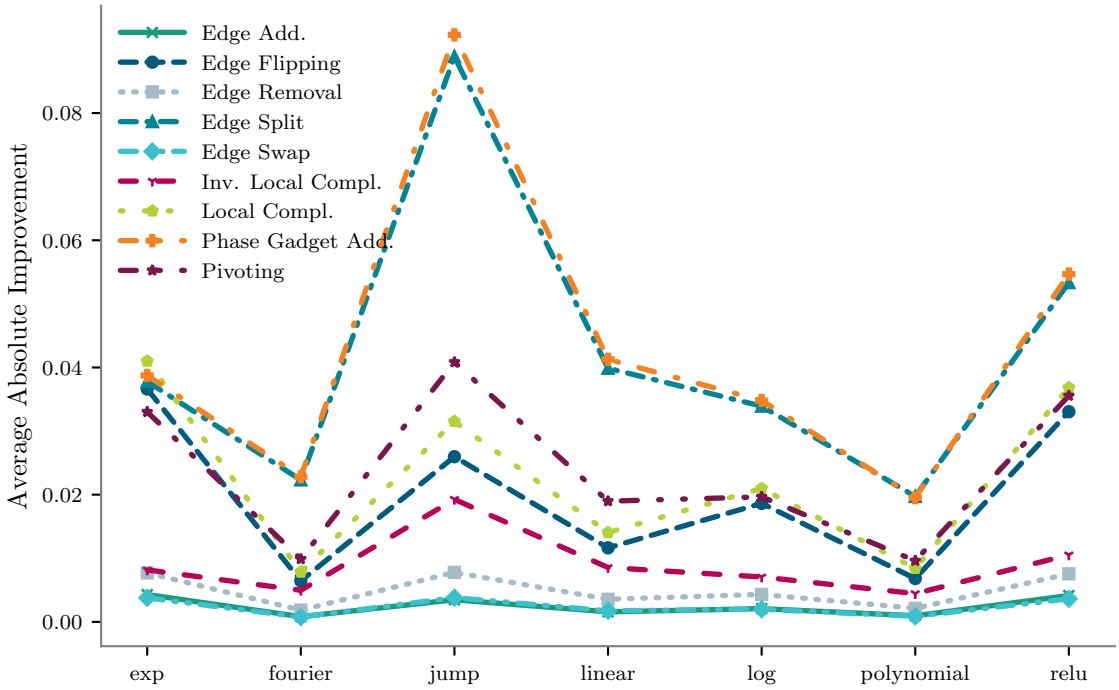


Figure 3.3: The absolute improvement in the approximation quality for the different test functions and mutations, averaged over all considered ZX-diagrams. We see significant differences between the different mutations and the different test functions respectively. But we see no strong evidence, that some mutations are especially well suited for certain test functions.

In the next step we analyze the generated dataset. First, we have a look at the average absolute improvement of the different mutations for the different test functions, it is visualized in figure 3.3. We directly see, that there is a big difference between the different mutations as well as between the different test functions. One important result is, that the edge splitting and the adding of a phase gadget behave very similarly and tend to have the largest effect on approximation quality of the ZX-diagram.

We do not see strong evidence, that specific mutations are especially well suited for specific functions. There is one notable anomaly, the local complementation, pivoting and the edge flip perform on the exponential function nearly as well as the phase gadget addition, but on the other test functions they perform worse. Nonetheless, we conclude, that it does not make sense to use specific mutations depending on the target function.

In the next step of the analysis we check, if the structure of the ZX-diagram has an influence on the performance of the mutations. To that end we calculated the correlation between the connectivity, the number of vertices and the number of wires, or qubits, to three different measures for the performance of the mutations. First to the success rate, then to the average absolute improvement while counting failed mutations as an improvement of zero and finally to the average absolute improvement while completely ignoring failed mutations. We calculated these correlations by training linear models to predict the three performance measures using the ZX-diagrams properties as features. By standardizing the features before the training we ensure that the intercept of the model

equals the average value and the coefficients equal the correlation of the feature with the target value. The results are visualized in figure 3.4. Additionally, the coefficients together with their p-values are shown in tables C.1 to C.3, most p-values are small enough for the detected correlation to be significant, and if they are not, the corresponding coefficient is almost zero anyway.

We see, not surprisingly, that the mutations that are guaranteed to preserve the gFlow, pivoting and local complementation, are highest in the success rate ranking. Pivoting only fails in the rare case, that the ZX-diagram does not have two suitable vertices for the mutation, while local complementation always succeeds. Another notable observation, the inverse local complementation, while very unlikely to succeed, if it succeeds has a significant impact. The mutations with the highest impact are the edge splitting and the phase gadget addition, which both are failing very rarely and have a big impact.

Seeing that some mutations have a big impact if they succeed, even while not succeeding often, motivates the strategy to repeatedly try a mutation until a success in algorithm 3.10.

All in all these results support the claim, that all mutations, we have analyzed, make sense to use, even if some of them have a bigger impact than others, none have a vanishing impact.

The results could be used in further research to use different mutations depending on the properties of the mutated graph, to increase the probability of successful mutations. Or depending on the approximation error, using mutations with a big impact as long as the error is large and switching to mutations with a small impact, once the error is smaller.

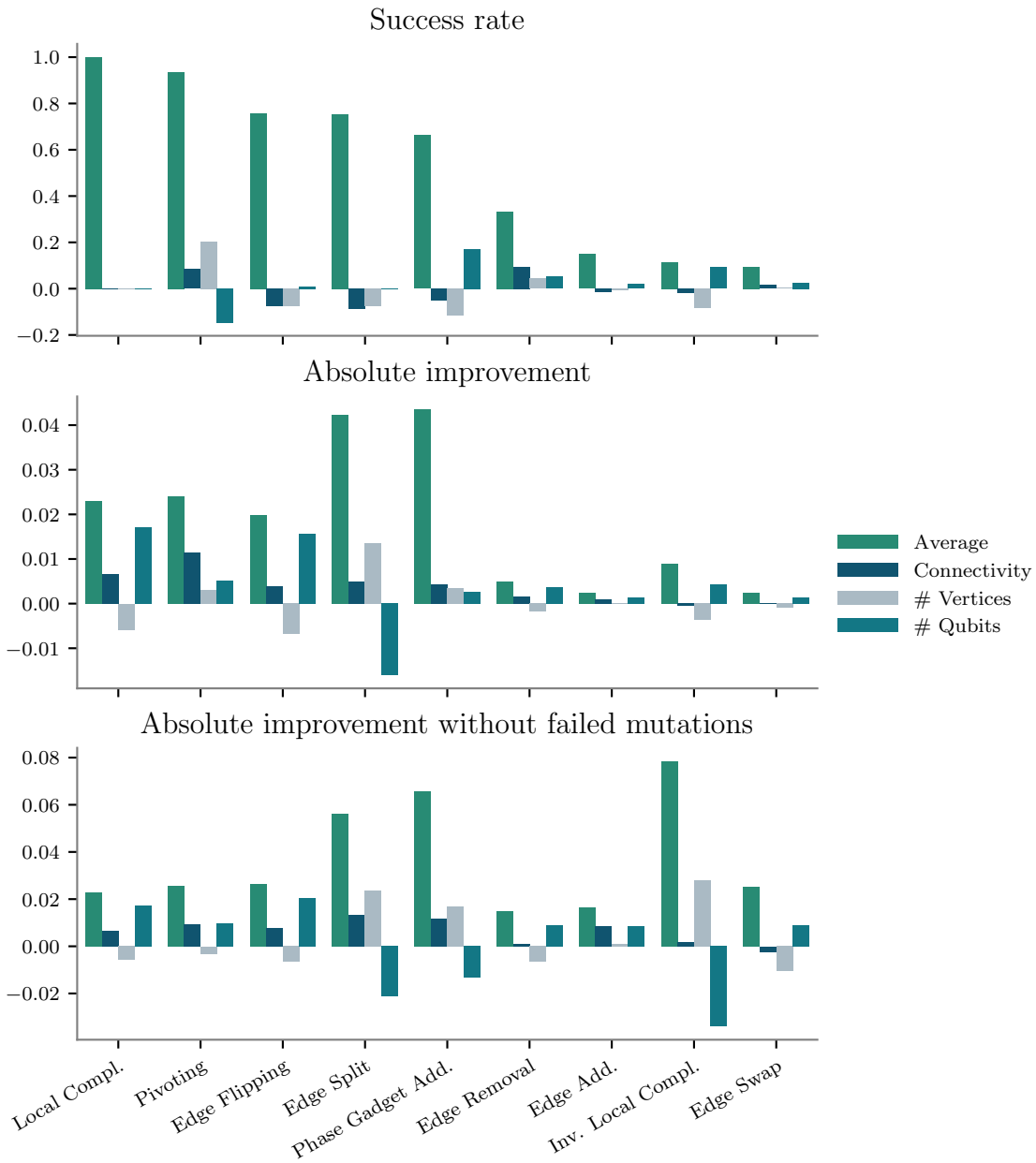


Figure 3.4: Influence of the ZX-diagrams' properties on the effect of the different mutations. In the first row on the success rate, in the second on the absolute improvement with failed mutations treated as zero improvement and in the third row on the absolute improvement without the failed mutations. Depicted are the coefficients of a linear model trained to predict the respective value using the standardized connectivity, number of vertices and qubits as regressors. As the variables are standardized, we can interpret the intercept of the model as the average value and the coefficients of the features as their correlation with the target value. The coefficients together with their respective p-values for all trained models are given in tables C.1 to C.3.

3.4 Numerical Results

In this section we will put the algorithms from the previous sections to use. As the goal of this thesis is pricing options we will learn the payoff functions of four different options, one of them in two different variations.

3.4.1 Implementation Details

Before we will go into the results of our numerical experiments we will give remarks on some implementation details.

First and foremost, everything shown in this work is calculated with classical computers, simulating quantum computers. If not stated otherwise, without any noise.

Our approach to QAS is computational very demanding, as for every evaluation of the fitness of a circuit we need to optimize the parameters of that circuit. This optimization process needs many evaluations of the loss of that circuit. Since we are doing regression, calculating the loss once means evaluating the circuit once for every data point. All this leads to a number of needed circuit evaluations, which is not feasible on currently available quantum hardware. In chapter 4 we will argue why the results are still relevant and can benefit calculations on a real quantum computer, even in settings that can no longer be simulated by classical computers.

For sake of completeness, if we would want to perform the complete optimization using actual quantum hardware, we would be limited to using optimizers that are either gradient free [55] or calculating gradients with the parameter shift rule [57]. One gradient free optimizer, that is deemed to be especially useful for optimizing PQCs, is the Simultaneous Perturbation Stochastic Approximation (SPSA) optimizer [62]. Both approaches have drawbacks, calculating gradients by the parameter shift rule needs additional circuit evaluations scaling with the number of trainable parameters, and gradient free optimizers usually need more evaluation of the loss function.

Simulators on the other hand, can make use of a powerful tool from classical machine learning, automatic differentiation [6]. This tool allows the calculation of exact gradients with little computational effort.

For our experiments we rely on the python library *jax* [11] for the calculation of gradients and the library *jaxopt* [10] for optimizers specifically designed to work with these gradients. For the circuit based approach, detailed in section 3.2, we further use the python library *pennylane* [7] to represent circuits and handle the parameterization of them.

The approach based on the ZX-calculus made use of the python library *pyzx* [41], which needed some adaptation to nicely work together with *sympy* [51], which, in turn, was needed to implement the parameterization of ZX-diagrams.

Under `https://gitlab.cc-asp.fraunhofer.de/itwm-fm-qc-public/cvqa` the code of our implementation can be found.

3.4.2 General Considerations

The goal of QAS is finding good circuits for a certain problem. What is meant by good, is not directly clear, and can also depend on the problem itself. The most obvious concern for regression problems, is the approximation quality, often measured by the MSE. We are in the easy situation, that we use this regression model to learn functions, thus, overfitting is not a concern for us. If it were, simply minimizing the MSE would not be the best strategy. But even without caring for overfitting, just minimizing the approximation error is not ideal.

The first reason is the same as in classical machine learning. Models that have a low approximation error are often the ones with many parameters, thus, they longer and more resources to train. Especially in quantum computing these resources are currently scarce.

The other reason, that is unique to quantum computing, is the noise of the devices. Current devices have, very broadly speaking, two kinds of noise. On the one hand unwanted disturbances of the quantum state, i.e., after applying a quantum algorithm the final state is not as expected. There is hope that this kind of noise can be mitigated [14] or even corrected in the future [18, 50]. On the other hand there is shot noise, which even idealized quantum computers will have. Shot noise comes from the stochastic nature of quantum computing. When measuring the final state of a quantum algorithm, the quantum state only dictates the probabilities of measuring the possible states. As the number of repeated measurements, so-called shots, is finite, the observed relative probabilities can only be approximations of the true probabilities.

With these sources of noise in mind, it does not make a lot of sense to use a highly complex model with a perfect, theoretical, approximation quality, which a quantum computer can never practically achieve. Especially on non-fault-tolerant quantum devices, the noise increases with the depth of the circuit and with the number of two-qubit interactions, making a balance between the complexity and performance of the model even more important.

Taking these considerations into account, we use the following fitness criteria for our multicriteria genetic QAS approach. For the approximation quality we use the MSE, which is a commonly used loss function, and fits to the theoretical results that some PQCs converge in L^2 norm to any sufficiently regular function [69]. To measure the model complexity we use three different measures, the circuit depth, the number of two-qubit gates and the number of input encoding gates. The reasoning for the first one is quite obvious, the shorter a circuit the easier it is to evaluate. Moreover, quantum devices have a maximal cohesion time, that puts a limit on the total number of gates that can be executed. The number of two-qubit gates is also important, as these gates usually have a higher error rate, than the single-qubit gates. Using the number of input encodings separately may seem a little strange at first glance, but our final goal with the learned circuits is to use them for option pricing via CPQCs introduced in section 2.4. When transforming a PQC into a CPQC the number of additional two-qubit gates scales linearly with the number of input encoding gates, thus, keeping their number low will benefit this procedure.

In the remainder of this chapter we show three different versions of algorithm 3.1 applied to five different functions with one to three input dimensions.

The three versions are the ZX-diagram-based approach from section 3.3 and two circuit-based versions from section 3.2. The first version, based on the concept of Variable Ansatz (VAns), suggested by Bilkis et al. [8], applies multiple mutation steps per generation, and we will call it VAns-based. The other approach based on circuits, which we will simply call circuit-based, only performs one addition of gates per generation.

For all of these methods we have to specify a starting population. We use a similar circuit for all three versions as starting point, a similar structure as seen in figure 2.6. We use one repetition for the two versions based on quantum circuits and two repetitions for the ZX-diagram-based version. Further, we use models with two to five qubits, and as measurement operator a Pauli Z on the first qubit and a Pauli Z on all qubits. The VAns-based approach also uses different values for the threshold while removing non-contributing quantum gates, i.e., how much loss increase can the removal of one gate cause and still be removed. We have used values between 0 and 0.1 for the relative loss increase.

3.4.3 Setup

We start with defining the test cases, they consist of the payoff function and a discretization. From that we can build regression problems following definition 2.26. Only the class of functions will be left open, as finding a good class is the goal of QAS.

Example 3.11 European Call

The first example will be a straight forward European call with a strike of 100, i.e., the payoff is given as

$$\Pi_c(S) := \max\{S - 100, 0\}.$$

We take $K = 400$ data points $\mathbf{x}_j \in [50, 200]$ that we choose equidistantly on the complete interval. The target points are then

$$y_j := \Pi_c(\mathbf{x}_j).$$

Example 3.12 European Basket Call

A possible extension of the normal call, is the basket call. As underlying it uses the weighted sum of multiple assets instead of a single asset. For this example we use only two assets that are weighted with 1/3 and 2/3 and again a strike of 100. This leads to the following payoff

$$\Pi_{bc}(S_1, S_2) := \max\left\{\left(\frac{1}{3}S_1 + \frac{2}{3}S_2\right) - 100, 0\right\}.$$

We take $K = 400$ data points $\mathbf{x}_j \in [50, 200] \times [50, 200]$ that we choose equidistantly. The target points are then

$$y_j := \Pi_{bc}(\mathbf{x}_j).$$

Example 3.13 European Basket Call with variable weight

The same as example 3.12, but with the weight as an extra parameter. The payoff then changes to

$$\Pi_{bcw}(S_1, S_2, w) := \max\{(wS_1 + (1-w)S_2) - 100, 0\}.$$

We take $K = 8000$ data points $\mathbf{x}_j \in [50, 200] \times [50, 200] \times [0, 1]$ that we choose equidistantly. The target points are then

$$y_j := \Pi_{bcw}(\mathbf{x}_j).$$

Example 3.14 European Rainbow Call

The rainbow call is similar to the basket call, but instead of the weighted mean of the underlying assets, the maximum is used, i.e.,

$$\Pi_{rc}(S_1, S_2) := \max\{\max\{S_1, S_2\} - 100, 0\}.$$

We take $K = 400$ data points $\mathbf{x}_j \in [50, 200] \times [50, 200]$ that we choose equidistantly. The target points are then

$$y_j := \Pi_{rc}(\mathbf{x}_j).$$

Example 3.15 Asian Barrier Spread

The Asian barrier spread is slightly more involved and differs in not looking at two separate assets, but at two time points of the same underlying. The barrier part says, it pays out nothing if the asset is at one time point above the barrier, we chose 180. Then we have a normal call on the average of the two time points with a strike of again 100. Finally, the payoff is limited from below by 50.

$$\Pi_{abs}(S_1, S_2) := \begin{cases} \min\left\{\max\left\{\frac{S_1+S_2}{2} - 100, 0\right\}, 50\right\} & \text{for } S_1, S_2 < 180 \\ 0 & \text{else} \end{cases}.$$

We take $K = 400$ data points $\mathbf{x}_j \in [50, 200] \times [50, 200]$ that we choose equidistantly. The target points are then

$$y_j := \Pi_{abs}(\mathbf{x}_j).$$

The last missing detail for a regression problem from definition 2.26 is the function class. We will be using PQCs, but how exactly they will look like is automatically learned with algorithm 3.1.

All variations have in common, that we train 48 models per generation. We chose this number, as all models of one generation can be trained in parallel and the machine this experiments was performed on is equipped with a 48 core processor. For the different examples the hard constraints and the allowed time differ and are shown in table 3.1. We use different values for the different problems, as they vary in the difficulty for a PQC to solve them, thus, we allow more complex circuits, higher errors and longer training times for the more complex problems.

Table 3.1: Available training time and used hard limits for the different examples. These parameters are the same for all three versions of QAS.

	Hours trained	Maximal circuit depth	Maximal MSE	Maximal # Two-qubit gates	Maximal # input gates
European Call 3.11	24	100	0.2	50	20
European Basket Call 3.12	24	75	2.0	30	20
European Rainbow Call 3.14	36	75	4.0	50	20
Asian Barrier Spread 3.15	72	100	50.0	50	30
European Basket Call with variable weight 3.13	72	100	7.0	50	20

3.4.4 Results

In total, we have analyzed three different versions of QAS for five different problems. For the sake of brevity, we only go into detail for the European call seen in example 3.11, the results for examples 3.12 to 3.15 are moved to the section D.

The output of one run of algorithm 3.1 is not one single PQC, but a population of non-dominated PQCs. From here a decision needs to be made, which fitness criteria are most important or which middle ground is most beneficial. This decision may vary for different applications. The non-dominated PQCs together form the Pareto front, which dimension equals the number of used fitness criteria. As we use four criteria, the Pareto front is four dimensional and visualization becomes a bit difficult. We chose to look at the projections of the four-dimensional Pareto front onto the three planes spanned by the MSE and the other three criteria, respectively. This will give a way to see the trade-off between the three criteria describing the circuit complexity and the approximation quality.

In figure 3.5 we see the Pareto front for example 3.11, the same plots for the other examples can be found in the appendix as figures D.1, D.5, D.9 and D.13. On the one hand, this visualization can support the decision, which model is best suited for a given problem. On the other hand, they allow us to compare the different versions of our QAS algorithm.

In general, we can observe, that the different versions have their strengths in different areas and perform differently well for the different problems. A general trend is, that the VAns-based approach reaches the lowest approximation errors, while the other two approaches perform better for models with lower complexity.

For the visualization of individual models we focus on the extreme cases, the models

that are best in one of the fitness criteria. For practical purposes, these choices may not be ideal because, although they excel in one area, they often pay for that by a poor performance in the other criteria. We tend to see diminishing returns in the gained approximation quality for increasing circuit complexity.

In table 3.2 we see a summary and comparison of the models we chose for visualization. In figures 3.6 to 3.8 we see the models predictions. We can see that although the MSE ranges roughly between 0.05 and 0.5, all approximations are quite good for this example. To get a feeling for the other fitness criteria see figures 3.9a to 3.9d, 3.10a to 3.10d and 3.11a to 3.11c for the respective circuits.

Table 3.2: Results for the European call seen in example 3.11. All versions used 24 hours for the training. The lowest achieved value over all models, i.e., the best model in each fitness criteria, is marked bold. The last column gives the figure number of the corresponding circuit.

	Trained generations	Minimized criterion	MSE	Circuit depth	# Two-qubit gates	# Input encodings	Figure
Circuit-based	486	MSE	0.0431	100	26	12	3.9a
		Circuit depth	0.1718	**17**	**3**	6	3.9b
		# Two-qubit gates	0.0547	25	**3**	11	3.9c
		# Input encodings	0.1429	31	**3**	6	3.9d
VAns-based	131	MSE	**0.0396**	100	23	13	3.10a
		Circuit depth	0.1598	19	**3**	8	3.10b
		# Two-qubit gates	0.0713	37	**3**	12	3.10c
		# Input encodings	0.1560	31	5	7	3.10d
ZX-graph-based	360	MSE	0.0795	42	11	8	3.11a
		Circuit depth	0.1153	**17**	5	8	3.11b
		# Two-qubit gates	0.1153	**17**	5	8	3.11b
		# Input encodings	0.1758	22	8	**5**	3.11c

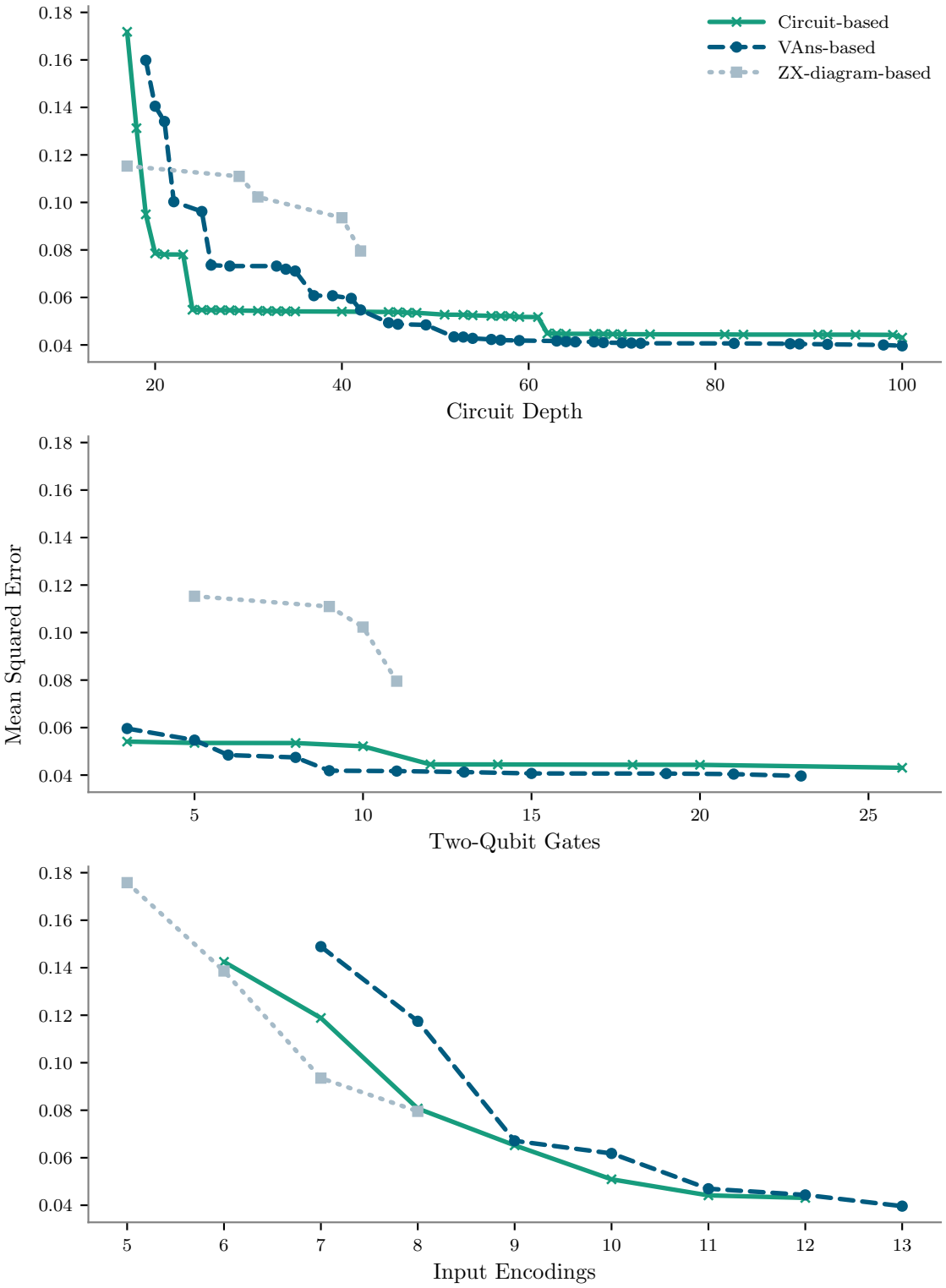
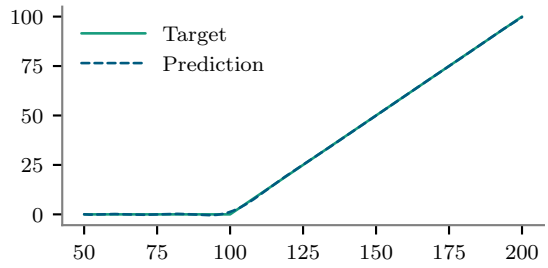
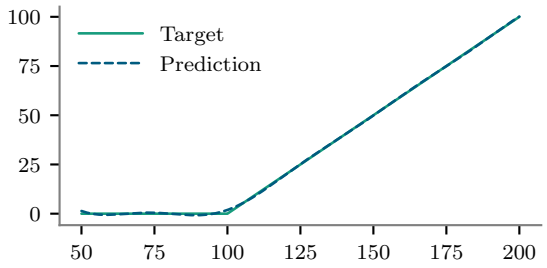


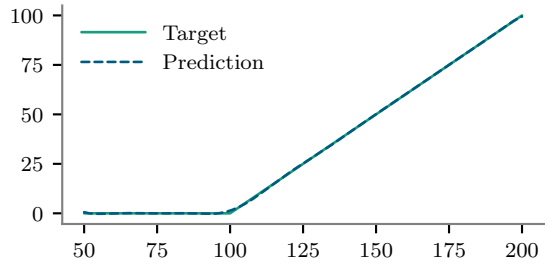
Figure 3.5: Result of the genetic QAS with algorithm 3.1 for the European call seen in example 3.11. Projection of the Pareto front into the planes spanned by the MSE and the other three fitness criteria. In every plot only the models are shown, that are on the Pareto front regarding the two displayed fitness-criteria.



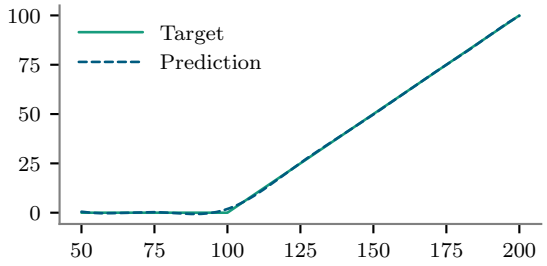
(a) MSE - figure 3.9a



(b) Circuit depth - figure 3.9b

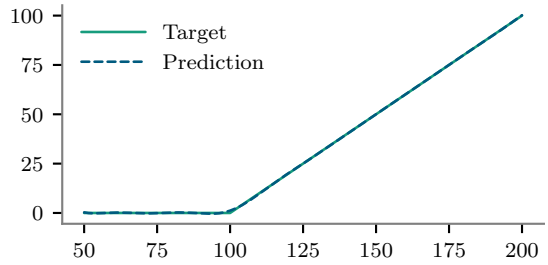


(c) Number of controlled gates - figure 3.9c

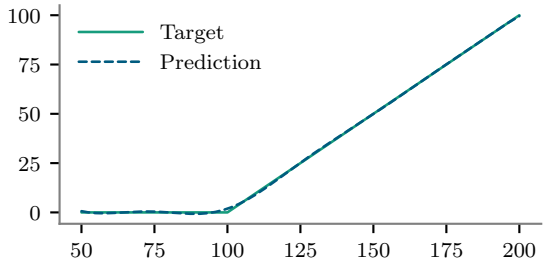


(d) Number of input gates - figure 3.9d

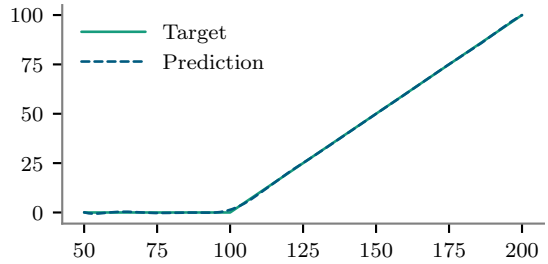
Figure 3.6: Predictions of four Pareto optimal models found by algorithm 3.1 in the circuit-based version for the European call, found in example 3.11. Each of the models shown is optimal regarding one of the four fitness criteria.



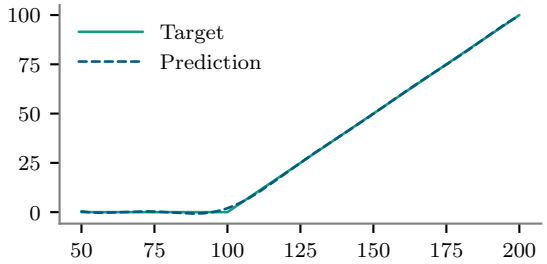
(a) MSE - figure 3.10a



(b) Circuit depth - figure 3.10b

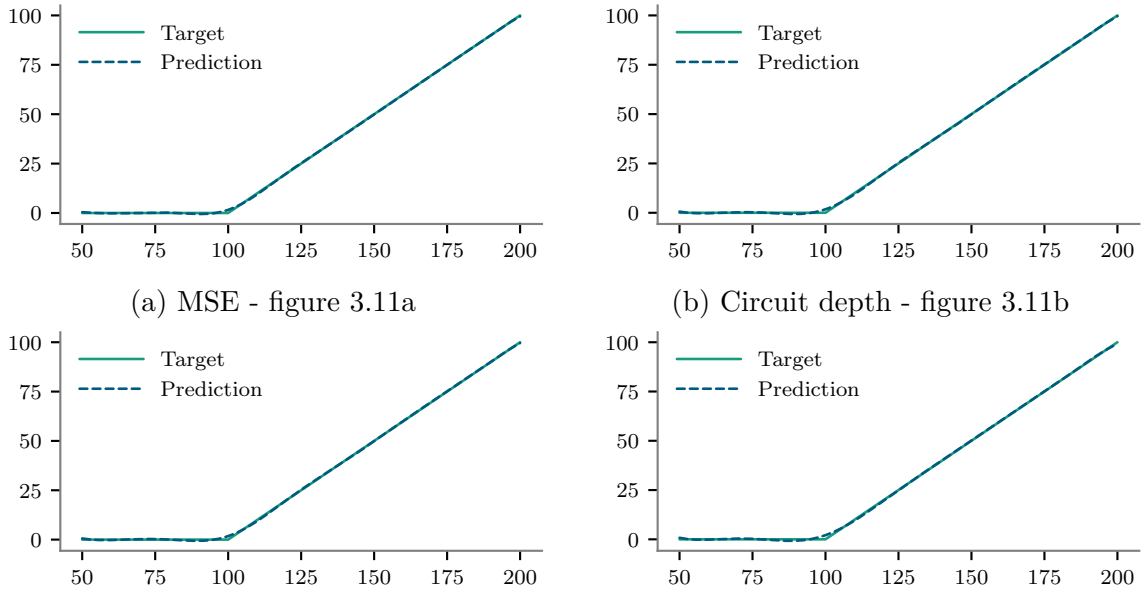


(c) Number of controlled gates - figure 3.10c



(d) Number of input gates - figure 3.10d

Figure 3.7: Predictions of four Pareto optimal models found by algorithm 3.1 in the VAns-based version for the European call, found in example 3.11. Each of the models shown is optimal regarding one of the four fitness criteria.



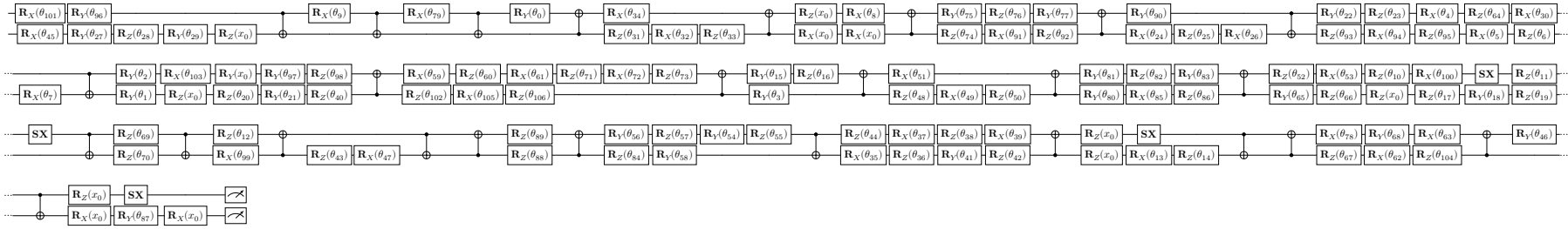
(a) MSE - figure 3.11a

(b) Circuit depth - figure 3.11b

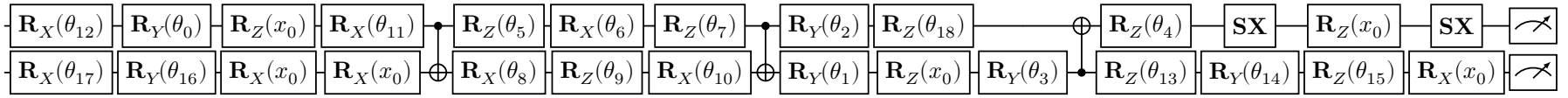
(c) Number of controlled gates - figure 3.11b

(d) Number of input gates - figure 3.11c

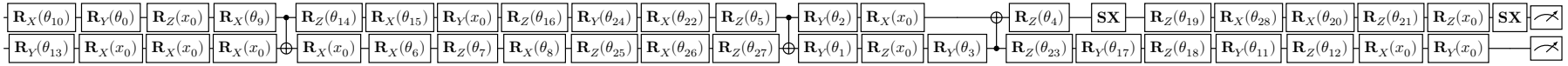
Figure 3.8: Predictions of four Pareto optimal models found by algorithm 3.1 in the ZX-diagram-based version for the European call, found in example 3.11. Each of the models shown is optimal regarding one of the four fitness criteria.



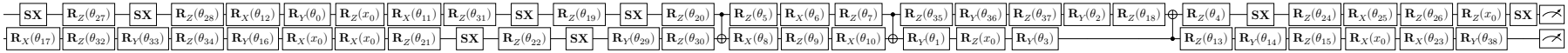
(a) Lowest MSE.



(b) Lowest depth.

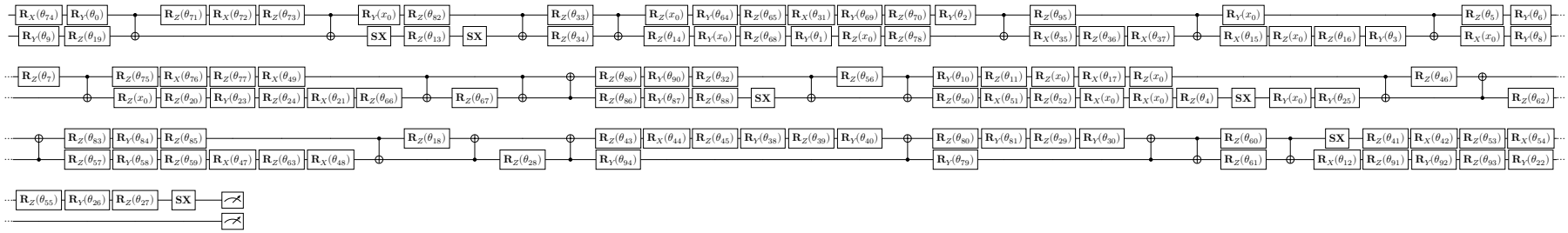


(c) Fewest two-qubit gates.

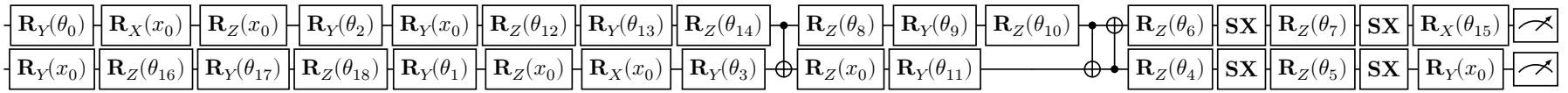


(d) Fewest input encodings.

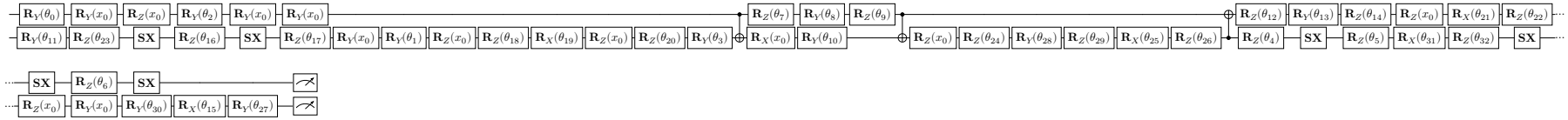
Figure 3.9: Circuits of the models optimal regarding the individual fitness criteria generated by the circuit-based version of algorithm 3.1 for the European call, found in example 3.11



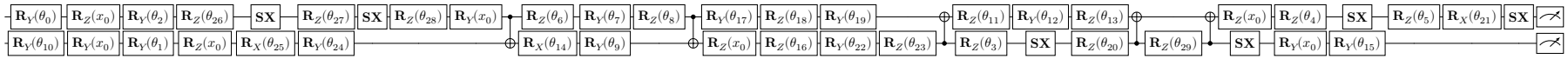
(a) Lowest MSE.



(b) Lowest depth.

