Capturing dynamics with noisy quantum computers

Proefschrift

ter verkrijging van
de graad van doctor aan de Universiteit Leiden,
op gezag van rector magnificus prof. dr. S. de Rijcke,
volgens besluit van het college voor promoties
te verdedigen op dinsdag 17 februari 2026
klokke 14:30 uur

door

David Simon Dechant

geboren te Fulda, Duitsland
in 1995

An electronic version of this thesis can be found
at https://openaccess.leidenuniv.nl

Cover designed by Zhenya Cheipesh.

Meinen Eltern, für ihre Unterstützung und Erziehung

Contents

CHAPTER 1

Introduction

1.1 Preface

In the decades that followed the foundation of quantum mechanics, scientists have been fascinated by the wide range of both practical and theoretical implications that this field has opened up. In particular, with the rise of classical computers, scientists like Yuri Manin, Richard Feynman and David Deutsch proposed the idea of using quantum mechanics at the core of computations, giving birth to the field of quantum information [1–3]. This new paradigm was believed to help in computations related to the simulation of systems described by quantum mechanics. In the 1990s, surprising theoretical results showed the possibilities that quantum computing has outside of applications in the quantum domain: Shor's algorithm demonstrates the ability to calculate the prime factors of a number, taking only time that scales polynomially in the size of that number, making the algorithm exponentially faster than known classical algorithms [4]. Grover's search algorithm can search any quantum database for a specific marked item quadratically faster than classical counterparts [5]. In the years since, the field of quantum algorithms has grown considerably both in the diversity of algorithms and the range of applications [6]. In parallel, the application of quantum information to other areas like communication and cryptography has created new insights [7, 8].

These theoretical results have motivated the search for hardware that could process quantum computations. At the current stage, many different physical implementations are being considered and researched as candidates for useful quantum computing [9, 10]: For example, superconducting qubits [11, 12], trapped ions [13, 14], neutral atoms [15], quantum dots (for example, spin-based systems [16]), photonic systems [17]. In recent years, considerable progress could be made on the experimental front, especially with platforms focusing on superconducting qubits. In 2019, for the first time, calculations were implemented that seemed unfeasible on classical computers [18] and in 2024 significant strides were made toward scalable error-correction [19]. However, so far no practical use case seems to be within reach of current quantum devices. Therefore, significant progress toward fault-tolerant quantum computing is still necessary, both in experimental setups and on the theoretical side, such as in quantum algorithms.

One of the approaches of quantum computing that might fit current and near-term platforms well, are variational quantum algorithms [20, 21], as they do not rely on fault-tolerant quantum hardware. They consist not solely of quantum computations, but as a classical-quantum feedback loop, alternating steps between classical and quantum processors. They face several challenges, like statistical noise, gate errors, decoherence of the qubits and measurement errors.

For these near-term devices, people have investigated applications in areas such as combinatorial optimization [22], quantum chemistry [23], finance [24], machine learning [25] and solving differential equations [26]. In this thesis, we will focus in particular on three application areas: Solving differential equations, quantum machine learning, and finance.

Differential equations describe dynamical systems, which is why they are at the core of most physical theories and are applied widely in several industries, constituting one of the most common computational problems on supercomputers. Solving them efficiently with quantum computers would therefore be very useful. Many ideas have been explored for how quantum computing can benefit to machine learning. Using quantum computing as subroutines in machine learning could provide benefits in universality, training and generalization for tasks like regression, classification and generative modeling. Furthermore, the finance industry, as a calculation-heavy sector, could offer additional applications of quantum computing in areas such as option pricing, portfolio optimization, and synthetic data generation.

In Sec. 1.2, we introduce certain basics of quantum computing, variational quantum algorithms, and shot noise. Afterwards, we introduce solving differential equations, quantum machine learning, and quantum finance as applications of variational quantum computing in Sec. 1.3. Finally, we present an overview of Chapters 2, 3, 4, and 5 of this thesis in Sec. 1.4.

1.2 Basics of quantum computing

In this section, we introduce several basics of quantum computing that are used in this thesis. We start by defining qubits, quantum circuits and quantum measurements, then proceed with variational quantum algorithms and end with an introduction to shot noise, a source of error arising in quantum measurements. The reader who is familiar with these topics can safely skip to Sec. 1.3.

1.2.1 Qubits, circuits and measurements

Quantum states, the fundamental objects in quantum mechanics, are unit vectors in a complex vector space with an inner product, a *Hilbert space* [9]. While they can be infinite-dimensional, we focus here on finite-dimensional Hilbert spaces of dimension $d \in \mathbb{N}$, which are isomorphic to $\mathbb{C}^d$. *Quantum computing* is a paradigm that uses quantum systems called *qubits*, whose states are described by vectors in two-dimensional Hilbert spaces, to perform calculations. We call $\{|0\rangle, |1\rangle\}$ the *computational basis*, which forms an orthonormal basis for this space. Any qubit $|\psi\rangle \in \mathbb{C}^2$ can be written in the following way:

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

where α and β are complex numbers that are normalized such that $|\alpha|^2 + |\beta|^2 = 1$. Equivalently, a general quantum state $|\psi\rangle$ is normalized via the *Hilbert-Schmidt inner product*: $\langle\psi|\psi\rangle = \langle\psi| \times |\psi\rangle = 1$, where $\langle\psi| = |\psi\rangle^\dagger$ is the conjugate transpose of $|\psi\rangle$.

We can describe several qubits together, which we call a *quantum register*. For n qubits $|\phi_i\rangle$, where $i \in \{1, \ldots, n\}$, the joint state $|\psi\rangle \in \mathbb{C}^{2^n}$ is formed by applying a tensor product between them:

$$|\psi\rangle = |\phi_1\rangle \otimes |\phi_2\rangle \otimes \cdots \otimes |\phi_n\rangle \ . \tag{1.2}$$

States of this form are called fully product states, but not every multipartite state can be expressed this way. The state $|\psi\rangle$ can be written in the computational basis of the joint Hilbert space, formed by tensor products of single-qubit basis states. Because of the possible combinations of $|0\rangle$ and $|1\rangle$, there are 2^n different basis states for $|\psi\rangle$, and we use the notation

$$|0...01\rangle = |0\rangle \otimes ... \otimes |0\rangle \otimes |1\rangle \ . \tag{1.3}$$

In general, we write

$$|\psi\rangle = \sum_{i=0}^{2^n-1} \alpha_i |i\rangle \ , \tag{1.4}$$

where the basis states are labeled as $\{|i\rangle : i \in \{0, ..., 2^n - 1\}\}$. Again, the amplitudes α_i are normalized such that $\sum_{i=0}^{n-1} |\alpha_i|^2 = 1$.

The *Hamiltonian* $\mathcal{H}$ is a linear operator that is *Hermitian* (which means $\mathcal{H} = \mathcal{H}^\dagger$) and acts on the Hilbert space of a quantum system. It determines the dynamics of the system, hence how it changes in time. This dynamics is described by the *Schrödinger equation*:

$$-i\frac{\partial}{\partial t}|\psi(t)\rangle = \mathcal{H}|\psi(t)\rangle \tag{1.5}$$

Solving this equation yields a unitary operator (which means $U^\dagger U = UU^\dagger = I$) given by $U = \exp(-i\mathcal{H}t)$. In quantum computing, such unitaries are implemented using *quantum circuits*, which are sequences of *quantum gates*, elementary unitary operators that act on one or a few qubits at a time [27]. Common examples include the single-qubit *Pauli gates* X, Y, and Z, their respective rotations $R_X(\theta)$, $R_Y(\theta)$ and $R_Z(\theta)$, the *Hadamard gate* H, and two-qubit gates such as the *controlled-NOT (CNOT) gate.* These gates form a universal set, meaning that arbitrary unitary operations can be approximated to any desired precision using sequences of such gates. They are defined in the following way:

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \quad Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}, \tag{1.6}$$

$$R_X(\theta) = e^{-i\theta X/2} \quad R_Y(\theta) = e^{-i\theta Y/2}, \quad R_Z(\theta) = e^{-i\theta Z/2} \tag{1.7}$$

$$H = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}, \quad \text{CNOT} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}. \tag{1.8}$$

The CNOT gates create *entanglement*, a quantum phenomenon where the state of a qubit can not be described independently of the states of the other qubits, and which is essential for generating non-classical correlations in the circuit.

Given an input quantum register, $|\psi_{\text{in}}\rangle$, a quantum circuit U transforms it to an output register $|\psi_{\text{out}}\rangle$ by the transformation:

$$|\psi_{\text{out}}\rangle = U|\psi_{\text{in}}\rangle \ . \tag{1.9}$$

The final state $|\psi_{\text{out}}\rangle$ resulting from the quantum circuit, exists only as a quantum state, inaccessible directly in a classical understanding, because observation of the state collapses the wave function [28]. Instead, one can perform *quantum measurements* described by the set $\{M_m\}$ of measurement operators. The index m describes the outcome of the measurement, and the probability $p(m)$ that it occurs is equal to

$$p(m) = \langle\psi_{\text{out}}| M_m^\dagger M_m |\psi_{\text{out}}\rangle . \tag{1.10}$$

A particularly relevant class of measurements are *observables*, Hermitian operators whose eigenvalues represent the possible outcomes and whose eigenstates define the measurement basis. An example of such an observable is the Hamiltonian, whose eigenvalues correspond to the system's energy spectrum. For single qubits, another example is the measurement in the computational (or Z) basis, with measurement operators $\{|0\rangle\langle 0|, |1\rangle\langle 1|\}$. In practice, tensor products of single-qubit Pauli operators, known as *Pauli strings*, are often chosen as the set of measurement operators. They are typically decomposed into single-qubit gates applied to ψ_{out} and subsequent measurements of each qubit in the computational basis. *Quantum state tomography* aims to reconstruct the classical description of the quantum state and can be implemented by measuring Pauli strings, and estimating their corresponding probabilities $p(m)$, or by more general measurement schemes. We describe a particular method for quantum state tomography in Chapter 4.

1.2.2 Variational quantum algorithms

A *quantum algorithm* is a sequence of instructions designed to run on a quantum computer, typically tailored to solve a particular computational problem, given inputs of a specific form. Those instructions contain specific quantum circuits as well as descriptions of measurements that determine the result of the computational problem. Algorithms designed for fault-tolerant hardware often make use solely of quantum computation, such as Shor's and Grover's algorithms [4, 5]. In contrast, *variational quantum algorithms* consist of alternating calculations on quantum and classical hardware, as sketched in Fig. 1.1. They typically consist of *parameterized quantum circuits* and classical optimizers.

Parameterized quantum circuits consist of gates that can be tuned through parameters. A typical parameterized quantum circuit is the *hardware efficient Ansatz* [21, 29], which consists of single-qubit Pauli rotations and CNOT gates on neighboring qubits (see Eqs. (1.7) and (1.8)). In Fig. 1.2, we show an example of a parameterized quantum algorithm.

This Ansatz is designed to make optimal use of current hardware, where connectivity (the ability to entangle qubits), the available gate set, and the coherence time of qubits are limited. It is particularly useful for shallow quantum circuits [30].

In variational quantum algorithms, a parameterized quantum circuit $U(\theta)$ is repeatedly executed, where the gates depend on a set of tunable parameters θ. After each execution, a quantum measurement is performed to extract an output f_θ, typically the expectation value of an observable relevant to the

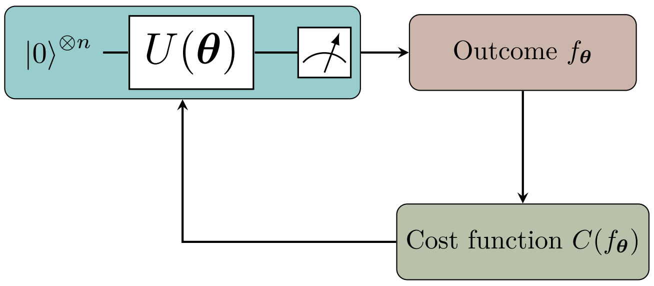

$|0\rangle^{\otimes n}$
$U(\boldsymbol{\theta})$
Outcome $f_{\boldsymbol{\theta}}$
Cost function $C(f_{\boldsymbol{\theta}})$

Figure 1.1: Sketch of a variational quantum algorithm. $U(\boldsymbol{\theta})$ represents a parameterized quantum circuit with variational parameters $\boldsymbol{\theta}$ that takes x as input, $f_{\boldsymbol{\theta}}$ is the expectation value of an observable and $C(f_{\boldsymbol{\theta}})$ is the cost function that is evaluated classically.

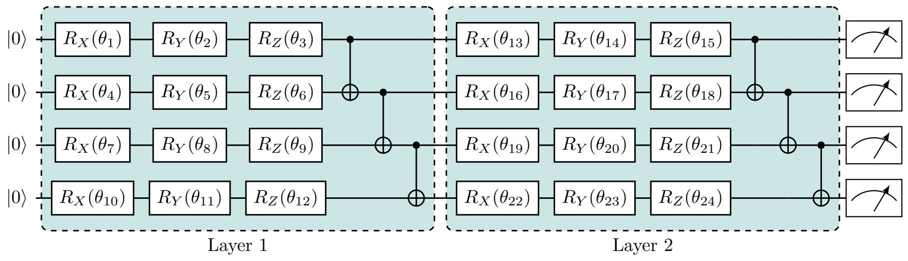

$|0\rangle$
$R_X(\theta_1)$ $R_Y(\theta_2)$ $R_Z(\theta_3)$
$R_X(\theta_4)$ $R_Y(\theta_5)$ $R_Z(\theta_6)$
$R_X(\theta_7)$ $R_Y(\theta_8)$ $R_Z(\theta_9)$
$R_X(\theta_{10})$ $R_Y(\theta_{11})$ $R_Z(\theta_{12})$
$R_X(\theta_{13})$ $R_Y(\theta_{14})$ $R_Z(\theta_{15})$
$R_X(\theta_{16})$ $R_Y(\theta_{17})$ $R_Z(\theta_{18})$
$R_X(\theta_{19})$ $R_Y(\theta_{20})$ $R_Z(\theta_{21})$
$R_X(\theta_{22})$ $R_Y(\theta_{23})$ $R_Z(\theta_{24})$
Layer 1
Layer 2

Figure 1.2: Example of a parameterized quantum circuit with 4 qubits and 2 layers, in the form of a hardware efficient Ansatz. Each layer consists of single-qubit Pauli rotations and CNOT gates. The single-qubit Pauli rotations are tunable with parameters θ_i.

problem. A *cost function*, defined to reflect the goal of the algorithm (e.g., energy minimization or quantum machine learning tasks such as described later), evaluates this measurement outcome. In most approaches, a classical optimizer updates the parameters θ based on this cost, which are then fed back to the parameterized quantum circuit. Another form of parameter updating that does not depend on the cost function is used for the quantum algorithms described in Chapter 3. Depending on the algorithm, this loop of quantum and classical calculations is repeated either for a preset number of iterations or until a specific accuracy is met.