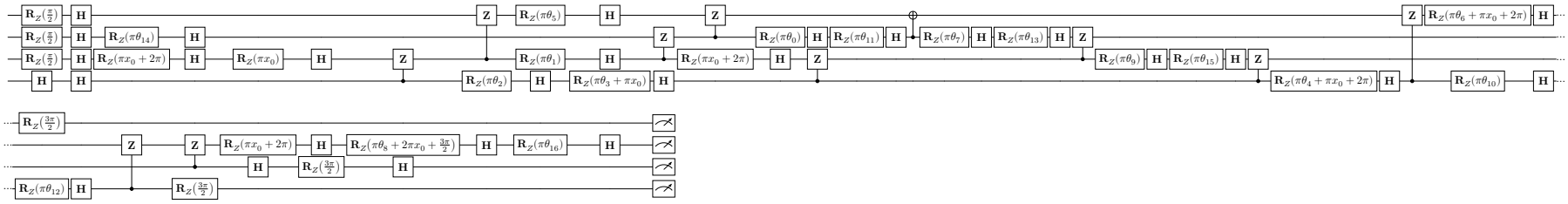


(c) Fewest two-qubit gates.

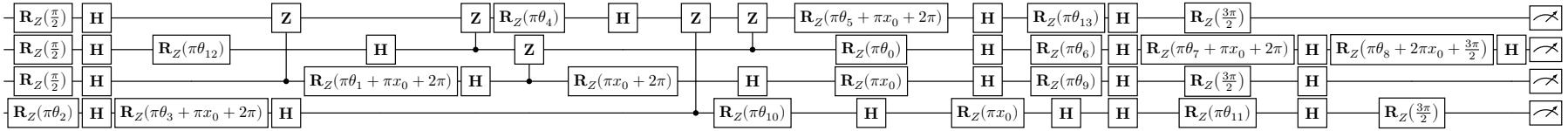


(d) Fewest input encodings.

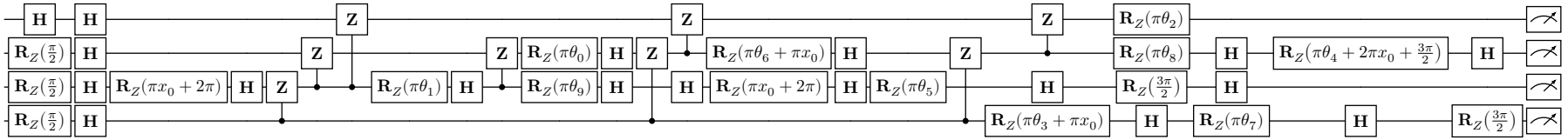
Figure 3.10: Circuits of the models optimal regarding the individual fitness criteria generated by the VAns-based version of algorithm 3.1 for the European call, found in example 3.11



(a) Lowest MSE.



(b) Lowest depth and fewest two-qubit gates.



(c) Fewest input encodings.

Figure 3.11: Circuits of the models optimal regarding the individual fitness criteria generated by the ZX-diagram-based version of algorithm 3.1 for the European call, found in example 3.11

3.4.5 Application to real hardware

Let us finish this chapter with some thoughts on how this approach can be extended for real hardware. Three main concerns come to mind when switching from simulators to actual hardware, noise, gate sets and topology.

With topology, we mean the layout and connectivity of qubits of a specific hardware. This limits which qubits can be directly entangled with each other. Simulators often assume all to all connectivity, i.e., we can directly entangle every qubit with any other. This is, depending on the physical implementation of the qubits, often far from the truth and translating a circuit that was designed for all to all connectivity to a topology with less connectivity, leads to many additional gates. To mitigate this problem, it would be nice, if our QAS algorithms would follow a given topology. While it is not clear to the author if and how this is possible for the ZX-diagram-based approach, it is quite straight forward for the two circuit based approaches. Whenever we add a two-qubit block and sample on which qubits this block will be applied, we restrict the possible choices to pairs of qubits, that are physically connected.

A similar approach can be employed to take a limited gate set into account. Every actual quantum computer has only a small set of gates in can physically perform. This needs to be a universal gate set, i.e., all other gates can be build from this set. But translating a circuit to such a base set again adds overhead. By only using gates from this available gate set in the first place, we could prevent this overhead.

The last aspect, the noise, is more involved. But one idea would be, as we already have a multi criteria optimization in place, to add the variance of the circuit over multiple shots as an additional fitness criteria.

Chapter 4

Contents

4 Option Pricing and Quantum Computing

In this chapter we will apply the theory from the previous chapters to option pricing. First we will demonstrate how CPQCs from section 2.4 can be used to price options. Then we will go, in some more detail, into option pricing with the QFT. Finally, we will benchmark the two versions against each other and against another approach from the literature.

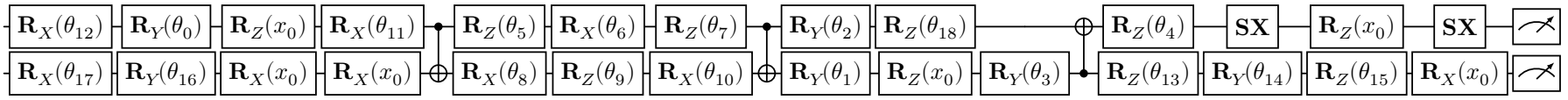
4.1 Option Pricing with Conditional Parameterized Quantum Circuits

This section will make use of the results from chapter 3 and transform the found PQCs into CPQCs from section 2.4 to calculate option prices. Conveniently, the examples from section 3.4 are all payoff functions of different options, thus, we can use the results directly. A question we want to answer in this section is, which of the Pareto optimal circuits found by the genetic QAS is the best to use for option pricing.

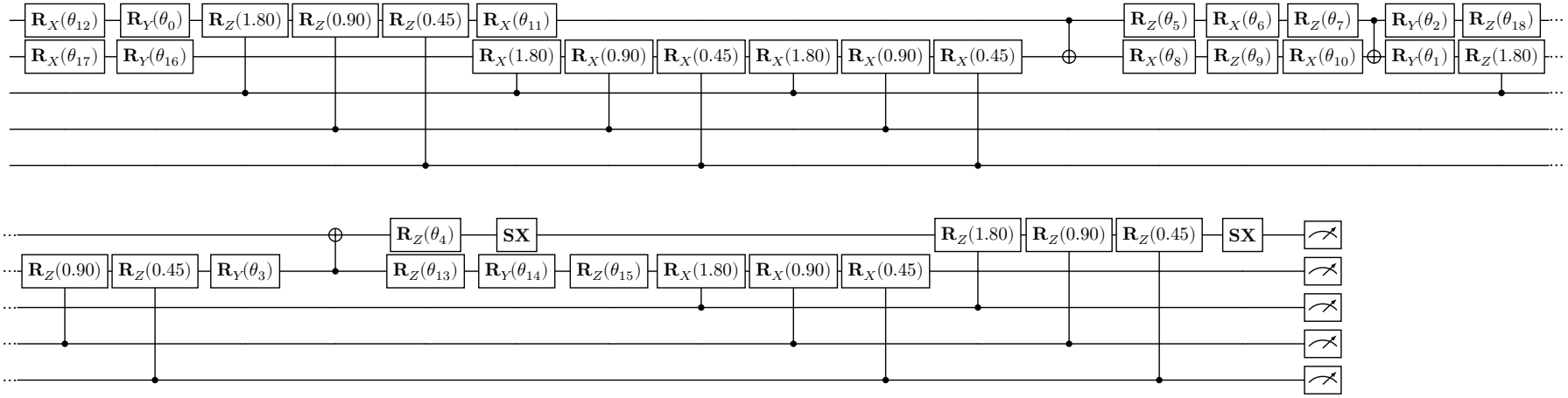
We will limit this analysis to the European call from example 3.11 in the Black-Scholes model from example 1.4 for two reasons. There is a closed formula available for the option price with which we can compare our results, and it is the simplest payoff function, thus, demanding the fewest resources while simulating the circuits.

To demonstrate our approach, we take one circuit for which we go through the steps from a PQC to an option price. To make the visualization as compact as possible, we use the circuit with the lowest depth on the generated Pareto front, generated by the circuit based approach, see figure 4.1a.

The first step is to build a controlled version, a CPQC out of the original circuit, by replacing every input encoding by a sequence of controlled but fixed rotations. Starting from the circuit depicted in figure 4.1a, the result is shown in figure 4.1b. We have used three qubits to discretize the input.



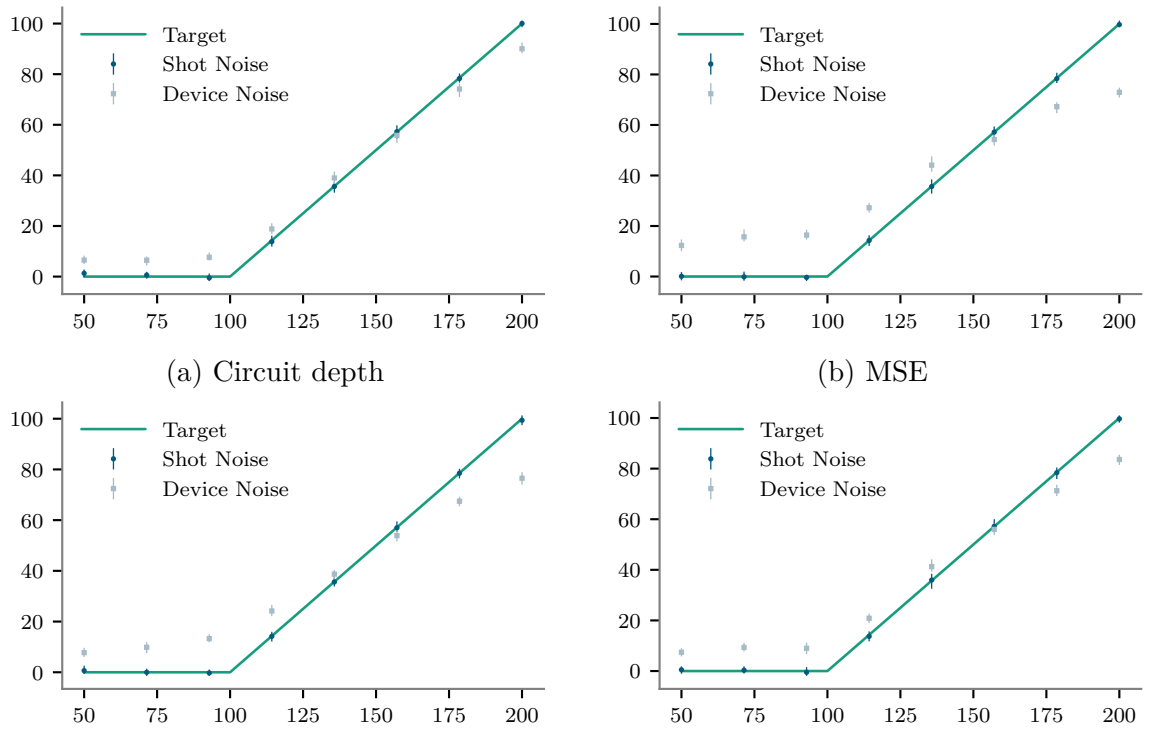
(a) Original circuit.



(b) Controlled circuit.

Figure 4.1: Circuit of the model with the lowest depth, generated by the circuit-based version of algorithm 3.1 for the European call, found in example 3.11. The original circuit (4.1a) and the controlled version (4.1b) it is transformed into. Note that for implementation reasons the target and control register are in the opposite order as presented in section 2.4.

These controlled circuits are now used differently than the original PQCs. The input no longer comes into the circuit via the parameterized input gates, but by the control register. We can still get predictions for individual input points, but only for the finitely many points on the discretization induced by size of the control register. In figure 4.2 we see the predictions for the discretization points together with the influence of noise. We see, that while the model with the best MSE performs very good as long as shot noise is the only noise, the model with the lowest circuit depth performs a lot better once device noise is simulated.



(a) Circuit depth

(b) MSE

(c) Two-qubit gates

(d) Input gates

Figure 4.2: Influence of shot and device noise on the approximation quality for the models optimal regarding the four fitness criteria. The circuit is evaluated for every possible basis state of the discretization register 1000 times, the point represents the average and the vertical bar the complete range of results. Each evaluation itself uses 1000 shots. All calculations are done on a simulator, for the shot noise the `StatevectorSimulator` from the qiskit-aer library is used and for the device noise the `GenericBackendV2` from the qiskit library with its default noise model [39].

In the next step we use theorem 2.36 to calculate option prices. This theorem states, that given a PQC **UR**, that implements the payoff of an option, and a discrete distribution $\mathcal{P}$, that represents the distribution of the underlying, $f_{\mathbf{C}^{\mathbf{UR}}_{\mathcal{D}},\mathbf{M},\mathcal{P}} \colon \mathbb{R}^P \to \mathbb{R}$ represents the non-discounted expected payoff of the option. After discounting, we get the option's fair price.

As a demonstration we will calculate the price of a European call in the Black-Scholes model as introduced in example 1.4. First, we have to approximate this continuous distribution with $\mathbb{R}_{\geq 0}$ as support by a discrete distribution over a finite range. We

have two choices to make, first, which discrete points do we select, and second, which probabilities do we assign to them. We use equidistant points, which leaves only the minimal and maximal value open. It is crucial, that this range is completely covered by the payoff function we have learned in the previous step. In this specific example we choose as minimal value 50 and as maximal value 200, the number of discretization points depends on the number of discretization qubits n.

Next we have to assign probabilities to the discrete points. For ease of implementation we have used the `LogNormalDistribution` class from the qiskit-finance library [39]. With the probability density function of the logarithm of the underlying's price f_{s_t}, and the discrete points $\left(x_j\right)_{0\leq j<2^n}$ the library's authors define the probabilities as

$$p_j := \frac{f_{s_t}\left(\log\left(x_j\right)\right)}{\sum_{k=0}^{2^n-1} p_k}.$$

Alternatively one could define the probabilities as

$$p'_j := \begin{cases} \mathbb{P}\left[S_t \leq x_j\right] & \text{for} \quad j = 0 \\ \mathbb{P}\left[x_{j-1} < S_t \leq x_j\right] & \text{for} \quad 0 < j < 2^n - 1 \\ \mathbb{P}\left[S_t > x_{j-1}\right] & \text{for} \quad j = 2^n - 1 \end{cases}.$$

There are even more possibilities how this can be done, see for example the work by Chakraborty [17] for an overview.

Once the discretization is calculated, the discretization register has to be prepared in a state with amplitudes $\sqrt{p_j}$. The resulting circuit needs to be evaluated, either by repeated execution or by AE and the result discounted to get the option's price.

There are different sources of error in this approximation. The payoff is only approximated by the PQC, the distribution of the underlying is cut off and discretized, and finally, the quantum computer introduces shot noise and device noise.

How much this error sources effect the result and how well this approach compares to other approaches, will be discussed in section 4.3.

4.2 Option Pricing with the Quantum Fourier Transform

In this section we will look at pricing options from a different perspective. We calculate the expectation in equation (1.1) by interpreting it as an integral and approximate that integral numerically. We will follow an idea by Carr and Madan [15] to efficiently do that approximation for a broad class of models for the underlying. For this approach to work we need an analytic expression of the characteristic function of the logarithm of the underlying.

Definition 4.1 Characteristic Function

Given a random variable X we call

$$\Phi_X(u) := \mathbb{E}[e^{\mathbf{i}uX}]$$

the characteristic function of X. If X has a density f_X we can write

$$\Phi_X(u) = \int_{-\infty}^{\infty} e^{\mathbf{i}us} f_X(s) \mathrm{d}s. \tag{4.1}$$

It is often beneficial to work with the logarithm of prices, whenever we do so, we use a small letter instead of a capital one. A small letter with a subscript represents the logarithm of the value represented by the capital letter with the same subscript. For the underlying we write $s_t := \ln(S_t)$. We assume that we know Φ_{s_T}, the characteristic function of the logarithm of the underlying at maturity under the risk-neutral measure. This assumption is valid for some important models such as the Black-Scholes model or the more complex Heston model. The Black-Scholes model assumes a log-normal distribution, i.e., the logarithm of the stock price is normally distributed. For the normal distribution the characteristic function is well known. For the Heston model see for example the work by Rouah [56] for the logarithms characteristic function.

This approach is tailored specifically to European calls. We will call the with strike Z and its logarithm z. A call pays out the difference between the value of the underlying and the strike at the time of maturity, as long as this difference is positive, i.e.,

$$\Pi(S_T) = \max\{0, S_T - Z\}. \tag{4.2}$$

We will denote the dependency of the option price on the logarithm of the strike by writing $C_T(z)$ for the price. The goal of this section is finding an approximation of this price, that we can actually calculate.

First we plug the payoff function from equation (4.2) into equation (1.1) to get

$$C_T(z) = e^{-rT}\mathbb{E}[\Pi(S_T)] = e^{-rT}\mathbb{E}[\max\{0, S_T - Z\}]. \tag{4.3}$$

If we now write the expectation as an integral over the density f_{s_T} of the logarithm of the underlying we get

$$C_T(z) = e^{-rT} \int_z^{\infty} (e^s - e^z) f_{s_T}(s) \mathrm{d}s. \tag{4.4}$$

The positive part of the payoff is incorporated by only integrating starting from z.

Note that we not necessarily know f_{s_T}, actually this approach brings a benefit only if we do not know it. We assumed in the beginning to know the characteristic function of s_T, thus, we try to express equation (4.4) in terms of that characteristic function instead of the unknown density.

4.2.1 Analytic transformation

To express the price in terms of the characteristic function, we apply the Fourier Transform (FT), do some simplifications in the transformed space, and transform back. We need absolute integrability to apply the FT. Unfortunately $C_T(z)$ is not absolutely integrable, i.e., $\int_{-\infty}^{\infty} |C_T(z)|\mathrm{d}z = \infty$, as it is not vanishing for z going to negative infinity. To fix this we introduce a dampening factor $\alpha > 0$ and write

$$c_T(z) := \mathrm{e}^{\alpha z} C_T(z) = \mathrm{e}^{\alpha z}\mathrm{e}^{-rT} \int_z^{\infty} (\mathrm{e}^s - \mathrm{e}^z) f_{s_T}(s)\mathrm{d}s. \tag{4.5}$$

Note that although we call it a dampening factor, it only makes the expression smaller for negative z. Now we look at the FT of $c_T(z)$

$$\psi_T(v) = \int_{-\infty}^{\infty} \mathrm{e}^{\mathbf{i}vz} c_T(z)\mathrm{d}z, \tag{4.6}$$

and plug in equation (4.5) to get

$$\psi_T(v) = \int_{-\infty}^{\infty}\int_z^{\infty} \mathrm{e}^{\mathbf{i}vz}\mathrm{e}^{\alpha z}\mathrm{e}^{-rT}(\mathrm{e}^s - \mathrm{e}^z) f_{s_T}(s)\mathrm{d}s\mathrm{d}z.$$

Now we want to apply the Fubini-Tonelli theorem to switch the order of integration. To do so we first have to make sure, that the inner range for the integral does not depend on the outer integration variable. We achieve this by introducing an indicator function

$$\psi_T(v) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \mathbb{1}_{[z,\infty]}(s)\mathrm{e}^{\mathbf{i}vz}\mathrm{e}^{\alpha z}\mathrm{e}^{-rT}(\mathrm{e}^s - \mathrm{e}^z) f_{s_T}(s)\mathrm{d}s\mathrm{d}z.$$

To make sure Fubini-Tonelli is applicable note that

$$\begin{aligned}
&\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \left|\mathbb{1}_{[z,\infty]}(s)\mathrm{e}^{\mathbf{i}vz}\mathrm{e}^{\alpha z}\mathrm{e}^{-rT}(\mathrm{e}^s - \mathrm{e}^z) f_{s_T}(s)\right|\mathrm{d}z\mathrm{d}s \\
&\quad = \mathrm{e}^{-rT}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \mathbb{1}_{[z,\infty]}(s)\mathrm{e}^{\alpha z}|\mathrm{e}^s - \mathrm{e}^z| f_{s_T}(s)\mathrm{d}z\mathrm{d}s \\
&\quad = \mathrm{e}^{-rT}\int_{-\infty}^{\infty} f_{s_T}(s)\int_{-\infty}^{\infty} \mathbb{1}_{[-\infty,s]}(z)\mathrm{e}^{\alpha z}|\mathrm{e}^s - \mathrm{e}^z|\mathrm{d}z\mathrm{d}s \\
&\quad = \mathrm{e}^{-rT}\int_{-\infty}^{\infty} f_{s_T}(s)\int_{-\infty}^{s} \mathrm{e}^{\alpha z}(\mathrm{e}^s - \mathrm{e}^z)\mathrm{d}z\mathrm{d}s \\
&\quad = \mathrm{e}^{-rT}\int_{-\infty}^{\infty} f_{s_T}(s)\int_{-\infty}^{s} \left(\mathrm{e}^{\alpha z+s} - \mathrm{e}^{(\alpha+1)z}\right)\mathrm{d}z\mathrm{d}s.
\end{aligned}$$

With

$$\frac{\mathrm{d}}{\mathrm{d}z}\left(\frac{\mathrm{e}^{\alpha z+s}}{\alpha} - \frac{\mathrm{e}^{(\alpha+1)z}}{\alpha+1}\right) = \mathrm{e}^{\alpha z+s} - \mathrm{e}^{(\alpha+1)z}$$

and

$$\lim_{z\to-\infty}\left(\frac{\mathrm{e}^{\alpha z+s}}{\alpha} - \frac{\mathrm{e}^{(\alpha+1)z}}{\alpha+1}\right) = 0$$

we have

$$\begin{aligned}
&\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\left|\mathbb{1}_{[z,\infty]}(s)\mathrm{e}^{\mathbf{i}vz}\mathrm{e}^{\alpha z}\mathrm{e}^{-rT}(\mathrm{e}^{s}-\mathrm{e}^{z})f_{s_T}(s)\right|\mathrm{d}z\mathrm{d}s\\
&\quad=\mathrm{e}^{-rT}\int_{-\infty}^{\infty}f_{s_T}(s)\left(\frac{\mathrm{e}^{\alpha s+s}}{\alpha}-\frac{\mathrm{e}^{(\alpha+1)s}}{\alpha+1}\right)\mathrm{d}s\\
&\quad=\mathrm{e}^{-rT}\left(\frac{1}{\alpha}-\frac{1}{\alpha+1}\right)\int_{-\infty}^{\infty}f_{s_T}(s)\mathrm{e}^{\alpha s+s}\mathrm{d}s\\
&\quad\overset{(4.1)}{=}\mathrm{e}^{-rT}\frac{1}{\alpha^2+\alpha}\Phi_{s_T}(-\mathbf{i}(\alpha+1)).
\end{aligned}$$

The expression $\Phi_{s_T}(-\mathbf{i}(\alpha+1))$ is closely related to the moment generating function of the underlying and is not generally finite. To be precise, to ensure that $\Phi_{s_T}(-\mathbf{i}(\alpha+1))$ is finite, we require that $\mathbb{E}\left[S_T^{\alpha+1}\right]$ is finite. This gives an upper bound for α depending on the chosen distribution for the underlying. For more details on the connection between moments and the characteristic function, see [47].

Now assuming, α is chosen in a way to allow the application of the Fubini-Tonelli theorem, switching the order of integration yields

$$\begin{aligned}
\psi_T(v)&=\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\mathbb{1}_{[z,\infty]}(s)\mathrm{e}^{\mathbf{i}vz}\mathrm{e}^{\alpha z}\mathrm{e}^{-rT}(\mathrm{e}^{s}-\mathrm{e}^{z})f_{s_T}(s)\mathrm{d}z\mathrm{d}s\\
&=\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\mathbb{1}_{[-\infty,s]}(z)\mathrm{e}^{\mathbf{i}vz}\mathrm{e}^{\alpha z}\mathrm{e}^{-rT}(\mathrm{e}^{s}-\mathrm{e}^{z})f_{s_T}(s)\mathrm{d}z\mathrm{d}s\\
&=\int_{-\infty}^{\infty}\int_{-\infty}^{s}\mathrm{e}^{\mathbf{i}vz}\mathrm{e}^{\alpha z}\mathrm{e}^{-rT}(\mathrm{e}^{s}-\mathrm{e}^{z})f_{s_T}(s)\mathrm{d}z\mathrm{d}s\\
&=\int_{-\infty}^{\infty}\mathrm{e}^{-rT}f_{s_T}(s)\int_{-\infty}^{s}\left(\mathrm{e}^{s}\mathrm{e}^{z(\alpha+\mathbf{i}v)}-\mathrm{e}^{z(1+\alpha+\mathbf{i}v)}\right)\mathrm{d}z\mathrm{d}s.
\end{aligned}$$

Noting that

$$\frac{\mathrm{d}}{\mathrm{d}z}\left(\frac{\mathrm{e}^{s}\mathrm{e}^{z(\alpha+\mathbf{i}v)}}{\alpha+\mathbf{i}v}-\frac{\mathrm{e}^{z(1+\alpha+\mathbf{i}v)}}{1+\alpha+\mathbf{i}v}\right)=\mathrm{e}^{s}\mathrm{e}^{z(\alpha+\mathbf{i}v)}-\mathrm{e}^{z(1+\alpha+\mathbf{i}v)}$$

and

$$\lim_{z\to-\infty}\left(\frac{\mathrm{e}^{s}\mathrm{e}^{z(\alpha+\mathbf{i}v)}}{\alpha+\mathbf{i}v}-\frac{\mathrm{e}^{z(1+\alpha+\mathbf{i}v)}}{1+\alpha+\mathbf{i}v}\right)=0$$

we can calculate the inner integral and simplify to

$$\begin{aligned}
\psi_T(v) &= \int_{-\infty}^{\infty} \mathrm{e}^{-rT} f_{s_T}(s) \left(\frac{\mathrm{e}^{s} \mathrm{e}^{s(\alpha+\mathbf{i}v)}}{\alpha+\mathbf{i}v} - \frac{\mathrm{e}^{s(1+\alpha+\mathbf{i}v)}}{1+\alpha+\mathbf{i}v} \right) \mathrm{d}s \\
&= \mathrm{e}^{-rT} \int_{-\infty}^{\infty} f_{s_T}(s) \mathrm{e}^{s(1+\alpha+\mathbf{i}v)} \left(\frac{1}{\alpha+\mathbf{i}v} - \frac{1}{1+\alpha+\mathbf{i}v} \right) \mathrm{d}s \\
&= \mathrm{e}^{-rT} \frac{1+\alpha+\mathbf{i}v-\alpha-\mathbf{i}v}{(\alpha+\mathbf{i}v)(1+\alpha+\mathbf{i}v)} \int_{-\infty}^{\infty} f_{s_T}(s) \mathrm{e}^{s(1+\alpha+\mathbf{i}v)} \mathrm{d}s \\
&= \frac{\mathrm{e}^{-rT}}{(\alpha+\mathbf{i}v)(1+\alpha+\mathbf{i}v)} \int_{-\infty}^{\infty} \mathrm{e}^{s(1+\alpha+\mathbf{i}v)} f_{s_T}(s) \mathrm{d}s \\
&\overset{(4.1)}{=} \frac{\mathrm{e}^{-rT} \Phi_{s_T}\left(\frac{1+\alpha+\mathbf{i}v}{\mathbf{i}}\right)}{(\alpha+\mathbf{i}v)(1+\alpha+\mathbf{i}v)}.
\end{aligned}$$

With that we now derive an expression for the option's price by applying the inverse FT to $c_T(z)$ and reverse the dampening factor

$$C_T(z) = \mathrm{e}^{-\alpha z} c_T(z) = \frac{\mathrm{e}^{-\alpha z}}{2\pi} \int_{-\infty}^{\infty} \mathrm{e}^{-\mathbf{i}vz} \psi_T(v) \mathrm{d}v. \tag{4.7}$$

At this point we derived a formula for the option price that no longer depends on the density of the underlying but the characteristic function instead.

4.2.2 Numerical Approximation

The next step is calculating the integral in equation (4.7). We want to do this by approximating it numerically. In this section we will present two different ways to do this. First the discretization that was presented by Carr and Madan [15] and afterwards one that is more fitting for a quantum device.

Both of them bring the discretization in a form that allows to calculate prices for multiple strikes at once using the DFT.

Version 1

We start this version by splitting the integral into its positive and negative part. For the negative part we invert the sign by substitution and get

$$C_T(z) = \frac{\mathrm{e}^{-\alpha z}}{2\pi} \left(\int_0^{\infty} \mathrm{e}^{-\mathbf{i}vz} \psi_T(v) \mathrm{d}v + \int_0^{\infty} \mathrm{e}^{\mathbf{i}vz} \psi_T(-v) \mathrm{d}v \right).$$

Since $C_T(z)$ is a real-valued function and therefore, also is $c_T(z)$, we know by the symmetry of the FT, that $\psi(v)$ is even symmetric. This gives us

$$C_T(z) = \frac{\mathrm{e}^{-\alpha z}}{2\pi} \left(\int_0^{\infty} \mathrm{e}^{-\mathbf{i}vz} \psi_T(v) \mathrm{d}v + \int_0^{\infty} \overline{\mathrm{e}^{-\mathbf{i}vz} \psi_T(v)} \mathrm{d}v \right).$$

Now we pull the complex conjugate out of the integral, since v is real-valued this is possible, and use the fact that $z + \bar{z} = 2\mathcal{R}[z]$ to get

$$C_T(z) = \frac{\mathrm{e}^{-\alpha z}}{\pi}\mathcal{R}\left[\int_0^\infty \mathrm{e}^{-\mathbf{i}vz}\psi_T(v)\mathrm{d}v\right]. \tag{4.8}$$

The next step is numerically approximating this integral. By choosing a step size for the discretization $\Delta_v^{(1)} \in \mathbb{R}_{\geq 0}$ and number of discretization steps $N^{(1)} \in \mathbb{N}_{\geq 0}$ we get

$$C_T(z) \approx \frac{\mathrm{e}^{-\alpha z}}{\pi}\Delta_v^{(1)}\mathcal{R}\left[\sum_{j=0}^{N^{(1)}-1} \mathrm{e}^{-\mathbf{i}\Delta_v^{(1)}jz}\psi_T\left(\Delta_v^{(1)}j\right)\right]. \tag{4.9}$$

This discretization introduces two sources of error, firstly the integral is cut off at $N^{(1)}\Delta_v^{(1)}$. Secondly the integrand is approximated by a step function. The latter can be mitigated by choosing a more involved quadrature rule, like the trapezoidal rule. As this is not the focus of this work we stick to the most simple quadrature.

In the next step we also discretize the log-strike, $z_k = z_0 + \Delta_z^{(1)}k$ for $0 \leq k < N^{(1)}$, to be able to calculate many option prices in one run. Here we can choose the smallest possible value for the strike $z_0 \in \mathbb{R}_{\geq 0}$ and again the step size $\Delta_z^{(1)} \in \mathbb{R}_{\geq 0}$. One possible choice for z_0 would be $z_0 = s_0 - {}^{N^{(1)}\Delta_z^{(1)}}/_2$. This centers the log strikes around the initial price of the underlying. Note that this gives equidistant log-strikes which leads to non-equidistant strikes. This especially leads to very small and very large strikes. With this the option price gets approximated as

$$\begin{aligned} C_T(z_k) &\approx \frac{\mathrm{e}^{-\alpha\left(z_0+\Delta_z^{(1)}k\right)}}{\pi}\Delta_v^{(1)}\mathcal{R}\left[\sum_{j=0}^{N^{(1)}-1} \mathrm{e}^{-\mathbf{i}\Delta_v^{(1)}j\left(z_0+\Delta_z^{(1)}k\right)}\psi_T\left(\Delta_v^{(1)}j\right)\right] \\ &= \frac{\mathrm{e}^{-\alpha\left(z_0+\Delta_z^{(1)}k\right)}}{\pi}\Delta_v^{(1)}\mathcal{R}\left[\sum_{j=0}^{N^{(1)}-1} \mathrm{e}^{-\mathbf{i}\Delta_v^{(1)}\Delta_z^{(1)}ju}\mathrm{e}^{-\mathbf{i}\Delta_v^{(1)}jz_0}\psi_T\left(\Delta_v^{(1)}j\right)\right]. \end{aligned}$$

To bring the summands into the right form for the DFT we fix the relation between the two discretizations to

$$\Delta_v^{(1)}\Delta_z^{(1)} = \frac{2\pi}{N^{(1)}} \tag{4.10}$$

and get

$$\begin{aligned} C_T(z_k) &\approx \frac{\mathrm{e}^{-\alpha\left(z_0+\Delta_z^{(1)}k\right)}}{\pi}\Delta_v^{(1)}\mathcal{R}\left[\sum_{j=0}^{N^{(1)}-1} \mathrm{e}^{-\mathbf{i}\frac{2\pi}{N^{(1)}}jk}x_j^{(1)}\right] \\ x_j^{(1)} &= \mathrm{e}^{-\mathbf{i}\Delta_v^{(1)}jz_0}\psi_T\left(\Delta_v^{(1)}j\right). \end{aligned} \tag{4.11}$$

With this we have reached the form of definition 2.1. Carr and Madan [15] calculate this sum by the FFT, a very efficient algorithm to calculate the DFT. See figures 4.3 and 4.4 for a demonstration of this approximation scheme for 5 and 15 qubits respectively.

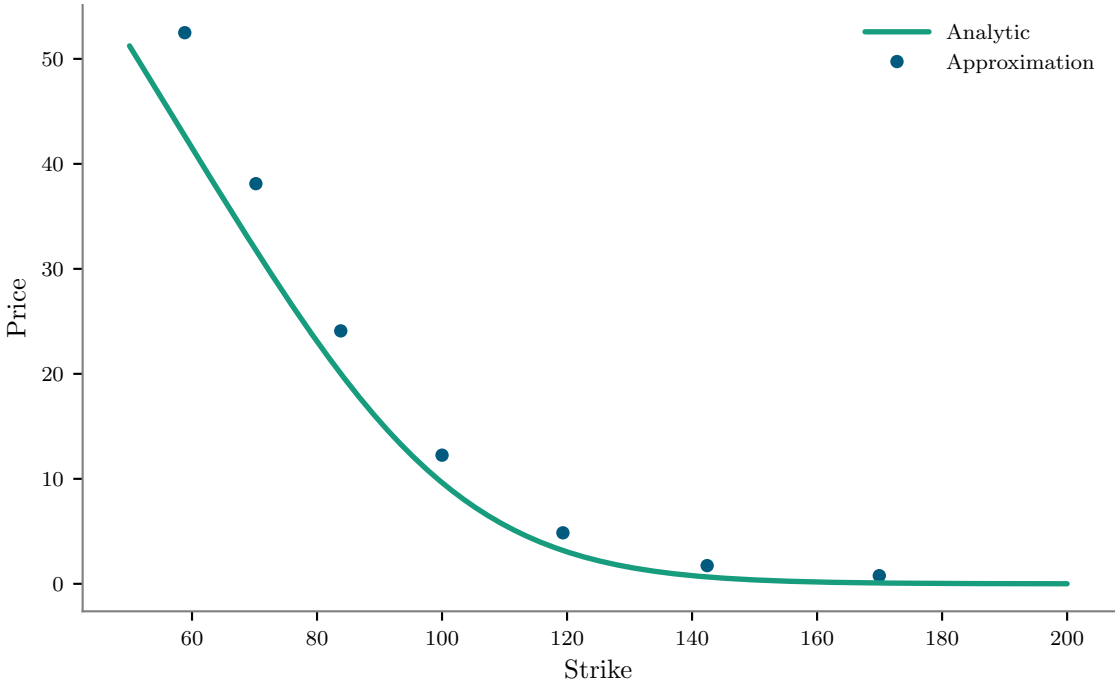


Figure 4.3: Prices of a European call in the Black-Scholes model, see example 1.4, with $2^5 = 32$ discretization steps and version 1 of the discretization from equation (4.11). Only prices for strikes in between 50 and 200 are plotted.

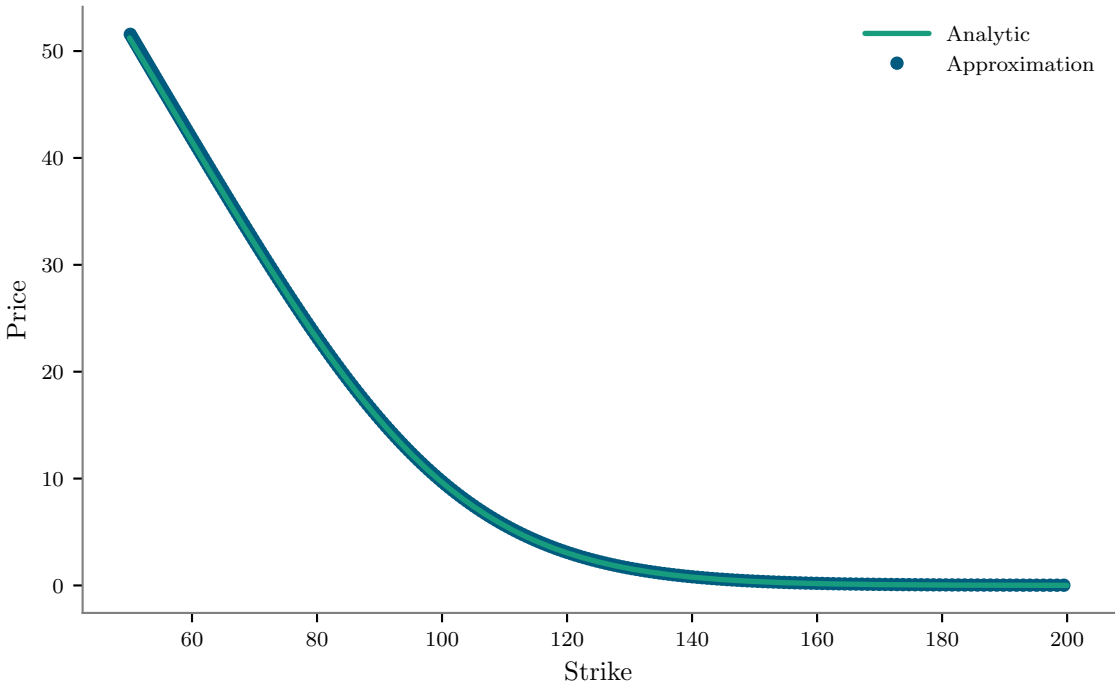


Figure 4.4: Prices of a European call in the Black-Scholes model, see example 1.4, with $2^{15} = 32768$ discretization steps and version 1 of the discretization from equation (4.11). Only prices for strikes in between 50 and 200 are plotted.

Version 2

For reasons explained in section 4.2.4 the real part in equation (4.11) is a problem when switching to a quantum computer. To get rid of the problem we go back to equation (4.7) and discretize without splitting the integral first. Like before we choose a step size $\Delta_v^{(2)} \in \mathbb{R}_{\geq 0}$ and a number of discretization steps $N^{(2)} \in \mathbb{N}_{\geq 0}$. This time we discretize both the positive and the negative part of the integral and get

$$C_T(z) \approx \frac{\mathrm{e}^{-\alpha z}}{2\pi} \sum_{j=-N^{(2)}}^{N^{(2)}-1} \mathrm{e}^{-\mathbf{i}\Delta_v^{(2)} jz} \psi_T\Big(\Delta_v^{(2)} j\Big)\Delta_v^{(2)}. \tag{4.12}$$

Now we bring the sum again in a form suitable for the DFT. First we shift the index to get

$$C_T(z) \approx \frac{\mathrm{e}^{-\alpha z}}{2\pi} \sum_{j=0}^{2N^{(2)}-1} \mathrm{e}^{-\mathbf{i}\Delta_v^{(2)}(j-N^{(2)})z} \psi_T\Big(\Delta_v^{(2)}\big(j-N^{(2)}\big)\Big)\Delta_v^{(2)}.$$

Next we introduce, as before, a discretization of the log-strike $z_k := z_0 + \Delta_z^{(2)} k$ with $0 \leq k < 2N^{(2)}$ while we fix

$$\Delta_z^{(2)}\Delta_v^{(2)} = \frac{\pi}{N^{(2)}}. \tag{4.13}$$

Note, that this is half of the value for version 1. With this we get

$$C_T(z_k) \approx \frac{\mathrm{e}^{-\alpha\left(z_0+\Delta_z^{(2)}k\right)}}{2\pi} \sum_{j=0}^{2N^{(2)}-1} \mathrm{e}^{-\mathbf{i}\Delta_v^{(2)}(j-N^{(2)})\left(z_0+\Delta_z^{(2)}k\right)} \psi_T\Big(\Delta_v^{(2)}\big(j-N^{(2)}\big)\Big)\Delta_v^{(2)}.$$

Note that

$$\begin{aligned}
&-\mathbf{i}\Delta_v^{(2)}\big(j-N^{(2)}\big)\big(z_0+\Delta_z^{(2)}k\big) \\
&\quad = \; -\mathbf{i}\Delta_v^{(2)} j\Delta_z^{(2)} k - \mathbf{i}\Delta_v^{(2)} jz_0 + \mathbf{i}\Delta_v^{(2)} N^{(2)} z_0 + \mathbf{i}\Delta_v^{(2)} N^{(2)}\Delta_z^{(2)} k \\
&\quad \overset{(4.13)}{=} -\mathbf{i}\frac{\pi}{N^{(2)}} jk \qquad - \mathbf{i}\Delta_v^{(2)} jz_0 + \mathbf{i}\Delta_v^{(2)} N^{(2)} z_0 + \mathbf{i}\pi k
\end{aligned}$$

and thus

$$\begin{aligned}
C_T(z_k) &\approx \frac{\mathrm{e}^{-\alpha\left(z_0+\Delta_z^{(2)}k\right)}}{2\pi}\Delta_v^{(2)}\mathrm{e}^{\mathbf{i}\pi k} \sum_{j=0}^{2N^{(2)}-1} \mathrm{e}^{-\mathbf{i}\frac{2\pi}{2N^{(2)}}jk} x_j^{(2)} \\
x_j^{(2)} &= \mathrm{e}^{-\mathbf{i}\Delta_v^{(2)} jz_0}\mathrm{e}^{\mathbf{i}\Delta_v^{(2)} N^{(2)} z_0}\psi_T\Big(\Delta_v^{(2)}\big(j-N^{(2)}\big)\Big).
\end{aligned} \tag{4.14}$$

With this we have again reached the form of definition 2.1. See figures 4.5 and 4.6 for examples for 5 and 15 qubits.

4.2.3 Convergence

In this section we will look at a numerical example to get an intuition on how the two numerical approximations converge to the true value. We do so purely numerical on a

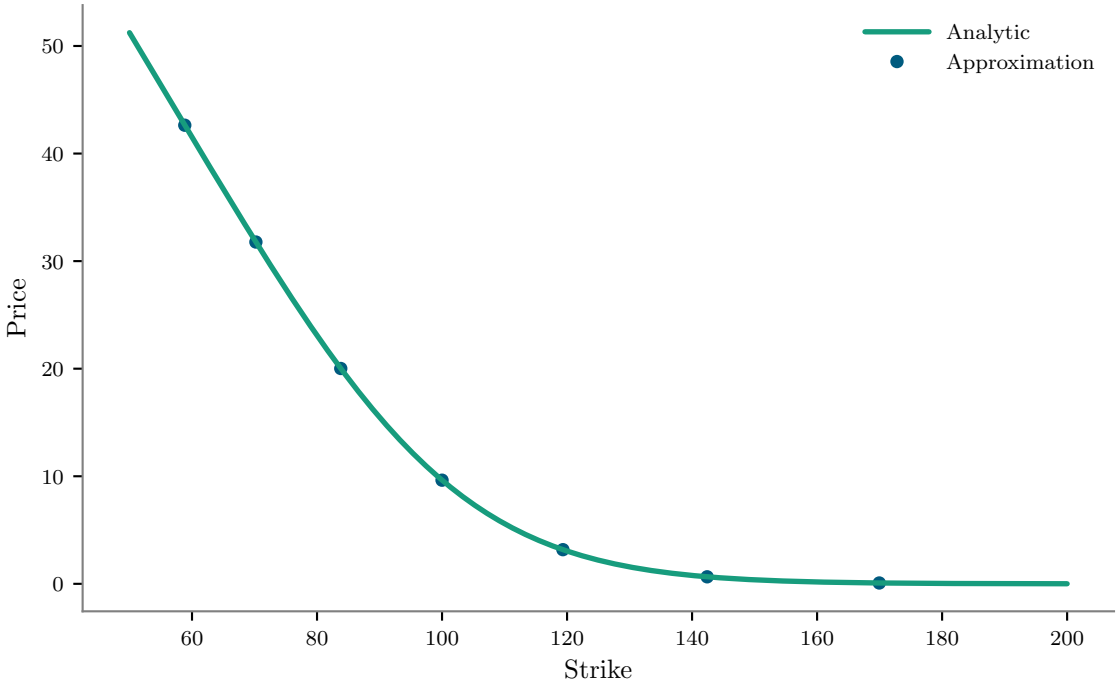


Figure 4.5: Prices of a European call in the Black-Scholes model, see example 1.4 with $2^5 = 32$ discretization steps and version 2 of the discretization from equation (4.14). Only prices for strikes in between 50 and 200 are plotted.

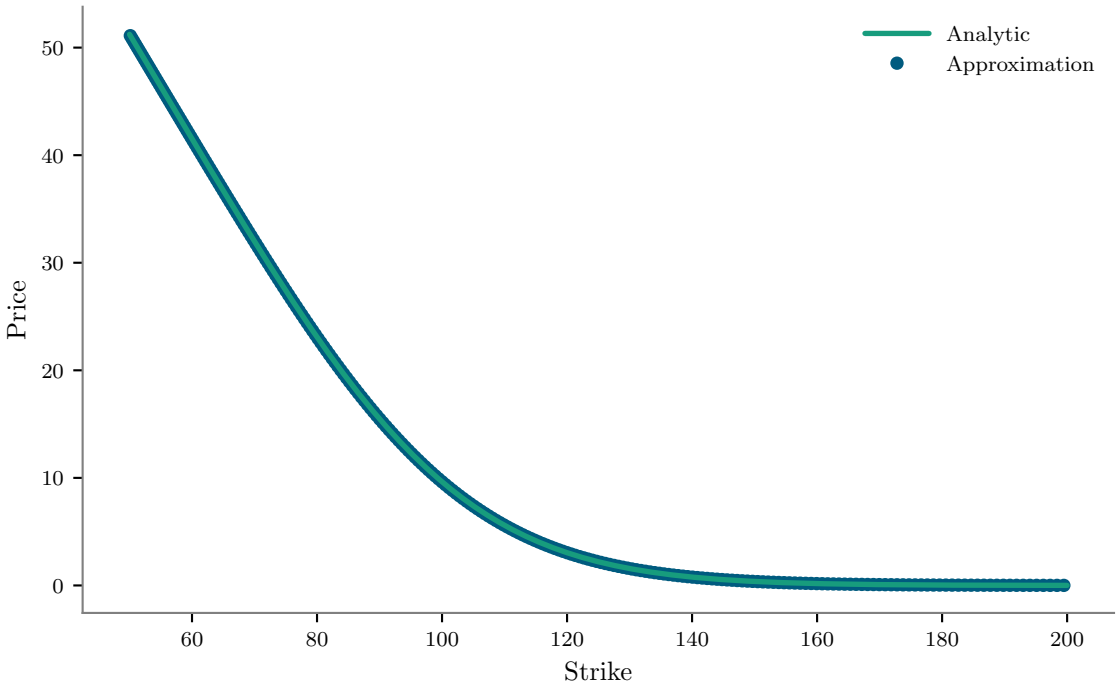


Figure 4.6: Prices of a European call in the Black-Scholes model, see example 1.4 with $2^{15} = 32768$ discretization steps and version 2 of the discretization from equation (4.14). Only prices for strikes in between 50 and 200 are plotted.

classical computer. We will look at equations (4.11) and (4.14) and fix $z_{\min}$ and $z_{\max}$, i.e., the range of strikes, and calculate prices for increasing number of qubits n.

Now let us have a look at the other parameters, that we can choose, starting with version 1. Using n qubits, we have $N^{(1)} = 2^n$ amplitudes available for the discretization. From equation (4.13) we know, that the product of $\Delta_v^{(1)}$ and $\Delta_z^{(1)}$ is fixed for a given $N^{(1)}$, so we have a choice how to balance the two discretizations against each other. Without a reason to have that balance shifted to either direction, we chose $\Delta_z^{(1)} = 1/\sqrt{N^{(1)}}$. From equation (4.13) then follows $\Delta_v^{(1)} = \Delta_z^{(1)} 2\pi/N^{(1)} = 2\pi/\sqrt{N^{(1)}}$. Finally, we decided to have the log-strikes evenly spaced around the starting value, thus, $z_0 = s_0 - \Delta_z^{(1)} N^{(1)}/2$.

For version 2 the main difference is, that we need twice the number of amplitudes for the same discretization, therefore we set $N^{(2)} = 2^{n-1}$. Similar considerations as for version 1 lead to

$$\Delta_z^{(2)} = \frac{1}{\sqrt{N^{(2)}}}, \qquad z_0 = s_0 - \frac{N^{(2)}}{2}\Delta_z^{(2)}, \qquad \Delta_v^{(2)} = \frac{\pi}{N^{(2)}\Delta_z^{(2)}} = \frac{\pi}{\sqrt{N^{(2)}}}.$$

Note that $N^{(1)} = 2N^{(2)}$, and since, for example, the sums in equations (4.11) and (4.14) mainly differ by one running from zero to $N^{(1)} - 1$ and the other until $2N^{(2)} - 1$, we use $N := N^{(1)} = 2N^{(2)}$ in these cases.

We will again use the Black-Scholes model from example 1.4, i.e., a log-normal distributed underlying. This choice has two benefits, first we know the options true price analytically and second, the logarithm of the underlying is normally distributed, i.e., all moments exist, and we have no restrictions on the choice of α.

Figures 4.7 and 4.8 show the MSEs over all calculated prices for strikes between 50 and 200 for increasing number of qubits and different values for α. The first shows the errors for equation (4.9) and the second for equation (4.12). We see that the second version of the discretization converges a lot faster, needing only about seven qubits to reach a point where discretization error seems to be no longer the dominating source of error. The first version reaches an accuracy of two decimal points with 16 qubits.

4.2.4 Option Pricing with the QFT

Now that we have two numerical approximations, this section will be about applying the QFT to calculate actual option prices for the two discretizations from section 4.2.2.

To use a quantum computer to calculate the sums from equations (4.11) and (4.14) we start with preparing the states $\left|x^{(l)}\right\rangle = \sum_{j=0}^{N-1} x_j^{(l)} |j\rangle$ for $l = 1, 2$. On the prepared state the inverse QFT is applied to get a final state that contains the calculated sums in its amplitudes. The complicated part is how to get these values from the quantum state. We have limited access to this state. One possibility is state tomography, the reconstruction of the quantum state by multiple measurements in different bases. Unfortunately we would need to measure in 4^n different bases for an n-qubit system [38], which makes this approach scale very badly.

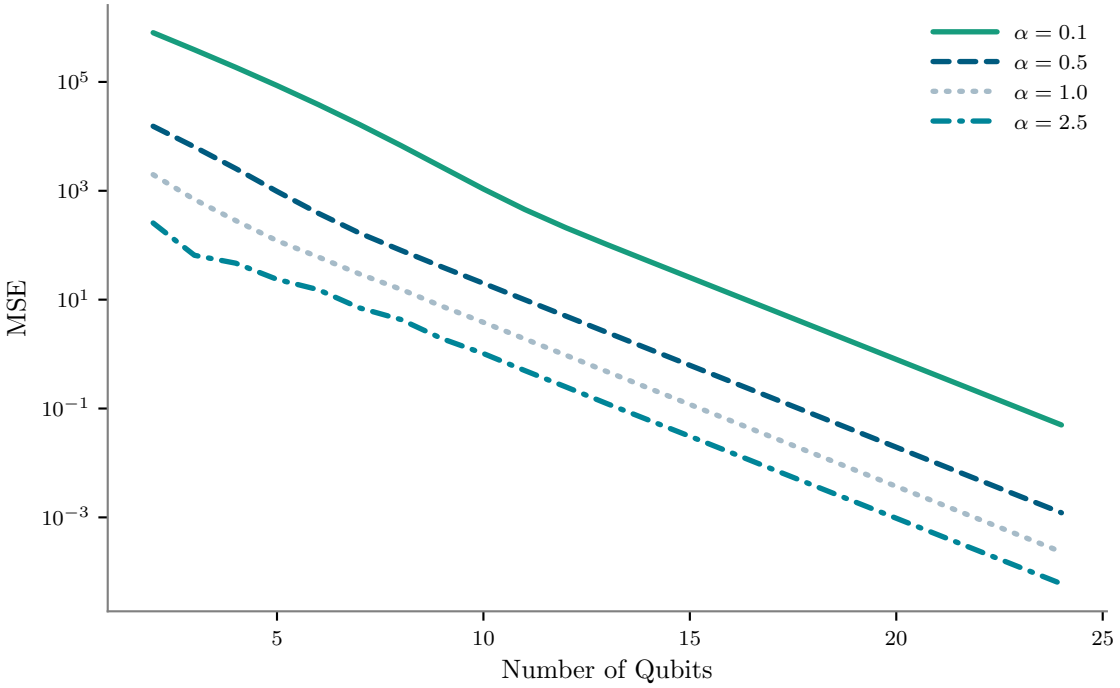


Figure 4.7: Approximation error of European call prices in the Black-Scholes model, see example 1.4, for increasing number of qubits with version 1 of the discretization from equation (4.11).

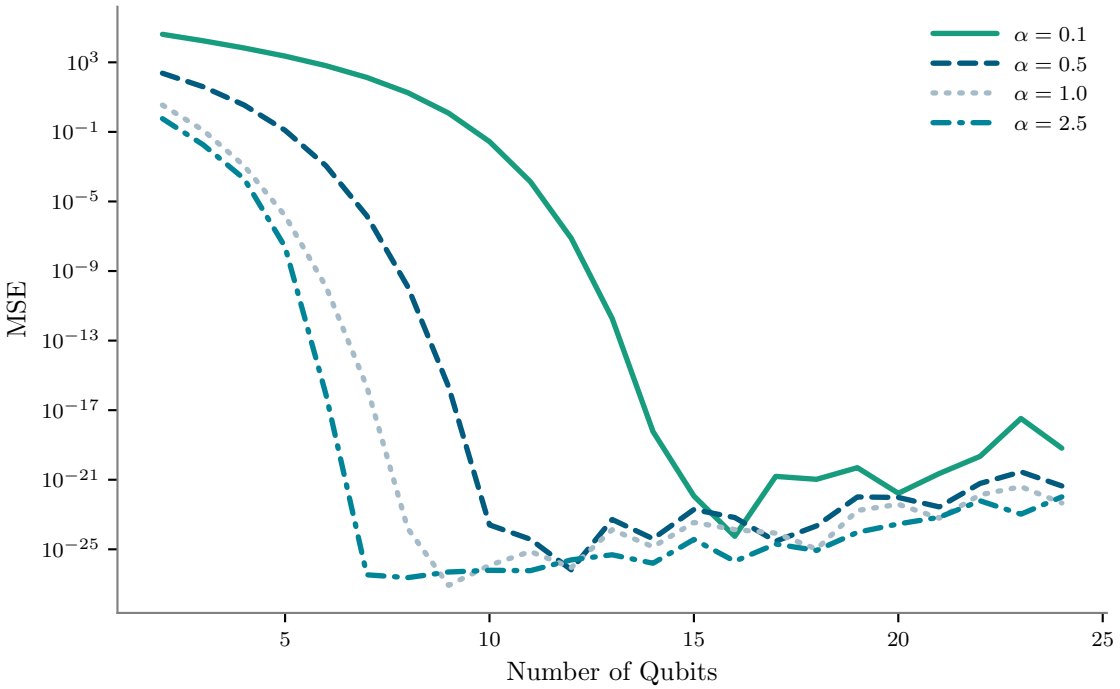


Figure 4.8: Approximation error of European call prices in the Black-Scholes model, see example 1.4, with version 2 of the discretization from equation (4.14). It looks like discretization error stops being the dominant source of error for more than 7 qubits.

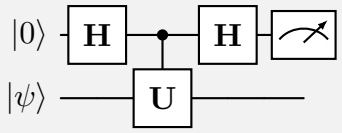
We will try to get the information we need by only repeatedly measuring the final state in the computational basis. This means we observe estimates for the amplitudes of each state, the squared absolute values of the state vector.

When looking at equation (4.11) we see, that we will need the real part of the final state. This is a problem, as we only have access to the amplitudes, i.e., the absolute values, which do not help us in finding the real part of the state vector.

In order to get that we use the so-called Hadamard test, a well known procedure in quantum computing [1].

Definition 4.2 Hadamard Test

Given a quantum algorithm represented by a unitary $\mathbf{U} \in \mathbb{C}^{2^n \times 2^n}$ and a quantum state $|\psi\rangle \in \mathbb{C}^{2^n}$ we call the circuit



the Hadamard test. This circuit can be represented by the unitary

$$\mathbf{HT}(\mathbf{U}) := (\mathbf{I} \otimes \mathbf{H})\Lambda\Big(0, \mathbf{U}^{(1,\dots,n)}\Big)(\mathbf{I} \otimes \mathbf{H}).$$

To describe this more mathematically we use the notations for controlled operations, introduced in example 1.18, and for multi-qubit measurements from definition 1.14.

Proposition 4.3

Given a quantum algorithm represented by a unitary $\mathbf{U} \in \mathbb{C}^{2^n \times 2^n}$ and a quantum state $|\psi\rangle \in \mathbb{C}^{2^n}$ applying the Hadamard test allows us to calculate the real part of $\langle\psi|\mathbf{U}|\psi\rangle$ by

$$\mathcal{R}[\langle\psi|\mathbf{U}|\psi\rangle] = \mathbb{P}\Big[\mathcal{M}_{\{0\}}(|\phi\rangle) = 0\Big] - \mathbb{P}\Big[\mathcal{M}_{\{0\}}(|\phi\rangle) = 1\Big],$$

where

$$|\phi\rangle := \mathbf{HT}(\mathbf{U})(|\psi\rangle \otimes |0\rangle).$$

Proof. We start with the state the Hadamard test circuit produces

$$\begin{aligned}
|\phi\rangle &= (\mathbf{I} \otimes \mathbf{H})\Lambda\Big(0, \mathbf{U}^{(1,\dots,n)}\Big)(\mathbf{I} \otimes \mathbf{H})(|\psi\rangle \otimes |0\rangle) \\
&\overset{(1.6)}{=} \frac{(\mathbf{I} \otimes \mathbf{H})\Lambda\Big(0, \mathbf{U}^{(1,\dots,n)}\Big)(|\psi\rangle \otimes (|0\rangle + |1\rangle))}{\sqrt{2}} \\
&= \frac{(\mathbf{I} \otimes \mathbf{H})\Lambda\Big(0, \mathbf{U}^{(1,\dots,n)}\Big)(|\psi\rangle \otimes |0\rangle + |\psi\rangle \otimes |1\rangle)}{\sqrt{2}}.
\end{aligned}$$

We can now apply the controlled operation, as the state it is applied to is nicely separated into the two cases

$$\begin{aligned}
|\phi\rangle &= \frac{(\mathbf{I}\otimes\mathbf{H})(|\psi\rangle\otimes|0\rangle + \mathbf{U}|\psi\rangle\otimes|1\rangle)}{\sqrt{2}} \\
&\overset{(1.6)}{=} \frac{|\psi\rangle\otimes(|0\rangle+|1\rangle) + \mathbf{U}|\psi\rangle\otimes(|0\rangle-|1\rangle)}{2} \\
&= \frac{(|\psi\rangle+\mathbf{U}|\psi\rangle)\otimes|0\rangle + (|\psi\rangle-\mathbf{U}|\psi\rangle)\otimes|1\rangle}{2} \\
&= \frac{(\mathbf{I}+\mathbf{U})|\psi\rangle\otimes|0\rangle + (\mathbf{I}-\mathbf{U})|\psi\rangle\otimes|1\rangle}{2} \\
&= (|\phi_1\rangle\otimes|0\rangle + |\phi_2\rangle\otimes|1\rangle).
\end{aligned}$$

Where

$$\begin{aligned}
|\phi_1\rangle &= \frac{(\mathbf{I}+\mathbf{U})|\psi\rangle}{2}, \\
|\phi_2\rangle &= \frac{(\mathbf{I}-\mathbf{U})|\psi\rangle}{2}.
\end{aligned}$$

Note that for an arbitrary state $|\varphi\rangle = |\varphi_1\rangle\otimes|0\rangle + |\varphi_2\rangle\otimes|1\rangle$ the probability to measure state $|0\rangle$ in the first qubit while ignoring the other qubits is, following definition 1.14, given as

$$\mathbb{P}\left[\mathcal{M}_{\{0\}}(|\varphi\rangle) = 0\right] = \langle\varphi|(\mathbf{I}\otimes|0\rangle\langle 0|)|\varphi\rangle\,,$$

which can be calculated as

$$\begin{aligned}
\mathbb{P}\left[\mathcal{M}_{\{0\}}(|\varphi\rangle) = 0\right] &= \langle\varphi|(\mathbf{I}\otimes|0\rangle\langle 0|)|\varphi\rangle \\
&= (\langle\varphi_1|\otimes\langle 0| + \langle\varphi_2|\otimes\langle 1|)(\mathbf{I}\otimes|0\rangle\langle 0|)(|\varphi_1\rangle\otimes|0\rangle + |\varphi_2\rangle\otimes|1\rangle) \\
&= (\langle\varphi_1|\otimes\langle 0|)(\mathbf{I}\otimes|0\rangle\langle 0|)(|0\rangle\otimes|\varphi_1\rangle) \\
&\quad + (\langle\varphi_1|\otimes\langle 0|)(\mathbf{I}\otimes|0\rangle\langle 0|)(|1\rangle\otimes|\varphi_2\rangle) \\
&\quad + (\langle\varphi_2|\otimes\langle 1|)(\mathbf{I}\otimes|0\rangle\langle 0|)(|0\rangle\otimes|\varphi_1\rangle) \\
&\quad + (\langle\varphi_2|\otimes\langle 1|)(\mathbf{I}\otimes|0\rangle\langle 0|)(|1\rangle\otimes|\varphi_2\rangle) \\
&= \langle\varphi_1|\varphi_1\rangle\otimes\langle 0|0\rangle\,\langle 0|0\rangle + \langle\varphi_1|\varphi_2\rangle\otimes\langle 0|0\rangle\,\langle 0|1\rangle \\
&\quad + \langle\varphi_2|\varphi_1\rangle\otimes\langle 1|0\rangle\,\langle 0|0\rangle + \langle\varphi_2|\varphi_2\rangle\otimes\langle 1|0\rangle\,\langle 0|1\rangle\,.
\end{aligned}$$

Now we can use that $|0\rangle$ and $|1\rangle$ are orthogonal and that $|0\rangle$ has a norm of one to get to

$$\mathbb{P}\left[\mathcal{M}_{\{0\}}(|\varphi\rangle) = 0\right] = \langle\varphi_1|\varphi_1\rangle\,.$$

With this fact we can calculate the probability to measure the first qubit in state $|0\rangle$ as

$$\begin{aligned}
\mathbb{P}\Big[\mathcal{M}_{\{0\}}(|\psi\rangle) = 0\Big] &= \langle\phi_1|\phi_1\rangle \\
&= \frac{\Big\langle\psi\Big|\Big(\mathbf{I}^\dagger + \mathbf{U}^\dagger\Big)(\mathbf{I}+\mathbf{U})\Big|\psi\Big\rangle}{4} \\
&= \frac{\Big\langle\psi\Big|\Big(\mathbf{I}+\mathbf{U}+\mathbf{U}^\dagger+\mathbf{U}^\dagger\,\mathbf{U}\Big)\Big|\psi\Big\rangle}{4} \\
&= \frac{\Big\langle\psi\Big|\Big(2\,\mathbf{I}+\mathbf{U}+\mathbf{U}^\dagger\Big)\Big|\psi\Big\rangle}{4} \\
&= \frac{2\,\langle\psi|\psi\rangle + \Big\langle\psi\Big|\mathbf{U}\,|\psi\rangle + \langle\psi|\,\mathbf{U}^\dagger\Big|\psi\Big\rangle}{4} \\
&= \frac{2\,\langle\psi|\psi\rangle + \langle\psi|\mathbf{U}|\psi\rangle + \overline{\langle\psi|\mathbf{U}|\psi\rangle}}{4} \\
&= \frac{2 + 2\mathcal{R}[\langle\psi|\mathbf{U}|\psi\rangle]}{4} \\
&= \frac{1 + \mathcal{R}[\langle\psi|\mathbf{U}|\psi\rangle]}{2}.
\end{aligned}$$

Obviously the probability to measure state $|1\rangle$ is just the complementary probability and thus, given as

$$\begin{aligned}
\mathbb{P}\Big[\mathcal{M}_{\{0\}}(|\psi\rangle) = 1\Big] &= 1 - \mathbb{P}\Big[\mathcal{M}_{\{0\}}(|\psi\rangle) = 0\Big] \\
&= 1 - \frac{1 + \mathcal{R}[\langle\psi|\mathbf{U}|\psi\rangle]}{2} \\
&= \frac{1 - \mathcal{R}[\langle\psi|\mathbf{U}|\psi\rangle]}{2}.
\end{aligned}$$

Together this yields the claim

$$\begin{aligned}
\mathbb{P}\Big[\mathcal{M}_{\{0\}}(|\psi\rangle) = 0\Big] - \mathbb{P}\Big[\mathcal{M}_{\{0\}}(|\psi\rangle) = 1\Big] &= \frac{1 + \mathcal{R}[\langle\psi|\mathbf{U}|\psi\rangle]}{2} - \frac{1 - \mathcal{R}[\langle\psi|\mathbf{U}|\psi\rangle]}{2} \\
&= \mathcal{R}[\langle\psi|\mathbf{U}|\psi\rangle].
\end{aligned}$$

□

With this tool we are able to extract the real part we need to calculate equation (4.11). Unfortunately the Hadamard test introduces a lot of controlled operations, which are very expensive and error-prone on actual hardware. This is the motivation for the second discretization, as we only need simple measurements to calculate equation (4.14).

Now we put the parts together to finally calculate option prices. For that lets recall the two different discretizations first

$$\begin{aligned}
C_T(z_k) &\approx \frac{\mathrm{e}^{-\alpha\left(z_0 + \Delta_z^{(1)} k\right)}}{\pi} \Delta_v^{(1)} \mathcal{R}\left[\sum_{j=0}^{N-1} \mathrm{e}^{-\mathbf{i}\frac{2\pi}{N} jk} x_j^{(1)}\right] \\
x_j^{(1)} &= \mathrm{e}^{-\mathbf{i}\Delta_v^{(1)} j z_0} \psi_T\Big(\Delta_v^{(1)} j\Big),
\end{aligned} \tag{4.11 revisited}$$

and

$$C_T(z_k) \approx \frac{\mathrm{e}^{-\alpha\left(z_0+\Delta_z^{(2)}k\right)}}{2\pi}\Delta_v^{(2)}\mathrm{e}^{\mathbf{i}\pi k}\sum_{j=0}^{N-1}\mathrm{e}^{-\mathbf{i}\frac{2\pi}{N}jk}x_j^{(2)} \qquad \text{(4.14 revisited)}$$
$$x_j^{(2)} = \mathrm{e}^{-\mathbf{i}\Delta_v^{(2)}jz_0}\mathrm{e}^{\mathbf{i}\Delta_v^{(2)}N^{(2)}z_0}\psi_T\Big(\Delta_v^{(2)}\big(j-N^{(2)}\big)\Big).$$

The first step is encoding the $x_j^{(l)}$'s into the amplitudes of the quantum system. Since the squares of the absolute amplitudes need to add up to one, we normalize first

$$\tilde{x}_j^{(l)} = \frac{x_j^{(l)}}{\sqrt{\sum_{k=0}^{N-1}\left|x_k^{(l)}\right|^2}}$$

The question on how to efficiently produce this arbitrary state is an open one, which will not be discussed here [64, 71]. Let's call the unitaries, that do the encodings, $\mathbf{E}^{(1)} \in \mathbb{C}^{N\times N}$ and $\mathbf{E}^{(2)} \in \mathbb{C}^{N\times N}$. In formulas this means

$$\mathbf{E}^{(l)}\left|0\right\rangle = \sum_{j=0}^{N-1}\tilde{x}_j^{(l)}\left|j\right\rangle.$$

Next, we apply the inverse QFT and get

$$y^{(l)} = \mathbf{QFT}_{\mathrm{n}}^{-1}\,\mathbf{E}^{(l)}\left|0\right\rangle.$$

Then we have

$$y_k^{(l)} = \frac{1}{\sqrt{N}}\sum_{j=0}^{N-1}\mathrm{e}^{\mathbf{i}\frac{2\pi}{N}kj}\tilde{x}_j^{(l)} = \frac{1}{\sqrt{N}}\frac{1}{\sqrt{\sum_{j=0}^{N-1}\left|x_j^{(l)}\right|^2}}\sum_{j=0}^{N-1}\mathrm{e}^{\mathbf{i}\frac{2\pi}{N}kj}x_j^{(l)}. \qquad (4.15)$$

With

$$\lambda^{(l)} := \frac{1}{\sqrt{N}}\frac{1}{\sqrt{\sum_{j=0}^{N-1}\left|x_j^{(l)}\right|^2}}$$

this simplifies to

$$y_k^{(l)} = \lambda^{(l)}\sum_{j=0}^{N-1}\mathrm{e}^{\mathbf{i}\frac{2\pi}{N}kj}x_j^{(l)}.$$

How we continue from here differs again for the two versions.

Version 1

For the first version we need to get the real part of the prepared state. To achieve this we employ the Hadamard test from definition 4.2. We set $\mathbf{U} = \mathbf{QFT}_{\mathrm{n}}^{-1}\,\mathbf{E}^{(1)}$ and $\left|\psi\right\rangle = \left|0\right\rangle$

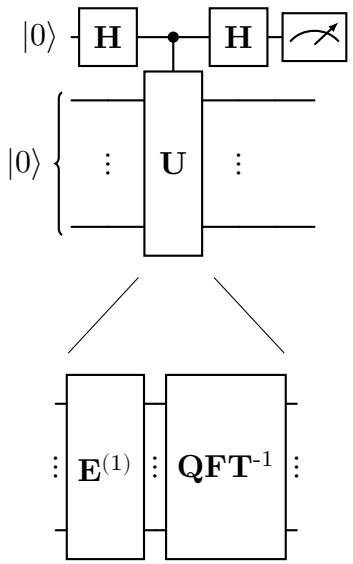


Figure 4.9: Hadamard test to calculate the option price for the first strike.

to get $\mathcal{R}\left[\langle 0|\,\mathbf{QFT}_{\mathrm{n}}^{-1}\,\mathbf{E}^{(1)}\,|0\rangle\right]$. This is the real part of the first entry of the state we are interested in. With that we can calculate the option price for the first strike as

$$C_T(z_0) \approx \frac{\mathrm{e}^{-\alpha z_0}}{\pi}\Delta_v^{(1)}\frac{1}{\lambda^{(1)}}\mathcal{R}\left[\langle 0|\,\mathbf{QFT}_{\mathrm{n}}^{-1}\,\mathbf{E}^{(1)}\,|0\rangle\right].$$

See figure 4.9 for a visualization of the circuit we need to run. Unfortunately, just switching to $|\psi\rangle = |e_k^n\rangle$ for some $0 < k < N$ does not allow us to calculate the option price for the k-th strike. It would give us $\mathcal{R}\left[\langle e_k^n|\,\mathbf{QFT}_{\mathrm{n}}^{-1}\,\mathbf{E}^{(1)}\,|e_k^n\rangle\right]$, which represents the real part of the k-th entry of the state $\mathbf{QFT}_{\mathrm{n}}^{-1}\,\mathbf{E}^{(1)}\,|e_k^n\rangle$. What we want instead is $\mathcal{R}\left[\langle k|\,\mathbf{QFT}_{\mathrm{n}}^{-1}\,\mathbf{E}^{(1)}\,|0\rangle\right]$.

To get that, we need to modify $\mathbf{U}$ and $|\psi\rangle$. We set $|\tilde{\psi}\rangle = |e_k^n\rangle$ and $\tilde{\mathbf{U}} = \mathbf{QFT}_{\mathrm{n}}^{-1}\,\mathbf{E}^{(1)}\mathbf{B}_k$, where the operator $\mathbf{B}_k$ is chosen such that $\mathbf{B}_k\,|e_k^n\rangle = |0\rangle$. This can easily be achieved by a circuit that performs a NOT gate on every qubit, where k has a one in its binary representation. See figure 4.10 for a visualization of the Hadamard test with this newly constructed operator. The result of the Hadamard test is then

$$\mathcal{R}\left[\left\langle\psi\middle|\tilde{\mathbf{U}}\middle|\psi\right\rangle\right] = \mathcal{R}\left[\langle e_k^n|\mathbf{QFT}_{\mathrm{n}}^{-1}\,\mathbf{E}^{(1)}\mathbf{B}_k|e_k^n\rangle\right] = \mathcal{R}\left[\langle e_k^n|\mathbf{QFT}_{\mathrm{n}}^{-1}\,\mathbf{E}^{(1)}|0\rangle\right] = \mathcal{R}\left[y_k^{(1)}\right],$$

exactly what we need.

We then run the circuit from figure 4.10 often enough to get a good estimate for the probability of states $|0\rangle$ and $|1\rangle$ and calculate the option's price as

$$C_T(z_k) \approx \frac{\mathrm{e}^{-\alpha z_0}}{\pi}\Delta_v^{(1)}\frac{1}{\lambda^{(1)}}\left(\mathbb{P}\left[\mathcal{M}_{\{0\}}\left(\mathbf{HT}\left(\tilde{\mathbf{U}}\right)|e_k^n\rangle\right)=0\right]-\mathbb{P}\left[\mathcal{M}_{\{0\}}\left(\mathbf{HT}\left(\tilde{\mathbf{U}}\right)|e_k^n\rangle\right)=1\right]\right).$$

Alternatively the states probabilities could be extracted by AE introduced in section 2.2.3.

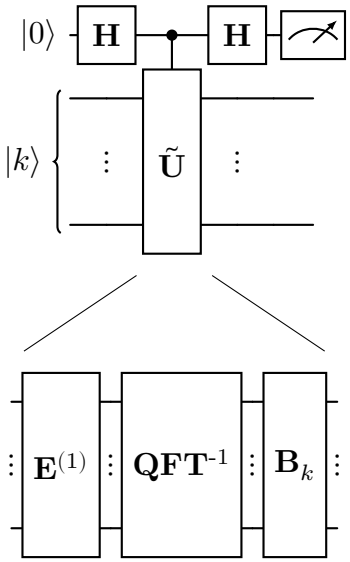


Figure 4.10: Hadamard test to calculate the option price for the k-th.

Version 2

For the second version we *just* need to get the complex values of the state vector. Unfortunately we have no direct access to them, but by equation (4.3) we know, that the price we want to approximate is purely real and positive. So we will accept a slight error by approximating

$$\mathrm{e}^{\mathbf{i}\pi k} y_k^{(2)} \approx \sqrt{\left|y_k^{(2)}\right|} = \sqrt{\mathbb{P}[\mathcal{M}(\mathbf{QFT}_{\mathrm{n}}^{\text{-}1}\, \mathbf{E}^{(2)}\, |0\rangle) = |e_k^n\rangle]}. \tag{4.16}$$

We get each state's probability by repeatedly measuring $\mathbf{QFT}_{\mathrm{n}}^{\text{-}1}\, \mathbf{E}^{(2)}\, |0\rangle$ and noting the relative frequencies of the different states as p_k. Then we can plug equation (4.15) into equation (4.14) and get

$$C_T(z_k) \approx \frac{\mathrm{e}^{-\alpha\left(z_0 + \Delta_z^{(2)} k\right)}}{2\pi} \Delta_v^{(2)} \mathrm{e}^{\mathbf{i}\pi k} \frac{1}{\lambda^{(2)}} y_k^{(2)}. \tag{4.17}$$

With equation (4.16) we get

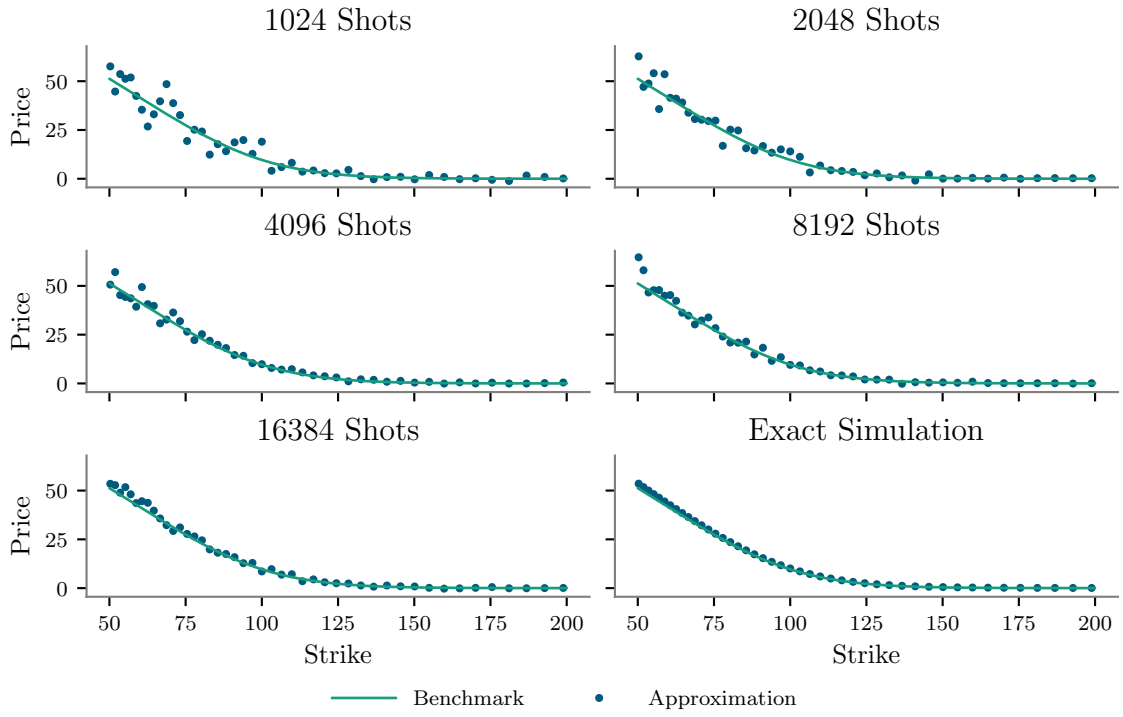
$$C_T(z_k) \approx \frac{\mathrm{e}^{-\alpha\left(z_0 + \Delta_z^{(2)} k\right)}}{2\pi} \Delta_v^{(2)} \frac{1}{\lambda^{(2)}} \sqrt{p_k}. \tag{4.18}$$

This we can calculate using the observed probabilities p_k.