1.2.3 Shot noise

The precision of quantum measurements, whether of observables or more general measurement schemes, is impacted by various sources of errors. Depending on the quantum hardware, errors can arise from gate noise, qubit decoherence, and measurement noise. Additionally, the inherent statistical nature of quantum measurements introduces errors known as *shot noise*, which is present even in perfectly noiseless quantum devices. For a simple example, assume the final quantum state is $|\psi_{\text{out}}\rangle = \alpha |0\rangle + \beta |1\rangle$, and we measure the Pauli-Z observable. In each measurement, the outcome is either $|0\rangle$ or $|1\rangle$ and the probability to observe the first is equal to $|\alpha|^2$, while the probability to observe the latter is $|\beta|^2$. Now suppose that a total number of N_{meas} measurements are performed on the quantum register. After each measurement, the quantum state has to be prepared again, since the act of measurement collapses the state. The expectation value of the observable will then be estimated as

$$E(Z) = \frac{1}{N_{\text{meas}}} \sum_{k=1}^{N_{\text{meas}}} m^{(k)}, \tag{1.11}$$

where $m^{(k)} = 1$ if at the k-th measurement, $|0\rangle$ is observed, and $m^{(k)} = -1$, if $|1\rangle$ is observed. Hence, the expectation value $E(Z)$ will lie between-1 and 1. The law of large numbers specifies that as the number of measurements N_{meas} increases, the expectation value $E(Z)$ converges to the true mean value $\mu(Z) = |\alpha|^2 - |\beta|^2$. This means that for a finite number of measurements, the expectation value will generally deviate from the true mean value. See Fig. 1.3 for a sketch of this deviation, which is commonly called the shot noise. Several statistical bounds, such as the *Chebyshev inequality*, provide estimates on the likelihood of this shot noise deviation. It states that for a chosen $\delta > 0$ and for finite and non-zero variance for a single measurement σ^2_{single}, the following inequality holds:

$$P(\|E(Z) - \mu(Z)\| \geq \delta) \leq \frac{\sigma^2_{single}}{\delta^2 N_{\text{meas}}} \,. \tag{1.12}$$

Since measurement outcomes are discrete ($+1$ or -1), the maximum variance per shot is 1. In other words, with probability at least $1 - \delta$, the deviation between the expectation value and the true mean is bounded by a shot noise error of at most $(\delta^2 N_{\text{meas}})^{-1}$. This implies that to reduce the shot noise by a factor of two, the number of measurements must be increased by a factor of four, since the statistical error δ scales as $1/\sqrt{N_{\text{meas}}}$. In practice, achieving very high precision can therefore become costly in time and resources.

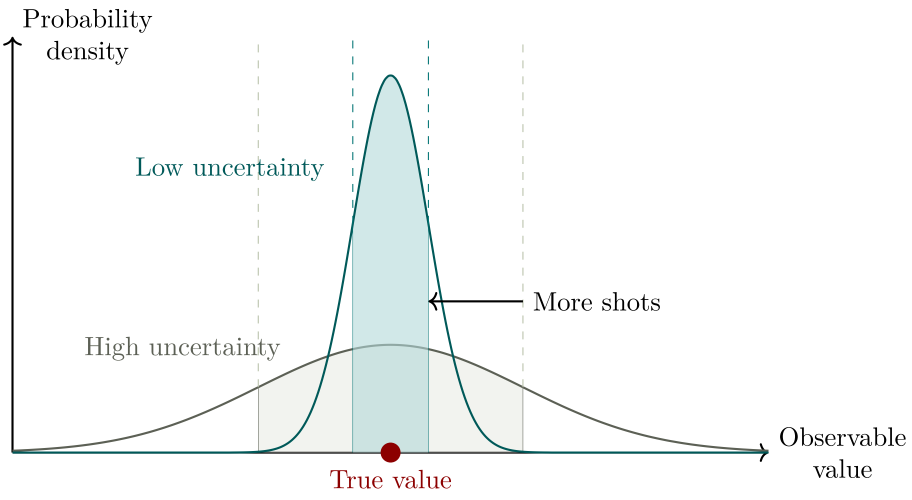


Figure 1.3: Increasing the number of shots in quantum measurements reduces the uncertainty of the measured observable. The shaded areas in this sketch show the probability density functions within standard deviation, indicating the area in which the true value lies with a probability of $\approx 68,27\%$. This interval becomes narrower with a higher number of shots taken, which reduces the uncertainty of the estimate.

1.3 Some applications of variational quantum computing

In this section, we introduce solving differential equations, quantum machine learning and finance as examples of applications of variational quantum computing.

1.3.1 Differential equations

Differential equations describe a wide range of phenomena across many fields. This is especially true for physics, where theories are often based on the description of physical systems with differential equations. Famous examples are Newton's laws of motion, Maxwell's equations, Einstein's field equations and the Schrödinger equation, providing the basis of classical mechanics, electrodynamics, relativity and quantum mechanics, respectively. They describe how quantities change depending on variables such as time (for dynamical systems) or space (for stationary systems). Due to these fundamental reasons, problems in industry can also be described by differential equations, such as the *Navier-Stokes equation* for the application in weather modeling and engineering or the *Black-Scholes equation* for applications in finance. In order to use these equations for making quantitative statements, it is necessary to find solutions that are in accordance with these differential equations.

In some cases, it is possible to find exact and explicit descriptions of these functions, known as *closed-form solutions*. However, this is in general only possible for very simple, artificial cases. In most cases, it is necessary to use numerical methods in order to approximate solutions to these differential equations. The computational cost to solve these can be prohibitively expensive. For example, solving the Navier-Stokes equation, which describes fluid dynamics, can be computationally very expensive [31]. This equation lies at the basis of applications such as weather and climate modeling or aircraft and car design. Therefore, finding efficient methods for solving differential equations is an important challenge in science and engineering.

Differential equations can be categorized in different ways [32]. Most commonly, they are grouped into *ordinary differential equations* (ODEs), *partial differential equations* (PDEs), and other types such as *stochastic differential equations* (SDEs) [33]. Further, they are distinguished into *linear* or *nonlinear differential equations*, depending on whether the terms of the equation are linear or nonlinear in the dependent variables and their derivatives.

ODEs are differential equations that consist of differentials with respect to a single independent variable. A simple example of a linear ODE is the following:

$$\frac{df(x)}{dx} = y(x) \ . \tag{1.13}$$

If the equation involves derivatives with respect to more than one independent variable, it is called a PDE. For instance, the *heat equation* that describes how heat spreads over time, is given by [34]:

$$\frac{df(\vec{x})}{dt} = \frac{d^2f(\vec{x})}{dx_1^2} + \frac{d^2f(\vec{x})}{dx_2^2} + \frac{d^2f(\vec{x})}{dx_3^2} \ . \tag{1.14}$$

Differential equations containing a stochastic process are called stochastic differential equations. SDEs are used to model systems with inherent randomness, such as Brownian motion describing the movement of pollen in water, or the Black-Scholes model describing the dynamics of financial markets. The SDE describing the dynamics of the stock price $S(t)$ in the latter model is [33]:

$$dS(t) = \mu S(t)dt + \sigma S(t)dW_t \ . \tag{1.15}$$

It is written in differential notation, where $dS(t)$, dt, and dW_t represent infinitesimal changes in the stock price, time, and Wiener process, respectively. The scalars μ and σ denote the drift and standard deviation (volatility) of the stock price.

Quantum algorithms for solving differential equations

Several methods have been proposed for using quantum computers to approximate solutions to differential equations, ranging from algorithms requiring fault-tolerant quantum hardware to variational methods suitable for near-term quantum devices. A special focus in research is on possible advantages compared to classical methods. Often, these advantages come from the fact that many methods are based on discretizing the differential equation and that this discretization can efficiently be stored on a quantum computer due to the exponential scaling of the dimension of the Hilbert space with the number of qubits (see Sec. 1.2.1). Some of the most common approaches using fault-tolerant quantum computers [35, 36], are based on the *Harrow-Hassidim-Lloyd (HHL) algorithm* [37]. It prepares, under certain conditions, a quantum state proportional to the solution of a linear system $A\vec{x} = \vec{b}$, where A is a sparse and well-conditioned matrix. Many classical approaches for solving differential equations are based on discretization, such as finite difference or finite element methods, and can be reduced to solving such linear systems. Furthermore, linearization techniques provide methods for transforming nonlinear differential equations into linear systems [38, 39]. However, the HHL algorithm comes with the significant caveat that the solution is encoded in the amplitudes of the output state of the quantum circuit, and one can not simply "read out" the state [28]. Instead, it is necessary to repeat the same circuit multiple times and to estimate observables or global properties of the solution by repeated measurements.

Given the challenges of fault-tolerant quantum computing, much recent attention has been given to developing variational quantum algorithms for solving differential equations that can be run on near-term quantum devices. Several of these variational approaches have been proposed:

- Instead of applying the HHL algorithm, *variational linear system solvers* [40] are methods that calculate the variational parameters $\boldsymbol{\theta}$ which minimize $\|A\vec{x} - \vec{b}\|$, where $\vec{x}$ is prepared as a quantum state depending on $\boldsymbol{\theta}$. As with the HHL algorithm, this serves as a subroutine for solving differential equations [41, 42].

- *Quantum neural networks and physics-informed neural networks* [43] calculate the variationally encoded solution of a differential equation by minimizing cost functions which directly encode the differential equations, including boundary conditions and initial values [44].

- Many differential equations can be mapped to the Schrödinger equation [45], which makes it possible to use techniques from *Hamiltonian simulation* for approximating the resulting dynamics. The solution is then obtained by calculations of the time evolution of the parameters of the PQC [26, 46].

- For SDEs, *Quantum Monte Carlo methods* can be applied that use methods such as amplitude estimation or quantum random walks [47].

Each of these approaches exhibits different trade-offs in terms of range of applicability, scalability and noise resilience.

Throughout this thesis, modeling dynamics with variational quantum algorithms is explored from several angles: Chapters 2 and 5 deal with current challenges of solving differential equations and of modeling dynamical systems with quantum machine learning approaches using parameterized quantum circuits. Chapter 3 encompasses a rigorous error and resource analysis of algorithms for solving differential equations that are based on Hamiltonian simulation. Chapter 4 examines quantum state tomography, which is an essential subroutine for many variational quantum algorithms, especially those which require partial reconstruction of the quantum state to evaluate cost functions.

1.3.2 Elements of quantum machine learning

Machine learning is dealing with algorithms that learn from data without explicit programming [48]. It is commonly divided into three categories: *supervised learning*, where the model learns from labeled data, *unsupervised learning*, where it learns from unlabeled data, and *reinforcement learning*, where a machine learning agent learns by interaction with an environment. The currently most-studied and effective approaches are based on *neural networks* (NNs). One distinguishes between *shallow neural networks*, which have a single hidden layer, and *deep neural networks*, which consist of multiple hidden layers. Each layer consists of several hidden units. The number of hidden units per layer is referred to as the *width* and the number of layers determines the *depth* of the NN. See Fig. 1.4 for a sketch. Classical machine learning based on neural networks has made remarkable advancements in the past years, exceeding expectations in areas such as natural language processing (e.g., large language models like ChatGPT [49]), protein folding (e.g. AlphaFold [50]) and image generation (e.g., diffusion models [48], DALL-E [51]).

Quantum Machine Learning

The field of *quantum-enhanced (or quantum) machine learning* studies how quantum computing can be used to enhance machine learning models. Several results have shown that quantum machine learning enables certain speedups compared to classical machine learning models [43, 52–56]. Most of the current promising quantum machine learning algorithms are based on variational approaches, alternating between classical and quantum computations. The two most prominent paradigms for quantum machine learning are *kernel methods* and *quantum neural networks*. In kernel methods, a quantum machine is used

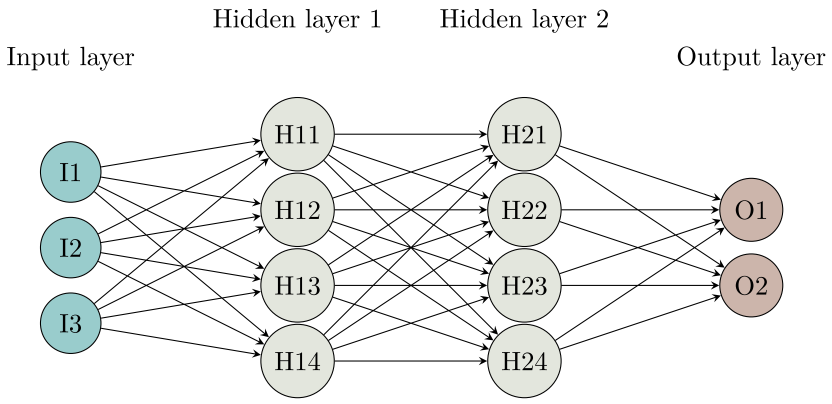


Figure 1.4: Sketch of a deep neural network with two hidden layers that each have four hidden units (H11-H14, H21-H24). The input layer (I1-I3) receives data and the output layer (O1, O2) produces the predictions of the network. The arrows show how information flows through the network, each corresponds to a trainable weight that is adjusted during training. Each layer is fully connected to the next.

to map input data into a quantum feature space, after which the inner product between the resulting vectors in the feature space are calculated [57]. The classically calculated cost function is then the weight of different inner products, and those weights are trained, while the quantum kernels remain untrained.

In addition to variational approaches, an influential class of quantum machine learning algorithms is based on linear algebraic techniques [58–61], building on the HHL algorithm [62]. Many of those have also motivated quantum-inspired classical algorithms, via the so-called *dequantization* [63, 64].

In this thesis, we focus on quantum neural networks [43]. A typical QNN is a variational quantum algorithm with a PQC that includes both fixed and tunable gates. While machine learning is typically divided into supervised, unsupervised and reinforcement learning, a significant amount of theoretical research deals with supervised learning. This is because it is easier to define what constitutes success in a learning task as there is an underlying ground truth (opposed to unsupervised learning) and because the agent does not interact with the training data (as is the case for reinforcement learning).

In theoretical analysis, machine learning is often characterized by three aspects: *Expressivity, training* and *generalization*. Here, we focus on two of them.