4.2.5 Scaling

We have introduced a way of calculating option prices with a quantum computer. The obvious next question is how performant this approach is and how it compares to the classical approach employing the FFT.

(a) For 7 qubits.



(b) For 10 qubits.

Figure 4.11: The error introduced to the option pricing solely by the shot noise, i.e., experiments performed on an error free quantum computer. The prices are calculated for a European call with strike 100 in the Black-Scholes model from example 1.4 with the discretization from equation (4.11).

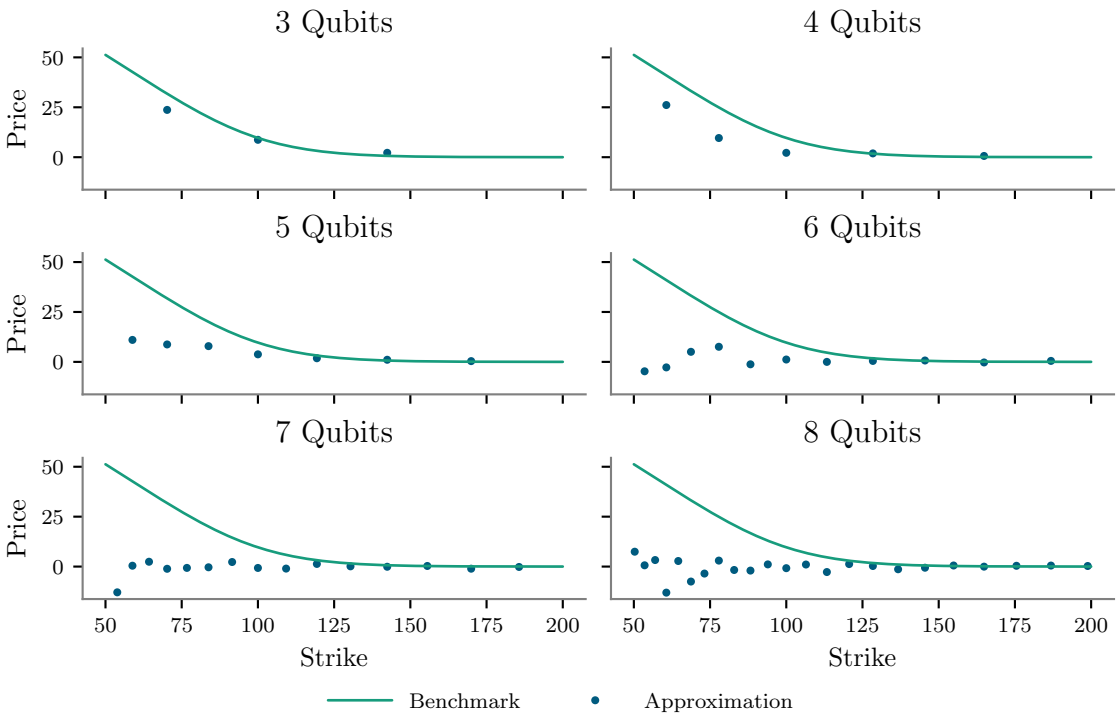


Figure 4.12: The error introduced by using a simulated Backend including device noise on top of shot noise. The prices are calculated for a European call with strike 100 in the Black-Scholes model from example 1.4 for increasing number of qubits and with the discretization from equation (4.11).

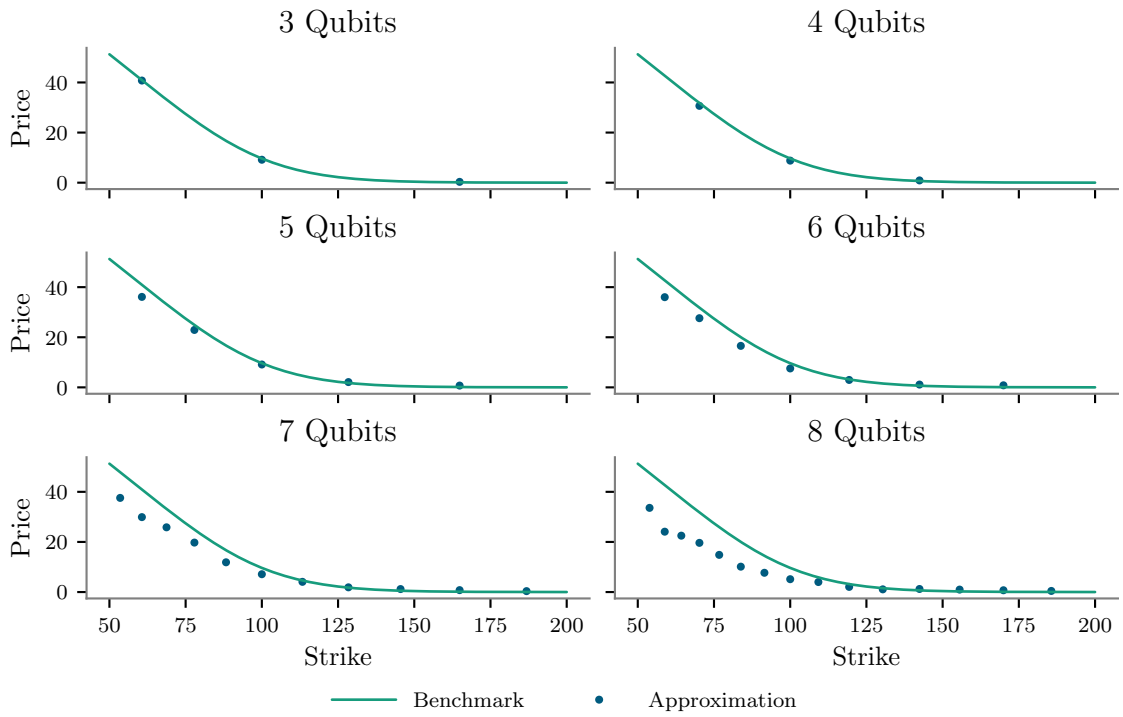


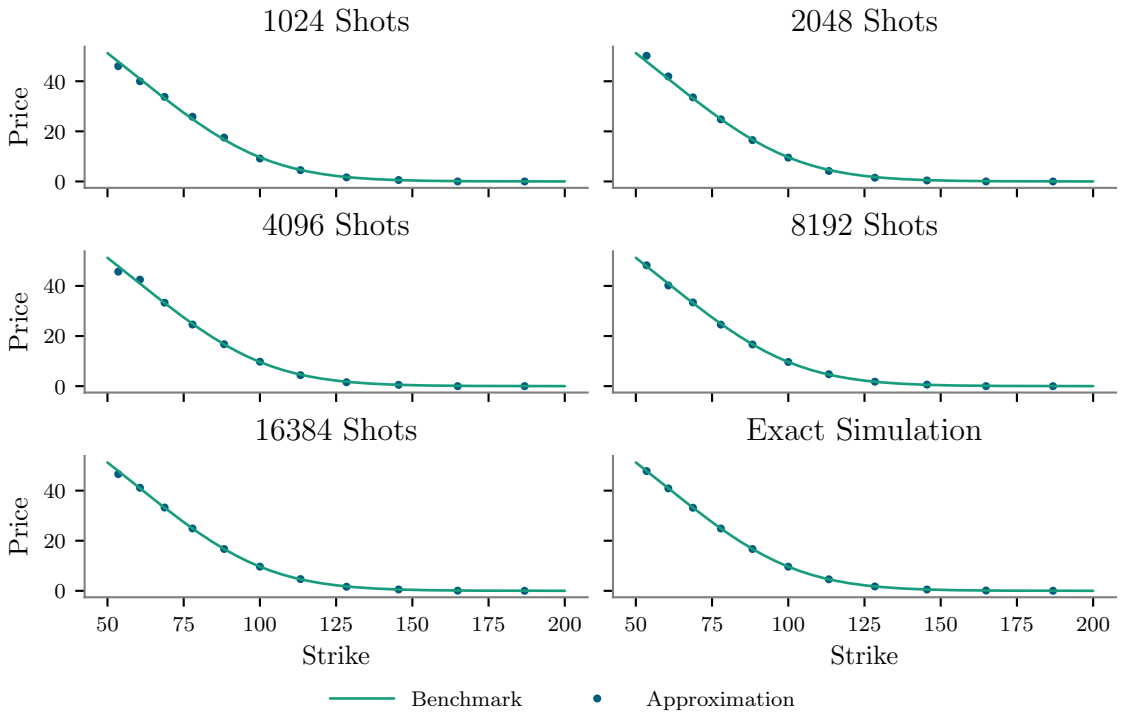
Figure 4.13: The error introduced by using a simulated Backend. The prices are calculated for a European call with strike 100 in the Black-Scholes model from example 1.4 for increasing number of qubits and with the discretization from equation (4.14).

We will analyze the scaling for increasing number of used qubits, translating to an increasing quality of approximation. Classically the algorithm is bottlenecked by the FFT which takes $\mathcal{O}(n2^n)$ steps. For the quantum version things are more complicated and differ for the two versions. The first step is the same for both, which is loading the initial vectors $x^{(l)}$ into the amplitudes of the quantum system. To the author's knowledge the best available algorithm does that with a circuit depth of $\mathcal{O}(2^n)$ [71]. The second step is applying the inverse QFT which adds a depth of $\mathcal{O}(n^2)$ [42]. Here lies the potential for a significant speedup as the $\mathcal{O}(n^2)$ is exponentially faster than $\mathcal{O}(n2^n)$ of the FFT.

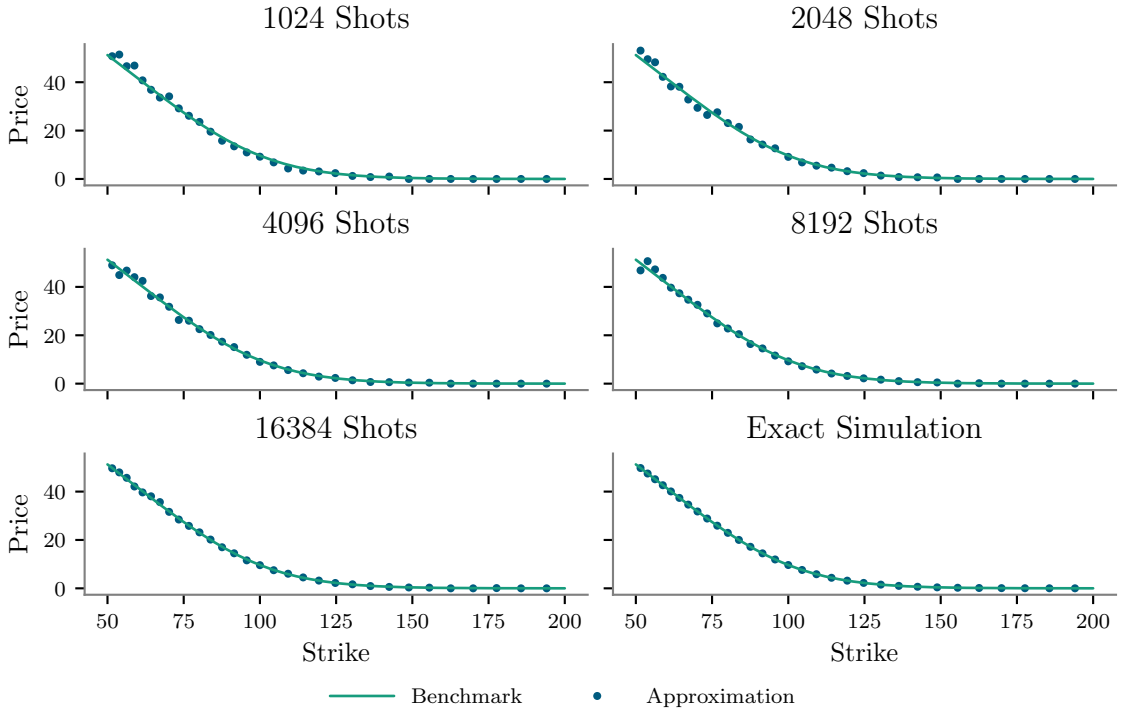
From hereon the two versions differ again.

Version 1

For this version we need the Hadamard test. That only adds a constant to the depth, but greatly increases the number of controlled gates, as all the previous operations need to be controlled. We also have up to $2n$ additional gates, and by the Hadamard test also CNOT gates, for calculating $C_T(z_k)$ for $0 < k < 2^n$. As the Hadamard test only allows to observe a single value, we have to apply the above approach once for every price we want to calculate, 2^n times if we would want to calculate all prices. Further we have to execute the circuit often enough to get good estimates for the probabilities.



(a) For 7 qubits.



(b) For 10 qubits.

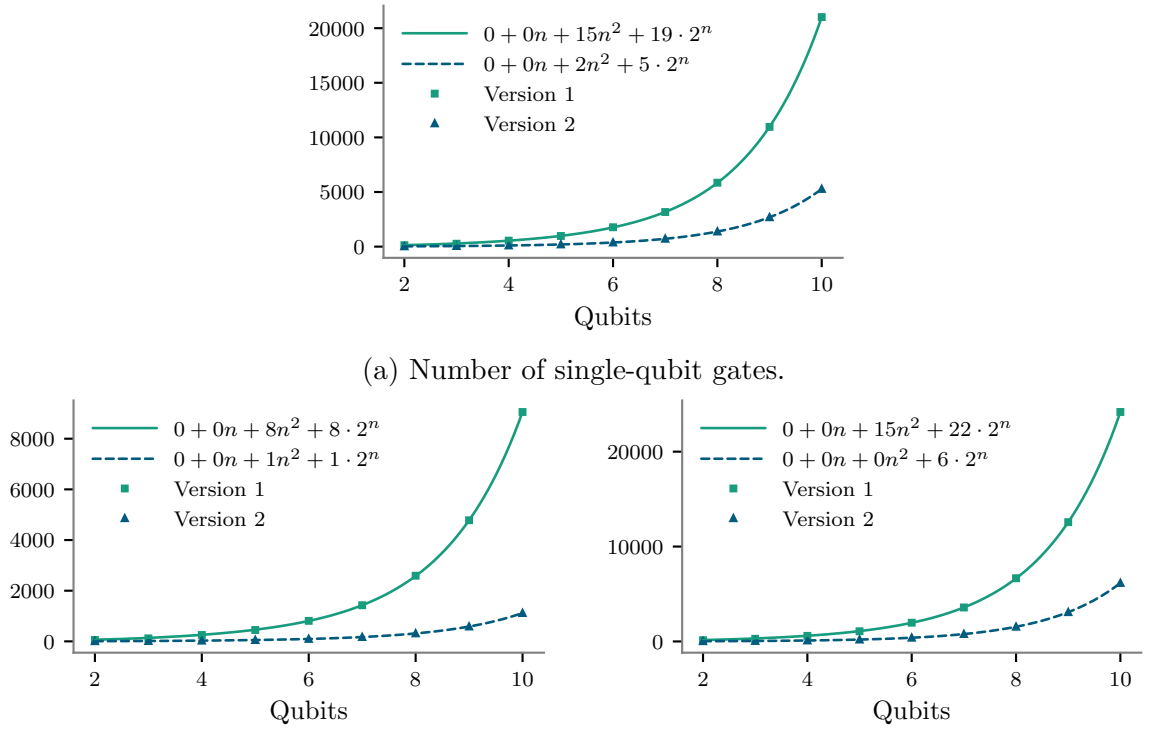
Figure 4.14: The error introduced to the option pricing solely by the shot noise, i.e., experiments performed on an error free quantum computer. The prices are calculated for a European call with strike 100 in the Black-Scholes model from example 1.4 with the discretization from equation (4.14).

Version 2

Here we only need to get estimates for the state's measurement probabilities. This can be done by just running many shots to simultaneous get all prices or by amplitude estimation if only one price is of interest.

Comparison

We performed a numerical study on this scaling, visualized in figure 4.15. We see that all analyzed measures scale, as expected, according to $\mathcal{O}(2^n)$. In all cases, version 2 has worse coefficients.



(a) Number of single-qubit gates.

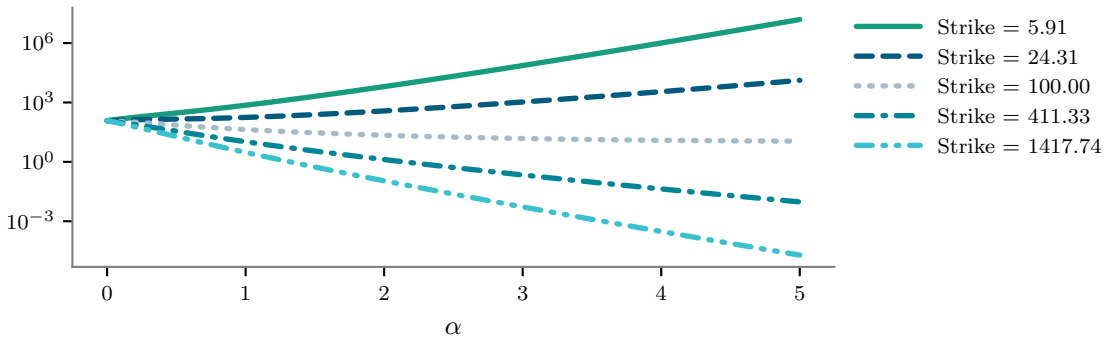
(b) Number of two-qubit gates.

(c) Circuit depth.

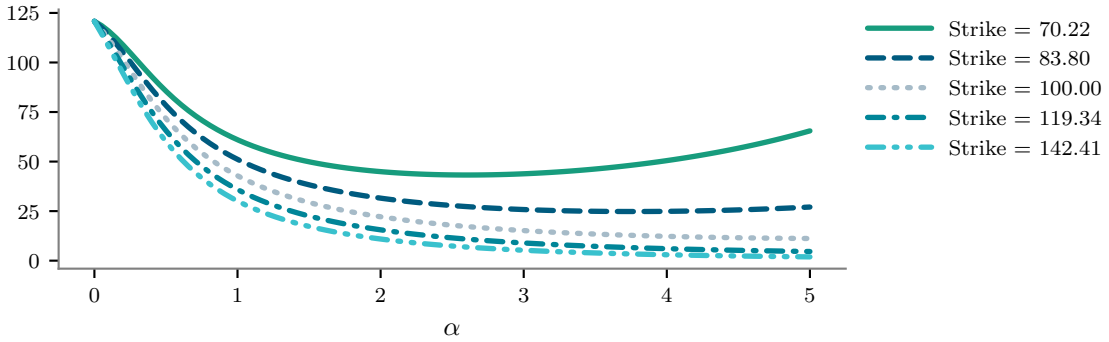
Figure 4.15: Scaling of the two versions of option pricing with the QFT, measured in the number of single-qubit gates (4.15a), number of two-qubit gates (4.15b) and circuit depth (4.2.5). The points are calculated by building the circuits and transpiling them to the gate set used by IBM. The lines are interpolations with a constant part, a linear part, a quadratic part and an exponential part. Version 1 only calculates a single price, while the circuit for version 2 can be used to calculate all prices.

4.2.6 Error Propagation

In this section, we will analyze how an error in the calculations of a quantum computer propagates to an error in the option price. As the previous comparison very clearly showed,



(a) Full range of values for the strike.



(b) Range of the strikes limited to a practical range.

Figure 4.16: The factor between the error done by the quantum device and the actual option price for different strikes and values for α.

that version 2 is superior to version 1, we will limit ourselves to analyzing version 2. We will have another look at equations (4.17) and (4.18) and see how a difference between $\sqrt{p_k}$ and $y_k^{(2)}$, i.e., the quantum computer's error, translates to difference in the calculated prices. Both equations depend linearly on $\sqrt{p_k}$ and $y_k^{(2)}$ respectively with the same prefactor. So if the difference between $\sqrt{p_k}$ and $y_k^{(2)}$ is ϵ, the absolute difference in approximation is

$$\begin{aligned}
\mathcal{E}(\alpha, k) &:= \frac{\mathrm{e}^{-\alpha\left(z_0 + k\Delta_z^{(2)}\right)}}{2\pi} \Delta_v^{(2)} \lambda^{(2)} \epsilon, \\
x_j^{(2)} &= \mathrm{e}^{-\mathbf{i}\Delta_v^{(2)} j z_0} \mathrm{e}^{\mathbf{i}\Delta_v^{(2)} 2^n z_0} \psi_T\left(\Delta_v^{(2)}\left(j - N^{(2)}\right)\right), \\
\psi_T(v) &= \frac{\mathrm{e}^{-rT} \Phi_{s_T}\left(\frac{1+\alpha+\mathbf{i}v}{\mathbf{i}}\right)}{(\alpha + \mathbf{i}v)(1 + \alpha + \mathbf{i}v)}.
\end{aligned} \tag{4.19}$$

This also means, the relative error is the same as the difference between $\sqrt{p_k}$ and $y_k^{(2)}$.

In figure 4.16 we see, that relation between this term and alpha, depends on the strike value we want to calculate the price for. But for strikes closer to the start value, which are the more interesting ones, we see that a value for α between 2 and 3 seems to perform well.

Let us assume we want the option price to be precise up to the cent, i.e., an error below

0.01. The factor lives in an order of magnitude of 100, so we would need the quantum computer to produce results that produce an error lower than 0.0001. Since the scaling factor cancels out for the relative error, this would translate to a relative error of 0.01 %.

4.3 Comparison

In this section we will compare different approaches for pricing options with a quantum computer. We will compare the approximation quality, the needed circuit depth and the possibility of extending the approaches to other models or options.

The first approach is the one introduced by Stamatopoulos et al. [63]. It uses a manually designed circuit that encodes the option price into the amplitude of a target qubit.

The second approach is the one introduced by Wolf, Ewen, and Turkalj [68] and explained in more detail in section 4.1. Here we use a machine learning approach to automatically find a circuit that implements the option's payoff and condition it on an additional register, that implements the distribution of the assets price.

The third approach, introduced by Ewen [26] and explained in more detail in section 4.2, uses the QFT.

4.3.1 Implementation Details

For the numerical comparison of these approaches we use the Black-Scholes model introduced in example 1.4 and price a European call with strike equal to the start value of the underlying, 100. Using this rather simple model and option allows us to calculate the true option price as a benchmark.

Qiskit's implementation

The approach introduced by Stamatopoulos et al. [63] has conveniently been implemented by the python library *Qiskit Finance*, a part of qiskit [39]. For the discretization of the asset price we have used a lower bound of 50 and an upper bound of 200. The only other choice we have is the rescaling factor, for which we chose, by comparing results, 0.001.

CPQC

For the discretization of the asset price we have used the same circuit as for the qiskit version, again with a lower bound of 50 and an upper bound of 200. We use circuits generated by the circuit-based and VAns-based QAS. We compared the performance of different models on the Pareto fronts and chose for the circuit-based version the model with the least input encodings, see figure 3.10d, and for the VAns-based approach the model with the lowest approximation error, see figure 3.9d.

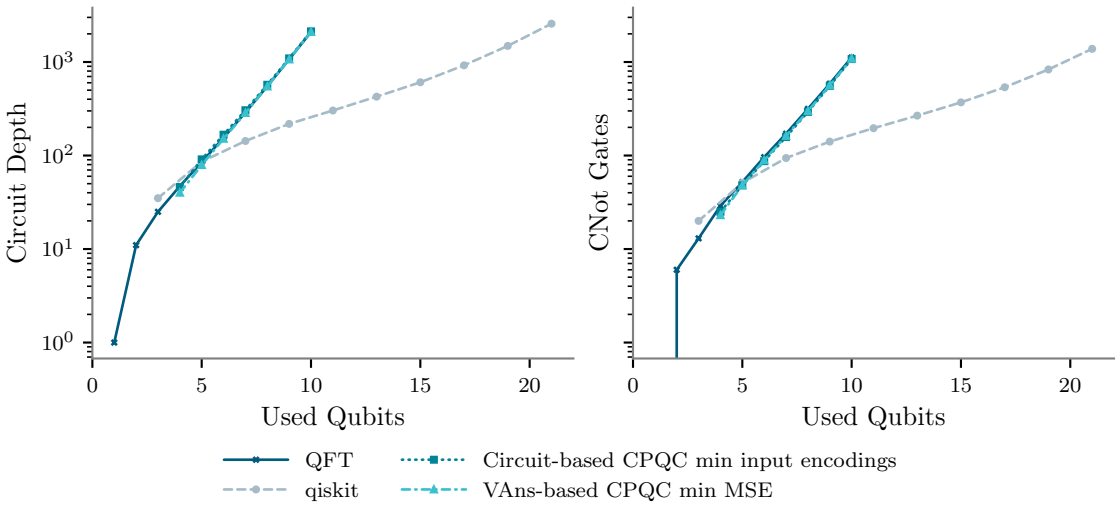


Figure 4.17: The circuit depth and number of two-qubit gates for increasing number of total used qubits for the different approaches to price options with a quantum computer.

QFT

For the QFT based version there is mainly the dampening factor α that needs to be chosen. By the numerical experiments done in section 4.2 we have chosen $\alpha = 2.5$.

4.3.2 Approximation Quality

The interesting question is how to compare the different approaches. One idea would be to look at the scaling of the circuit depth and the number of two-qubit gates for increasing number of used qubits as is often done in the literature to compare quantum algorithms. In figure 4.17 we can see that the QFT based approach and the CPQC based approaches perform similarly and the qiskit based approach has a significantly better scaling.

If we only look at the qubits used for the discretization, see figure 4.18, the picture looks differently. Here the qiskit based approach and the QFT based approach seem to have similar asymptotic behavior, while the CPQC based approach uses significantly more gates.

Although it is common for quantum algorithms to be compared in their scaling for increasing number of qubits, this is not what we are actually interested in. What we actually care for, is the approximation quality of the option price and what circuit complexity we need to achieve these qualities. Looking at figures 4.19 and 4.20 one can clearly see, that the QFT version reaches much better error rates than the other approaches. The other three approaches perform very similarly, with a slight advantage for the VAns-based CPQC version.

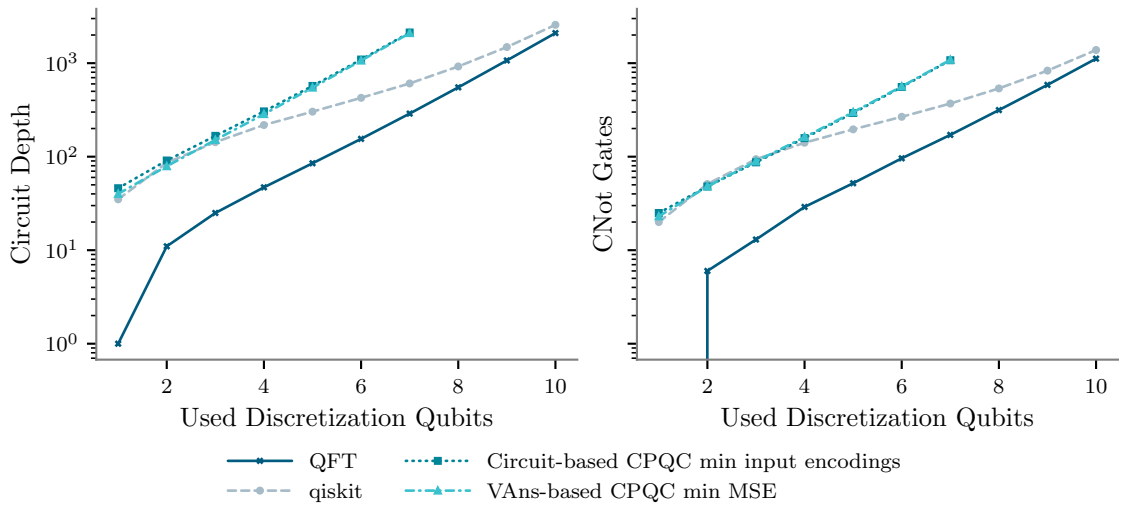


Figure 4.18: The circuit depth and number of two-qubit gates for increasing number of discretization qubits for the different approaches to price options with a quantum computer.

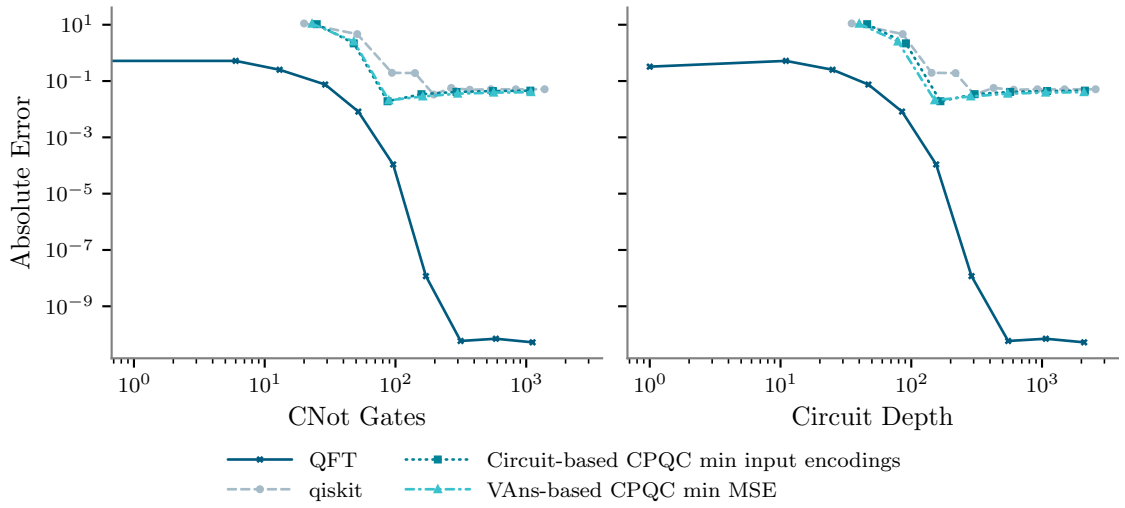


Figure 4.19: The absolute error in the option prices plotted against the number of two-qubit gates and the circuit depth for the different approaches to price options with a quantum computer.

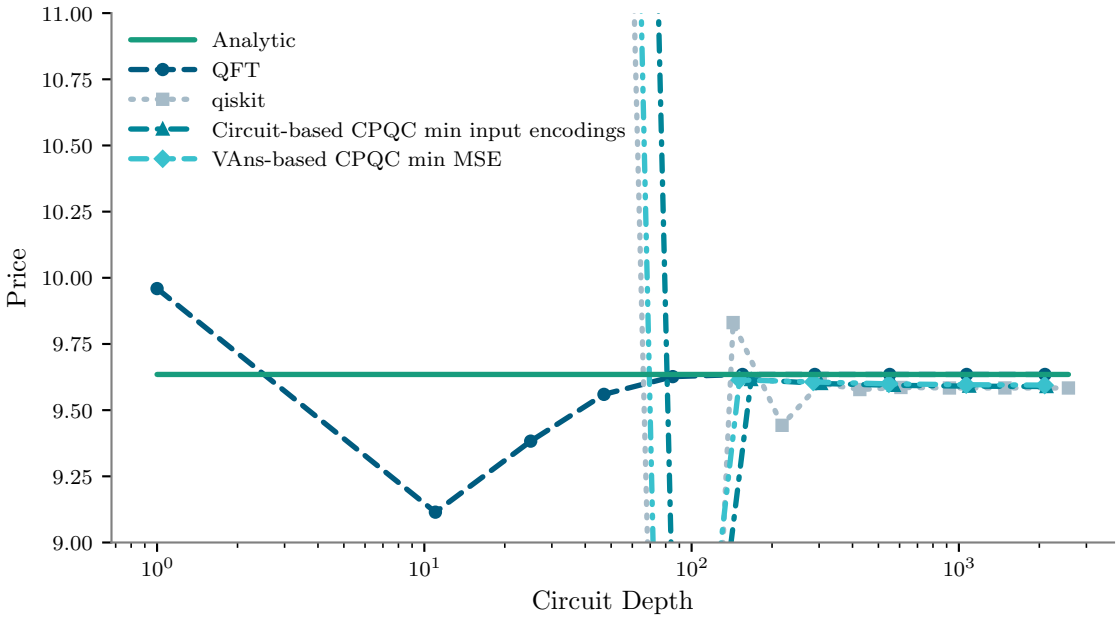


Figure 4.20: The option price plotted against the circuit depth for the different approaches to price options with a quantum computer.

4.3.3 Flexibility and Extensions

For the comparison we have only looked at the very simple example of a European call in the Black-Scholes model. In this short section we will discuss how well the different approaches can be extended to more complex models and options.

The QFT based approach has the disadvantage, that it is specifically designed for European calls. Since the payoff function comes into the formulas at the very beginning, all calculations heavily depend on it. Thus, it is not clear if the same methodology would work for other payoff function. On the other hand this method has the benefit, compared to the other approaches, that prices can be calculated for various strikes at once. It also allows for more complex models for the underlying. The restriction is, that the characteristic function of the logarithm of the underlying must be known analytically and that at least some moments exist.

The approach provided by qiskit implements the payoff function directly, thus, it cannot be directly adapted to other payoff functions. But the basic idea works for all piece wise linear payoff functions. Regarding the underlying, we can use every model, that allows us to calculate a discretized distribution of the underlying's value. This could also be done approximately or even without assuming a specific model by using quantum Generative Adversarial Networks (GANs) [72].

The CPQC based approach is the most flexible one. Regarding the underlying it has the same liberties as the qiskit based approach and, as demonstrated in the examples of chapter 3, can be used for various payoff functions. It is especially easy to extend to multi asset options, as CPQCs allow learning functions of multiple inputs by using one register for every input.

Conclusion and Outlook

The goal of this work was to price options with a quantum computer. We have reached that goal by following two paths, via the QFT and via AE.

The former version is the more direct adaptation of previously known, classical methods. Here we analyzed the effect of replacing the classical FFT by the QFT on the performance, the scalability and the formulas of the method itself. We presented an adaptation of the classical formulas, that improve the performance, on a quantum device, significantly. Regarding the performance, we came to the conclusion, that this method needs basically fault-tolerant quantum computers to deliver acceptable pricing errors. Furthermore, we need an efficient solution to the state preparation task to scale this method to sizes, where classical computers reach their limits. But we have demonstrated that, if these two significant requirements are given, this method can outperform the classical approach.

The latter approach is based on AE and aims at improving classical MC methods. This method needed some more steps. Before this work, the state of the literature was building handcrafted quantum circuits to approximate the payoff profile of simple options. Even in this setting, we are only talking about approximations, as quantum computers are very inefficient in simple arithmetic, and we use trigonometric functions to approximate the piecewise linear payoff profiles of simple options. We put considerable effort into replacing the handcrafting step by a multi-criteria genetic algorithm, that automatically finds good parameterized circuits for a specific problem. The result of this process, often called QAS, is a set of Pareto optimal circuits with different tradeoffs between approximation quality and complexity that implement option payoffs. Another contribution of this work is transforming these circuits into controlled versions, we call CPQCs, which no longer take the numeric input as parameters for the input encoding gates, but use discretization registers, which can encode an entire distribution for the input values. Using such a controlled version allows us to calculate the expected payoff of an option with regard to an arbitrary distribution for the underlying. In the last step AE can be used to efficiently extract that expected value. In theory, AE has a quadratic speedup compared to classical MC. But to really harness this advantage, we would need similar prerequisites on the quantum hardware as for the QFT based approach. We do not need arbitrary state preparation, but distribution loading, which is also not solved efficiently yet. Also, we need less error-prone hardware, to be actually able to employ AE.

During the work on QAS, we have tried two different representations for quantum algorithms, the typical, gate based one, and a less intuitive, ZX-diagram-based one. While using ZX-diagrams seems promising, especially for low complexity models, the translation into CPQCs is more complicated and was not done in this work.

We want to close with some thoughts on which aspects further research could be directed at. We mentioned earlier, that the usage of CPQCs for option pricing allows taking physical properties of quantum devices, e.g., the available gate set and connectivity or the actual error rates of the individual qubits, into account. Experiments on actual hardware could be directed at finding out, how much of an improvement this would bring.

Further, the loading of distributions for the used underlying, is a step in our approaches, that could use some improvement. One idea is that loading an approximation of the desired distribution could reduce the complexity of doing so. As we already introduce an approximation error by discretizing the distribution and also approximate the payoff function, one would expect, that an additional approximation error, if it is small enough, does not significantly decrease overall performance.

Further research is also needed for the ZX-diagram-based QAS. One could try to construct more gFlow preserving mutations or look into translating these diagrams into CPQCs.

For the QFT based option pricing, further research could be directed at the structure of the initial vectors, that need to be loaded into the quantum computer. Finding a way of more efficiently loading this distribution would greatly benefit this approach. Additionally, one could try to repeat the calculations we did for other types of options.

Acronyms

AA Amplitude Amplification . . . 41
AE Amplitude Estimation . . . 3
CPQC Conditional Parameterized Quantum Circuit . . . 68
DFT Discrete Fourier Transform . . . 31
FFT Fast Fourier Transform . . . 3
FT Fourier Transform . . . 118
GAN Generative Adversarial Network . . . 142
gFlow general flow . . . 28
GP Genetic Programming . . . 83
MC Monte-Carlo . . . 3
MSE Mean Squared Error . . . 84
NISQ Noisy Intermediate-Scale Quantum . . . 62
PQC Parameterized Quantum Circuit . . . 62
QAS Quantum Architecture Search . . . 83
QFT Quantum Fourier Transform . . . 3
QMCI Quantum Monte-Carlo Integration . . . 68
SDE Stochastic Differential Equation . . . 7
SPSA Simultaneous Perturbation Stochastic Approximation . . . 98
VAns Variable Ansatz . . . 100
VQA Variational Quantum Algorithm . . . 62

Bibliography

[1] Dorit Aharonov, Vaughan Jones, and Zeph Landau. "A Polynomial Quantum Algorithm for Approximating the Jones Polynomial". In: *Algorithmica* 55.3 (Nov. 1, 2009), pp. 395–421. ISSN: 1432-0541. DOI: 10.1007/s00453-008-9168-0. URL: https://doi.org/10.1007/s00453-008-9168-0 (visited on 01/15/2025).

[2] Miriam Backens, Hector Miller-Bakewell, Giovanni de Felice, Leo Lobski, and John van de Wetering. "There and Back Again: A Circuit Extraction Tale". In: *Quantum* 5 (Mar. 25, 2021), p. 421. ISSN: 2521-327X. DOI: 10.22331/q-2021-03-25-421. arXiv: 2003.01664 [quant-ph]. URL: http://arxiv.org/abs/2003.01664 (visited on 10/13/2023).