Expressivity

Expressivity deals with how well the model can represent the target function or distribution. Depending on the choice of model, its ability to approximate the ground truth varies. This property is called expressivity and can be mathematically formulated with *universal approximation theorems*. In 1989, George Cybenko proved that classical neural networks with a single hidden layer of arbitrary width can approximate any square-integrable function [65]. Later it was shown that deep neural networks can achieve universal approximation for a bounded width as well, by increasing the depth of the neural network [66].

In 2021, universality was proven for PQC as well, for one layer parameterized quantum circuits [67]. Further results have been given both for multivariate functions and for constant width multi-layer PQC [68–71]. While highly expressive quantum models are desirable for approximating a wide class of functions, increased expressivity often comes at the cost of trainability, leading to issues such as *local minima* and *barren plateaus* during optimization [25, 72].

Generalization

Generalization deals with how well a trained model performs on unseen data [48]. In supervised learning, the model is trained on a finite dataset $\mathcal{I}$, composed of input-target pairs $(x, f^*(x))$. As described in Sec. 1.2, for variational quantum algorithms such as those employed in quantum machine learning, a cost function is chosen depending on the problem at hand. Here, it is equivalent to the so-called *empirical risk*, which, given the training data, is minimized during training:

$$D(\hat{f}, f) = \frac{1}{|\mathcal{I}|} \sum_{(x, f^*(x)) \in \mathcal{I}} \ell(\hat{f}(x), f^*(x)) \;, \tag{1.16}$$

where $\ell(\cdot, \cdot)$ is a task-specific loss function (e.g., squared error) and $\hat{f}(x)$ are the predictions of the quantum model.

In contrast, the *(expected) risk* quantifies the average loss between the model's predictions and the true outputs over new data drawn from the underlying distribution $\mathcal{P}$:

$$\mathcal{E}_{\text{gen}}(\hat{f}, f) = \mathbb{E}_{x \sim \mathcal{P}} \left[\ell(\hat{f}(x), f^*(x)) \right] . \tag{1.17}$$

Understanding and bounding the difference between the expected risk and the empirical risk, $\mathcal{E}_{\text{gen}}(\hat{f}, f) - D(\hat{f}, f)$, also called the *generalization error*, is essential for reliably deploying quantum machine learning models [73].

To reduce the generalization error and avoid overfitting (the scenario in which the empirical risk is low, but the expected risk remains high), *regularization* techniques are often employed by adding a penalty term $\Omega(\hat{f})$, which effectively constrains the model function $\hat{f}$ [48]. This leads to the objective of minimizing the *regularized empirical risk*:

$$D_{\text{reg}}(\hat{f}, f) = D(\hat{f}, f) + \lambda\Omega(\hat{f}), \tag{1.18}$$

where $\lambda > 0$ is a hyperparameter.

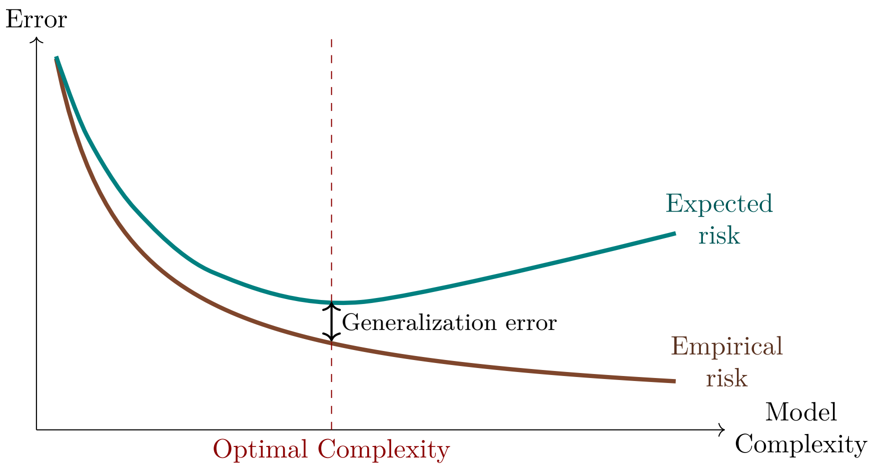


Figure 1.5: Illustration of empirical risk and expected risk as functions of model complexity. Here, *model complexity* refers to the expressiveness or capacity of the model class. This depends on factors such as the number of trainable parameters, circuit depth, or the number of qubits. As complexity increases, the empirical risk typically decreases, while the expected risk often exhibits a U-shaped curve due to overfitting. The optimal complexity minimizes the expected risk and the generalization error, defined as the gap between expected and empirical risk. Generalization bounds establish upper bounds on this gap.

1.3.3 Applications of quantum computing to finance

The financial industry is fundamentally driven by extensive calculations [74]. Many decisions depend on the outcomes of a wide range of computational tasks. Key examples are portfolio optimization, risk management, algorithmic trading and option pricing. Optimizing these tasks can yield substantial competitive advantages, motivating significant efforts to improve computational efficiency.

In light of this, several proposals have been made to apply quantum computing for computations in finance, motivated by the data-intense nature, probabilistic models, and applications of differential equations in finance.

In this chapter, we shall introduce the concepts of *financial time series* and *option pricing*, and outline how quantum computing can be leveraged in these contexts.

Time series

In finance, the traded products are called *financial instruments*, which, for example, are stocks, indices of stocks, or derivative contracts. Their prices over time are captured in a *time series*, a set of data points spaced equally in time. A sketch of such a time series is given in Fig. 1.6. An example is the daily closing value of the *S&P 500 index*, which is an index capturing the 500 largest publicly traded companies in the USA [75].

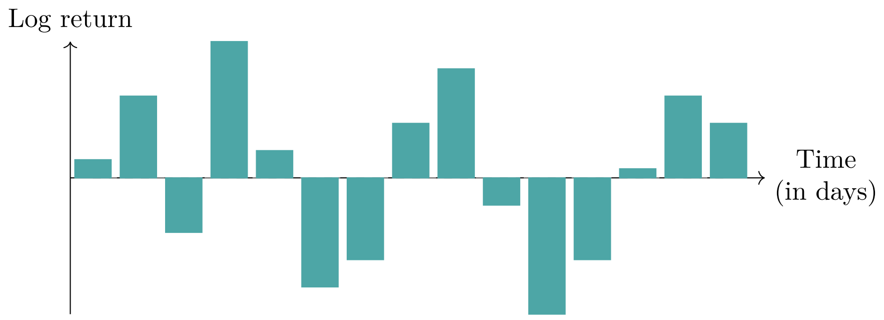


Figure 1.6: Sketch of a financial time series of log returns over several days, illustrating typical fluctuations around zero.

Although probabilistic in nature, financial time series share several consistent statistical properties, also called *stylized facts*. They are typically observed on the log return, which for a stock price S_t at time t is defined as

$$r_t = \log \frac{S_t}{S_{t-1}} \ . \tag{1.19}$$

Plotting the distribution of these log returns exhibits so-called *non-Gaussianity*. Compared to a Gaussian distribution, the tails are heavier, and the peak around the mean is sharper. Moreover, financial time series exhibit several temporal correlations: *volatility clustering, absence of linear autocorrelation* and the *leverage effect* [76]. Those properties will be explained in more detail in Chapter 5.

In recent years, the use of neural networks has been explored in modeling these time series [77]. However, their training requires access to large volumes of high-quality training data. But in practice, we only ever observe one realization of financial processes, namely the historical evolution of the market, which is further limited by data availability and the age of the market. To address this limitation, researchers have proposed methods of generating synthetic time series, which resemble the statistical properties of financial time series, and can be used in the training of neural networks [78]. Quantum computing has been proposed as a potential enabler for such tasks, due to its probabilistic nature and because quantum circuits have been proven to enable sampling from distributions which are intractable for classical circuits [43, 55, 56, 79]. Therefore, the set of distributions they access is in general different than what classical models access, and thus it is expected they may be more effective with some classes of distributions. In Chapter 5, we explore quantum machine learning models for the generation of synthetic financial time series.

Option pricing

Financial derivatives are instruments traded in the financial market [80]. They are contracts based on a specific asset, the underlying, and constitute a fundamental element of the risk management of trading agents. An example that is easy to analyze is the so-called *European call option*, which is a contract that grants the buyer the right, but not the obligation, to buy the underlying from the seller of the option for a specific price K at a specific time t_{final} in the future. As their fair price depends on the future price of the underlying, its calculation is cumbersome. In 1973, economists Black and Scholes introduced a simple mathematical model to describe the price of the European call option [81], for which the Nobel prize in economics in 1997 was awarded [82]. It relies on the simplifying assumption that the price of the underlying stock $S(t)$ behaves as a geometric Brownian motion, modeled as the following stochastic differential equation:

$$dS(t) = \mu S(t)dt + \sigma S(t)dW_t \ , \tag{1.20}$$

with a drift μ, volatility σ and the random variable dW_t, which is a Wiener process. It follows that the price V of a European call option satisfies the following partial differential equation, which can be derived from Eq. (1.20) via *Itô calculus*:

$$\frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} + rS\frac{\partial V}{\partial S} = rV \ , \tag{1.21}$$

where r is the risk-free rate, S denotes the current price of the underlying asset, and σ its volatility. For an agreed maturity t_{final} and strike price K, the solution

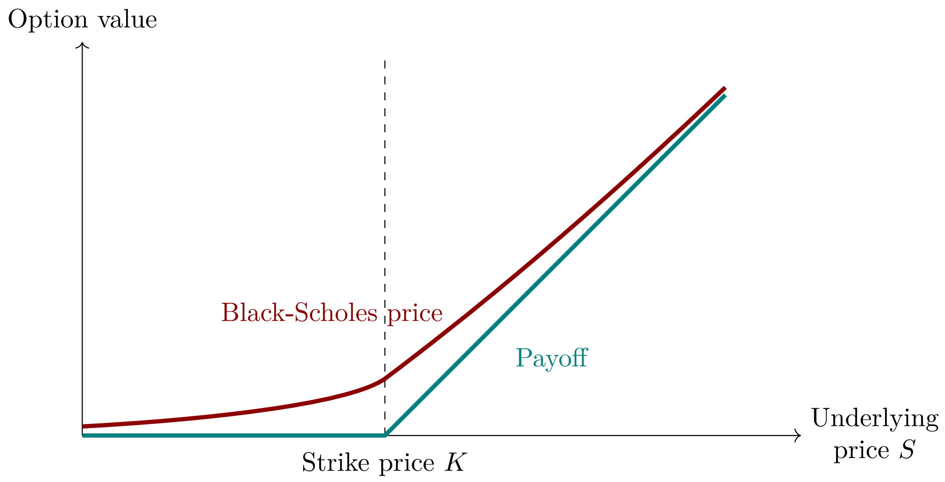


Figure 1.7: Sketch of the payoff (blue) and the Black-Scholes price (red) of a European call option at a specific time and with strike price K as a function of the underlying price S. The option price equals the payoff at maturity, and lies above it before maturity due to time value.

of this PDE together with the boundary condition

$$V(t_{\text{final}}, S) = \max\{S(t_{\text{final}}) - K, 0\} \ , \tag{1.22}$$

yields the fair price of the option. In Chapter 2, these concepts will be described in more detail.

Note that the choice of the dynamics (differential equation) is independent of the choice of the financial derivative (boundary condition). The Black Scholes equation (1.21) has an analytical solution for European options (boundary condition as in Eq. (1.22)) [81]. Nevertheless, solving it numerically is a toy model often used for benchmarking. However, the Black-Scholes model has several limitations for practical use. The most obvious one is the choice of the volatility σ to be constant, even though in reality, it varies in time. More complex dynamical models, such as the Heston model, the SABR model or the Merton jump-diffusion model, as well as a variety of derivative contracts, like Basket options and American options, build on the Black-Scholes model [74, 83].

There are several computational methods for solving the resulting differential equations, which, in this context, means approximating the function $V(S)$ describing the price of an option. Based on the stochastic formalism, it is possible to use methods like *binomial trees, Monte-Carlo methods*, or *spectral methods*, like the *Carr-Madan method* [74, 84, 85]. Based on the partial differential equation formalism, one can derive analytical solutions in rare cases, or use discretization methods such as the *finite difference method* [84]. Furthermore, in recent years, the use of neural networks has been explored [86].

Motivated by the variety of classical methods for option pricing, several quantum algorithms have been proposed, in the quest of finding possible advantages. As with solving differential equations, these advantages are motivated by properties of quantum computing like the exponential scaling of the Hilbert space. Typically, one can distinguish between approaches based on quantum versions of Monte-Carlo simulations and approaches based on solving the differential equations. The former approach is based on work by [87], which calculates option prices from payoff functions such as Eq. (1.22), based on stock prices modeled as stochastic processes, such as in the Black-Scholes model in Eq. (1.20).

Quantum approaches for solving differential equations for option pricing vary in methodology: There exist methods including quantum finite difference methods, based on quantum linear system solvers, and methods based on Hamiltonian simulation.

The latter approach is based on original insights from Baaquie [45] who demonstrated a mapping between the Black-Scholes differential equation and the Schrödinger equation, and we will study it in Chapter 3.

1.4 Outline of this thesis

The remaining chapters of this thesis shed light on several aspects of capturing dynamics with quantum computers. In Chapter 2, we examine universality and generalization (see Sec.1.3.2) in regression tasks where training data includes derivatives of functions. Since dynamical systems can mathematically be described by differential equations, the ability of parameterized quantum circuits (PQCs) to approximate and generalize not only functions but also their derivatives is essential for modeling those systems. In Chapter 3, we conduct an error and resource estimation of specific quantum algorithms proposed for solving differential equations. In particular, we consider the effects of shot noise, an inherent error source of many quantum computations, and evaluate the real-world applicability of these quantum algorithms. In Chapter 4, we present a method for mitigating the effects of shot noise in quantum state tomography, which is an important subroutine for many variational quantum algorithms. Finally, in Chapter 5, we explore QGANs (see 1.3.2) for generating synthetic financial time series, whose dynamics is stochastic in nature.

In the following, we present the research questions that are covered in Chapters 2, 3, 4, and 5.

While there exist results exploring the expressivity of parameterized quantum circuits in approximating square-integrable functions under the L^2 distance [67], the approximation for other function spaces and under other distances has been less explored. As we will see in Chapter 2, these existing results generally do not guarantee that both a function and its derivatives can be simultaneously approximated. However, in some applications such as solving differential equation, it is important to use models that are expressive enough for learning the derivative of the function as well. This motivates the first research question:

Research Question 1: *Can parameterized quantum circuits approximate functions as well as their derivatives?*

In Chapter 2, we answer this research question, based on the previously published work in Ref. [88]. We show that PQCs can approximate the space of continuous functions, p-integrable functions and the H^k Sobolev spaces under specific distances. The Sobolev spaces contain functions which are bounded with respect to the L^2 distance, and whose partial derivatives up to order k are bounded under this distance as well. However, for the approximation of continuous functions and those in the Sobolev space, it is necessary to rescale the input data. These results therefore answer the research question by proving that it is possible to approximate functions as well as their derivatives with PQCs, if the input data is rescaled accordingly.

Given these results, in the following research question we ask how we can train a model such that it is guaranteed to approximate the function and its derivatives well.

Research Question 2: *How does an augmentation of the training data with derivatives of the target function influence generalization in quantum machine learning with parameterized quantum circuits?*

We answer this question as well in Chapter 2, based on the results published in Ref. [88]. We prove generalization bounds that connect different function spaces and distances. In particular, we prove that it is possible to bound the generalization error such that it approximates both the function and its derivatives, if the training data includes not only labels of the target function, but also labels of its derivatives. Furthermore, we prove that including data of the derivatives of the target function also guarantees generalization bounds with respect to the supremum distance and the L^p distances. We find that the higher the dimension of the function, the more higher-order derivatives are required in order to achieve these bounds. These theorems give a theoretical explanation of earlier numerical findings that suggested improved generalization with classical neural networks [89], therefore also impact classical machine learning.

The results of research questions 1 and 2 provide a theoretical basis for different applications of PQCs, for example for solving differential equations. Furthermore, they provide us with new insight on the role of the data normalization in PQCs and of loss functions which better suit the specific needs of the users.

A focus of recent research in quantum computing has been on developing quantum algorithms for solving differential equations using variational methods on near-term quantum devices. A promising approach involves variational algorithms, which combine classical Runge-Kutta methods with quantum computations. However, a rigorous error analysis, essential for assessing real-world feasibility, has so far been lacking. We therefore ask the following research question:

Research Question 3: *What is the total error arising in variational quantum algorithms for solving differential equations based on Runge-Kutta methods and which Runge-Kutta order minimizes the number of circuit evaluations needed?*

In Chapter 3, we provide an extensive analysis of error sources and determine the resource requirements needed to achieve specific target errors, based on results published in Ref. [46]. In particular, we derive analytical error and resource estimates for scenarios with and without shot noise, examining shot noise in quantum measurements and truncation errors in Runge-Kutta methods. Our analysis does not take into account representation errors and hardware noise, as these are specific to the instance and the used device. We evaluate the implications of our results by applying them to two scenarios: classically solving a 1D ordinary differential equation and solving an option pricing linear partial differential equation with the variational algorithm, showing that the most resource-efficient methods are of order 4 and 2, respectively. This work provides a framework for optimizing quantum resources when applying Runge-Kutta methods, enhancing their efficiency and accuracy in both solving differential equations and simulating quantum systems. Furthermore, this work plays a crucial role in assessing the suitability of these variational algorithms for fault-tolerant quantum computing. The results may also be of interest to the numerical analysis community as they involve the accumulation of errors in the function description, a topic that has hardly been explored even in the context of classical differential equation solvers.

Reduced density matrices (RDMs) are fundamental in quantum information processing, allowing the computation of local observables, such as energy and correlation functions, without the exponential complexity of fully characterizing quantum states. In the context of near-term quantum computing, RDMs provide sufficient information to effectively design variational quantum algorithms. However, their experimental estimation is challenging, as it involves preparing and measuring quantum states in multiple bases, which is a resource-intensive process susceptible to producing non-physical RDMs due to shot noise from limited measurements. Motivated by this influence of shot noise, we ask the following research question:

Research Question 4: *Is it possible to mitigate the effects of shot noise in the quantum state tomography of reduced density matrices by enforcing physicality conditions organized as semidefinite programs?*

In Chapter 4, we answer this question, which is addressed in Ref. [90]. We propose a method to mitigate shot noise by re-enforcing certain physicality constraints on RDMs. These constraints are, first, the requirement that RDMs be compatible with higher-dimensional RDMs, which we call enhanced-compatibility. Second, overlapping-compatibility requires that RDMs overlapping on a common subsystem be consistent on that subsystem. We organize these compatibility constraints in semidefinite programs to reconstruct RDMs from simulated data. Our approach yields, on average, tighter bounds for the same number of measurements compared to tomography without compatibility constraints. We demonstrate the versatility and efficacy of our method by integrating it into an algorithmic cooling procedure to prepare low-energy states of local Hamiltonians.

Generative adversarial networks have been investigated as a method for generating synthetic data with the goal of augmenting training data sets for neural networks. As we described in Sec. 1.3.3, this is especially relevant for financial time series, as the availability of data is very limited. However, for classical generative adversarial networks it has been shown that generated data may (often) not exhibit desired properties (also called stylized facts), such as matching a certain distribution or showing specific temporal correlations [78]. As quantum computers have been shown to be able to capture distributions that classical models can not [43, 55, 56, 79], it has been proposed to replace the classical generator with a quantum circuit, and to generate synthetic data with such a quantum generative adversarial network. Typically, the ability of generative models to reproduce time series with desired properties is concluded qualitatively. We thus state the following research question:

Research Question 5: *Can quantum generative adversarial networks capture the distribution and stylized facts of financial time series on a qualitative level?*

In Chapter 5, we present our answer to this question, based on the results shown in Ref. [91]. We train QGANs, composed of a quantum generator and a classical discriminator, and investigate two classical simulation approaches for the quantum generator: a full simulation of the quantum circuits, and an approximate simulation using matrix product states. We test the effect of the choice of circuit depths and bond dimensions of the matrix product state simulation on the generated time series. Our QGANs successfully showcase most temporal correlations in generating synthetic financial time series, but the quality differs between different properties, depending on the chosen hyperparameters of the QGAN.

In Chapter 6, we conclude the thesis, come back to the research questions posed here, and outline future work.

CHAPTER 2

Approximation and Generalization Capacities of Parameterized Quantum Circuits for Functions in Sobolev Spaces

2.1 Introduction

Machine learning has gained significant attention in recent years for its practical applications and transformative impact in various fields. As a consequence, there has been a rising interest in exploring the use of quantum circuits as machine learning models, capitalizing on the advancements in both fields to unlock new possibilities and potential breakthroughs. Among the various possibilities for leveraging quantum circuits in machine learning, our particular focus lies in parameterized quantum circuits (PQC). These quantum circuits consist of both fixed and adjustable (hence 'parameterized') gates. When used for a learning task such as learning a function [92], a classical optimizer updates the parameters of the PQC in order to minimize a cost function depending on measurement results from this quantum circuit (see Fig. 2.1).

In this context, a growing line of research studies the expressivity of PQCs. More precisely, the capacity of PQCs to approximate any function belonging to a particular *function space* defined in a prescribed *domain* up to arbitrary

The contents of this chapter have been published in Ref. [88].

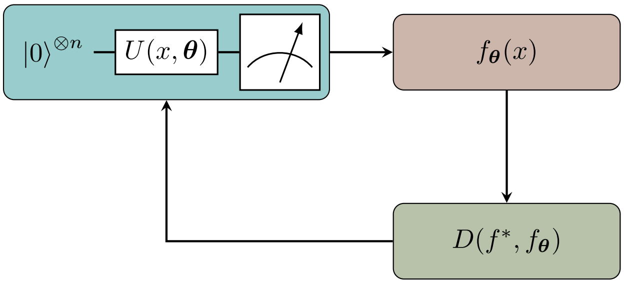

$|0\rangle^{\otimes n}$
$U(x, \boldsymbol{\theta})$
$f_{\boldsymbol{\theta}}(x)$
$D(f^*, f_{\boldsymbol{\theta}})$

Figure 2.1: Sketch of a hybrid variational algorithm. $U(x, \theta)$ represents a quantum circuit that takes x as input and with variational parameters θ, $f_\theta(x)$ is the expected value of some observable and $D(f^*, f_\theta)$ is the expected loss that we want to minimize.

precision with respect to a specific *distance*. In [67], they showed that PQCs can be written as a generalized trigonometric series in the following way:

$$\begin{aligned} f_{\boldsymbol{\theta}}(\boldsymbol{x}) &= \langle 0| U^\dagger(\boldsymbol{x}; \boldsymbol{\theta}) M U(\boldsymbol{x}; \boldsymbol{\theta}) |0\rangle \qquad (2.1) \\ &= \sum_{\boldsymbol{\omega} \in \Omega} c_{\boldsymbol{\omega}}(\theta) e^{i \boldsymbol{\omega} \boldsymbol{x}} \ . \qquad (2.2) \end{aligned}$$

We would like to emphasize that although similar, the form of the PQC in above equation is more general than a Fourier series. This will become relevant for the results of this chapter. Using this formulation, it was further shown in [67] that, if the PQC is chosen carefully, the increase of its depth and number of parameters can arbitrarily reduce the L^2 distance between the expected value $f_{\boldsymbol{\theta}}(\boldsymbol{x})$ of the PQC and any square-integrable function with the domain $[0, 2\pi]^N$. Throughout the chapter, we will refer to the PQC as the one approximating the functions to make the text more fluent, although technically it is the expectation value of the PQC that approximates the function.

This result had a significant impact on the motivation to study PQC-based QML, analogous to the impact that the famous Universality theorem for neural networks of Cybenko [65] had on the domain of classical machine learning. Previously, different results on universality for PQC have been established. In [93], the power of PQCs in expressing matrix product states and instantaneous quantum polynomial circuits was shown. Later, the universal approximation of PQCs was studied in regression problems, for single-qubit circuits with multiple layers [68, 69] and for both single- and multiple-qubit circuits [70, 71]. However, as it turns out, there are numerous different notions of universality, and not all are useful for all applications. For instance, as will be discussed later, in the context of Physics-Inspired Neural Networks (PINN) the "vanilla" universality does not suffice. This raises the question of whether PQCs can approximate functions belonging to other function spaces or in terms of other distances.

In this chapter, we present two novel results. The first result of this chapter is that PQCs can arbitrarily approximate the space of continuous functions, the space of p-integrable functions and the H^k space, which is the set of functions whose derivatives up to order k are square-integrable.

Furthermore, we explain how these properties can be easily achieved in practice by a simple min-max feature rescaling (see Eq. (2.21)) of the input data. In practice, this leads to an improved expressivity of PQCs, if the input data is normalized accordingly.

The second main result of the chapter concerns generalization bounds that connect distances with loss functions which are not built via the discretization of the integrals present in the definition of the distance. To make it more clear, we recall that in a machine learning problem one needs to choose an architecture, which defines the class of functions that can be approximated, and a target distance, which is intimately connected with the generalization error[1]. However, in general it is not possible to compute the target distance, as we would need to have available infinitely many data points. Instead, one chooses a different distance function which can be computed from the available data: a loss function. This loss function is a different function than the target distance but it should be chosen in such a way that we call *consistent* with the target distance, i.e., that the minimization of the expectation value of the loss function (i.e., the expected loss) yields the minimization of the target distance up to an error which asymptotically tends to zero *when the number of samples and the expressivity* (here meant architecturally, as e.g. depth) of the PQC increases. For example, the mean square error (as a loss function) is consistent with the L^2 error (as a target distance) but is inconsistent with the supremum distance. The usual generalization bounds connect target distances which are continuous with expected losses which are their discrete version.

The generalization bounds that we derive give a mapping *across* different distances and loss functions, i.e., they relate distances with loss functions which are not built via the discretization of the integrals present in the definition of the distance. A particular loss function we shall define, denoted ℓ_{h^1}, which consists of the sum of the mean square errors of the values of the functions and its derivative, is consistent with the supremum distance in one-dimensional problems. In the described case, this allows us to reduce the supremum distance while choosing a loss function which is differentiable.

Our results apply in many settings. For example, our first result has a direct consequence in that it allows one to approximate not just most, but all function values with satisfying quality. For instance, the minimization of the ubiquitous L^2 distance may allow functions to dramatically differ from the target function in some regions where we have plenty of data points available,

[1]In practice we may not explicitly think about the target distance, i.e. with respect to which distance we wish to approximate the "true" labeling function. But this decision is implicitly made, once the loss is chosen.

whereas the minimization of the supremum norm in Thm. 2.3 will force the PQC to converge for any given point in the domain of the target function. This is of high relevance in cases where we are interested in having a good approximation at any given point. For instance, when learning the shape of a probability distribution from samples, a good fit in the bulk of the distribution but a poor fit in its tails can lead to significant underestimation or overestimation of the probability of extreme events. In real-world applications, this could have severe consequences in risk assessment applications, where accurate estimation of tail probabilities is essential for developing appropriate contingency measures against rare but significant events, such as the COVID-19 pandemic or the 2008 economic crisis. Our second result has direct applications, e.g., in settings where we have access to data of the function and its derivatives. One case where this is standard is in settings where differential equations are solved. For example in physics-informed neural networks (PINN) problems [94] and differential machine learning (DML) [89], both function values and derivatives are accessible and in fact critical

This chapter is organized as follows: in Sec. 2.2 we explain the new results on the expressivity of PQCs. In Sec. 2.3 we discuss the proposed generalization bounds. Then, in Sec. 2.4 we illustrate the theoretical result of Secs. 2.2 and 2.3 by means of some numerical experiments. Lastly, in Sec. 2.5 we wrap up with the conclusions.

During the final stages of our work, we became aware of the paper [95] which overlaps in some parts with our own results in Sec. 2.2. However, the results presented here were developed independently and follow a different line of reasoning.

2.2 PQCs and universal approximation

In this section, we will review the established result on universality in [67] and then present our new universality results in Thms. 2.2, 2.3 and 2.4.