[3] A. Barenco, C. H. Bennett, R. Cleve, D. P. DiVincenzo, N. Margolus, P. Shor, T. Sleator, J. Smolin, and H. Weinfurter. "Elementary Gates for Quantum Computation". In: *Physical Review A* 52.5 (Nov. 1, 1995), pp. 3457–3467. ISSN: 1050-2947, 1094-1622. DOI: 10.1103/PhysRevA.52.3457. arXiv: quant-ph/9503016. URL: http://arxiv.org/abs/quant-ph/9503016 (visited on 12/06/2024).

[4] Kenton Barnes. "The Genetic Evolution of Quantum Programs Using The ZX-Calculus". Sept. 2020. URL: http://www.cs.ox.ac.uk/people/aleks.kissinger/theses/barnes-thesis.pdf (visited on 11/08/2023).

[5] Sunitha Basodi, Chunyan Ji, Haiping Zhang, and Yi Pan. "Gradient Amplification: An Efficient Way to Train Deep Neural Networks". In: *Big Data Mining and Analytics* 3.3 (Sept. 2020), pp. 196–207. ISSN: 2097-406X. DOI: 10.26599/BDMA.2020.9020004. URL: https://ieeexplore.ieee.org/document/9142152/?arnumber=9142152 (visited on 01/23/2025).

[6] Atılım Gunes Baydin, Barak A Pearlmutter, Alexey Andreyevich Radul, and Jeffrey Mark Siskind. "Automatic Differentiation in Machine Learning: A Survey". In: *Journal of machine learning research* 18 (Apr. 2018), pp. 1–43.

[7] Ville Bergholm, Josh Izaac, Maria Schuld, Christian Gogolin, Shahnawaz Ahmed, Vishnu Ajith, M. Sohaib Alam, Guillermo Alonso-Linaje, B. AkashNarayanan, Ali Asadi, Juan Miguel Arrazola, et al. *PennyLane: Automatic Differentiation of Hybrid Quantum-Classical Computations*. July 29, 2022. DOI: 10.48550/arXiv.1811.04968. arXiv: 1811.04968 [physics, physics:quant-ph]. URL: http://arxiv.org/abs/1811.04968 (visited on 04/24/2024). Pre-published.

[8] M. Bilkis, M. Cerezo, Guillaume Verdon, Patrick J. Coles, and Lukasz Cincio. "A Semi-Agnostic Ansatz with Variable Structure for Variational Quantum Algorithms". In: *Quantum Machine Intelligence* 5.2 (Nov. 18, 2023), p. 43. ISSN: 2524-4914. DOI: 10.1007/s42484-023-00132-1. URL: https://doi.org/10.1007/s42484-023-00132-1 (visited on 12/05/2023).

[9] Kostas Blekos, Dean Brand, Andrea Ceschini, Chiao-Hui Chou, Rui-Hao Li, Komal Pandya, and Alessandro Summer. "A Review on Quantum Approximate Optimization Algorithm and Its Variants". In: *Physics Reports* 1068 (June 2024), pp. 1–66. ISSN: 03701573. DOI: 10.1016/j.physrep.2024.03.002. arXiv: 2306.09198 [quant-ph]. URL: http://arxiv.org/abs/2306.09198 (visited on 12/12/2024).

[10] Mathieu Blondel, Quentin Berthet, Marco Cuturi, Roy Frostig, Stephan Hoyer, Felipe Llinares-López, Fabian Pedregosa, and Jean-Philippe Vert. "Efficient and Modular Implicit Differentiation". 2021. arXiv: 2105.15183.

[11] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. *JAX: Composable Transformations of Python+NumPy Programs*. Version 0.3.13. 2018. URL: http://github.com/jax-ml/jax.

[12] G. Brassard and P. Hoyer. "An Exact Quantum Polynomial-Time Algorithm for Simon's Problem". In: *Proceedings of the Fifth Israeli Symposium on Theory of Computing and Systems*. Proceedings of the Fifth Israeli Symposium on Theory of Computing and Systems. June 1997, pp. 12–23. DOI: 10.1109/ISTCS.1997.595153.

[13] Gilles Brassard, Peter Hoyer, Michele Mosca, and Alain Tapp. "Quantum Amplitude Amplification and Estimation". May 15, 2000. arXiv: quant-ph/0005055. URL: http://arxiv.org/abs/quant-ph/0005055 (visited on 04/16/2020).

[14] Zhenyu Cai, Ryan Babbush, Simon C. Benjamin, Suguru Endo, William J. Huggins, Ying Li, Jarrod R. McClean, and Thomas E. O'Brien. "Quantum Error Mitigation". In: *Reviews of Modern Physics* 95.4 (Dec. 13, 2023), p. 045005. ISSN: 0034-6861, 1539-0756. DOI: 10.1103/RevModPhys.95.045005. arXiv: 2210.00921 [quant-ph]. URL: http://arxiv.org/abs/2210.00921 (visited on 01/13/2025).

[15] Peter Carr and Dilip Madan. "Option Valuation Using the Fast Fourier Transform". In: *The Journal of Computational Finance* 2.4 (1999), pp. 61–73. ISSN: 14601559. DOI: 10.21314/JCF.1999.043. URL: http://www.risk.net/journal-of-computational-finance/technical-paper/2160495/option-valuation-fast-fourier-transform (visited on 12/02/2022).

[16] M. Cerezo, Andrew Arrasmith, Ryan Babbush, Simon C. Benjamin, Suguru Endo, Keisuke Fujii, Jarrod R. McClean, Kosuke Mitarai, Xiao Yuan, Lukasz Cincio, and Patrick J. Coles. "Variational Quantum Algorithms". In: *Nature Reviews Physics* 3.9 (Sept. 2021), pp. 625–644. ISSN: 2522-5820. DOI: 10.1038/s42254-021-00348-9. URL: https://www.nature.com/articles/s42254-021-00348-9 (visited on 12/09/2024).

[17] Subrata Chakraborty. "Generating Discrete Analogues of Continuous Probability Distributions-A Survey of Methods and Constructions". In: *Journal of Statistical Distributions and Applications* 2.1 (Aug. 5, 2015), p. 6. ISSN: 2195-5832. DOI: 10.1186/s40488-015-0028-6. URL: https://doi.org/10.1186/s40488-015-0028-6 (visited on 01/08/2025).

[18] Rui Chao and Ben W. Reichardt. "Fault-Tolerant Quantum Computation with Few Qubits". In: *npj Quantum Information* 4.1 (Sept. 12, 2018), pp. 1–8. ISSN: 2056-6387. DOI: 10.1038/s41534-018-0085-z. URL: https://www.nature.com/articles/s41534-018-0085-z (visited on 01/13/2025).

[19] Bob Coecke and Ross Duncan. "Interacting Quantum Observables". In: *Automata, Languages and Programming*. Ed. by Luca Aceto, Ivan Damgård, Leslie Ann Goldberg, Magnús M. Halldórsson, Anna Ingólfsdóttir, and Igor Walukiewicz. Vol. 5126. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 298–310. ISBN: 978-3-540-70582-6 978-3-540-70583-3. DOI: 10.1007/978-3-540-70583-3_25. URL: http://link.springer.com/10.1007/978-3-540-70583-3_25 (visited on 05/14/2024).

[20] Bob Coecke and Ross Duncan. "Interacting Quantum Observables: Categorical Algebra and Diagrammatics". In: *New Journal of Physics* 13.4 (Apr. 2011), p. 043016. ISSN: 1367-2630. DOI: 10.1088/1367-2630/13/4/043016. URL: https://dx.doi.org/10.1088/1367-2630/13/4/043016 (visited on 07/08/2024).

[21] D. Coppersmith. *An Approximate Fourier Transform Useful in Quantum Factoring*. Jan. 16, 2002. DOI: 10.48550/arXiv.quant-ph/0201067. arXiv: quant-ph/0201067. URL: http://arxiv.org/abs/quant-ph/0201067 (visited on 07/25/2023). Pre-published.

[22] Alexander Cowtan, Silas Dilkes, Ross Duncan, Will Simmons, and Seyon Sivarajah. "Phase Gadget Synthesis for Shallow Circuits". In: *Electronic Proceedings in Theoretical Computer Science* 318 (May 1, 2020), pp. 213–228. ISSN: 2075-2180. DOI: 10.4204/EPTCS.318.13. URL: http://arxiv.org/abs/1906.01734v2 (visited on 11/27/2024).

[23] P. A. M. Dirac. "A New Notation for Quantum Mechanics". In: *Mathematical Proceedings of the Cambridge Philosophical Society* 35.3 (July 1939), pp. 416–418. ISSN: 0305-0041, 1469-8064. DOI: 10.1017/S0305004100021162. URL: https://www.cambridge.org/core/product/identifier/S0305004100021162/type/journal_article (visited on 10/24/2024).

[24] Yuxuan Du, Tao Huang, Shan You, Min-Hsiu Hsieh, and Dacheng Tao. "Quantum Circuit Architecture Search for Variational Quantum Algorithms". In: *npj Quantum Information* 8.1 (May 23, 2022), pp. 1–8. ISSN: 2056-6387. DOI: 10.1038/s41534-022-00570-y. URL: https://www.nature.com/articles/s41534-022-00570-y (visited on 04/24/2024).

[25] Ross Duncan, Aleks Kissinger, Simon Perdrix, and John van de Wetering. "Graph-Theoretic Simplification of Quantum Circuits with the ZX-calculus". In: *Quantum* 4 (June 4, 2020), p. 279. ISSN: 2521-327X. DOI: 10.22331/q-2020-06-04-279. arXiv: 1902.03178 [quant-ph]. URL: http://arxiv.org/abs/1902.03178 (visited on 10/11/2023).

[26] Tom Ewen. *Pricing of European Calls with the Quantum Fourier Transform*. Apr. 22, 2024. DOI: 10.48550/arXiv.2404.14115. arXiv: 2404.14115 [quant-ph, q-fin]. URL: http://arxiv.org/abs/2404.14115 (visited on 06/13/2024). Pre-published.

[27] Tom Ewen, Ivica Turkalj, Patrick Holzer, and Mark-Oliver Wolf. "Application of ZX-calculus to Quantum Architecture Search". In: *Quantum Machine Intelligence* 7.1 (Mar. 4, 2025), p. 34. ISSN: 2524-4914. DOI: 10.1007/s42484-025-00264-6. URL: https://doi.org/10.1007/s42484-025-00264-6 (visited on 03/06/2025).

[28] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. *A Quantum Approximate Optimization Algorithm*. Nov. 14, 2014. DOI: 10.48550/arXiv.1411.4028. arXiv: 1411.4028 [quant-ph]. URL: http://arxiv.org/abs/1411.4028 (visited on 12/12/2024). Pre-published.

[29] Richard P. Feynman. "Simulating Physics with Computers". In: *International Journal of Theoretical Physics* 21.6 (June 1, 1982), pp. 467–488. ISSN: 1572-9575. DOI: 10.1007/BF02650179. URL: https://doi.org/10.1007/BF02650179 (visited on 10/24/2024).

[30] Dmitry Grinko, Julien Gacon, Christa Zoufal, and Stefan Woerner. "Iterative Quantum Amplitude Estimation". In: *npj Quantum Information* 7.1 (Dec. 2021), p. 52. ISSN: 2056-6387. DOI: 10.1038/s41534-021-00379-1. arXiv: 1912.05559 [quant-ph]. URL: http://arxiv.org/abs/1912.05559 (visited on 09/21/2022).

[31] Lov K. Grover. "A Fast Quantum Mechanical Algorithm for Database Search". In: *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing - STOC '96*. The Twenty-Eighth Annual ACM Symposium. Philadelphia, Pennsylvania, United States: ACM Press, 1996, pp. 212–219. ISBN: 978-0-89791-785-8. DOI: 10.1145/237814.237866. URL: http://portal.acm.org/citation.cfm?doid=237814.237866 (visited on 05/22/2020).

[32] Zoë Holmes, Kunal Sharma, M. Cerezo, and Patrick J. Coles. "Connecting Ansatz Expressibility to Gradient Magnitudes and Barren Plateaus". In: *PRX Quantum* 3.1 (Jan. 24, 2022), p. 010313. DOI: 10.1103/PRXQuantum.3.010313. URL: https://link.aps.org/doi/10.1103/PRXQuantum.3.010313 (visited on 07/04/2023).

[33] Patrick Holzer and Ivica Turkalj. *Spectral Invariance and Maximality Properties of the Frequency Spectrum of Quantum Neural Networks*. Feb. 22, 2024. DOI: 10.48550/arXiv.2402.14515. arXiv: 2402.14515 [quant-ph, stat]. URL: http://arxiv.org/abs/2402.14515 (visited on 02/26/2024). Pre-published.

[34] Roger A. Horn and Charles R. Johnson. *Topics in Matrix Analysis*. Transferred to digital printing. Cambridge: Cambridge Univ. Press, 2010. 607 pp. ISBN: 978-0-521-46713-1 978-0-521-30587-7.

[35] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Weinberger. *Deep Networks with Stochastic Depth*. July 28, 2016. DOI: 10.48550/arXiv.1603.09382. arXiv: 1603.09382 [cs]. URL: http://arxiv.org/abs/1603.09382 (visited on 01/23/2025). Pre-published.

[36] Philip Intallura, Georgios Korpas, Sudeepto Chakraborty, Vyacheslav Kungurtsev, and Jakub Marecek. *A Survey of Quantum Alternatives to Randomized Algorithms: Monte Carlo Integration and Beyond*. Mar. 8, 2023. DOI: 10.48550/arXiv.2303.04945. arXiv: 2303.04945 [quant-ph, stat]. URL: http://arxiv.org/abs/2303.04945 (visited on 09/10/2024). Pre-published.

[37] Peter Jäckel. *Monte Carlo Methods in Finance*. Repr. Wiley Finance Series. Chichester Weinheim: Wiley, 2010. 222 pp. ISBN: 978-0-471-49741-7.

[38] Daniel F. V. James, Paul G. Kwiat, William J. Munro, and Andrew G. White. "Measurement of Qubits". In: *Physical Review A* 64.5 (Oct. 16, 2001), p. 052312. DOI: 10.1103/PhysRevA.64.052312. URL: https://link.aps.org/doi/10.1103/PhysRevA.64.052312 (visited on 04/08/2024).

[39] Ali Javadi-Abhari, Matthew Treinish, Kevin Krsulich, Christopher J. Wood, Jake Lishman, Julien Gacon, Simon Martiel, Paul D. Nation, Lev S. Bishop, Andrew W. Cross, Blake R. Johnson, et al. *Quantum Computing with Qiskit*. 2024. DOI: 10.48550/arXiv.2405.08810. arXiv: 2405.08810 [quant-ph].

[40] Phillip Kaye, Raymond Laflamme, and Michele Mosca. *An Introduction to Quantum Computing*. 1. publ. Oxford: Oxford University Press, 2007. 274 pp. ISBN: 978-0-19-857000-4 978-0-19-857049-3.

[41] Aleks Kissinger and John van de Wetering. "PyZX: Large Scale Automated Diagrammatic Reasoning". In: *Electronic Proceedings in Theoretical Computer Science* 318 (May 1, 2020), pp. 229–241. ISSN: 2075-2180. DOI: 10.4204/EPTCS.318.14. arXiv: 1904.04735 [quant-ph]. URL: http://arxiv.org/abs/1904.04735 (visited on 04/23/2024).

[42] A. Yu Kitaev. "Quantum Measurements and the Abelian Stabilizer Problem". Nov. 20, 1995. arXiv: quant-ph/9511026. URL: http://arxiv.org/abs/quant-ph/9511026 (visited on 06/17/2020).

[43] Ralf Korn and Elke Korn. *Option Pricing and Portfolio Optimization: Modern Methods of Financial Mathematics*. Graduate Studies in Mathematics 31. Providence, RI: American Mathematical Society, 2001. 253 pp. ISBN: 978-0-8218-2123-7.

[44] John R Koza. "Genetic Programming as a Means for Programming Computers by Natural Selection". In: *Statistics and Computing* 4.2 (June 1994). ISSN: 0960-3174, 1573-1375. DOI: 10.1007/BF00175355. URL: http://link.springer.com/10.1007/BF00175355 (visited on 05/08/2024).

[45] G. Ganesh Kumar, Subhendu K. Sahoo, and Pramod Kumar Meher. "50 Years of FFT Algorithms and Applications". In: *Circuits, Systems, and Signal Processing* 38.12 (Dec. 1, 2019), pp. 5665–5698. ISSN: 1531-5878. DOI: 10.1007/s00034-019-01136-8. URL: https://doi.org/10.1007/s00034-019-01136-8 (visited on 11/25/2024).

[46] En-Jui Kuo, Yao-Lung L. Fang, and Samuel Yen-Chi Chen. *Quantum Architecture Search via Deep Reinforcement Learning*. Apr. 15, 2021. DOI: 10.48550/arXiv.2104.07715. arXiv: 2104.07715 [quant-ph]. URL: http://arxiv.org/abs/2104.07715 (visited on 04/24/2024). Pre-published.

[47] Eugene Lukacs. *Characteristic Functions*. 2nd ed., revised & enlarged. London: Griffin, 1970. 350 pp. ISBN: 978-0-85264-170-5.

[48] Yuri I. Manin. *Classical Computing, Quantum Computing, and Shor's Factoring Algorithm*. Mar. 2, 1999. arXiv: quant-ph/9903008. URL: http://arxiv.org/abs/quant-ph/9903008 (visited on 10/24/2024). Pre-published.

[49] Jarrod R. McClean, Sergio Boixo, Vadim N. Smelyanskiy, Ryan Babbush, and Hartmut Neven. "Barren Plateaus in Quantum Neural Network Training Landscapes". In: *Nature Communications* 9.1 (Nov. 16, 2018), p. 4812. ISSN: 2041-1723. DOI: 10.1038/s41467-018-07090-4. URL: https://www.nature.com/articles/s41467-018-07090-4 (visited on 12/13/2024).

[50] Anette Messinger, Valentin Torggler, Berend Klaver, Michael Fellner, and Wolfgang Lechner. *Fault-Tolerant Quantum Computing with the Parity Code and Noise-Biased Qubits*. Apr. 17, 2024. DOI: 10.48550/arXiv.2404.11332. arXiv: 2404.11332 [quant-ph]. URL: http://arxiv.org/abs/2404.11332 (visited on 01/13/2025). Pre-published.

[51] Aaron Meurer, Christopher P. Smith, Mateusz Paprocki, Ondřej Čertík, Sergey B. Kirpichev, Matthew Rocklin, AmiT Kumar, Sergiu Ivanov, Jason K. Moore, Sartaj Singh, Thilina Rathnayake, et al. "SymPy: Symbolic Computing in Python". In: *PeerJ Computer Science* 3 (Jan. 2017), e103. ISSN: 2376-5992. DOI: 10.7717/peerj-cs.103. URL: https://doi.org/10.7717/peerj-cs.103.

[52] K. Mitarai, M. Negoro, M. Kitagawa, and K. Fujii. "Quantum Circuit Learning". In: *Physical Review A* 98.3 (Sept. 10, 2018), p. 032309. DOI: 10.1103/PhysRevA.98.032309. URL: https://link.aps.org/doi/10.1103/PhysRevA.98.032309 (visited on 05/03/2022).

[53] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information*. 10th anniversary edition. Cambridge: Cambridge university press, 2010. ISBN: 978-1-107-00217-3.

[54] Adrián Pérez-Salinas, Alba Cervera-Lierta, Elies Gil-Fuster, and José I. Latorre. "Data Re-Uploading for a Universal Quantum Classifier". In: *Quantum* 4 (Feb. 6, 2020), p. 226. ISSN: 2521-327X. DOI: 10.22331/q-2020-02-06-226. arXiv: 1907.02085. URL: http://arxiv.org/abs/1907.02085 (visited on 05/03/2022).

[55] Michael JD Powell. "A View of Algorithms for Optimization without Derivatives". In: *Mathematics Today-Bulletin of the Institute of Mathematics and its Applications* 43.5 (2007), pp. 170–174.

[56] Fabrice Rouah. *The Heston Model and Its Extensions in Matlab and C#*. 1st ed. Wiley Finance Ser. Somerset: John Wiley & Sons, Incorporated, 2013. 1 p. ISBN: 978-1-118-54825-7 978-1-118-69518-0.

[57] Maria Schuld, Ville Bergholm, Christian Gogolin, Josh Izaac, and Nathan Killoran. "Evaluating Analytic Gradients on Quantum Hardware". In: *Physical Review A* 99.3 (Mar. 21, 2019), p. 032331. ISSN: 2469-9926, 2469-9934. DOI: 10.1103/PhysRevA.99.032331. arXiv: 1811.11184 [quant-ph]. URL: http://arxiv.org/abs/1811.11184 (visited on 08/18/2023).

[58] Maria Schuld, Alex Bocharov, Krysta M. Svore, and Nathan Wiebe. "Circuit-Centric Quantum Classifiers". In: *Physical Review A* 101.3 (Mar. 6, 2020), p. 032308. DOI: 10.1103/PhysRevA.101.032308. URL: https://link.aps.org/doi/10.1103/PhysRevA.101.032308 (visited on 08/02/2022).

[59] Maria Schuld, Ryan Sweke, and Johannes Jakob Meyer. "Effect of Data Encoding on the Expressive Power of Variational Quantum-Machine-Learning Models". In: *Physical Review A* 103.3 (Mar. 24, 2021), p. 032430. DOI: 10.1103/PhysRevA.103.032430. URL: https://link.aps.org/doi/10.1103/PhysRevA.103.032430 (visited on 08/02/2022).

[60] Vivek V. Shende, Stephen S. Bullock, and Igor L. Markov. "Synthesis of Quantum Logic Circuits". In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 25.6 (June 2006), pp. 1000–1010. ISSN: 0278-0070, 1937-4151. DOI: 10.1109/TCAD.2005.855930. arXiv: quant-ph/0406176. URL: http://arxiv.org/abs/quant-ph/0406176 (visited on 12/09/2024).

[61] Peter W. Shor. "Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer". In: *SIAM Journal on Computing* 26.5 (Oct. 1997), pp. 1484–1509. ISSN: 0097-5397, 1095-7111. DOI: 10.1137/S0097539795293172. arXiv: quant-ph/9508027. URL: http://arxiv.org/abs/quant-ph/9508027 (visited on 11/14/2024).

[62] James C Spall. "An Overview of the Simultaneous Perturbation Method for Efficient Optimization". In: *Johns Hopkins apl technical digest* 19.4 (1998), pp. 482–492.

[63] Nikitas Stamatopoulos, Daniel J. Egger, Yue Sun, Christa Zoufal, Raban Iten, Ning Shen, and Stefan Woerner. "Option Pricing Using Quantum Computers". Feb. 16, 2020. arXiv: 1905.02666 [quant-ph]. URL: http://arxiv.org/abs/1905.02666 (visited on 04/09/2020).

[64] Xiaoming Sun, Guojing Tian, Shuai Yang, Pei Yuan, and Shengyu Zhang. "Asymptotically Optimal Circuit Depth for Quantum State Preparation and General Unitary Synthesis". In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* (2023), pp. 1–1. ISSN: 1937-4151. DOI: 10.1109/TCAD.2023.3244885.

[65] Yohichi Suzuki, Shumpei Uno, Rudy Raymond, Tomoki Tanaka, Tamiya Onodera, and Naoki Yamamoto. "Amplitude Estimation without Phase Estimation". In: *Quantum Information Processing* 19.2 (Feb. 2020), p. 75. ISSN: 1570-0755, 1573-1332. DOI: 10.1007/s11128-019-2565-2. URL: http://link.springer.com/10.1007/s11128-019-2565-2 (visited on 04/22/2020).

[66] John van de Wetering. *ZX-calculus for the Working Quantum Computer Scientist*. Dec. 27, 2020. DOI: 10.48550/arXiv.2012.13966. arXiv: 2012.13966 [quant-ph]. URL: http://arxiv.org/abs/2012.13966 (visited on 10/11/2023). Pre-published.

[67] Peter J. M. Van Laarhoven and Emile H. L. Aarts. *Simulated Annealing: Theory and Applications*. Dordrecht: Springer Netherlands, 1987. ISBN: 978-90-481-8438-5 978-94-015-7744-1. DOI: 10.1007/978-94-015-7744-1. URL: http://link.springer.com/10.1007/978-94-015-7744-1 (visited on 01/23/2025).

[68] Mark-Oliver Wolf, Tom Ewen, and Ivica Turkalj. "Quantum Architecture Search for Quantum Monte Carlo Integration via Conditional Parameterized Circuits with Application to Finance". In: *2023 IEEE International Conference on Quantum Computing and Engineering (QCE)*. Vol. 01. Sept. 2023, pp. 560–570. DOI: 10.1109/QCE57702.2023.00070. URL: https://ieeexplore.ieee.org/document/10313631 (visited on 09/10/2024).

[69] Zhan Yu, Hongshun Yao, Mujin Li, and Xin Wang. *Power and Limitations of Single-Qubit Native Quantum Neural Networks*. Sept. 29, 2022. arXiv: 2205.07848 [cond-mat, physics:math-ph, physics:quant-ph]. URL: http://arxiv.org/abs/2205.07848 (visited on 12/21/2022). Pre-published.

[70] Shi-Xin Zhang, Chang-Yu Hsieh, Shengyu Zhang, and Hong Yao. "Differentiable Quantum Architecture Search". In: *Quantum Science and Technology* 7.4 (Oct. 1, 2022), p. 045023. ISSN: 2058-9565. DOI: 10.1088/2058-9565/ac87cd. arXiv: 2010.08561 [quant-ph]. URL: http://arxiv.org/abs/2010.08561 (visited on 04/24/2024).

[71] Xiao-Ming Zhang, Tongyang Li, and Xiao Yuan. "Quantum State Preparation with Optimal Circuit Depth: Implementations and Applications". In: *Physical Review Letters* 129.23 (Nov. 30, 2022), p. 230504. DOI: 10.1103/PhysRevLett.129.230504. URL: https://link.aps.org/doi/10.1103/PhysRevLett.129.230504 (visited on 05/05/2023).

[72] Christa Zoufal, Aurélien Lucchi, and Stefan Woerner. "Quantum Generative Adversarial Networks for Learning and Loading Random Distributions". In: *npj Quantum Information* 5.1 (Dec. 2019), p. 103. ISSN: 2056-6387. DOI: 10.1038/s41534-019-0223-2. URL: http://www.nature.com/articles/s41534-019-0223-2 (visited on 06/21/2022).

Appendix

Contents

A Kronecker Product

Throughout this work we often have used the Kronecker Product depicted by the operator $\otimes$. In this section we will briefly introduce it and mention some properties. For more details see for example the book by Horn and Johnson [34].

Definition A.1 Kronecker Product

Given two matrices $A \in \mathbb{R}^{j\times k}$ and $B \in \mathbb{R}^{l\times m}$ the Kronecker product is defined as

$$\mathbb{R}^{(jl)\times(km)} \ni A \otimes B = \begin{pmatrix} A_{1,1}B & \cdots & A_{1,k}B \\ \vdots & & \\ A_{j,1}B & \cdots & A_{j,k}B \end{pmatrix}.$$

The Kronecker product is associative, i.e.,

$$A \otimes (B + C) = A \otimes B + A \otimes C,$$
$$(B + C) \otimes A = B \otimes A + C \otimes A,$$

but in general not commutative.

The complex conjugation is distributive over the Kronecker product, i.e.,

$$(A \otimes B)^\dagger = A^\dagger \otimes B^\dagger.$$

If the dimensions of the involved matrices match, such that AC and BD are defined, the regular product and the Kronecker product mix, i.e.,

$$(A \otimes B)(C \otimes D) = AC \otimes BD. \tag{A.1}$$

B Trigonometric Identities and Inequalities

In this section we collect some trigonometric identities we need during this work. For $x, y \in \mathbb{R}$ we have

$$\cos(x) = \cos(-x), \tag{B.1}$$
$$\sin(x) = -\sin(-x), \tag{B.2}$$
$$\sin(x+y) = \sin(x)\cos(y) + \cos(x)\sin(y), \tag{B.3}$$
$$\cos(x+y) = \cos(x)\cos(y) - \sin(x)\sin(y), \tag{B.4}$$
$$\sin^2(x) + \cos^2(x) = 1, \tag{B.5}$$
$$\sin(2x) = 2\sin(x)\cos(x), \tag{B.6}$$
$$\cos(2x) = \cos^2(x) - \sin^2(x), \tag{B.7}$$
$$\cos(2x) = 1 - 1\sin^2(x), \tag{B.8}$$
$$\sin^2(x) = \frac{1-\cos(2x)}{2}. \tag{B.9}$$

For $x \geq 0$ we have

$$\sin(x) \leq x. \tag{B.10}$$

For $x \in [0, \pi/2]$ we have, by the concavity of the sine,

$$\sin(\pi x) \geq \sin(0) + (\pi x - 0)\frac{\sin(\frac{\pi}{2}) - \sin(0)}{\frac{\pi}{2} - 0} = \pi x \frac{2}{\pi} = 2x. \tag{B.11}$$

C ZX Mutation experiments

Here we give the detailed numerical results that are visualized in figure 3.4. We can summarize, that most p-values are sufficiently small, and in the cases they are not, the coefficient is very small anyway.

Table C.1: Coefficients and p-values for success rate.

	Connectivity		# Vertices		# Qubits	
	coef.	p-value	coef.	p-value	coef.	p-value
Edge Flipping	−0.07538	0.00000	−0.07433	0.00000	0.00986	0.04711
Edge Add.	−0.01324	0.01617	−0.00472	0.40182	0.01952	0.00000
Phase Gadget Add.	−0.05172	0.00000	−0.11665	0.00000	0.17120	0.00000
Edge Split	−0.08764	0.00000	−0.07401	0.00000	0.00031	0.95030
Edge Removal	0.09338	0.00000	0.04299	0.00000	0.05185	0.00000
Local Compl.	0.00000	0.00001	0.00000	0.00000	0.00000	0.00000
Inv. Local Compl.	−0.01969	0.00005	−0.08466	0.00000	0.09158	0.00000
Pivoting	0.08365	0.00000	0.20294	0.00000	−0.14619	0.00000
Edge Swap	0.01448	0.00132	0.00334	0.46895	0.02569	0.00000

Table C.2: Coefficients and p-values for absolute improvement.

	Connectivity		# Vertices		# Qubits	
	coef.	p-value	coef.	p-value	coef.	p-value
Edge Flipping	0.00392	0.00013	−0.00678	0.00000	0.01554	0.00000
Edge Add.	0.00092	0.01388	0.00007	0.84728	0.00143	0.00000
Phase Gadget Add.	0.00436	0.00849	0.00347	0.04087	0.00261	0.03553
Edge Split	0.00493	0.00220	0.01345	0.00000	−0.01602	0.00000
Edge Removal	0.00158	0.00242	−0.00170	0.00137	0.00369	0.00000
Local Compl.	0.00656	0.00000	−0.00584	0.00000	0.01713	0.00000
Inv. Local Compl.	−0.00055	0.48471	−0.00364	0.00001	0.00428	0.00000
Pivoting	0.01140	0.00000	0.00306	0.01766	0.00513	0.00000
Edge Swap	0.00005	0.89513	−0.00089	0.01868	0.00141	0.00000

Table C.3: Coefficients and p-values for absolute improvement without fails.

	Connectivity		# Vertices		# Qubits	
	coef.	p-value	coef.	p-value	coef.	p-value
Edge Flipping	0.00777	0.00000	−0.00655	0.00000	0.02026	0.00000
Edge Add.	0.00847	0.00047	0.00084	0.73266	0.00839	0.00001
Phase Gadget Add.	0.01167	0.00000	0.01690	0.00000	−0.01323	0.00000
Edge Split	0.01325	0.00000	0.02350	0.00000	−0.02117	0.00000
Edge Removal	0.00110	0.48081	−0.00665	0.00002	0.00894	0.00000
Local Compl.	0.00656	0.00000	−0.00584	0.00000	0.01713	0.00000
Inv. Local Compl.	0.00194	0.74979	0.02796	0.00000	−0.03399	0.00000
Pivoting	0.00922	0.00000	−0.00315	0.03035	0.00970	0.00000
Edge Swap	−0.00236	0.53122	−0.01034	0.00572	0.00910	0.00184

D Numerical Results

Here we will have a detailed look at the results from the algorithms presented section 3.4. The results for the European call where already presented in section 3.4, here we will see the results for the other options.

First the European basket call from example 3.12, we see the Pareto fronts in figure D.1. For every version of our QAS algorithm we visualize in detail the four models that are optimal regarding the four fitness criteria. The models found by the circuit-based approach are seen in figure D.2, the ones found by the VAns-based approach in figure D.3 and the ones found by the ZX-diagram-based approach in figure D.4. An overview of the corresponding circuits with their respective fitness values, together with references to the circuits themselves, is given in table D.1.

For the European rainbow call from example 3.14 the Pareto fronts are seen in figure D.5, the predictions in figures D.6 to D.8 and the circuit overview in table D.2.

For the Asian barrier spread from example 3.15 the Pareto fronts are seen in figure D.9, the predictions in figures D.10 to D.12 and the circuit overview in table D.3.

For the European basket call with variable weight from example 3.13 the Pareto fronts are seen in figure D.13, the predictions in figures D.14 to D.16 and the circuit overview in table D.4.

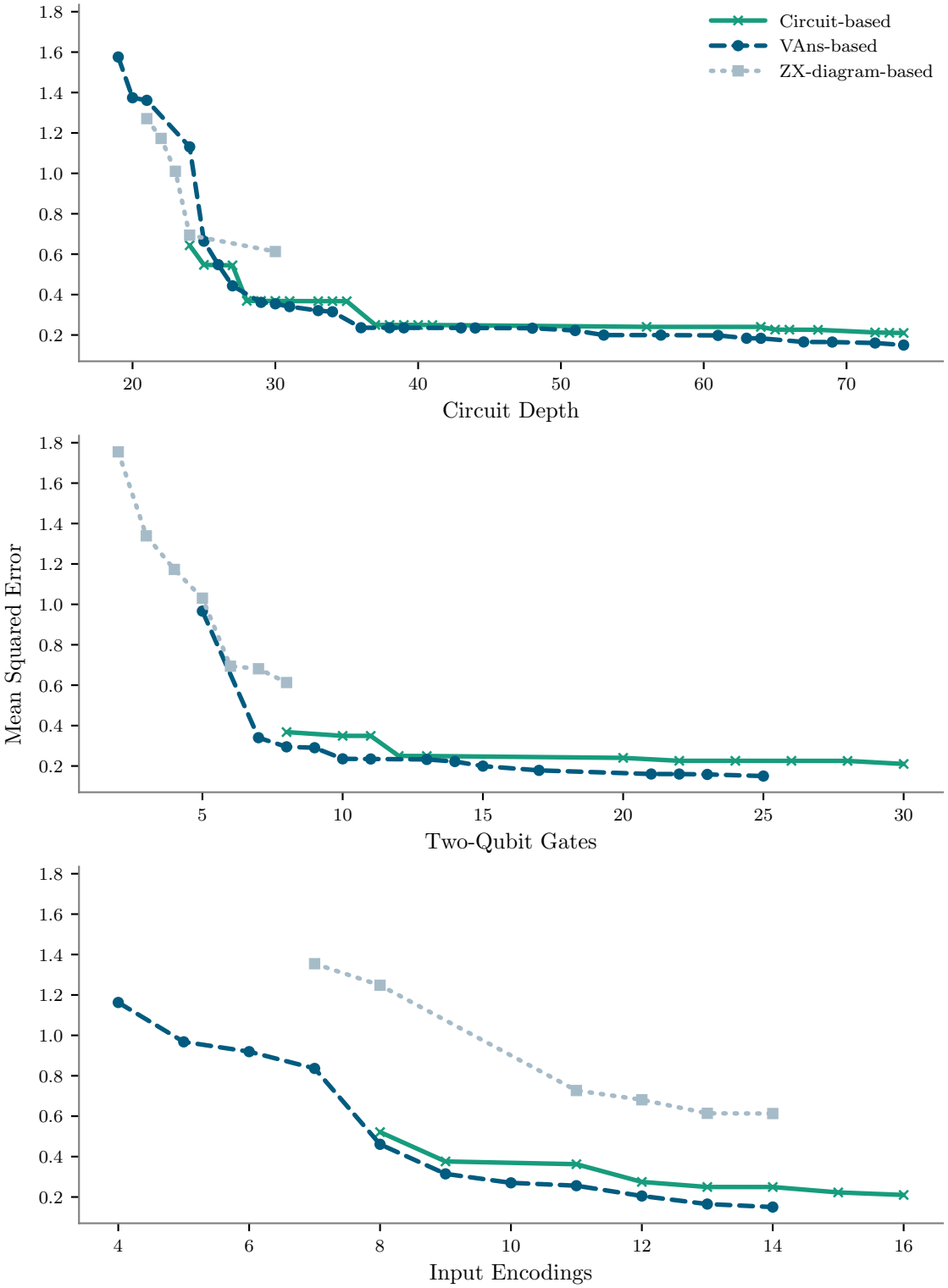
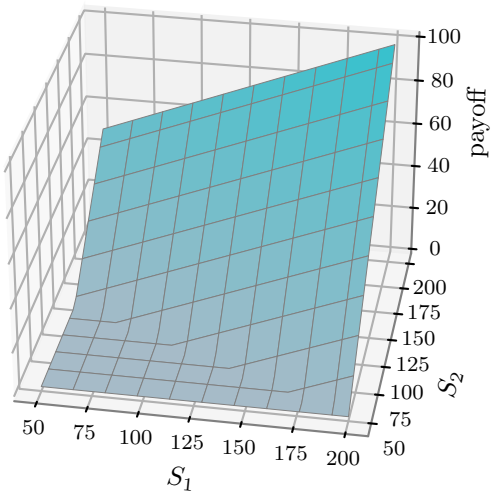
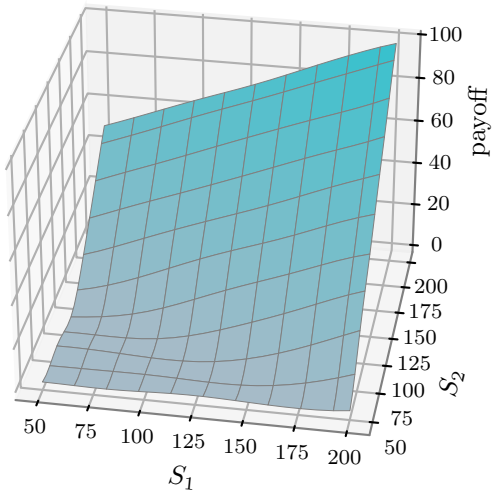


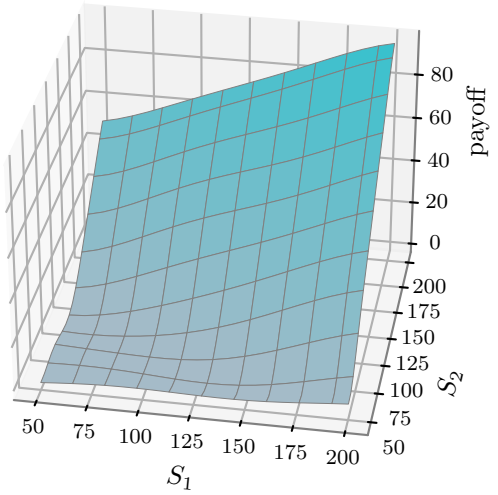
Figure D.1: Result of the genetic QAS with algorithm 3.1 for the European basket call seen in example 3.12. Projection of the Pareto front into the planes spanned by the MSE and the other three fitness criteria.