Schuld et al. showed in [67], how a quantum machine learning model of the form $f_\theta(x) = \langle 0| U^\dagger(x;\boldsymbol{\theta}) M U(x;\boldsymbol{\theta}) |0\rangle$ can be written as a univariate generalized trigonometric series:

$$\langle 0| U^\dagger(x;\boldsymbol{\theta}) M U(x;\boldsymbol{\theta}) |0\rangle = f_m(x;\boldsymbol{\theta}) \tag{2.3}$$

$$= \sum_{\boldsymbol{\omega}\in\Omega} c_{\boldsymbol{\omega}}(\theta) e^{i\boldsymbol{\omega x}}, \tag{2.4}$$

where M is an observable, $U(x;\boldsymbol{\theta})$ is a quantum circuit modeled as a unitary that depends on input x and the variational parameters $\boldsymbol{\theta} = (\theta_0, \theta_1, ..., \theta_T)$. In the above, $\boldsymbol{\omega} \in \Omega$ denotes the set of available frequencies which always contain 0. The quantum circuit consists of L layers each consisting of a trainable circuit

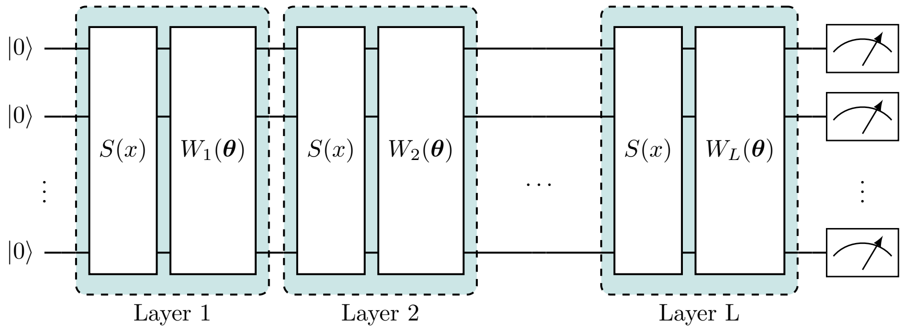


Figure 2.2: Parameterized quantum circuit that can be written as a generalized trigonometric series as in Eq. (2.1). It consists of L layers, each layer is composed by a trainable circuit block $W_i(\boldsymbol{\theta}), i \in \{1, ..., L+1\}$ and a data encoding block $S(x)$. The data encoding blocks $S(x)$ are identical for all layers, they determine which frequencies $\boldsymbol{\omega}$ are accessible and are implemented as Pauli rotations. The blocks $W_i(\boldsymbol{\theta})$ can be built from local rotation gates and $CNOT$ gates. They determine the coefficients $c_{\boldsymbol{\omega}}(\theta)$.

block $W_i(\boldsymbol{\theta}), i \in \{1, ..., L+1\}$ and a data encoding block $S(x)$ as shown in Fig. 2.2. The data encoding blocks determine which frequencies $\boldsymbol{\omega}$ are accessible in the sum and are implemented as Pauli rotations. The blocks $W(\boldsymbol{\theta})$ can be built from single-qubit rotation gates and CNOT gates and they determine the coefficients $c_{\boldsymbol{\omega}}(\theta)$ of the sum. It is possible to both implement this model with $L > 1$ layers, such as data re-uploading PQC [96, 97], where the encoding is repeated on the same subsystems in sequence, or with parallel encodings [59] and $L = 1$, where the encoding is repeated on several different subsystems.

For the needs of our discussion, we will briefly describe a more specific set-up under which the authors of [67] proved a universality theorem of these quantum models for the multivariate case with inputs $\boldsymbol{x} = (x_0, x_1, ..., x_N)$.

Let us construct a model of the form in Eq. (2.1), with the measurement M and a quantum circuit of one layer, $L = 1$:

$$f_{\boldsymbol{\theta}} = \langle 0| U^\dagger(\boldsymbol{\theta}, \boldsymbol{x}) M U(\boldsymbol{\theta}, \boldsymbol{x}) |0\rangle \,, \text{ with} \tag{2.5}$$

$$U(\boldsymbol{\theta}, \boldsymbol{x}) = W^{(2)}(\boldsymbol{\theta}^{(2)}) S(\boldsymbol{x}) W^{(1)}(\boldsymbol{\theta}^{(1)}), \tag{2.6}$$

where $\boldsymbol{\theta}^{(1)}$ and $\boldsymbol{\theta}^{(2)}$ are those parameters in $\boldsymbol{\theta}$ that affect $W^{(1)}$ and $W^{(2)}$, respectively. Let us further make the following two assumptions: Firstly, we assume that the data-encoding blocks $S(\boldsymbol{x})$ are written in the following way:

$$S(\boldsymbol{x}) = e^{-x_1 H} \otimes \cdots \otimes e^{-x_N H} \tag{2.7}$$

$$=: S_H(\boldsymbol{x}), \tag{2.8}$$

where H is a Hamiltonian that we specify later. Secondly, we assume that the trainable circuit blocks $W^{(1)}(\boldsymbol{\theta}^{(1)})$ and $W^{(2)}(\boldsymbol{\theta}^{(2)})$ are able to represent arbitrary global unitaries. In practice, this may require a circuit depth that scales exponentially in the number of qubits. With this assumption, we drop the dependence on $\boldsymbol{\theta}$ and reformulate the assumption as being able to prepare an arbitrary initial state $|\Gamma\rangle := W^{(1)}(\boldsymbol{\theta}^{(1)})\,|0\rangle$ and by absorbing $W^{(2)}(\boldsymbol{\theta}^{(2)})$ into the measurement M. We can then write the above quantum model as:

$$f(\boldsymbol{x}) = \langle\Gamma|\, S_H^\dagger(\boldsymbol{x}) M S_H(\boldsymbol{x})\,|\Gamma\rangle \ . \tag{2.9}$$

Let us further present the notion of a universal Hamiltonian family, as defined in [67]:

Definition 1. *Let $\{H_m | m \in \mathbb{N}\}$ be a Hamiltonian family where H_m acts on m subsystems of dimension s.*

Such a Hamiltonian family gives rise to a family of models $\{f_m\}$ in the following way:

$$f_m(\boldsymbol{x}) = \langle\Gamma|\, S_{H_m}^\dagger(\boldsymbol{x}) M S_{H_m}(\boldsymbol{x})\,|\Gamma\rangle \ . \tag{2.10}$$

Further, we call the set

$$\Omega_{H_m} := \{\lambda_j - \lambda_k | j,k \in \{1, ..., s^m\}\} \tag{2.11}$$

where $\{\lambda_1, ..., \lambda_{s^m}\}$ are the eigenvalues of H_m, the frequency spectrum of H_m.

Remark. *We call a Hamiltonian family $\{H_m\}$ a universal Hamiltonian family, if for all $K \in \mathbb{N}$, there exists an $m \in \mathbb{N}$, such that:*

$$\mathbb{Z}_K = \{-K, ..., 0, ..., K\} \subseteq \Omega_{H_m}, \tag{2.12}$$

hence if the frequency spectrum of $\{H_m\}$ asymptotically contains any integer frequency.

As shown in [67], a simple example of a universal Hamiltonian family is one which consists of tensor products of single-qubit Pauli gates:

$$H_m = \sum_{i=1}^{m} \sigma_q^{(i)}, \tag{2.13}$$

with $\sigma_q^{(i)}$, $q \in \{X, Y, Z\}$ and $s = 2$. The scaling of the frequency spectrum for this example goes as $K = m$. With these definitions, we can give the following theorem:

Theorem 2.1 (Convergence in L^2). *[67] Let $\{H_m\}$ be a universal Hamiltonian family, and $\{f_m\}$ the associated quantum model family, defined via Eq. (2.10). For all functions $f^* \in L^2\left([0,2\pi]^N\right)$, and for all $\epsilon > 0$, there exists some $m' \in \mathbb{N}$, some state $|\Gamma\rangle \in \mathbb{C}^{m'}$ and some observable M such that*

$$\|f_{m'} - f^*\|_{L^2} < \epsilon. \tag{2.14}$$

Here, we clearly see that there are two conditions on the target function f^* that must be fulfilled in order for the theorem to work properly. The first condition is that f^* belongs to L^2. This is not surprising, we need to assume certain regularity on the target function to make the theorem work. The second condition is that the target function f^* needs to be restricted to the domain $[0,2\pi]^N$. However, as suggested in the original paper [67], if the function f^* does not belong to this domain, we can easily map $[a,b]^N$ to the required domain $[0,2\pi]^N$ (or $[-\pi,\pi]^N$ equivalently).

We would like to highlight the fact that the distance we use to bring the approximator closer to the target function is the L^2 distance. Note that convergence in the L^2 sense does not imply other modes of convergence. For example, this does not give us information about the general case of L^p-distances, with $1 \leq p < \infty$. We explicitly address this more general case in the following theorem:

Theorem 2.2 (Convergence in L^p). *Let $\{H_m\}$ be a universal Hamiltonian family, and $\{f_m\}$ the associated quantum model family, defined via Eq. (2.1). For all functions $f^* \in L^p\left([0,2\pi]^N\right)$ where $1 \leq p < \infty$, and for all $\epsilon > 0$, there exists some $m' \in \mathbb{N}$, some state $|\Gamma\rangle \in \mathbb{C}^{m'}$, and some observable M such that:*

$$\|f_{m'} - f^*\|_{L^p} < \epsilon. \tag{2.15}$$

The proof of Thm. 2.2 is given in Appendix 2.A.

Let us emphasize the difference between Thms. 2.1 and 2.2: The target function can belong to any L^p space with $1 \leq p < \infty$ in contrast to the previous requirement of being square-integrable (L^2). This is essentially achieved by the fact that PQCs are not only able to represent Fourier series as it is discussed in [67] but they are also able to represent more general trigonometric series. This allow us to identify the expectation value of the quantum circuit with the Cèsaro summation of the partial Fourier series of f^* and leverage the power of Fejér-like theorems [98]. See Appendix 2.A for more details.

Nevertheless, the ability to approximate functions in L^p does not prevent us from having arbitrarily big errors in certain points. Intimately related to this problem is the so-called Gibbs phenomenon [99]. Namely, the approximation of a continuous, but non-periodic function by a Fourier series is increasingly better in the interior of the domain but increasingly poorer on its boundaries.

That leads to the fundamental question if we can approximate f^* in a stronger sense, so that we ensure that the target function f^* is well approximated in any given point. We answer this question in the next theorem.

A set $U \subset \mathbb{R}^N$ is compactly contained in another set $V \subset \mathbb{R}^N$, if the closure of U is compact and contained in the interior of V.

Theorem 2.3 (Convergence in C^0)**.** *Let $\{H_m\}$ be a universal Hamiltonian family, and $\{f_m\}$ the associated quantum model family, defined via Eq. (2.1). For all functions $f^* \in C^0(U)$ where U is compactly contained in the closed cube $[0, 2\pi]^N$, and for all $\epsilon > 0$, there exists some $m' \in \mathbb{N}$, some state $|\Gamma\rangle \in \mathbb{C}^{m'}$, and some observable M such that $f_{m'}$ converges uniformly to f^*:*

$$\|f_{m'} - f^*\|_{C^0} < \epsilon, \tag{2.16}$$

with [2]

$$\|f_{m'} - f^*\|_{C^0} := \sup_{\mathbf{x} \in [0,2\pi]^N} \|f_{m'}(\mathbf{x}) - f^*(\mathbf{x})\| \,. \tag{2.17}$$

The proof of Thm. 2.3 can be found in Appendix 2.A.

Simply stated, this theorem means that $f_{m'}$ converges uniformly to f^*. In other words, if we select a given target error ϵ we are always able to find a finite PQC such that the error on any point is smaller than the prescribed ϵ. Let us emphasize again the key differences between Thms. 2.1 and 2.3. The first difference is that the function f^* has to be defined in a domain U which is compactly contained in $[0, 2\pi]^N$. A simple example of U is the interval $\left(\left[-\frac{\pi}{2}, \frac{\pi}{2}\right]^N\right)$ (or $\left([0, \pi]^N\right)$, equivalently). By restricting ourselves to half of the original space we can always find a C^0 extension of the function f^* in $\mathbb{T}^N$. The second difference is that the target function now belongs to the class of continuous functions in contrast to the previous requirement of being square-integrable (L^2).

A last result that we will show in this regard is about the approximation of the function and its derivatives by the parameterized quantum circuit. This might seem as a purely synthetic question but it has many implications in practice. When we approximate a target function, in many occasions we not only want to recover its value but also its dynamics. This is particularly relevant for problems in physics, where we typically have a differential equation which describes the behavior of the system. As we will see in the following theorem, the universality results translate to functions defined in the Sobolev space H^k as well:

[2]Since f^* is defined on a compact domain U, the supremum is equivalent to the maximum in this case.

Definition 2. *The Sobolev space $H^k(\Omega)$ is defined as the space of square-integrable functions on a domain $\Omega \subseteq \mathbb{R}^N$ which derivatives up to order k are also square-integrable:*

$$f^\alpha = D^\alpha f, \text{ and } \left\| f^{(\alpha)} \right\|_2 < \infty, \tag{2.18}$$

for all $0 \le |\alpha| \le k$ and $D^\alpha := \frac{\partial^{|\alpha|}}{\partial x_1^{\alpha_1} \dots \partial x_N^{\alpha_N}}$.

The Sobolev norm $\|\cdot\|_{H^k}$ is defined as

$$\|f\|_{H^k} := \left(\sum_{|\alpha| \le k} \int_\Omega |D^\alpha f|^2 \right)^{1/2}. \tag{2.19}$$

Theorem 2.4 (Convergence in H^k)**.** *Let $\{H_m\}$ be a universal Hamiltonian family, and $\{f_m\}$ the associated quantum model family, defined via Eq. (2.1). For all functions $f^* \in H^k(U)$ where U is compactly contained in the closed cube $[0, 2\pi]^N$, and for all $\epsilon > 0$, there exists some $m' \in \mathbb{N}$, some state $|\Gamma\rangle \in \mathbb{C}^{m'}$, and some observable M such that $f_{m'}$ is ϵ-close to f^* with respect to the H^k-distance:*

$$\|f_{m'} - f^*\|_{H^k} < \epsilon. \tag{2.20}$$

The proof of Thm. 2.4 is given in Appendix 2.A.

As in Thms. 2.3 and 2.4, we require that the target function is defined on a compactly contained subset of $[0, 2\pi]^N$, we propose to perform a min-max feature scaling of the input data:

$$\boldsymbol{x} = (x_1, \dots, x_n) \longrightarrow \tilde{\boldsymbol{x}} = (\tilde{x}_1, \dots, \tilde{x}_n), \tag{2.21}$$

where $\boldsymbol{x} \in [a, b]^N$, $\tilde{\boldsymbol{x}} \in \left[-\frac{\pi}{2}, \frac{\pi}{2}\right]^N$, and

$$\tilde{\boldsymbol{x}} = \left(-\frac{\pi}{2} + \pi \frac{x_1 - a}{b - a}, \dots, -\frac{\pi}{2} + \pi \frac{x_n - a}{b - a} \right). \tag{2.22}$$

This simple recipe allows the PQC to approximate a much wider set of function spaces as shown throughout this section. This normalization strategy works very well in practice as can be seen in Sec. 2.4. However, we would like to emphasize that this particular normalization is not the only choice. The classical strategy in machine learning of normalizing the input data to lie in the $[-1, 1]^N$ domain is also completely valid.

Throughout this section, we have discussed the expressive power of PQCs, but when we do machine learning, we have more ingredients that we need to take into account. In the next section we will discuss the role that the loss function plays in accordance with the type of approximation that our PQC can get.