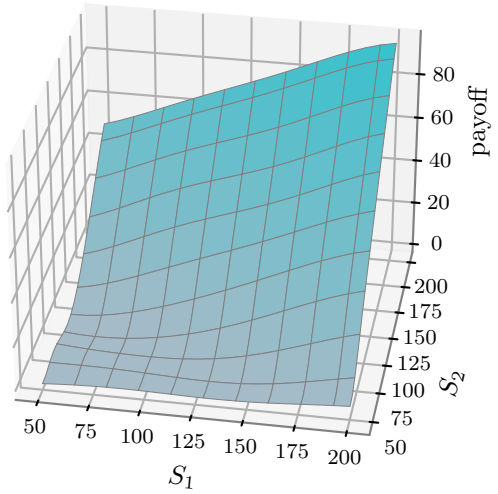
(a) Target



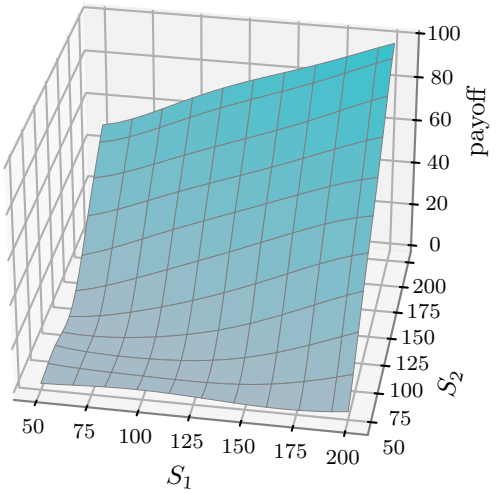
(b) MSE - figure D.17



(c) Circuit depth - figure D.18

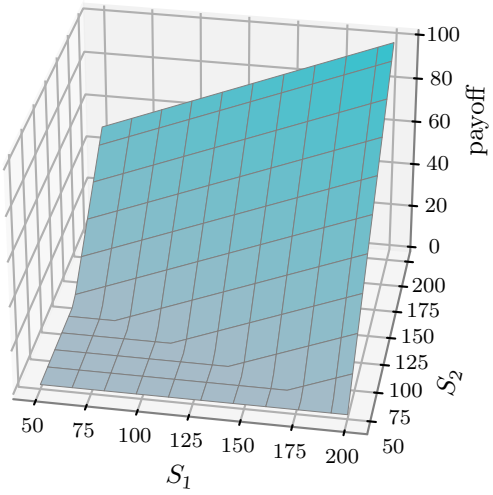


(d) Number of controlled gates - figure D.19

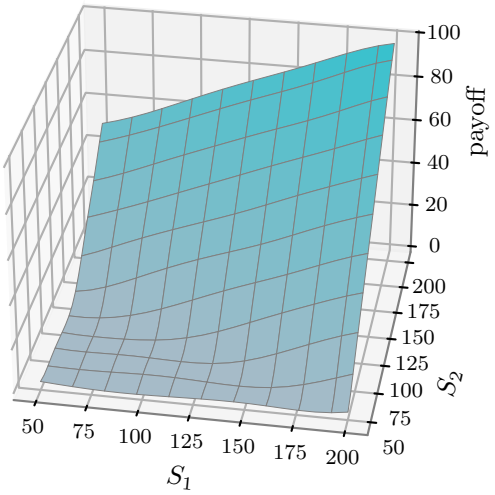


(e) Number of input gates - figure D.20

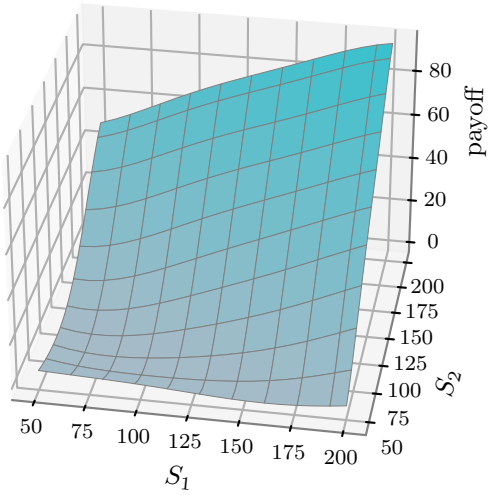
Figure D.2: Predictions of four Pareto optimal models found by algorithm 3.1 in the circuit-based version for the European basket call seen in example 3.12. Each of the models shown is optimal regarding one of the four fitness criteria.



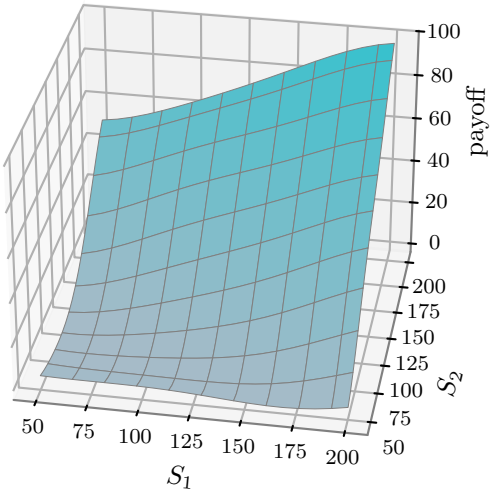
(a) Target



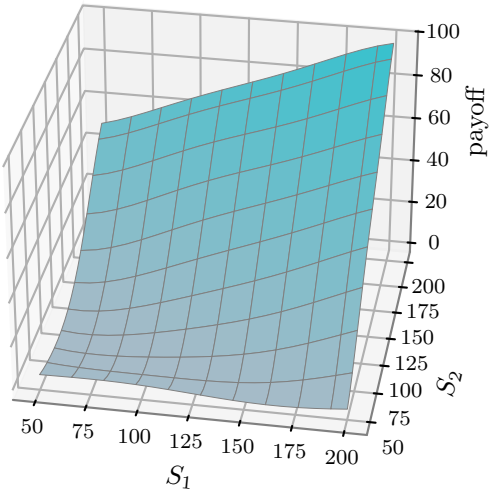
(b) MSE - figure D.21



(c) Circuit depth - figure D.22

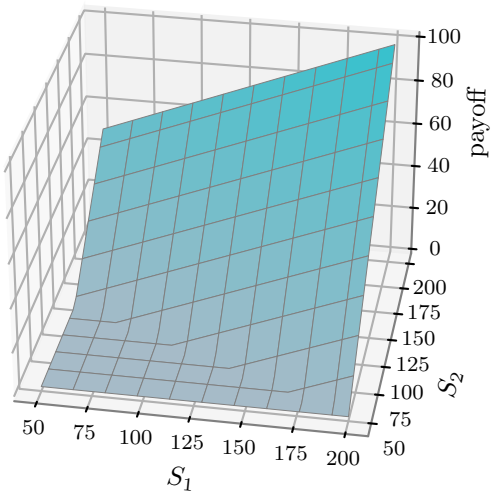


(d) Number of controlled gates - figure D.23

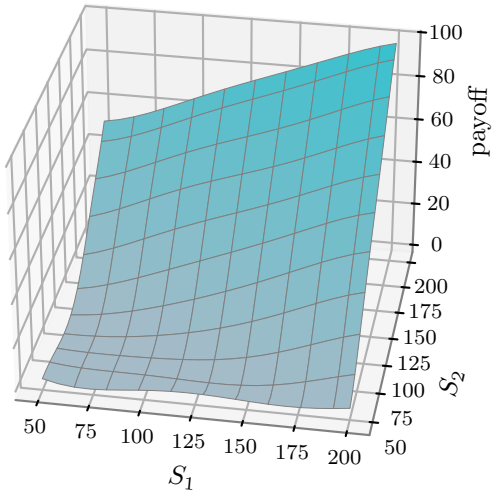


(e) Number of input gates - figure D.24

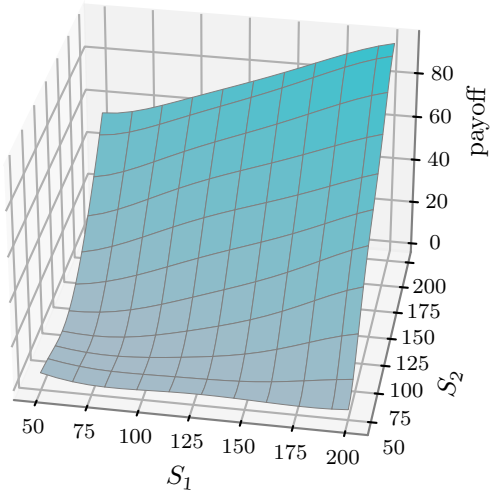
Figure D.3: Predictions of four Pareto optimal models found by algorithm 3.1 in the VAns-based version for the European basket call seen in example 3.12. Each of the models shown is optimal regarding one of the four fitness criteria.



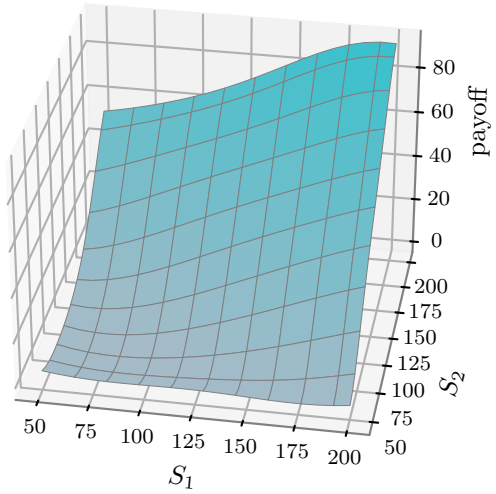
(a) Target



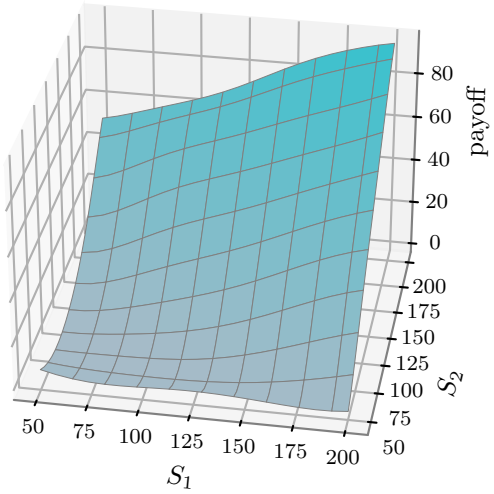
(b) MSE - figure D.25



(c) Circuit depth - figure D.26



(d) Number of controlled gates - figure D.27



(e) Number of input gates - figure D.28

Figure D.4: Predictions of four Pareto optimal models found by algorithm 3.1 in the ZX-diagram-based version for the European basket call seen in example 3.12. Each of the models shown is optimal regarding one of the four fitness criteria.

Table D.1: Results for the European basket call seen in example 3.12. All versions used 24 hours for the training. The lowest achieved value over all models is marked bold.

	Trained generations	Minimized criterion	MSE	Circuit depth	# Two-qubit gates	# Input encodings	Figure
Circuit-based	353	MSE	0.2101	74	30	16	D.17
		Circuit depth	0.6438	24	9	11	D.18
		# Two-qubit gates	0.6290	25	8	11	D.19
		# Input encodings	0.5300	56	12	8	D.20
VAns-based	129	MSE	**0.1503**	74	25	14	D.21
		Circuit depth	1.5758	**19**	5	**4**	D.22
		# Two-qubit gates	1.1310	24	5	5	D.23
		# Input encodings	1.1887	28	5	**4**	D.24
ZX-graph-based	224	MSE	0.6130	30	8	14	D.25
		Circuit depth	1.2718	21	5	12	D.26
		# Two-qubit gates	1.8633	22	**2**	11	D.27
		# Input encodings	1.3546	37	13	7	D.28

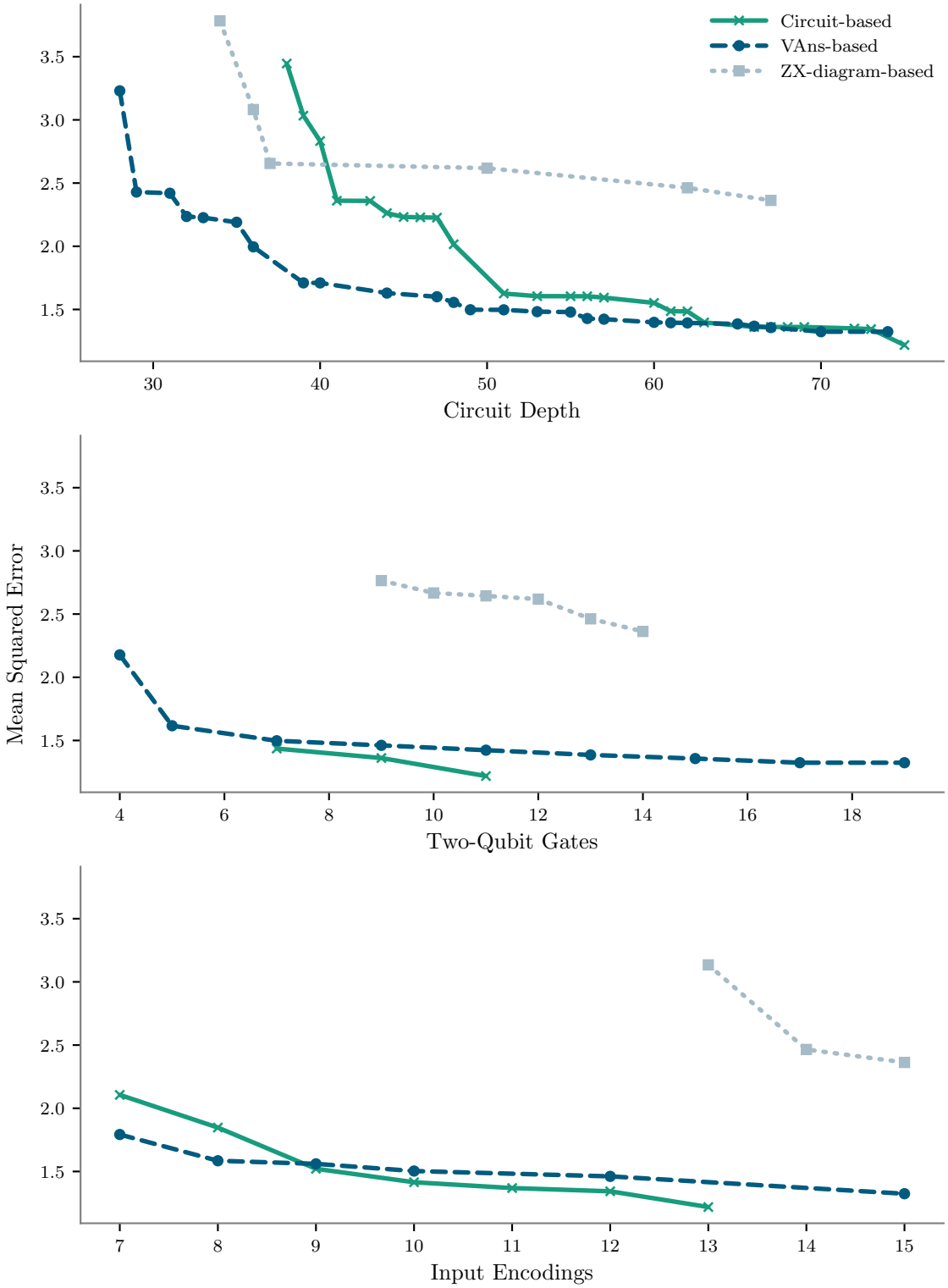
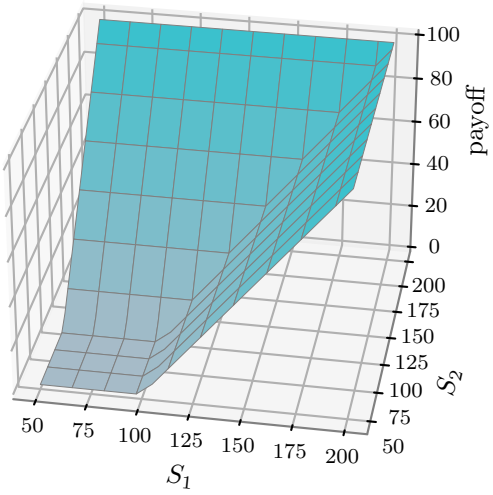
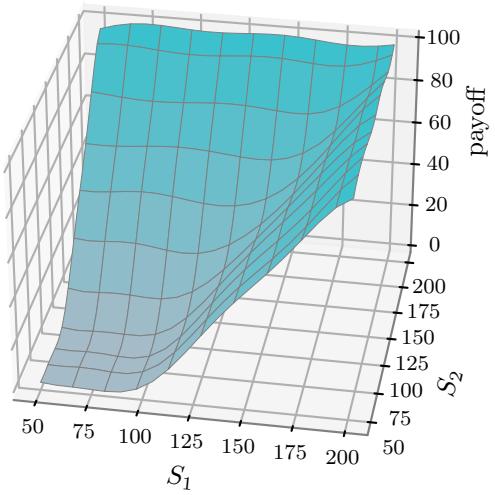


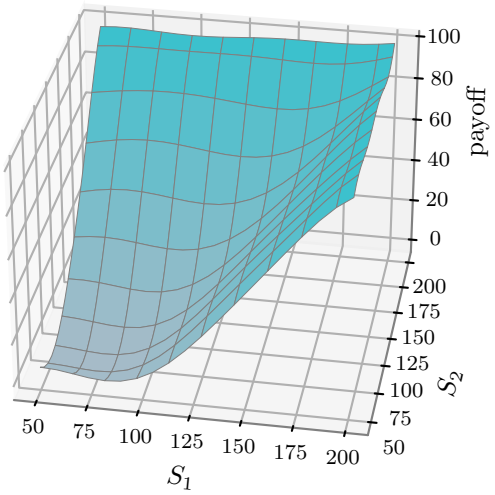
Figure D.5: Result of the genetic QAS with algorithm 3.1 for the European rainbow call seen in example 3.14. Projection of the Pareto front into the planes spanned by the MSE and the other three fitness criteria.



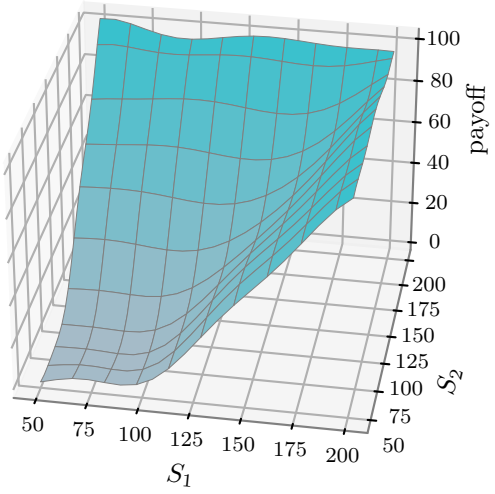
(a) Target



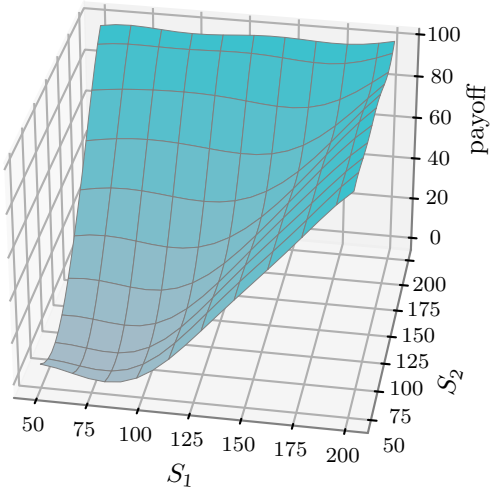
(b) MSE - figure D.41



(c) Circuit depth - figure D.42

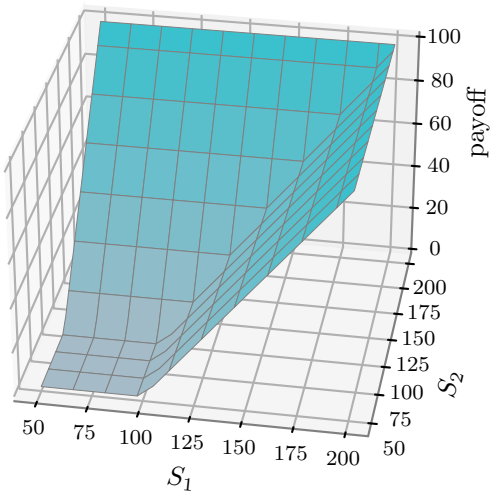


(d) Number of controlled gates - figure D.43

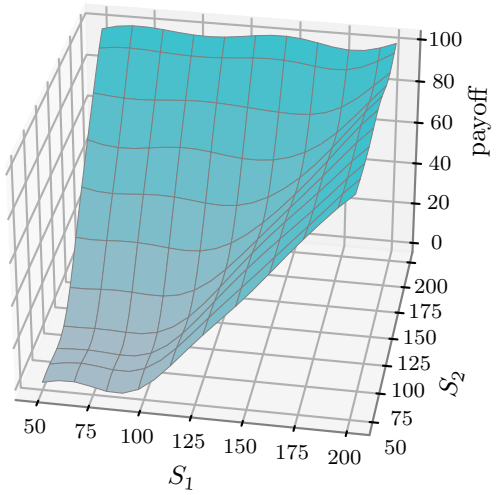


(e) Number of input gates - figure D.44

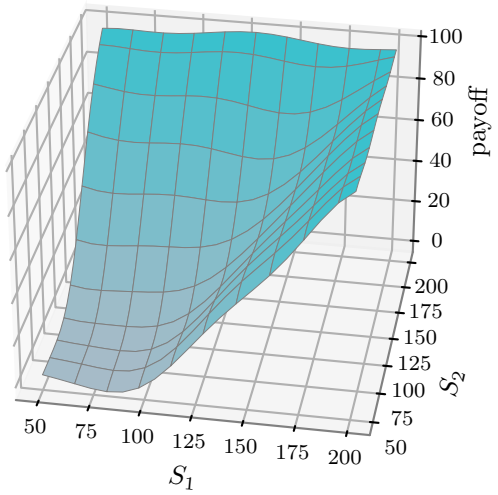
Figure D.6: Predictions of four Pareto optimal models found by algorithm 3.1 in the circuit-based version for the European rainbow call seen in example 3.14. Each of the models shown is optimal regarding one of the four fitness criteria.



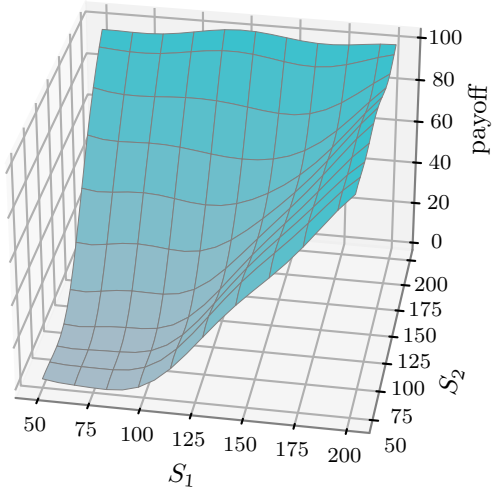
(a) Target



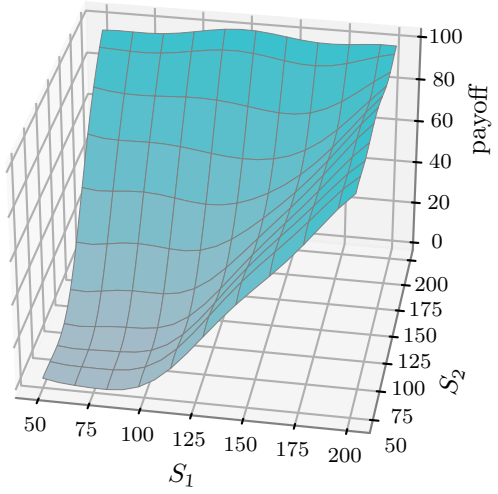
(b) MSE - figure D.45



(c) Circuit depth - figure D.46

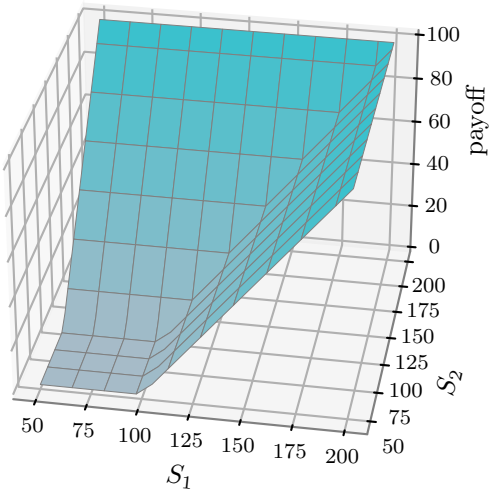


(d) Number of controlled gates - figure D.47

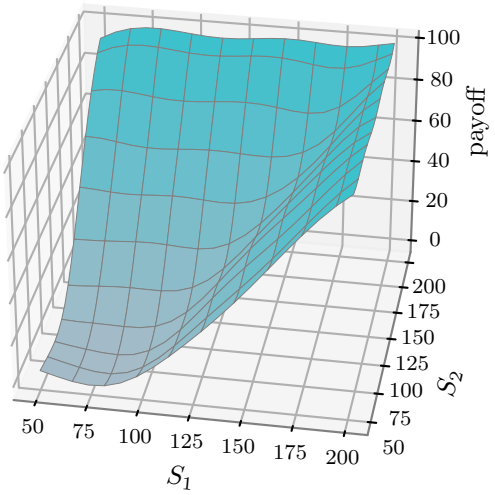


(e) Number of input gates - figure D.48

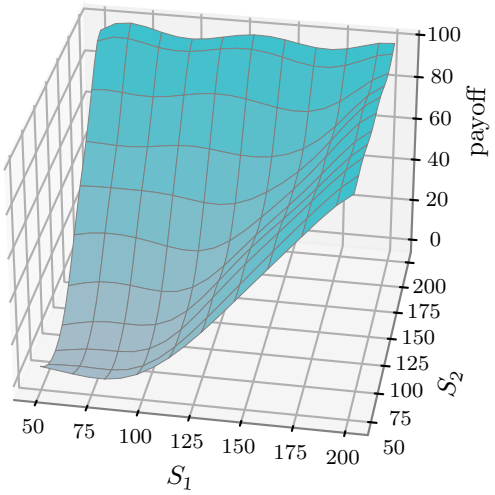
Figure D.7: Predictions of four Pareto optimal models found by algorithm 3.1 in the VAns-based version for the European rainbow call seen in example 3.14. Each of the models shown is optimal regarding one of the four fitness criteria.



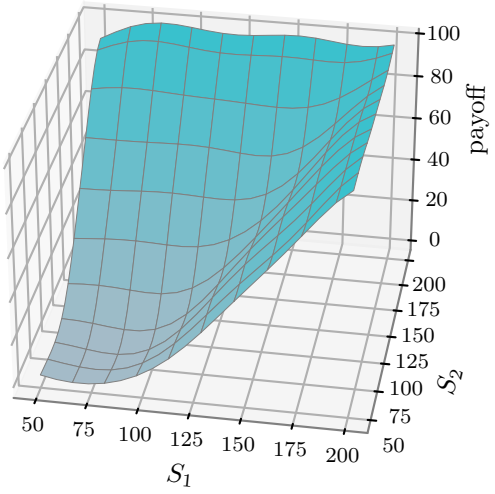
(a) Target



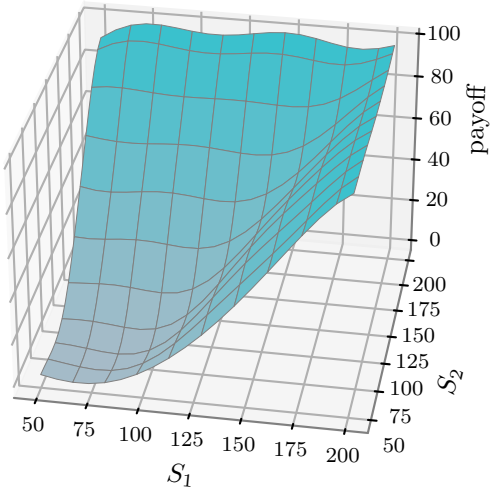
(b) MSE - figure D.49



(c) Circuit depth - figure D.50



(d) Number of controlled gates - figure D.51



(e) Number of input gates - figure D.52

Figure D.8: Predictions of four Pareto optimal models found by algorithm 3.1 in the ZX-diagram-based version for the European rainbow call seen in example 3.14. Each of the models shown is optimal regarding one of the four fitness criteria.

Table D.2: Results for the European rainbow call seen in example 3.14. All versions used 24 hours for the training. The lowest achieved value over all models is marked bold.

	Trained generations	Minimized criterion	MSE	Circuit depth	# Two-qubit gates	# Input encodings	Figure
Circuit-based	572	MSE	**1.2185**	75	11	13	D.41
		Circuit depth	3.4454	38	7	**7**	D.42
		# Two-qubit gates	2.2289	46	7	12	D.43
		# Input encodings	3.0042	53	7	**7**	D.44
VAns-based	369	MSE	1.3240	74	19	15	D.45
		Circuit depth	3.2298	**28**	5	**7**	D.46
		# Two-qubit gates	2.8985	32	**4**	9	D.47
		# Input encodings	2.9463	32	**4**	**7**	D.48
ZX-graph-based	262	MSE	2.3627	67	14	15	D.49
		Circuit depth	3.7832	34	14	15	D.50
		# Two-qubit gates	2.7647	51	9	16	D.51
		# Input encodings	3.1344	46	9	13	D.52

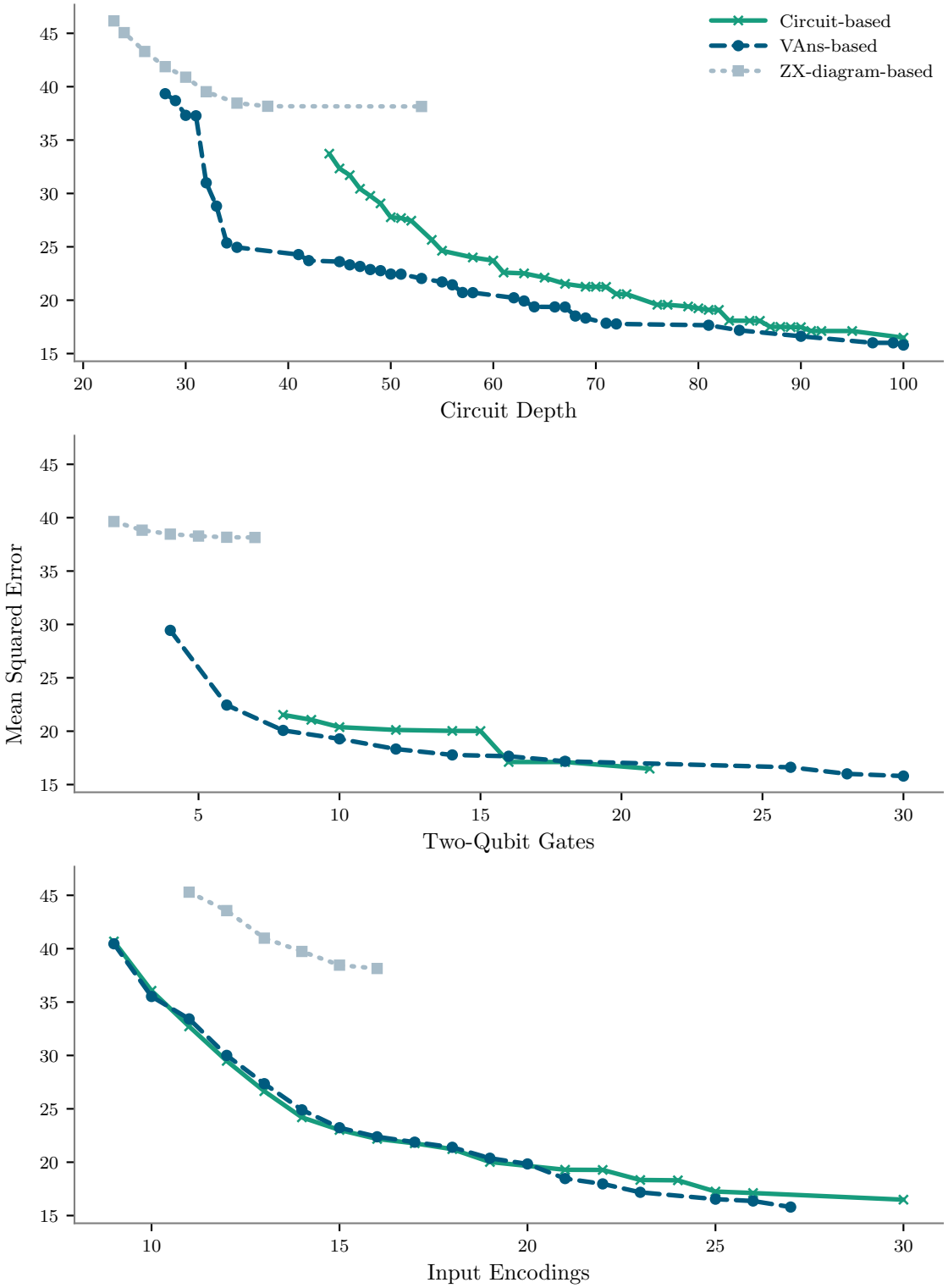
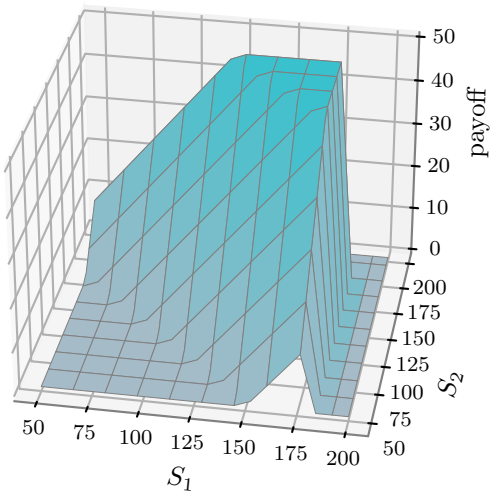
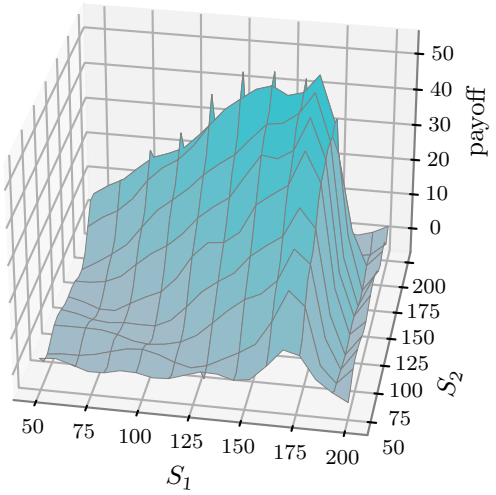


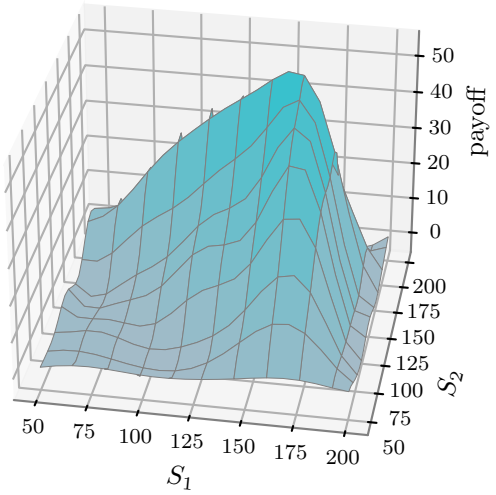
Figure D.9: Result of the genetic QAS with algorithm 3.1 for the Asian barrier spread seen in example 3.15. Projection of the Pareto front into the planes spanned by the MSE and the other three fitness criteria.



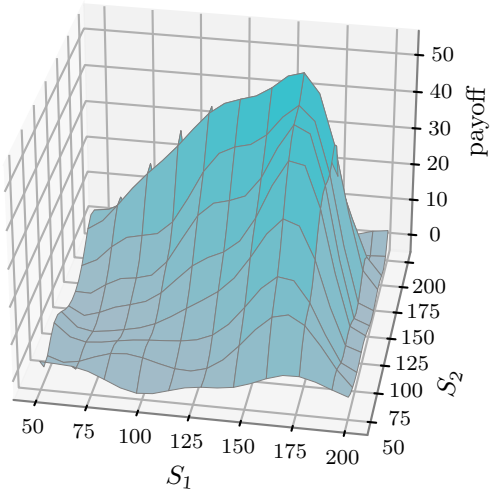
(a) Target



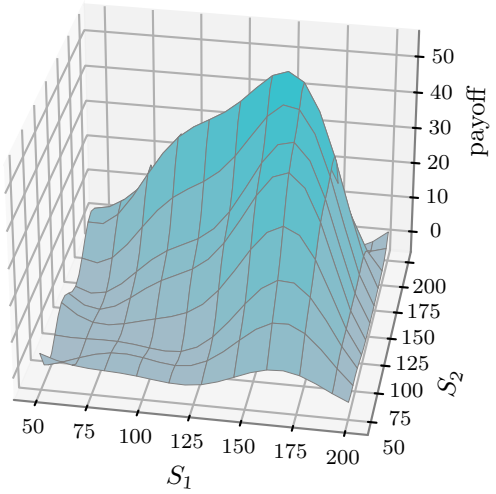
(b) MSE - figure D.53



(c) Circuit depth - figure D.54

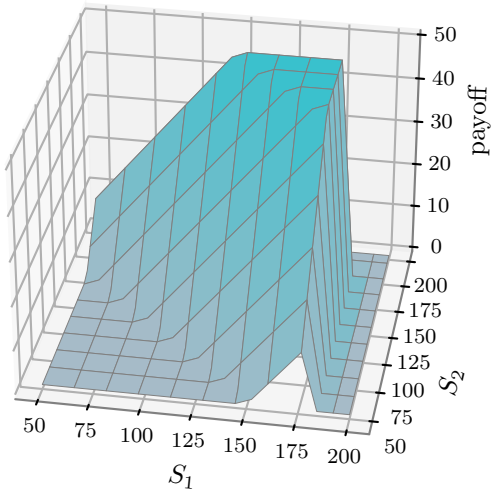


(d) Number of controlled gates - figure D.55

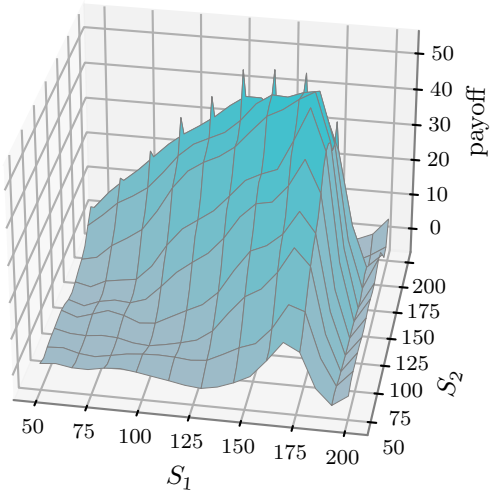


(e) Number of input gates - figure D.56

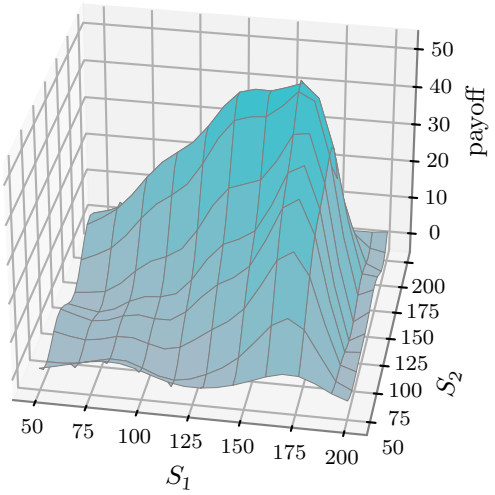
Figure D.10: Predictions of four Pareto optimal models found by algorithm 3.1 in the circuit-based version for the Asian barrier spread seen in example 3.15. Each of the models shown is optimal regarding one of the four fitness criteria.