2.3 Connections between different generalization bounds

As we have seen in the previous section, the notion of approximation depends on a prescribed distance. This distance is not given by the problem itself, but rather chosen by the user, this is why we refer to it as target distance. In general, it is however not possible to compute the target distance, which for example is the case for the L^p and H^k distances. This is why one needs to choose a distance function which can be computed from data, a loss function. It has to be chosen in such a way that it is consistent with the target function. To discuss the topic in more depth, let us formally introduce the continuous regression problem, which is the problem that we are most interested in.

In general, we can describe the continuous regression problem in the following way: assume that there is some target function $f^* \in \mathcal{F} \subseteq H^k$ mapping inputs $x \in \mathcal{X}$ to target labels $y \in \mathcal{Y}$. Moreover, assume that the points in $\mathcal{X}$ are sampled according to a bounded[3] density function p. Our goal is to find the best approximation $f \in \mathcal{M} \subseteq H^k$ of the target function f^*.

The notion of what is understood as a "good" approximation as clarified, allows for some freedom. For this reason, one has to make a choice by specifying a functional $D : H^k \times H^k \longrightarrow \mathbb{R}^+ \cup \{0\}$ which defines a distance between the elements of $\mathcal{F}$ and $\mathcal{M}$. The problem can then be stated as:

$$f = \underset{\hat{f} \in \mathcal{M}}{\arg\min}\, D(f^*, \hat{f}). \tag{2.23}$$

The most common distance in the literature for continuous regression problems is the one induced by the $L^2(\mathcal{X}, P)$ norm:

$$\begin{aligned} D_{L^2}(f^*, f) &= ||f^* - f||_{L^2} && (2.24)\\ &= \left(\int_{\mathcal{X}} (f^*(\boldsymbol{x}) - f(\boldsymbol{x}))^2 dP \right)^{\frac{1}{2}}. && (2.25) \end{aligned}$$

However, in regression we do not typically have access to the full information (i.e., we cannot compute the integral). It is for this reason that instead we work with the empirical risk minimization problem, which uses the discrete version l^2 of the L^2 distance as a loss function. The difference with the previous setup is that, for the empirical risk minimization problem, we are given a finite training

[3] It is possible to have more general density functions. However, we restrict ourselves with this one since it simplifies the analysis.

set S of I inputs sampled from the same probability density p, together with their target labels $\{(x_1, y_1), ..., (x_I, y_I)\}$ with $(x, y) \in \mathcal{X} \times \mathcal{Y}$, according to the target function $f^* : \mathcal{X} \to \mathcal{Y}$, $f \in \mathcal{F}$. Now, instead of minimizing a continuous functional, we will minimize a discrete one. We call

$$D_\ell(f^*, f) = \frac{1}{I} \sum_{i=0}^{I-1} \ell\left(f^*(\boldsymbol{x}^i), f(\boldsymbol{x}^i)\right) \tag{2.26}$$

the expected loss according to a loss function $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$. Similarly to the continuous case, we are concerned with the expected loss of the l^2 distance, which is defined as:

$$D_{l^2}(f^*, f) := \left(\frac{1}{I} \sum_{i=0}^{I-1} \left(f^*(\boldsymbol{x}^i) - f(\boldsymbol{x}^i)\right)^2\right)^{\frac{1}{2}}, \tag{2.27}$$

with $\boldsymbol{x}^i$ denoting the i-th input.

Although we are solving the minimization problem associated with the expected loss defined in Eq. (2.27), in general we are interested in the generalization performance, i.e., the distance in terms of Eq. (2.24). Using generalization bounds [100] we can relate the performance in terms of the distance given by Eq. (2.27) with the distance given by Eq. (2.24). However, these classical results in machine learning do in general not relate the l^2 distance with other distances, like the C^0 distance. In other words, even a solution which, as the model and the number of points grow larger asymptotically makes the D_{L^2} go to zero, does not necessarily make the D_{C^0} distance vanish, which is defined as:

$$D_{C^0}(f^*, f) := \sup_{\mathbf{x} \in \mathcal{X}} |f^*(\mathbf{x}) - f(\mathbf{x})|. \tag{2.28}$$

In such cases, we could find points where there is an arbitrarily large discrepancy between the solution and the target function.

One possible solution would be to use a different distance than D_{l^2}. For example one could try the discrete form of the D_{C^0} distance:

$$D_{l^\infty} = \max_{i \in \{0, \ldots, I-1\}} \left|f^*(\mathbf{x}^i) - f(\mathbf{x}^i)\right|, \tag{2.29}$$

but this distance is not differentiable, making the optimization process much harder.

Thus, we identify two desirable features for a distance in order to be able to approximate with the C^0 distance. The first requirement is that the solution of the minimization problem that it defines, tends uniformly to the target function f^* as we increase the number of given points I and we increase the size of our PQC. The second one is that it has to be differentiable in order to make minimization easier.

The solution that we propose here is to use a distance motivated by discretizing the Sobolev distance H^k on a fixed finite training set

$$\Big\{ \big(x_1, \{D^\alpha f^*(\boldsymbol{x}^1)\}_{0\le|\alpha|\le k}\big), \dots, \big(x_I,\ \{D^\alpha f^*(\boldsymbol{x}^I)\}_{0\le|\alpha|\le k}\big) \Big\},$$
$$(x, y) \in \mathcal{X} \times \mathcal{Y},$$

according to the target function $f^* : \mathcal{X} \to \mathcal{Y}$, $f \in \mathcal{F}$. The sets $\{D^\alpha f(\boldsymbol{x})\}_{0\le|\alpha|\le k}$ and $\{D^\alpha f^*(\boldsymbol{x})\}_{0\le|\alpha|\le k}$ consists of the function values $f(\boldsymbol{x})$ or $f^*(\boldsymbol{x})$ and their $M(N,k) := \sum_{\alpha=1}^{k} \binom{\alpha+N-1}{N-1}$ different partial derivatives up to order k evaluated at point $\boldsymbol{x}$, respectively. We write N for the input dimension. Note that for being able to apply this distance, one needs to have access to training data containing the required partial derivatives additionally to the function values.

We show the expected loss of the discretized version of H^1 and H^k, respectively, in the following two equations:

$$D_{h^1}(f^*, f) := \left[\frac{1}{I} \sum_{i=0}^{I-1} \big(f^*(\boldsymbol{x}^i) - f(\boldsymbol{x}^i)\big)^2 + \sum_{j=0}^{N-1} \sum_{i=0}^{I-1} \frac{1}{I} \left(\frac{\partial f^*}{\partial x_j}(\boldsymbol{x}^i) - \frac{\partial f}{\partial x_j}(\boldsymbol{x}^i) \right)^2 \right]^{\frac{1}{2}}, \tag{2.30}$$

$$D_{h^k}(f^*, f) := \left[\frac{1}{I} \sum_{i=0}^{I-1} \big(f^*(\boldsymbol{x}^i) - f(\boldsymbol{x}^i)\big)^2 + \sum_{|\alpha|\le k} \sum_{i=0}^{I-1} \frac{1}{I} \big(D^\alpha f^*(\boldsymbol{x}^i) - D^\alpha f(\boldsymbol{x}^i)\big)^2 \right]^{\frac{1}{2}}. \tag{2.31}$$

The expected loss as given in Eq. (2.30) was first introduced in [89] and gives rise to a new subfield of machine learning known in the literature as differential machine learning (DML). Its generalization, the discretization of the distance H^k, is given in Eq. (2.31), and can be applied when the required higher-dimensional derivatives are available as well. The derivatives $\frac{\partial^{(p)} f^*}{\partial x_j^{(p)}}$ and $\frac{\partial^{(p)} f}{\partial x_j^{(p)}}$ are the p-th order derivative functions in direction x_j of f^* and f, respectively. The corresponding loss function is thus defined as

$$\ell_{h^k} : \mathbb{R}^{M(N,k)+1} \times \mathbb{R}^{M(N,k)+1} \to \mathbb{R}, \tag{2.32}$$
$$\big(f(\boldsymbol{x}), \{D^\alpha f(\boldsymbol{x})\}_{|\alpha|\le k}, f^*(\boldsymbol{x}), \{D^\alpha f^*(\boldsymbol{x}) :\}_{|\alpha|\le k}\big) \mapsto$$
$$\ell_{h^k}(f(\boldsymbol{x}), f^*(\boldsymbol{x})) = (f^*(\boldsymbol{x}) - f(\boldsymbol{x}))^2 \tag{2.33}$$
$$+ \sum_{|\alpha|\le k} \left(D^\alpha f^*(\boldsymbol{x}) - D^\alpha f(\boldsymbol{x})\right)^2 .$$

With classical neural networks, DML has proven to yield better generalization results in terms of the D_{l^2} distance than the solution of the D_{l^2} itself. This means that, if we take the solutions f_{h^1} and f_{l^2} of the minimization problems defined by Eqs. (2.30) with the same number of labels and Eq. (2.27) respectively and evaluate their performance in terms of the D_{L^2}, in practice f_{h^1} performs better than f_{l^2}:

$$D_{L^2}\left(f^*, f_{h^1}\right) \leq D_{L^2}\left(f^*, f_{l^2}\right). \tag{2.34}$$

However, to the best of our knowledge there is no theoretical explanation in the literature on why this happens or under which condition we might expect this behavior. In the following theorems we present generalization bounds that shed some light onto it.

Before stating them, we will define two function families to which the generalization bounds apply:

Definition 3. *[73] By $\mathcal{F}_\Omega^B$, we denote the function family defined as*

$$\mathcal{F}_\Omega^B = \Big\{[0,2\pi]^N \ni \boldsymbol{x} \mapsto f(\boldsymbol{x}) = \sum_{\boldsymbol{\omega}\in\Omega} c_{\boldsymbol{\omega}} \exp(-i\boldsymbol{\omega}\cdot\boldsymbol{x}) : \\ \{c_{\boldsymbol{\omega}}\}_{\boldsymbol{\omega}\in\Omega} \ s.t. \ \|f\|_\infty \leq B \ and \ |\Omega| < \infty\Big\}.$$

By $\mathcal{H}_\Omega^{\tilde{B}}$, we denote the function family defined as

$$\mathcal{H}_\Omega^{\tilde{B}} = \Big\{[0,2\pi]^N \ni \boldsymbol{x} \mapsto \frac{a_0}{2} \\ + \sum_{\boldsymbol{\omega}\in\Omega_+} \left(a_{\boldsymbol{\omega}}\cos(\boldsymbol{\omega}\cdot\boldsymbol{x}) + b_{\boldsymbol{\omega}}\sin(\boldsymbol{\omega}\cdot\boldsymbol{x})\right) : \\ \sqrt{a_0^2 + \sum_{\boldsymbol{\omega}\in\Omega_+} a_{\boldsymbol{\omega}}^2 + b_{\boldsymbol{\omega}}^2} \leq \tilde{B} \ and \ |\Omega_+| < \infty\Big\},$$

where the frequency set Ω is divided into the disjoint parts $\Omega = \Omega_+ \cup \Omega_- \cup \{0\}$, where $\Omega_+ \cap \Omega_- = \emptyset$ and such that for every $\boldsymbol{\omega} \in \Omega_+$, it holds that $-\boldsymbol{\omega} \in \Omega_-$.

According to [73], both of these function families can be modeled by the quantum model given in Eq.(2.10). As can be seen by this equation, the bounds B and $\tilde{B}$ depend on the chosen circuit and observable, and they determine the scaling in the following generalization bounds. Note as well that the truncated Fourier series as defined in $\mathcal{F}_\Omega^B$ and $\mathcal{H}_\Omega^{\tilde{B}}$ are differentiable, and their derivatives form truncated Fourier series as well. If one chooses the frequency set Ω' and the bounds B' and $\tilde{B}'$ large enough, for a given function family $\mathcal{F}$, both the functions and their derivatives belong to $\mathcal{F}_{\Omega'}^{B'}$ and $\mathcal{H}_{\Omega'}^{\tilde{B}'}$.

Theorem 2.5 (Generalization bound for H^k)**.** *Let $f^* \in \mathcal{F} \subseteq H^k([0,2\pi]^N)$ be a target function, and let there be a $B > 0$ and a $\tilde{B} > 0$, such that $\mathcal{F}_\Omega^B \subseteq \mathcal{H}_\Omega^{\tilde{B}}$ is a suitable model family. Let us further assume that $\ell_{h^k}(f_1(\boldsymbol{x}), f_2(\boldsymbol{x})) \leq c$ for all $\mathbf{x} \in [0,2\pi]^N$, and for all $f_1, f_2 \in \mathcal{F}_\Omega^B$ or $\mathcal{F}$. For any $\delta \in (0,1)$ and the empirical risk $D_{h^k}(f^*, f)$ trained on an i.i.d. training data S with size I and containing data of ξ partial derivatives, the following holds for all functions $f \in \mathcal{F}_\Omega^B$ with probability at least $1-\delta$:*

$$D_{H^k}(f^*, f) \leq D_{h^k}(f^*, f) + r(|\Omega|, \xi, B, \tilde{B}, c, I, \delta), \tag{2.35}$$

where $r(|\Omega|, \xi, B, \tilde{B}, c, I, \delta) \to 0$ as $I \to \infty$.

Theorem 2.6 (Generalization bound for L^p)**.** *Let $f^* \in \mathcal{F} \subseteq H^k([0,2\pi]^N)$ be a target function, and let there be a $B > 0$ and a $\tilde{B} > 0$, such that $\mathcal{F}_\Omega^B \subseteq \mathcal{H}_\Omega^{\tilde{B}}$ is a suitable model family. Let us further assume that $\ell_{h^k}(f_1(\boldsymbol{x}), f_2(\boldsymbol{x})) \leq c$ for all $\mathbf{x} \in [0,2\pi]^N$, and for all $f_1, f_2 \in \mathcal{F}_\Omega^B$ or $\mathcal{F}$. Assume that $k, p \in \mathbb{N}$ satisfy one of the two following cases:*

1. *$N\left(\frac{1}{2} - \frac{1}{p}\right) < k < N/2$ and $1 \leq p < N$.*
2. *$k \geq N/2$ and $1 \leq p < \infty$.*

For any $\delta \in (0,1)$ and the empirical risk $D_{h^k}(f^*, f)$ trained on an i.i.d. training data S with size I and containing data of ξ partial derivatives, the following holds for all functions $f \in \mathcal{F}_\Omega^B$ with probability at least $1-\delta$:

$$\frac{1}{C} D_{L^p}(f^*, f) \leq D_{h^k}(f^*, f) + r(|\Omega|, \xi, B, \tilde{B}, c, I, \delta), \tag{2.36}$$

where C is a constant and $r(|\Omega|, \xi, B, \tilde{B}, c, I, \delta) \to 0$ as $I \to \infty$.