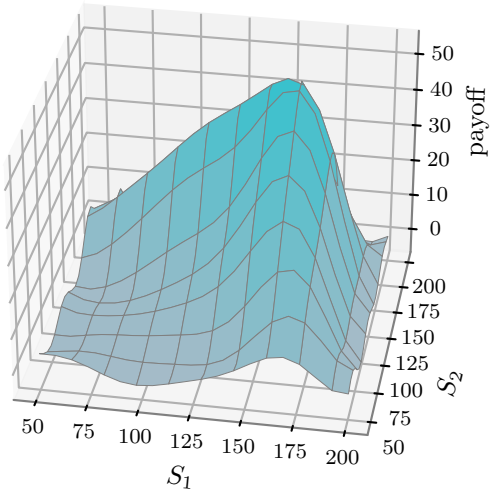
(a) Target



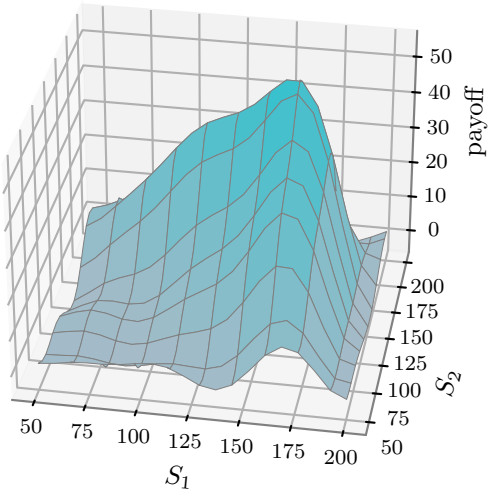
(b) MSE - figure D.57



(c) Circuit depth - figure D.58

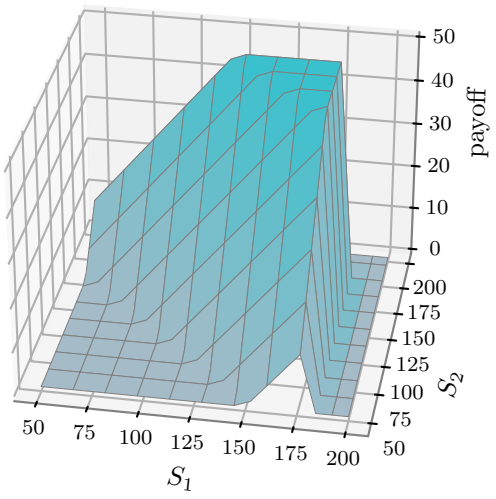


(d) Number of controlled gates - figure D.59

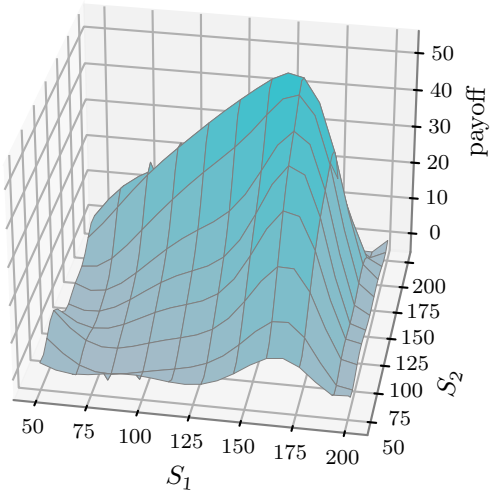


(e) Number of input gates - figure D.60

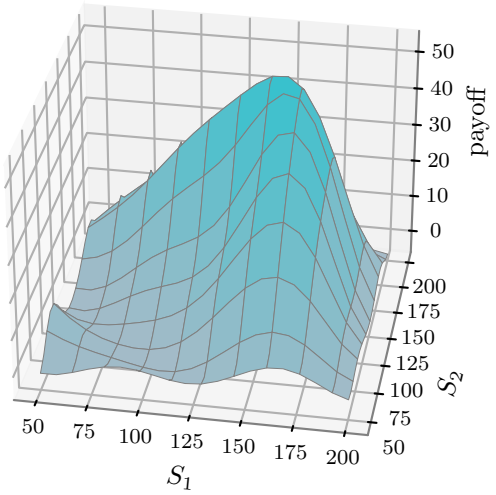
Figure D.11: Predictions of four Pareto optimal models found by algorithm 3.1 in the VAns-based version for the Asian barrier spread seen in example 3.15. Each of the models shown is optimal regarding one of the four fitness criteria.



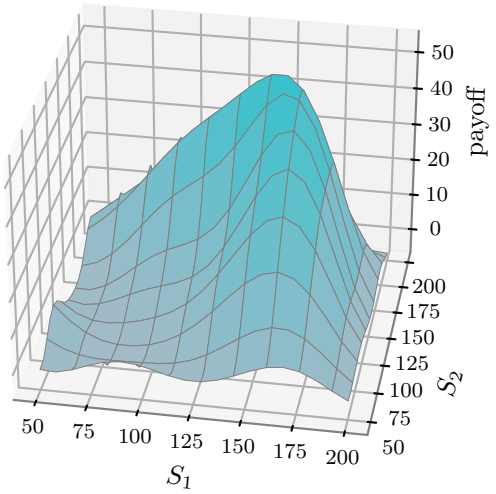
(a) Target



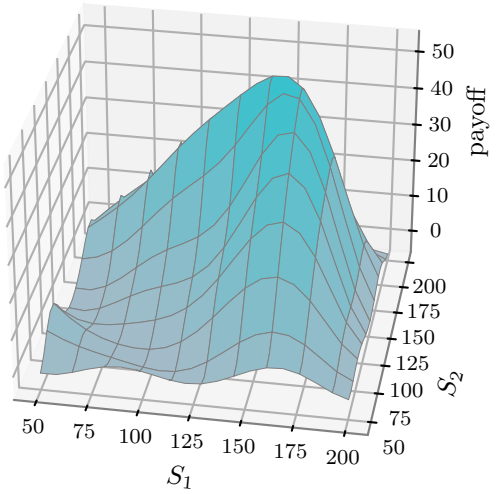
(b) MSE - figure D.61



(c) Circuit depth - figure D.62



(d) Number of controlled gates - figure D.63



(e) Number of input gates - figure D.64

Figure D.12: Predictions of four Pareto optimal models found by algorithm 3.1 in the ZX-diagram-based version for the Asian barrier spread seen in example 3.15. Each of the models shown is optimal regarding one of the four fitness criteria.

Table D.3: Results for the Asian barrier spread seen in example 3.15. All versions used 48 hours for the training. The lowest achieved value over all models is marked bold.

	Trained generations	Minimized criterion	MSE	Circuit depth	# Two-qubit gates	# Input encodings	Figure
Circuit-based	803	MSE	16.489	100	21	30	D.53
		Circuit depth	33.724	44	8	18	D.54
		# Two-qubit gates	31.708	46	8	16	D.55
		# Input encodings	41.669	73	11	**9**	D.56
VAns-based	479	MSE	**15.798**	100	30	27	D.57
		Circuit depth	39.341	28	6	11	D.58
		# Two-qubit gates	43.514	32	4	10	D.59
		# Input encodings	41.459	48	8	**9**	D.60
ZX-graph-based	725	MSE	38.145	53	7	16	D.61
		Circuit depth	46.163	**23**	3	11	D.62
		# Two-qubit gates	45.515	28	**2**	11	D.63
		# Input encodings	46.163	**23**	3	11	D.64

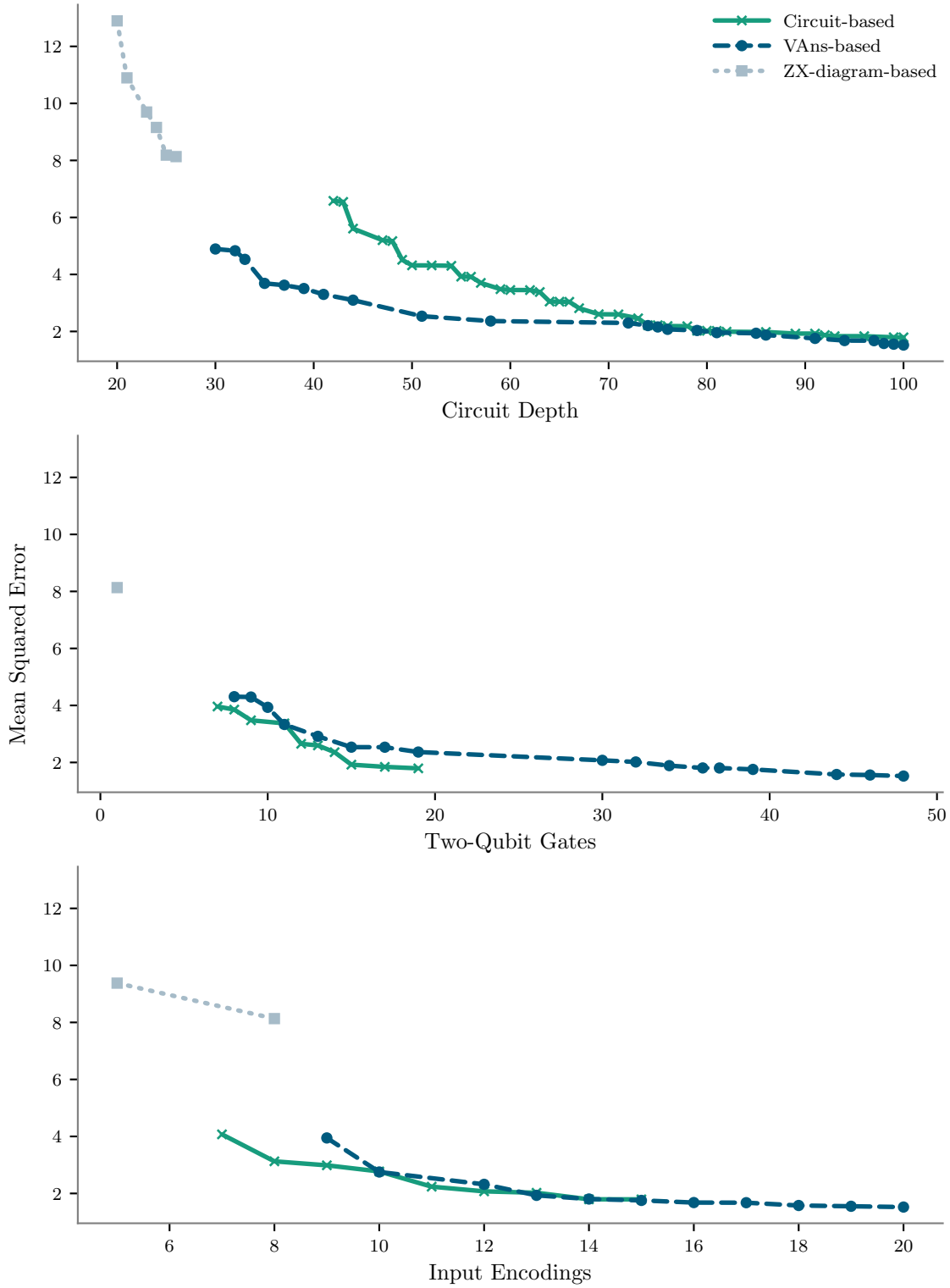


Figure D.13: Result of the genetic QAS with algorithm 3.1 for the Asian barrier spread seen in example 3.13. Projection of the Pareto front into the planes spanned by the MSE and the other three fitness criteria.

Table D.4: Results for the European basket call with variable weight seen in example 3.13. All versions used 72 hours for the training.

	Trained generations	Minimized criterion	MSE	Circuit depth	# Two-qubit gates	# Input encodings	Figure
Circuit-based	769	MSE	1.793	100	19	15	D.29
		Circuit depth	6.581	42	7	7	D.30
		# Two-qubit gates	5.335	51	7	9	D.31
		# Input encodings	5.448	51	8	7	D.32
VAns-based	86	MSE	**1.522**	100	48	20	D.33
		Circuit depth	5.661	30	9	9	D.34
		# Two-qubit gates	5.469	41	8	11	D.35
		# Input encodings	5.661	30	9	9	D.36
ZX-graph-based	822	MSE	8.133	26	**1**	8	D.37
		Circuit depth	12.895	**20**	**1**	7	D.38
		# Two-qubit gates	10.765	26	**1**	7	D.39
		# Input encodings	9.378	30	4	**5**	D.40

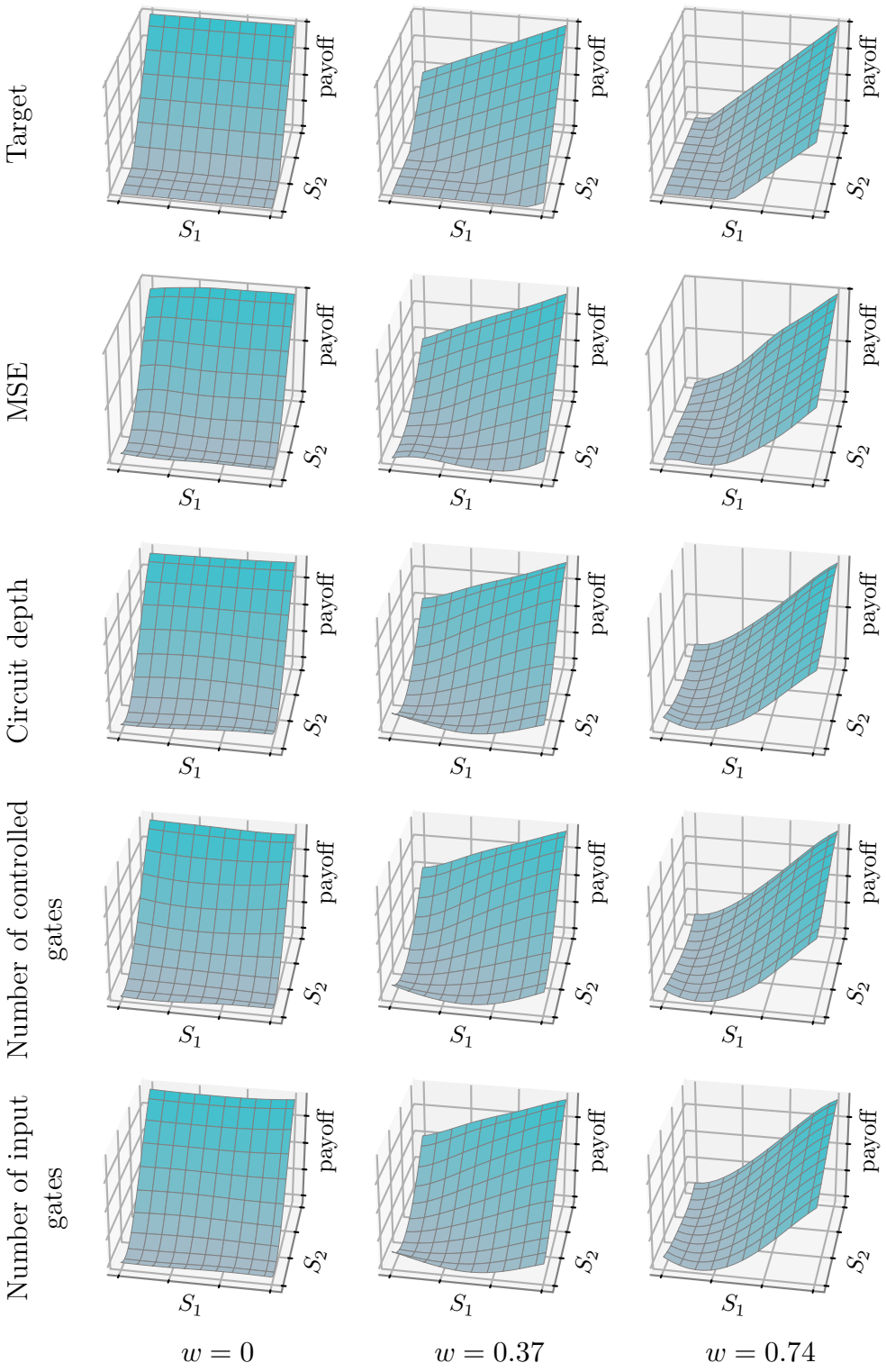


Figure D.14: Predictions of four Pareto optimal models found by algorithm 3.1 in the circuit-based version for the European basket call with variable weight seen in example 3.15. Each row of the grid represents one model that is optimal regarding one of the four fitness criteria and each column represents one value for the weight of the first asset. The first row is the target the models where trained to approximate.

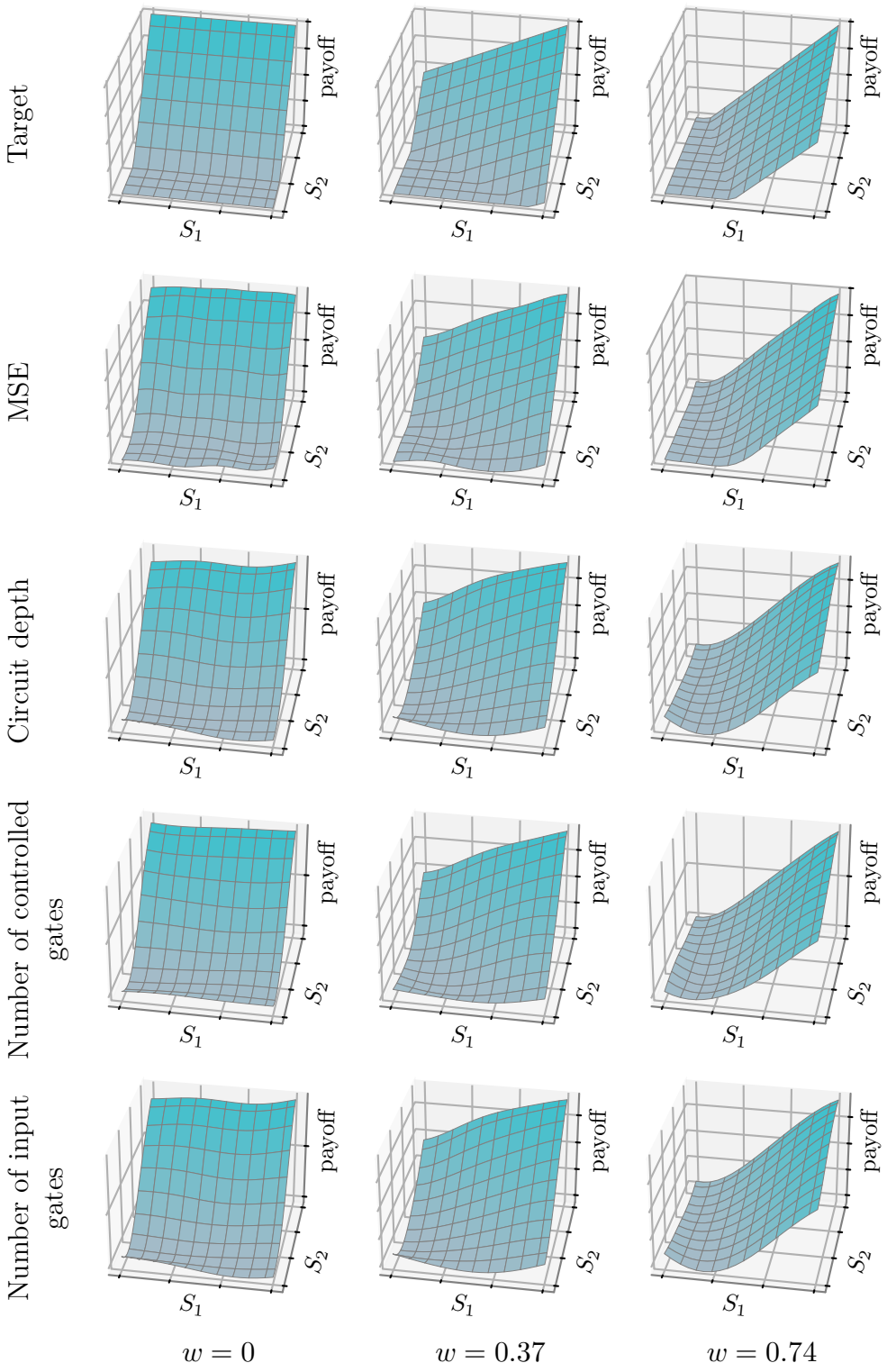


Figure D.15: Predictions of four Pareto optimal models found by algorithm 3.1 in the VAns-based version for the European basket call with variable weight seen in example 3.15. Each row of the grid represents one model that is optimal regarding one of the four fitness criteria and each column represents one value for the weight of the first asset. The first row is the target the models where trained to approximate.

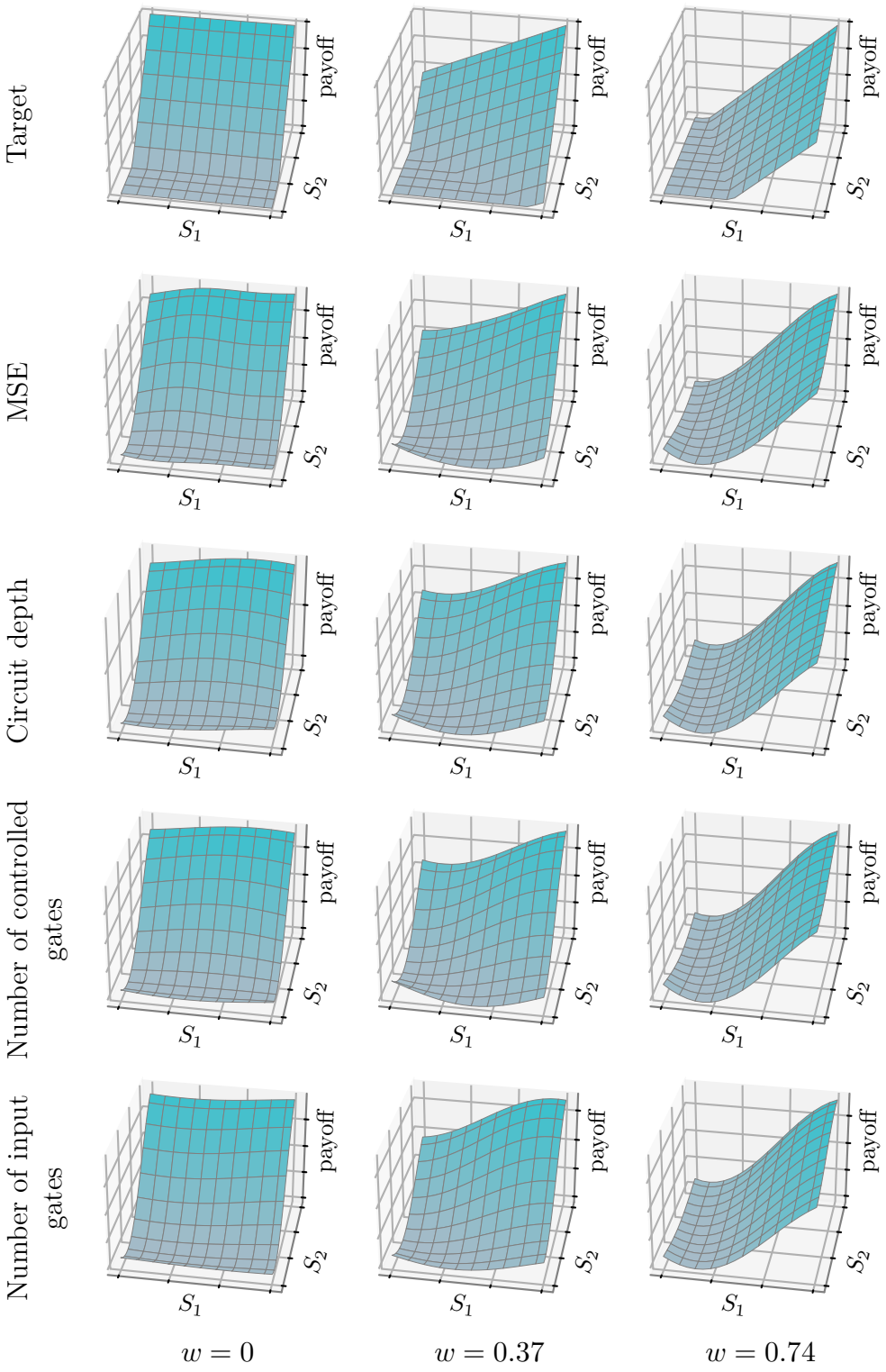


Figure D.16: Predictions of four Pareto optimal models found by algorithm 3.1 in the ZX-diagram-based version for the European basket call with variable weight seen in example 3.15. Each row of the grid represents one model that is optimal regarding one of the four fitness criteria and each column represents one value for the weight of the first asset. The first row is the target the models where trained to approximate.

Table D.5: Figure numbers and pages of the circuits generated by algorithm 3.1.

		Minimal MSE	Minimal circuit depth	Minimal # two-qubit gates	Minimal # input encodings
European call example 3.11	Circuits	3.9a, 107	3.9b, 107	3.9c, 107	3.9d, 107
	VAns	3.10a, 108	3.10b, 108	3.10c, 108	3.10d, 108
	ZX-diagrams	3.11a, 109	3.11b, 109	3.11b, 109	3.11b, 109
European basket call example 3.12	Circuits	D.17, 183	D.18, 183	D.19, 184	D.20, 184
	VAns	D.21, 185	D.22, 185	D.23, 185	D.24, 185
	ZX-diagrams	D.25, 186	D.26, 186	D.27, 186	D.28, 187
European basket call with variable weight example 3.13	Circuits	D.29, 187	D.30, 187	D.31, 188	D.32, 188
	VAns	D.33, 189	D.34, 189	D.35, 190	D.36, 190
	ZX-diagrams	D.37, 191	D.38, 191	D.39, 191	D.40, 191
European rainbow call example 3.14	Circuits	D.41, 192	D.42, 192	D.43, 192	D.44, 193
	VAns	D.45, 193	D.46, 193	D.47, 194	D.48, 194
	ZX-diagrams	D.49, 194	D.50, 195	D.51, 195	D.52, 195
Asian barrier spread example 3.15	Circuits	D.53, 196	D.54, 196	D.55, 196	D.56, 197
	VAns	D.57, 197	D.58, 197	D.59, 198	D.60, 198
	ZX-diagrams	D.61, 198	D.62, 198	D.63, 199	D.64, 199

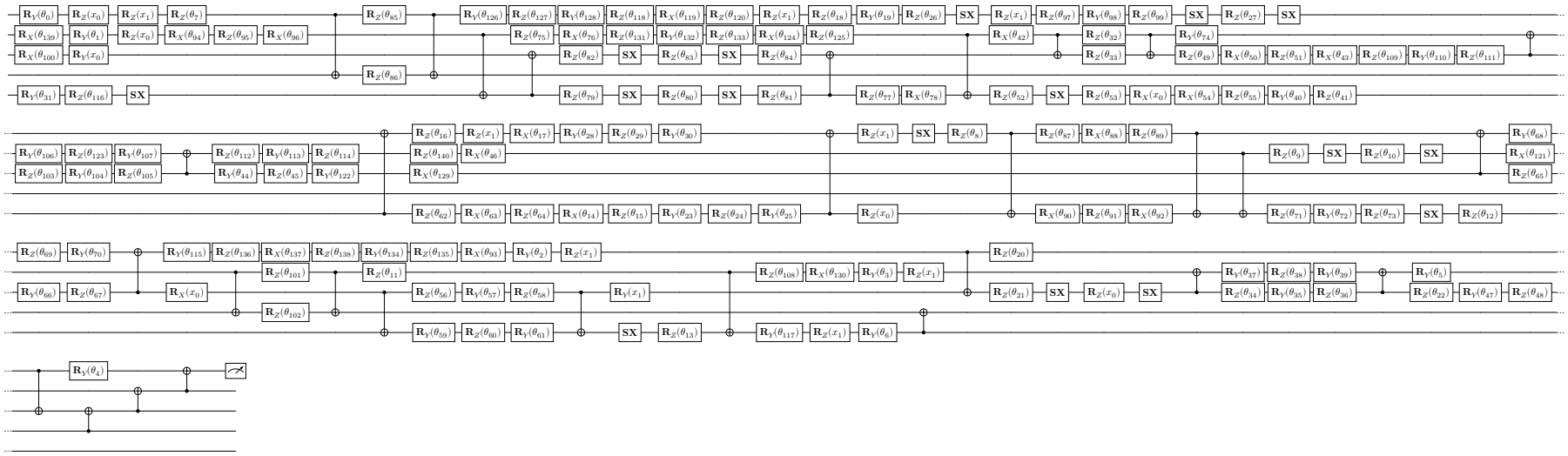


Figure D.17: Model with the lowest error generated by the circuit-based version of algorithm 3.1 for the European basket call seen in example 3.12.

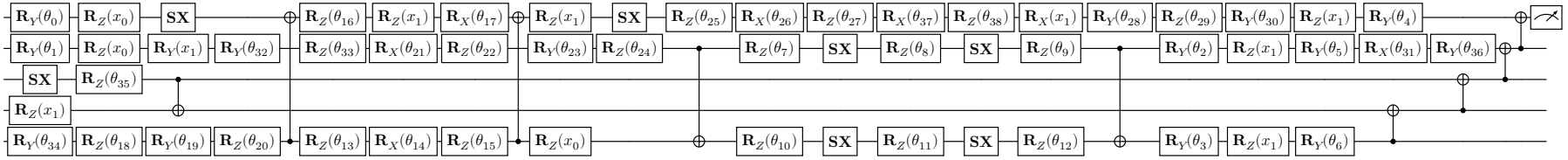


Figure D.18: Model with the lowest depth generated by the circuit-based version of algorithm 3.1 for the European basket call seen in example 3.12.

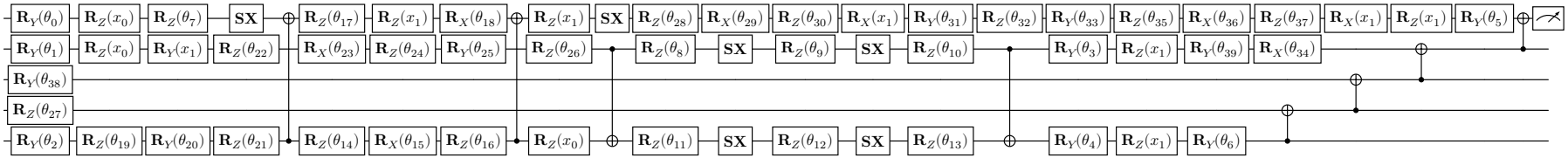


Figure D.19: Model with the fewest two-qubit gates generated by the circuit-based version of algorithm 3.1 for the European basket call seen in example 3.12.

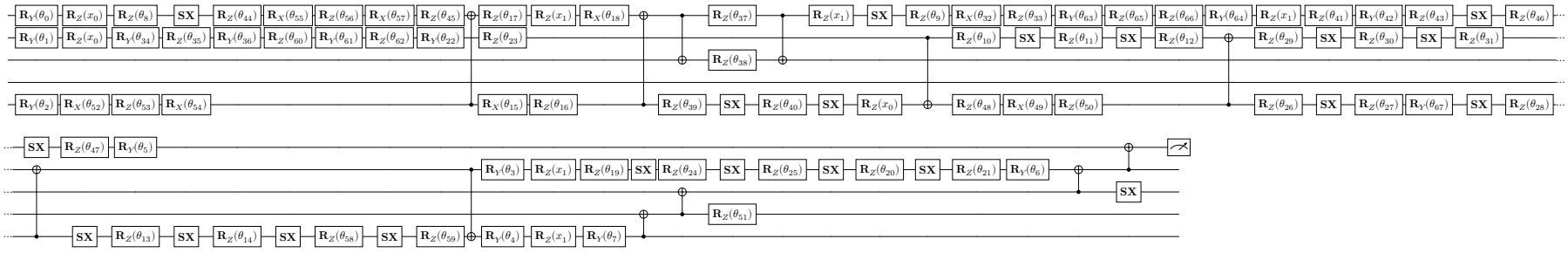


Figure D.20: Model with the fewest input encodings generated by the circuit-based version of algorithm 3.1 for the European basket call seen in example 3.12.

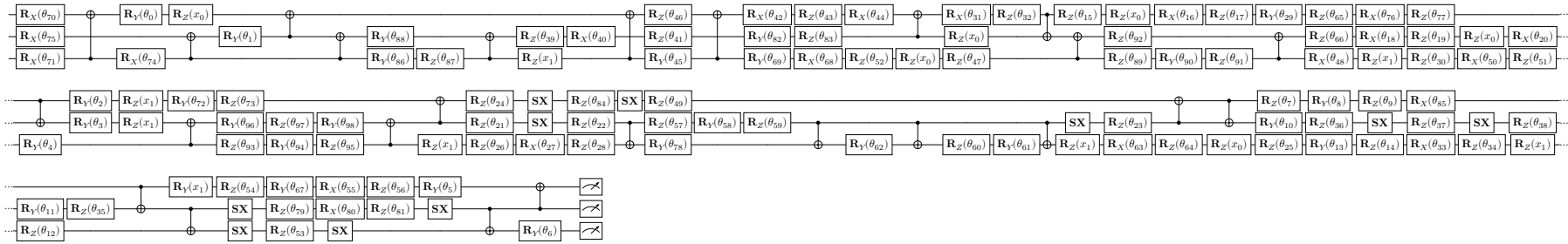


Figure D.21: Model with the lowest error generated by the VAns-based version of algorithm 3.1 for the European basket call seen in example 3.12.

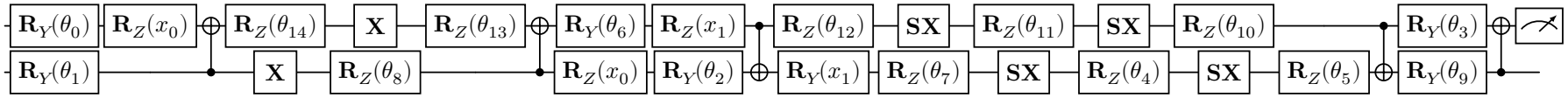


Figure D.22: Model with the lowest depth generated by the VAns-based version of algorithm 3.1 for the European basket call seen in example 3.12.

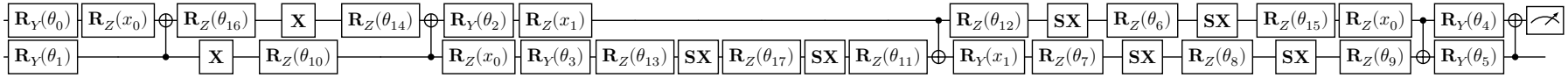


Figure D.23: Model with the fewest two-qubit gates generated by the VAns-based version of algorithm 3.1 for the European basket call seen in example 3.12.

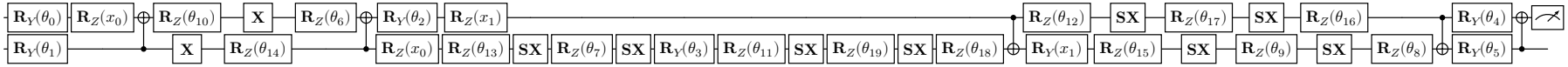


Figure D.24: Model with the fewest input encodings generated by the VAns-based version of algorithm 3.1 for the European basket call seen in example 3.12.

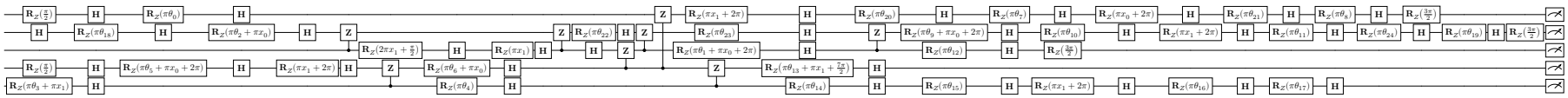


Figure D.25: Model with the lowest error generated by the ZX-diagram-based version of algorithm 3.1 for the European basket call seen in example 3.12.

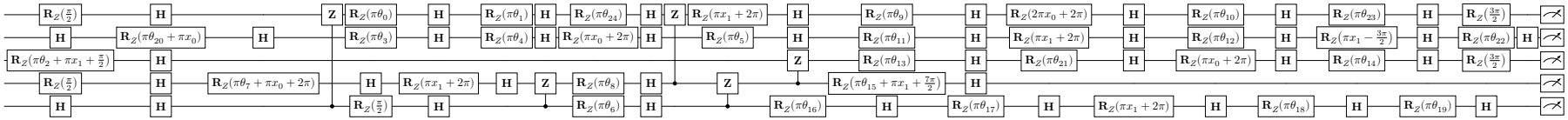


Figure D.26: Model with the lowest depth generated by the ZX-diagram-based version of algorithm 3.1 for the European basket call seen in example 3.12.

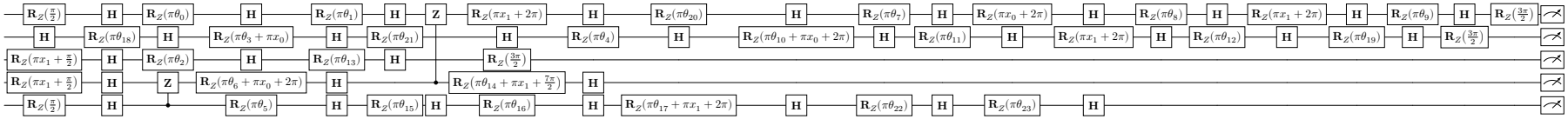


Figure D.27: Model with the fewest two-qubit gates generated by the ZX-diagram-based version of algorithm 3.1 for the European basket call seen in example 3.12.