Theorem 2.7 (Generalization bound for C^0)**.** *Let $f^* \in \mathcal{F} \subseteq H^k([0,2\pi]^N)$ be a target function, and let there be a $B > 0$ and a $\tilde{B} > 0$, such that $\mathcal{F}_\Omega^B \subseteq \mathcal{H}_\Omega^{\tilde{B}}$ is a suitable model family. Let us further assume that $\ell_{h^k}(f_1(\boldsymbol{x}), f_2(\boldsymbol{x})) \leq c$ for all $\mathbf{x} \in [0,2\pi]^N$, and for all $f_1, f_2 \in \mathcal{F}_\Omega^B$ or $\mathcal{F}$ and that $\|f\|_\infty \leq B$ for all $f \in \mathcal{F}_\Omega^B$. Assume, that $k \in \mathbb{N}$ satisfies $k > N/2$. For any $\delta \in (0,1)$ and the empirical risk $D_{h^k}(f^*, f)$ trained on an i.i.d. training data S with size I and containing data of ξ partial derivatives, the following holds for all functions $f \in \mathcal{F}_\Omega^B$ with probability at least $1-\delta$:*

$$\frac{1}{C} D_{C^0}(f^*, f) \leq D_{h^k}(f^*, f) + r(|\Omega|, \xi, B, \tilde{B}, c, I, \delta), \tag{2.37}$$

where C is a constant and $r(|\Omega|, \xi, B, \tilde{B}, c, I, \delta) \to 0$ as $I \to \infty$.

The proofs of Thms. 2.5, 2.6 and 2.7 can be found in Appendix 2.B.

A consequence of Thm. 2.7 is that, if the order of the derivatives that we have at our disposal are higher than half the input dimension ($k > N/2$), our solution of the D_{h^k} problem is also a solution of the D_{C^0} problem, corresponding to uniform convergence. It means that training with the ℓ_{h^k} loss function (for $k > N/2$), which sums the ℓ_{l^2} losses of function and derivative values, is sufficient for an approximation in C^0. This would not be possible by a training with ℓ_{l^2} loss function and more practical than the training with the ℓ_{l^∞} loss function, as described above.

Note that we face a curse of dimensionality-like phenomenon as the dimension of the input grows. In this case, the number of terms that go into the ℓ_{h^k} loss function grows exponentially with k, as we have to take into account mixed derivatives. Hence, for high dimensional problems the demand on data of partial derivatives is higher and only if they are available, this generalization bound holds.

Further, the requirement of quantum resources for evaluating $D_{h^1}(f^*, f)$ is higher than for the evaluation of $D_{l^2}(f^*, f)$. If we use the parameter shift rule for the evaluation of the derivatives, we need to evaluate $I(1 + 2N)$ different PQCs. Similar to the demand on training data, this number of PQCs to evaluate $D_{h^k}(f^*, f)$ grows exponentially in k. However, even if the amount of training data is the same (and implying an increase of required PQC evaluations up to a factor of 2), the training with the ℓ_{h^k} loss function shows the promised advantages, as presented in [89].

The last property we wish to highlight is the fact that the generalization bounds connect the empirical risk with the full risk, but they do not give us information of whether they can both tend to zero. In order to tackle that question we need to combine the results of the theorems in this section with the ones present in Sec. 2.2. For example, if we try to fit a one-dimensional function which is not periodic on $[0, 2\pi]$, using model families $\mathcal{F}_\Omega^B$ and $\mathcal{H}_\Omega^{\tilde{B}}$ and the ℓ_{h^1} loss function, as we increase the number of sample points both sides of the inequality will tend to the same constant but they will not converge to zero. In this regard, observe that the fact that the empirical risk goes to zero is a sufficient but not a necessary condition for the target distance to also tend to zero. Following the same example, if instead of training the model using the ℓ_{h^1} loss function we trained the model using the ℓ_{l^2} loss function, then the L^2 distance will vanish. This idea will become apparent in Fig. 2.5. The bottom line is that more information in the training data does not always equate to a better approximation, if we are not very careful with the necessary data normalization.

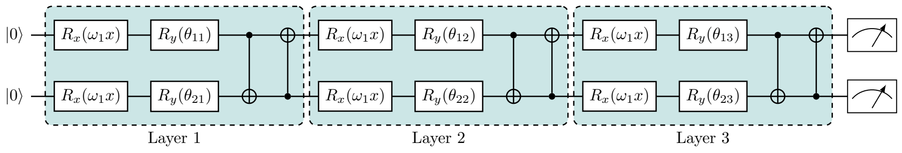


Figure 2.3: Architecture $U(x, \boldsymbol{\theta})$ used in the experiments. The parameters θ_{ij} are variational parameters. Each qubit is measured in the Pauli-Z basis.

2.4 Numerical experiments

In this section we illustrate the theoretical discussion of Secs. 2.2 and 2.3 with an illustrative example: the approximation of function $f^*(x) = \frac{x}{2\pi}$, $x \in [-\pi, \pi]$ by the PQC in Fig. 2.3.

We conduct two different numerical experiments and show them in Figs. 2.4 and 2.5. We chose a linear function to show that even in this simple case, the numerical tests fail completely if the results of Secs. 2.2 and 2.3 are not applied.

All simulations have been performed using 10 points (10 for the labels plus 10 for the derivative values when they are present) uniformly distributed along the domain for the training phase. Each experiment has been repeated 100 times and we depict the 25, 50 and 75 percentiles in colored solid lines in Fig. 2.4. The legends call the result of the PQCs as $f_\bullet(\cdot)$, where the subscript denotes under which loss function we have done the training and in the parentheses we indicate which normalization we have chosen.

In Fig. 2.4 we compare the performance of our PQC under different normalizations. We normalize the data to lie in the domains $\left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$, $[-\pi, \pi]$ and $[-2\pi, 2\pi]$, respectively. When we normalized our data to lie in the range $\left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$ we get the best results, as we expected due to Thm. 2.3.

In contrast, when the data is normalized to lie in the range $[-2\pi, 2\pi]$ we obtain very poor approximation results, because in this case, it is not possible to approximate with the C^0-distance or the L^2-distance. The intermediate regime happens when we normalize the data to lie in the range $[-\pi, \pi]$, here we obtain a reasonable approximation except for the boundaries. This is a consequence of approximating with the L^2-distance instead of the C^0-distance: we cannot guarantee that the error will be reduced on any given point. This behavior remains even when we increase the size of the circuit and the number of given points.

In Fig. 2.5, we study the impact of the different loss functions with different normalizations in the learning problem. We simulated the regression using two different loss functions, ℓ_{h^1} and ℓ_{l^2} under two different normalizations, with the domains $\left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$ and $[-\pi, \pi]$. The first noticeable phenomenon that we can see is that using the h^1 norm instead of the l^2 norm when the data is normalized to lie in the interval $\left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$ not only reduces the variance stemming

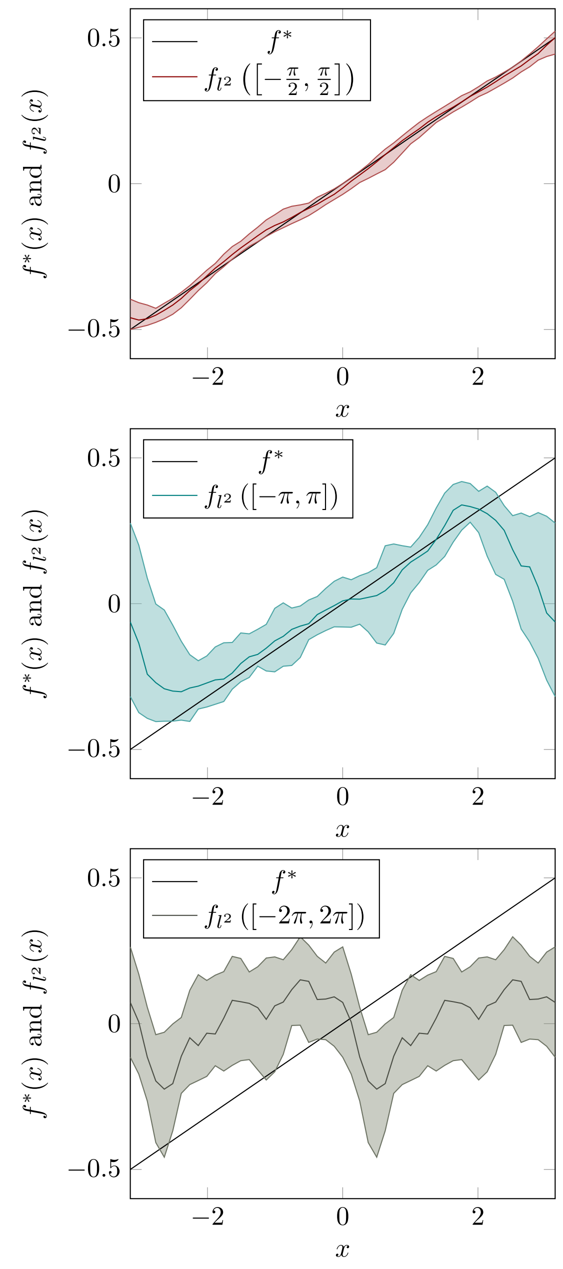


Figure 2.4: In this picture we have trained the PQC of Fig. 2.3 to approximate the function $f^*(x) = \frac{x}{2\pi}$. We have used 10 training points, the ℓ_{l^2} loss function and 100 epochs with the Adam optimizer. The experiments have been repeated 100 times. In the first panel we have normalized the data to lie in the interval $\left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$. In the second panel we have normalized the data to lie in the interval $[-\pi, \pi]$. In the third panel we have normalized the data to lie in the interval $[-2\pi, 2\pi]$.

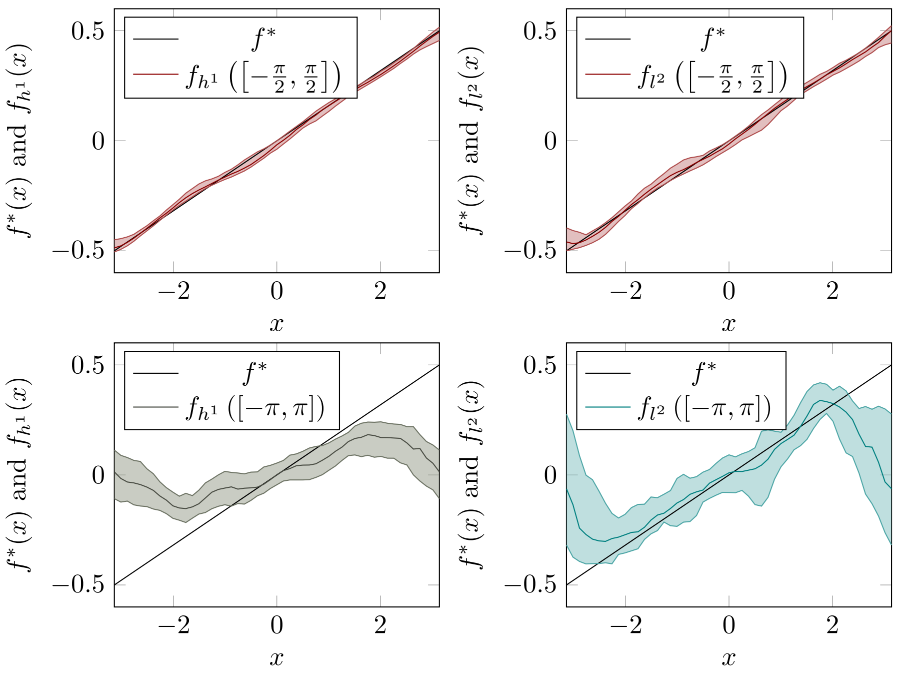


Figure 2.5: In this picture we have trained the PQC of Fig. 2.3 to approximate the function $f^* = \frac{x}{2\pi}$, using the two different loss functions ℓ_{h^1} and ℓ_{l^2}. We have used 10 training points (10 for the labels plus 10 for the derivative values when they are present) and 100 epochs with the Adam optimizer. The experiments have been repeated 100 times. In the upper panel, we have normalized the data to lie in the interval $\left[-\frac{\pi}{2},\frac{\pi}{2}\right]$. In the lower panel we have normalized the data to lie in the interval $[-\pi,\pi]$.

from repeating the experiments 100 times, but also has some impact on the bias. What might be more surprising is the effect of the h^1 norm when the data is normalized to lie in the interval $[-\pi,\pi]$. Instead of getting a better approximation w.r.t. the l^2 we worsen it. We explain it with the fact that, when we normalize the data to lie in the interval $[-\pi,\pi]$, our PQC is not an approximator of H^1 but it is an approximator of L^2, i.e., it can approximate the function but it cannot simultaneously approximate the function and the derivatives. Thus, in the minimization process the PQC tries to find a balance between the error in the function and the error in the derivatives, worsening the results with respect to the quality of the function approximation.

2.5 Conclusions

In this chapter, we have developed a broader theory of approximation capacities of PQCs. We have shown how an appropriate choice of the data normalization greatly improves the expressivity of the PQCs. More specifically, we showed that a min-max feature scaling that normalizes the input data along each dimension to lie in the range $[-\frac{\pi}{2}, \frac{\pi}{2}]$ makes PQCs universal approximators in the L^p space with $1 \leq p < \infty$, the continuous function space and the H^k space.

Moreover, since with this normalization we are able to approximate functions in the sense of the L^p, the C^0 and the H^k distance, we discussed that a loss function which is consistent with those distances in training the models might be more appropriate than other choices. In particular, the natural choice for the C^0 would be the l^∞ distance. However, since the l^∞ distance is not differentiable, which makes the optimization of PQCs harder, we leveraged Sobolev inequalities to show that the h^1 distance is consistent with the C^0 distance in $\mathbb{R}$ while being differentiable. We showed further, that the h^k distances are consistent with the L^p and the H^k distances.

Lastly, we performed some numerical experiments to illustrate how this simple choice of normalization and loss function can vastly improve the results in practice.

The data normalization technique can be seen as a complementary result to the work of [67]. Nevertheless, there is still much work to do in this direction. For example, if instead of only taking a min-max feature scaling, we can combine it with a mapping of the form $\tilde{x} = \arcsin(x)$ to end up with a series that closely resembles Chebyshev polynomials, which are better suited for certain problems. In analogy with neural networks, the data encoding strategy is playing a similar role to that of the activation functions.

The relation between the ℓ_{h^k} loss functions and the L^p generalization bounds can be seen as a complementary result to differential machine learning [89] and to generalization bounds for PQCs as derived in [73]. This is the first work that gives some insight on why differential machine learning leads to better generalization results. From the relations that we derived, one would expect this technique to fail as we increase the input dimension. However, in practice it has demonstrated very good results, as shown in [89], where a 7-dimensional Basket option was trained using the ℓ_{h^1} loss function. An interesting line of research would be to study the threshold at which differential machine learning starts to fail.