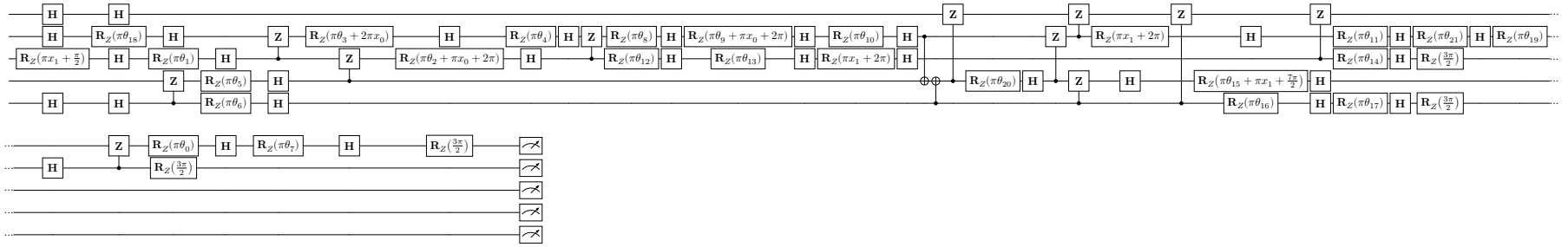


Figure D.28: Model with the fewest input encodings generated by the ZX-diagram-based version of algorithm 3.1 for the European basket call seen in example 3.12.

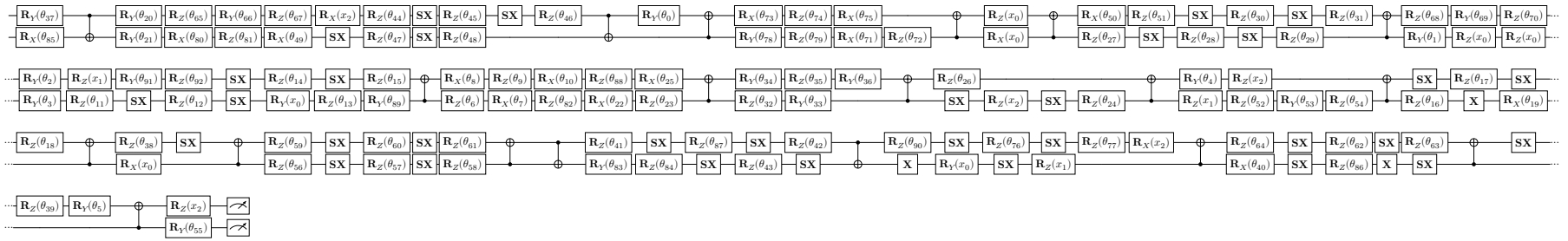


Figure D.29: Model with the lowest error generated by the circuit-based version of algorithm 3.1 for the European basket call with variable weight seen in example 3.13.

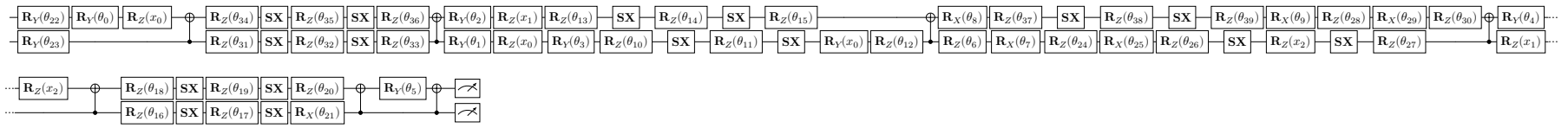


Figure D.30: Model with the lowest depth generated by the circuit-based version of algorithm 3.1 for the European basket call with variable weight seen in example 3.13.

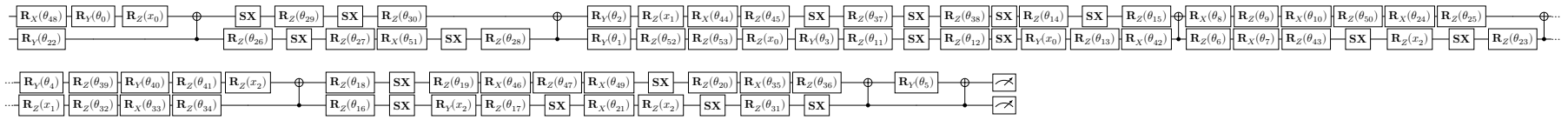


Figure D.31: Model with the fewest two-qubit gates generated by the circuit-based version of algorithm 3.1 for the European basket call with variable weight seen in example 3.13.

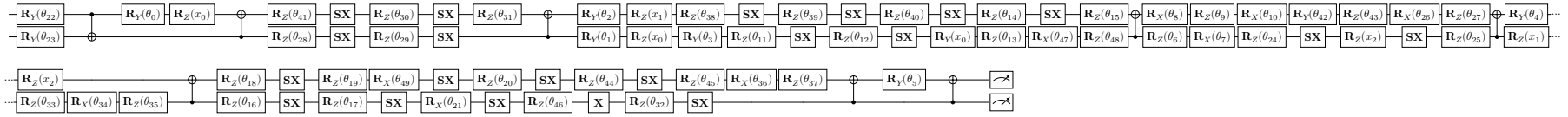


Figure D.32: Model with the fewest input encodings generated by the circuit-based version of algorithm 3.1 for the European basket call with variable weight seen in example 3.13.

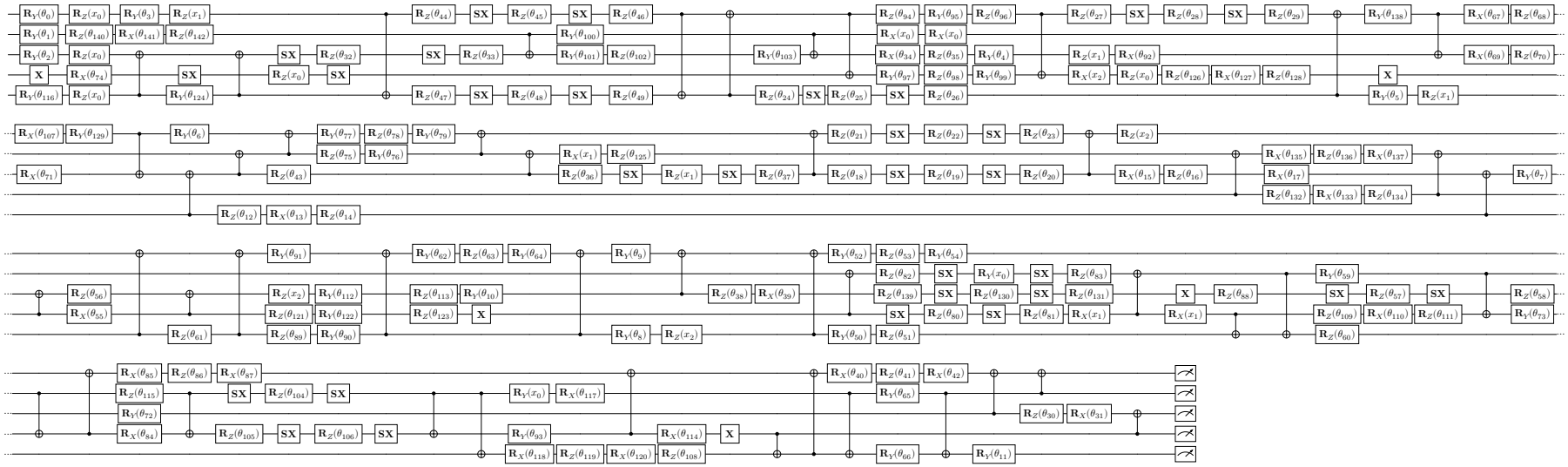


Figure D.33: Model with the lowest error generated by the VAns-based version of algorithm 3.1 for the European basket call with variable weight seen in example 3.13.

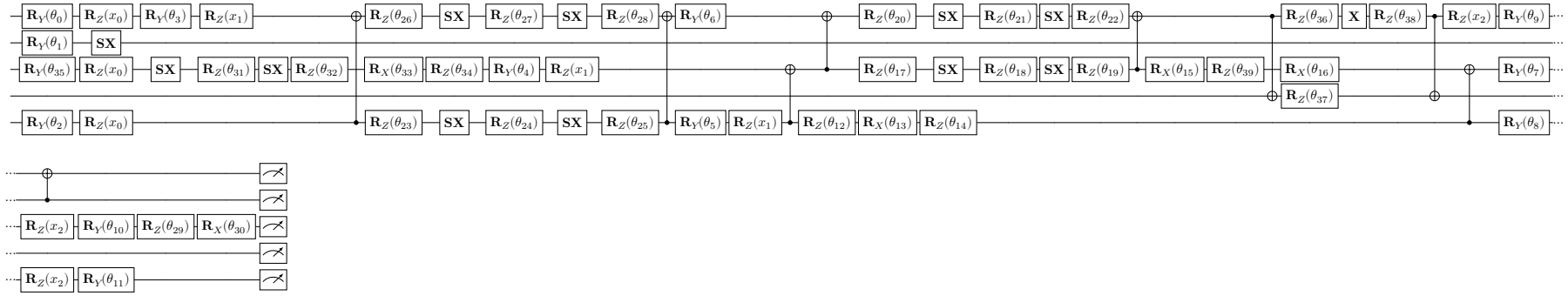


Figure D.34: Model with the lowest depth generated by the VAns-based version of algorithm 3.1 for the European basket call with variable weight seen in example 3.13.

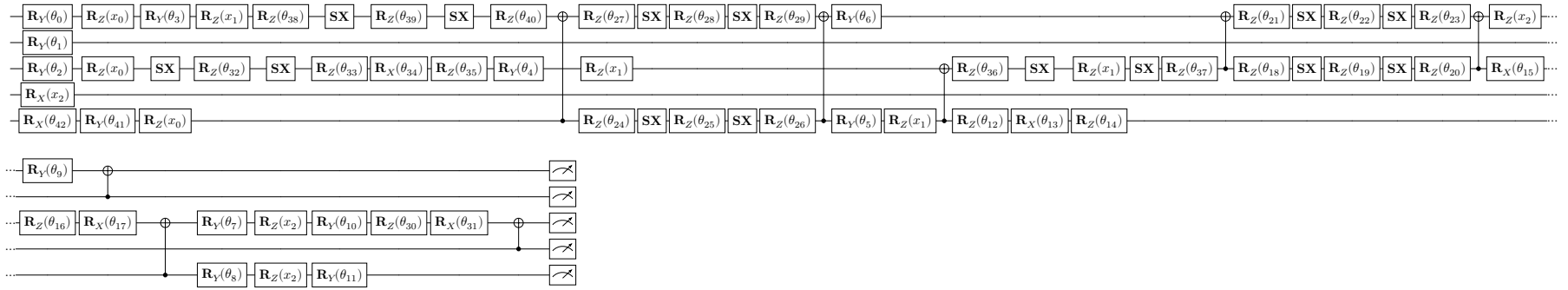


Figure D.35: Model with the fewest two-qubit gates generated by the VAns-based version of algorithm 3.1 for the European basket call with variable weight seen in example 3.13.

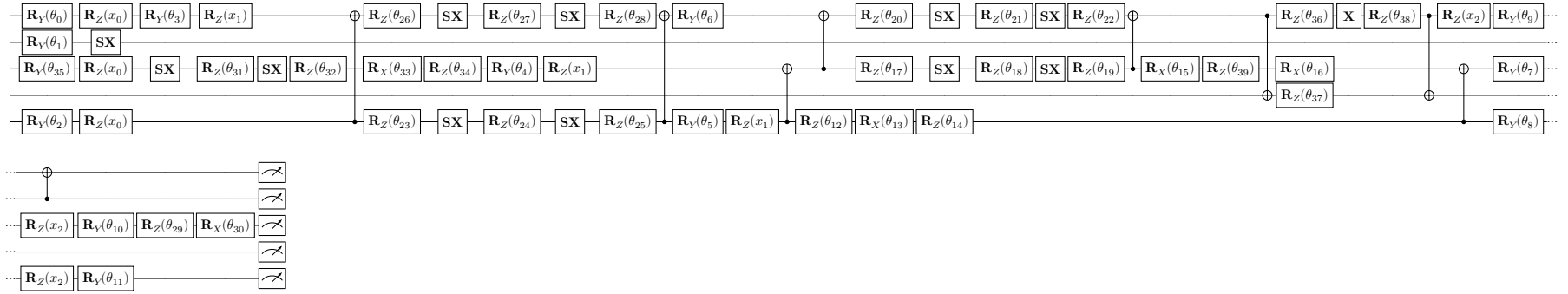


Figure D.36: Model with the fewest input encodings generated by the VAns-based version of algorithm 3.1 for the European basket call with variable weight seen in example 3.13.

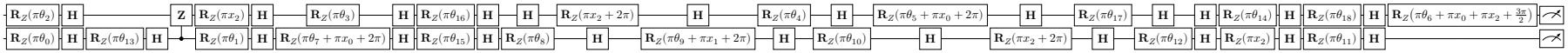


Figure D.37: Model with the lowest error generated by the ZX-diagram-based version of algorithm 3.1 for the European basket call with variable weight seen in example 3.13.

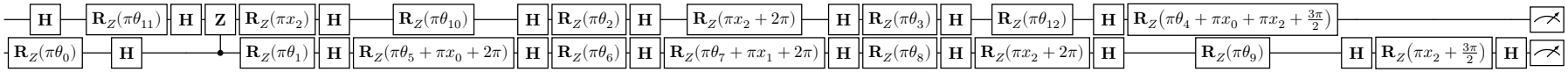


Figure D.38: Model with the lowest depth generated by the ZX-diagram-based version of algorithm 3.1 for the European basket call with variable weight seen in example 3.13.

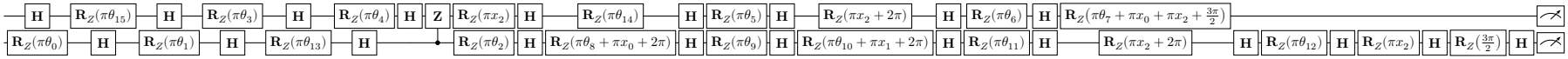


Figure D.39: Model with the fewest two-qubit gates generated by the ZX-diagram-based version of algorithm 3.1 for the European basket call with variable weight seen in example 3.13.

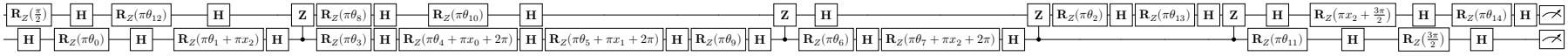


Figure D.40: Model with the fewest input encodings generated by the ZX-diagram-based version of algorithm 3.1 for the European basket call with variable weight seen in example 3.13.

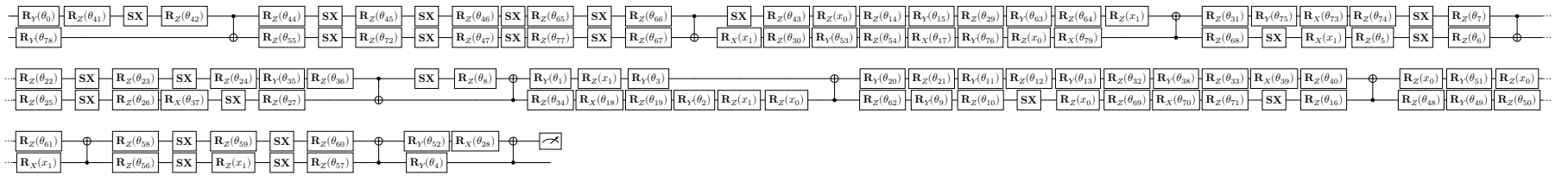


Figure D.41: Model with the lowest error generated by the circuit-based version of algorithm 3.1 for the European rainbow call seen in example 3.14.

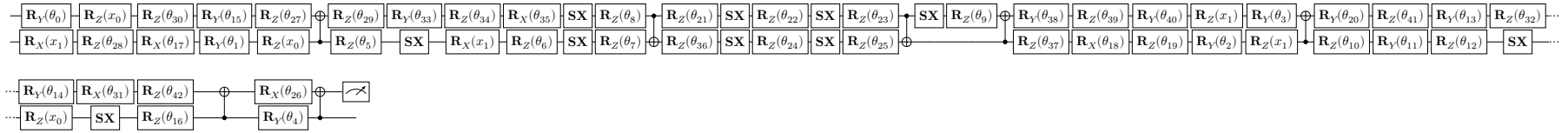


Figure D.42: Model with the lowest depth generated by the circuit-based version of algorithm 3.1 for the European rainbow call seen in example 3.14.

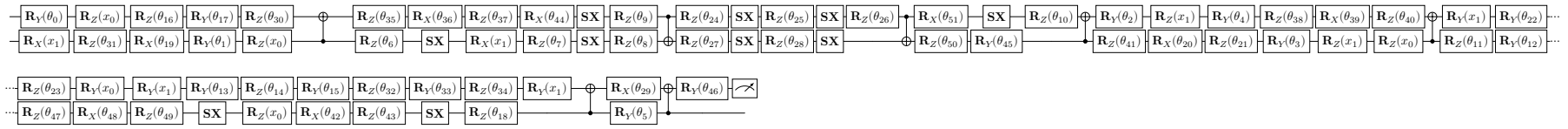


Figure D.43: Model with the fewest two-qubit gates generated by the circuit-based version of algorithm 3.1 for the European rainbow call seen in example 3.14.

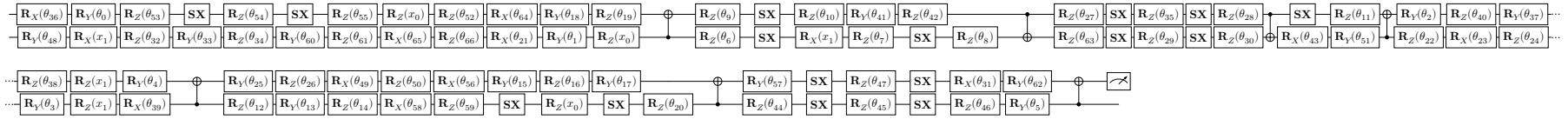


Figure D.44: Model with the fewest input encodings generated by the circuit-based version of algorithm 3.1 for the European rainbow call seen in example 3.14.

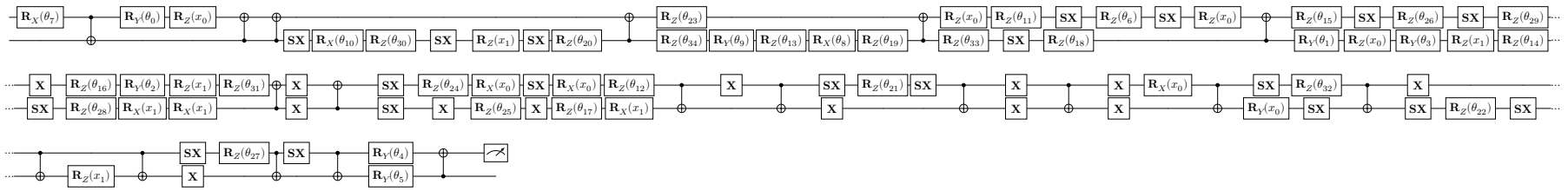


Figure D.45: Model with the lowest error generated by the VAns-based version of algorithm 3.1 for the European rainbow call seen in example 3.14.

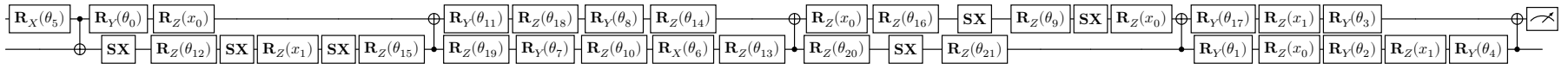


Figure D.46: Model with the lowest depth generated by the VAns-based version of algorithm 3.1 for the European rainbow call seen in example 3.14.

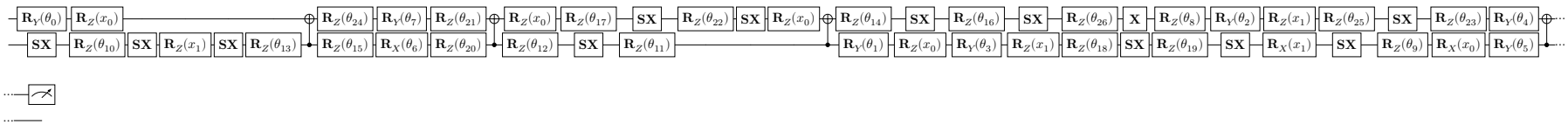


Figure D.47: Model with the fewest two-qubit gates generated by the VAns-based version of algorithm 3.1 for the European rainbow call seen in example 3.14.

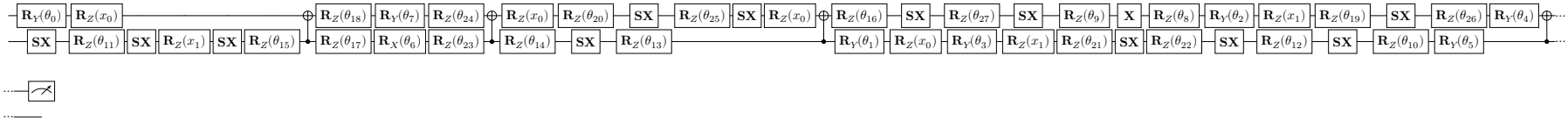


Figure D.48: Model with the fewest input encodings generated by the VAns-based version of algorithm 3.1 for the European rainbow call seen in example 3.14.

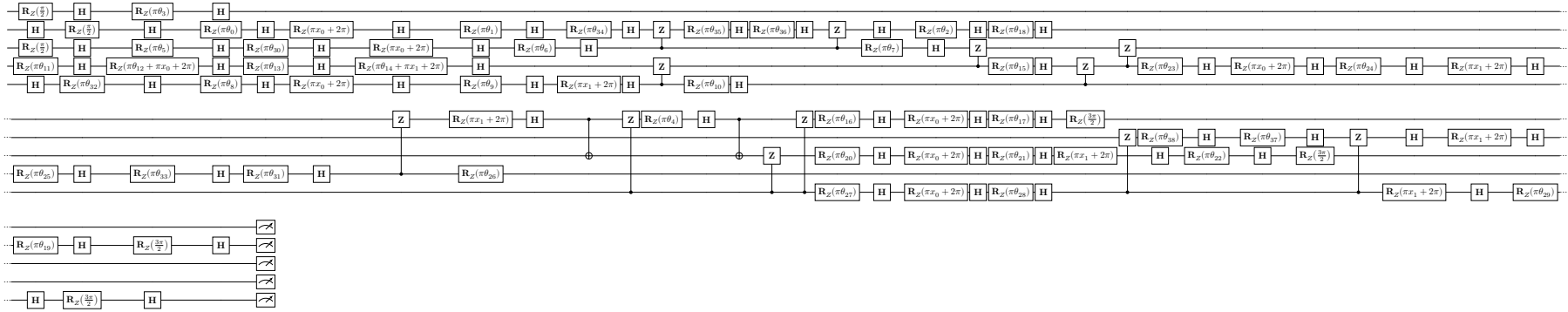


Figure D.49: Model with the lowest error generated by the ZX-diagram-based version of algorithm 3.1 for the European rainbow call seen in example 3.14.

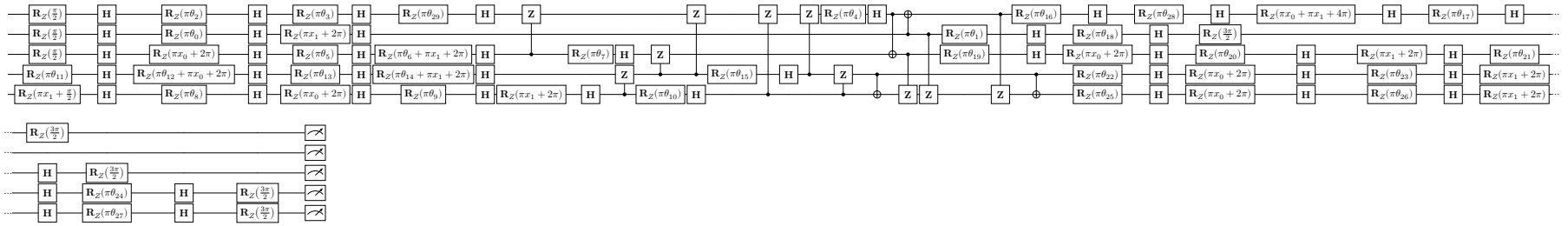


Figure D.50: Model with the lowest depth generated by the ZX-diagram-based version of algorithm 3.1 for the European rainbow call seen in example 3.14.

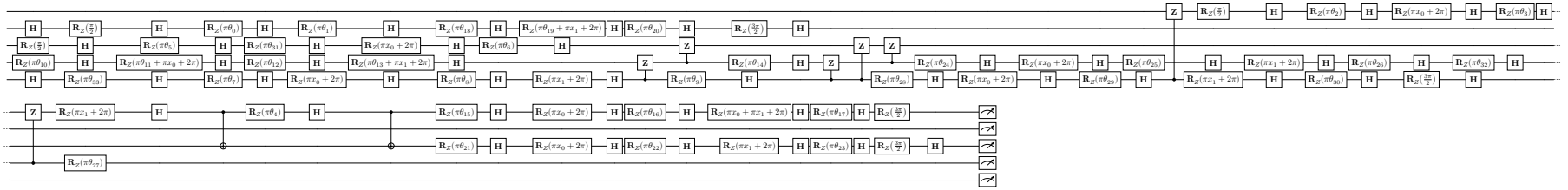


Figure D.51: Model with the fewest two-qubit gates generated by the ZX-diagram-based version of algorithm 3.1 for the European rainbow call seen in example 3.14.

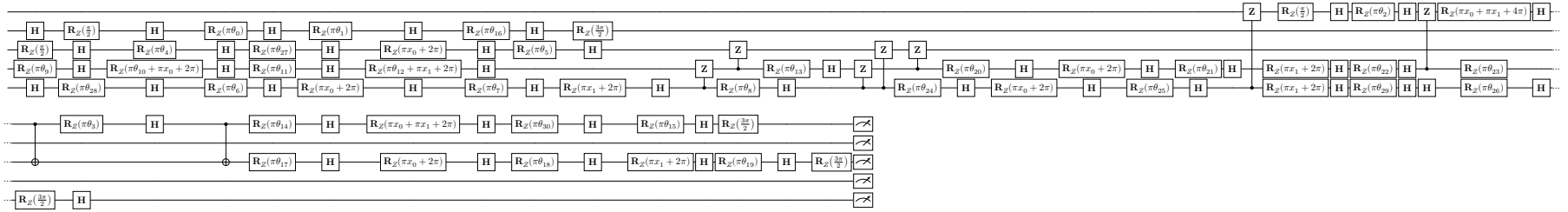


Figure D.52: Model with the fewest input encodings generated by the ZX-diagram-based version of algorithm 3.1 for the European rainbow call seen in example 3.14.

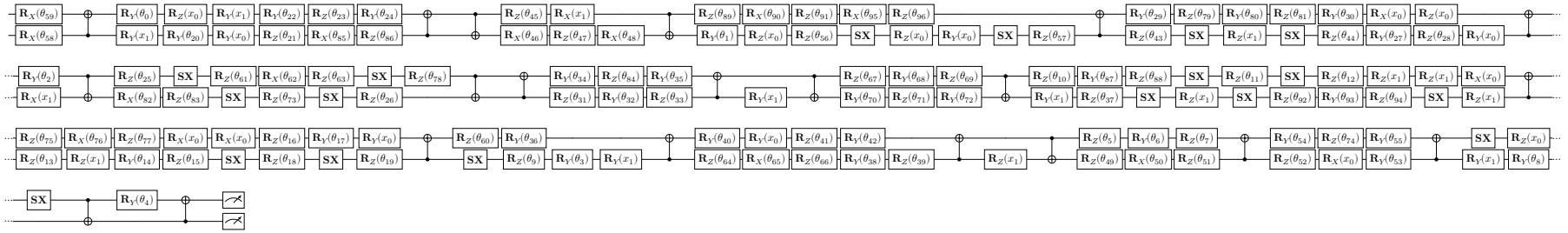


Figure D.53: Model with the lowest error generated by the circuit-based version of algorithm 3.1 for the Asian barrier spread seen in example 3.15.

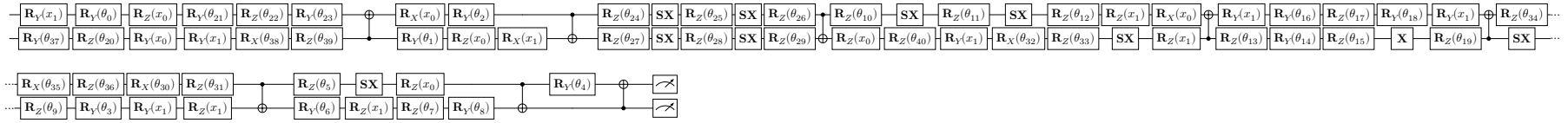


Figure D.54: Model with the lowest depth generated by the circuit-based version of algorithm 3.1 for the Asian barrier spread seen in example 3.15.

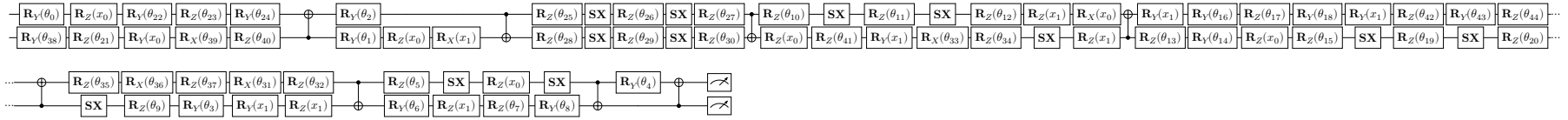


Figure D.55: Model with the fewest two-qubit gates generated by the circuit-based version of algorithm 3.1 for the Asian barrier spread seen in example 3.15.

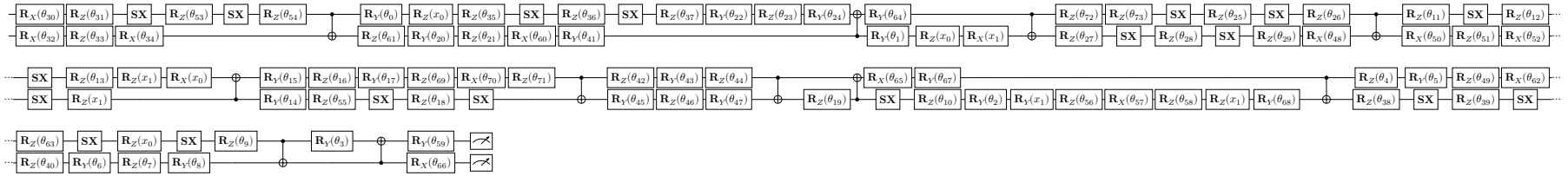


Figure D.56: Model with the fewest input encodings generated by the circuit-based version of algorithm 3.1 for the Asian barrier spread seen in example 3.15.

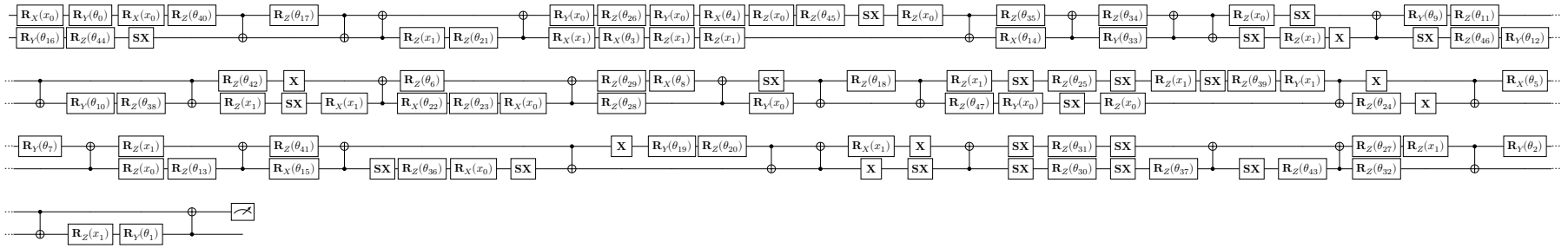


Figure D.57: Model with the lowest error generated by the VAns-based version of algorithm 3.1 for the Asian barrier spread seen in example 3.15.

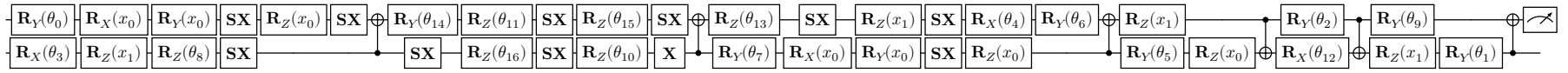


Figure D.58: Model with the lowest depth generated by the VAns-based version of algorithm 3.1 for the Asian barrier spread seen in example 3.15.

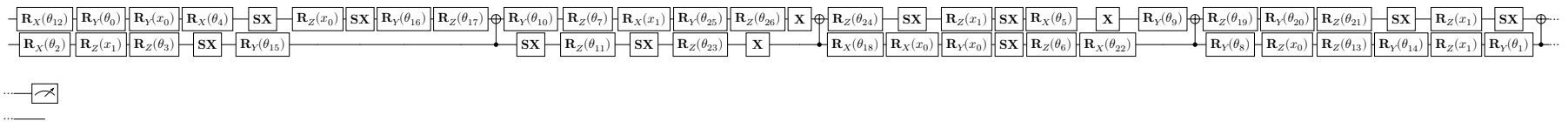


Figure D.59: Model with the fewest two-qubit gates generated by the VAns-based version of algorithm 3.1 for the Asian barrier spread seen in example 3.15.

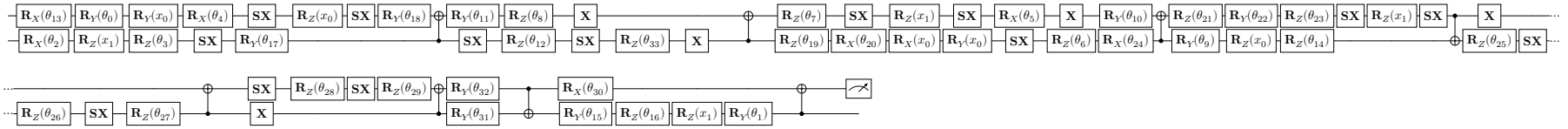


Figure D.60: Model with the fewest input encodings generated by the VAns-based version of algorithm 3.1 for the Asian barrier spread seen in example 3.15.

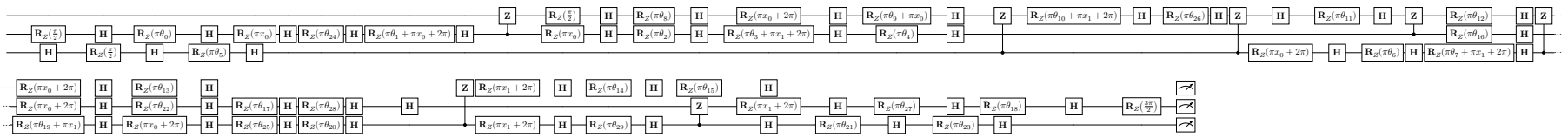


Figure D.61: Model with the lowest error generated by the ZX-diagram-based version of algorithm 3.1 for the Asian barrier spread seen in example 3.15.

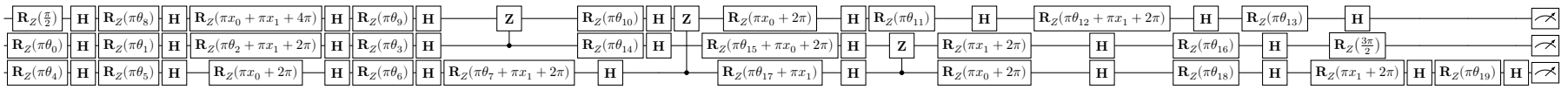


Figure D.62: Model with the lowest depth generated by the ZX-diagram-based version of algorithm 3.1 for the Asian barrier spread seen in example 3.15.

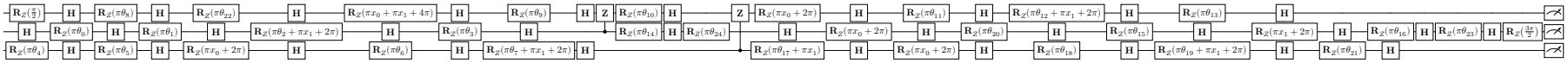


Figure D.63: Model with the fewest two-qubit gates generated by the ZX-diagram-based version of algorithm 3.1 for the Asian barrier spread seen in example 3.15.

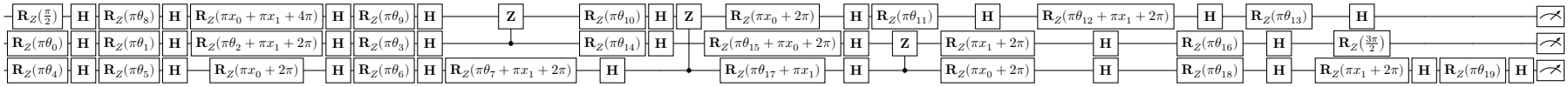


Figure D.64: Model with the fewest input encodings generated by the ZX-diagram-based version of algorithm 3.1 for the Asian barrier spread seen in example 3.15.

Akademischer Lebenslauf

01/2022 - 03/2025	Promotion am Fachbereich Mathematik der RPTU Kaiserslautern-Landau *Quantum Computing in Option Pricing*.
04/2018 - 03/2025	Wissenschaftlicher Mitarbeiter am Fraunhofer Institut für Techno- und Wirtschaftsmathematik (ITWM) in der Abteilung Finanzmathematik.
03/2018	Master of Science in Mathematik mit Vertiefung Finanzmathematik und Nebenfach Wirtschaftswissenschaften an der TU Kaiserslautern.
11/2015	Bachelor of Science in Mathematik mit Nebenfach Wirtschaftswissenschaften an der TU Kaiserslautern.
09/2015 - 03/2018	Wissenschaftliche Hilfskraft am Fraunhofer ITWM in der Abteilung Finanzmathematik.
2012	Abitur am Lichtenberg Gymnasium Darmstadt.

Academic Curriculum Vitae

01/2022 - 03/2025	PhD at the department mathematics of the RPTU Kaiserslautern-Landau *Quantum Computing in Option Pricing*.
04/2018 - 03/2025	Scientific associate at the Fraunhofer Institute for Industrial Mathematics ITWM in the department Financial Mathematics.
03/2018	Master of Science in Mathematics with a specialization in Financial Mathematics and a minor in Economics at TU Kaiserslautern.
11/2015	Bachelor in Mathematics with a minor in Economics at TU Kaiserslautern.
09/2015 - 03/2018	Research Assistant at Fraunhofer ITWM in the Department of Financial Mathematics.
2012	Abitur at Lichtenberg Gymnasium Darmstadt.