Since a natural application are physical systems governed by differential equations where data on the derivatives of a target function are available, another open question remains as to how our approach compares to standard differential equation solvers in these scenarios.

2.A Proof of Theorems 2.2, 2.3 and 2.4

For proving Thms. 2.2, 2.3 and 2.4, we need two preliminary results. Firstly, we need to show that a quantum circuit can realize the ℓ^1-Fejér's mean of $C^0\left(\mathbb{T}^N\right)$ and $L^p\left(\mathbb{T}^N\right), \forall 1 \leq p < \infty$ functions. Secondly, we need to prove that we can define periodic extensions of functions belonging to $C^0\left(U\right)$ and $H^k\left(U\right), \forall 1 \leq k < \infty$, where U is compactly contained in $\mathbb{T}^N$ to functions belonging to $C^0\left(\mathbb{T}^N\right)$ and $H^k\left(\mathbb{T}^N\right), \forall 1 \leq k < \infty$ respectively . The combination of both results plus Fejér's theorem in multiple dimensions naturally yields Thms. 2.2 and 2.3. Thm. 2.4 can be proven by a standard approximation theorem of the Fourier series.

2.A.1 Féjer's mean

We call the function

$$\sigma_{NK}(f_{m'}) = \sum_{\mathbf{j}\in\mathbb{Z}_K^N} \left(1 - \frac{\|\mathbf{j}\|_1}{NK}\right) \hat{f}_{\mathbf{j}} e^{i\mathbf{x}\cdot\mathbf{j}}, \tag{2.38}$$

where $\hat{f}_{\mathbf{j}}$ is the $\mathbf{j}$-th Fourier coefficient of $f_{m'}$, the ℓ^1-Fejér's mean of $f_{m'}$.

We will show that our PQC can realize the Fejér's mean of any well-defined function. In Appendix C of [67], the authors showed that the quantum model family $f_{m'}$ can be written as a generalized trigonometric series of the form

$$f_{m'}(\mathbf{x}) = \sum_{\mathbf{j}\in\mathbb{Z}_K^N} c_{\mathbf{j}} e^{i\mathbf{x}\cdot\mathbf{j}}, \tag{2.39}$$

where $\mathbb{Z}_K^N = \{-K, -K+1, ..., 0, ..., K-1, K\}^N$ is contained in the Cartesian product of the frequency spectrum associated with H_m, as defined in Definition 1 and that the coefficients $c_{\mathbf{j}}$ are completely determined by the observable, up to the complex-conjugation symmetry that guarantees that the model output is a real-valued function. Note that we can choose the coefficients $c_{\mathbf{j}}$ as:

$$c_{\mathbf{j}} = \left(1 - \frac{\|\mathbf{j}\|_1}{NK}\right) \hat{f}_{\mathbf{j}}, \tag{2.40}$$

which are the coefficients of the ℓ^1-Fejér's mean in Eq.(2.38).

2.A.2 Periodic extension for C^0 functions

By the Tietze extension theorem [101], there exists a function $g_1 \in C^0(\mathbb{R}^N)$ with $g_1|_U = f^*$. Then, we define a function $g_2 \in C^0(\mathbb{R}^N)$ with $g_2|_{\overline{U}} = 1$ and

$g_2|_{\mathbb{R}^N\setminus V} = 0$, where V is defined as $\overline{U} \subset V \subset (0, 2\pi)^N$. This set V exists since U is compactly contained in $[0, 2\pi]^N$.

We can explicitly construct the function g_2 in the following way: Let $\delta > 0$, such that the closure $\overline{\omega_{2\delta}}$ of the 2δ-neighborhood of ω, is contained in $[0, 2\pi]^N$, which is possible due to U being compactly contained in $[0, 2\pi]^N$. We define $V := \omega_{2\delta}$ and a function $\psi_\delta \in C^0(\mathbb{R}^N)$, supported on the δ Ball in $\mathbb{R}^N$ centered around 0 and normalized as $\int_{\mathbb{R}^N} \psi_\delta(x)dx = 1$. Then, we define g_2 as the convolution of $\mathbb{1}_{U_\delta}$ and ψ_δ:

$$g_2(x) = \int_{\mathbb{R}^N} \mathbb{1}_{U_\delta}(\tau)\psi_\delta(\tau - x)d\tau \ . \tag{2.41}$$

With this construction, g_2 satisfies the required properties. We define the extension f_{ext} as the product $g_1 g_2$, which yields a function f^*_{ext} with

$$f^*_{ext}|_U = f^*, \tag{2.42}$$
$$f^*_{ext}|_{\mathbb{R}^N\setminus V} = 0, \text{ hence} \tag{2.43}$$
$$f^*_{ext}(x) = f^*_{ext}(y) \quad \forall x, y \in \partial\mathbb{T}^N \ . \tag{2.44}$$

The extension f^*_{ext} defined in this way is therefore an element of $C^0([0, 2\pi]^N)$ with periodic boundary conditions, so it can be viewed as a function on the N-dimensional torus $\mathbb{T}^N$.

2.A.3 Periodic extension for H^k functions

By the extension theorems for Sobolev functions [102, Theorem 2.2, Part 2], there exists a function $g_1 \in H^k(\mathbb{R}^N)$ with $g_1|_U = f^*$. Then, we define a function $g_2 \in H^k(\mathbb{R}^N)$ with $g_2|_{\overline{U}} = 1$ and $g_2|_{\mathbb{R}^N\setminus V} = 0$, where V is defined as $\overline{U} \subset V \subset (0, 2\pi)^N$. This set V exists since U is compactly contained in $[0, 2\pi]^N$.

We can explicitly construct the function g_2 in the following way: Let $\delta > 0$, such that the closure $\overline{\omega_{2\delta}}$ of the 2δ-neighborhood of ω, is contained in $[0, 2\pi]^N$, which is possible due to U being compactly contained in $[0, 2\pi]^N$. We define $V := \omega_{2\delta}$ and a function $\psi_\delta \in H^k(\mathbb{R}^N)$, supported on the δ Ball in $\mathbb{R}^N$ centered around 0 and normalized as $\int_{\mathbb{R}^N} \psi_\delta(x)dx = 1$. Then, we define g_2 as the convolution of $\mathbb{1}_{U_\delta}$ and ψ_δ:

$$g_2(x) = \int_{\mathbb{R}^N} \mathbb{1}_{U_\delta}(\tau)\psi_\delta(\tau - x)d\tau \ . \tag{2.45}$$

With this construction, g_2 satisfies the asked properties. We define the extension

f_{ext} as the product $g_1 g_2$, which yields a function f^*_{ext} with

$$f^*_{ext}|_U = f^*, \tag{2.46}$$

$$f^*_{ext}|_{\mathbb{R}^N \setminus V} = 0, \text{ hence} \tag{2.47}$$

$$f^*_{ext}(x) = f^*_{ext}(y) \quad \forall x, y \in \partial \mathbb{T}^N . \tag{2.48}$$

The such defined extension f^*_{ext} is thus an element of $H^k([0, 2\pi]^N)$ with periodic boundary conditions, so we can map it onto the N-dimensional torus $\mathbb{T}^N$.

2.A.4 Proof of Theorems 2.2, 2.3 and 2.4

The final step leverages the power of Fejèr's theorem in multiple dimensions:

Theorem 2.8. *[103, Theorem 2] For all functions $f^* \in L^p\left(\mathbb{T}^N\right)$ with $1 \le p < \infty$, and for all $\epsilon > 0$, there exists some $t \in \mathbb{N}$, such that*

$$\left\|\sigma_t\left(f\right) - f^*\right\|_{L^p} < \epsilon. \tag{2.49}$$

Combining Thm. 2.8 with the fact that quantum circuits can recover any ℓ^1-Fejér's mean as shown in Appendix 2.A.1 directly implies Thm. 2.2.

Similarly, for continuous functions we have another version of Fejér's theorem for continuous functions:

Theorem 2.9. *[103, Theorem 2] For all functions $f^* \in C^0\left(\mathbb{T}^N\right)$, and for all $\epsilon > 0$, there exists some $t \in \mathbb{N}$, such that*

$$\left\|\sigma_t\left(f\right) - f^*\right\|_{\infty} < \epsilon. \tag{2.50}$$

Combining Thm. 2.9 with the fact that quantum circuits can recover any ℓ^1-Fejér's mean as shown in Appendix 2.A.1 and the fact that we can extend any function in $C^0\left(U\right), \forall 1 \le p < \infty$ where U is compactly contained in $\mathbb{T}^N$ to a function in $C^0\left(\mathbb{T}^N\right), \forall 1 \le p < \infty$ as shown in Appendix 2.A.2 directly implies Thm. 2.3.

We finally prove Thm. 2.4, which uses the setup in [67] as described in Sec. 2.2: We note firstly that the quantum model family $f_{m'}$ generates a truncated Fourier series $\tilde{f}$ in the domain $[0, 2\pi]^N$ of the form

$$\tilde{f}(\mathbf{x}) = \sum_{\mathbf{j} \in \mathbb{Z}_K^N} c_{\mathbf{j}} e^{i\mathbf{x}\cdot\mathbf{j}}, \tag{2.51}$$

where $\mathbb{Z}_K^N = \{-K, -K+1, ..., 0, ..., K-1, K\}^N$ is contained in the Cartesian product of the frequency spectrum associated with H_m, as defined in Definition 1. The proof of that is written in Appendix C of [67].

Secondly, we can extend the function f^* defined on U to a periodic function f^*_{ext} on $[0,2\pi]^N$ via the construction shown in Appendix 2.A.3. As written in Thm. 1.1 in [104], the Fourier series of f^*_{ext}, which we can write in the form of Eq. (2.51), converges in the H^k-distance to f^*_{ext}. As $f^*_{ext}(x) = f^*(x)$ for all $x \in U$, the Fourier series of f^*_{ext} converges in the H^k-distance to f^* on U. This implies Thm. 2.4.

2.B Proof of Theorems 2.5, 2.6 and 2.7

In this appendix, we prove Thms. 2.5, 2.6 and 2.7, for which we need several preliminary definitions and results:

Definition 4 (L-Lipschitz loss function)**.** *Let $(\mathcal{Y}, d_{\mathcal{Y}})$ be a metric space with metric $d_{\mathcal{Y}}$ and let $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ be a loss function. We call it $L-$ Lipschitz with regard to a fixed $y \in \mathcal{Y}$, if there exists a constant $L \geq 0$, such that for all $z_1, z_2 \in \mathbb{R}$,*

$$d_{\mathcal{Y}}\left(\ell(y,z_1), \ell(y,z_2)\right) \leq L\left|z_1 - z_2\right| \ . \tag{2.52}$$

Theorem 2.10 (Generalization bound for general trigonometric series)**.** *[73, Theorem 11] Let $N, I \in \mathbb{N}$. Let $B > 0$ and $\tilde{B} > 0$ be such that $\mathcal{F}^B_\Omega \subseteq \mathcal{H}^{\tilde{B}}_\Omega$, for the function families $\mathcal{F}^B_\Omega$ and $\mathcal{H}^{\tilde{B}}_\Omega$ as defined in Definition 3. Let $\ell : \mathbb{R} \times \mathbb{R} \to [0,c]$ be a bounded loss function such that $\mathbb{R} \ni z \mapsto \ell(y,z)$ is L- Lipschitz for all $y \in \mathbb{R}$. For any $\delta \in (0,1)$ and for any probability measure P on $[0,2\pi]^N \times \mathbb{R}$, with probability at least $1-\delta$ over the choice of i.i.d. training data $S \in ([0,2\pi]^N \times \mathbb{R})^I$ of size I, for every $f \in \mathcal{F}^B_\Omega$, the generalization error can be upper-bounded as*

$$\int_{[0,2\pi]^N \times \mathbb{R}} \ell(f^*(\boldsymbol{x}), f(\boldsymbol{x})) dP((\boldsymbol{x}), f^*(\boldsymbol{x})) - \frac{1}{|I|} \sum_{\boldsymbol{x}_i, f(\boldsymbol{x}_i) \in S} \ell(f^*(\boldsymbol{x}_i), f(\boldsymbol{x}_i)) \tag{2.53}$$

$$\leq \mathcal{O}\left(BL\sqrt{\frac{|\Omega|(\log(|\Omega|) + \log(\tilde{B}))}{I}} + c\sqrt{\frac{\log(1/\delta)}{I}} \right), \tag{2.54}$$

for a target function $f^* : [0,2\pi]^N \to \mathbb{R}$.

This theorem is written for loss functions that take two real values as an input, which is the case for most loss functions. We show that the theorem holds as well for the loss function ℓ_{h^k}:

Lemma 1. *Thm. 2.10 holds as well for the loss function $\ell_{h^k} : \mathbb{R}^{\binom{N}{k}+1} \times \mathbb{R}^{\binom{N}{k}+1} \to [0,c]$ with $N, k \in \mathbb{N}$ by choosing the frequency set Ω and the bounds*

B and $\tilde{B}$ large enough, such that both the functions f of a considered function family $\mathcal{F}$ and their derivatives $D^\alpha f$ for $|\alpha| \leq k$ are contained in the families $\mathcal{F}_\Omega^B \subseteq \mathcal{H}_\Omega^{\tilde{B}}$.

Proof. The proof goes analogous to the proof of Thm. 11 in [73]. There are two points which require special care:

Firstly, we need to adapt the application of Talagrand's lemma which is used to upper bound the Rademacher complexity. Let us use the ℓ_{l^2} loss function

$$\ell_{l^2}(f^*(\boldsymbol{x}), f(\boldsymbol{x})) = (f^*(\boldsymbol{x}) - f(\boldsymbol{x}))^2\,, \tag{2.55}$$

which is related to the loss function ℓ_{h^k} by

$$\ell_{h^k}(f^*(\boldsymbol{x}), f(\boldsymbol{x})) = \sum_{|\alpha|\leq k} \ell_{l^2}\left(D^\alpha f^*(\boldsymbol{x}), D^\alpha f(\boldsymbol{x})\right)\;. \tag{2.56}$$

By using the reverse triangle inequality, we can prove the Lipschitzness of the loss function ℓ_{l^2}, for a fixed $f^*(\boldsymbol{x}) \in L^2([0, 2\pi]^N)$:

$$\begin{aligned}
&\Big|\ell_{l^2}(f_1(\boldsymbol{x}), f^*(\boldsymbol{x})) - \ell_{l^2}(f_2(\boldsymbol{x}), f^*(\boldsymbol{x}))\Big| \\
=&\Big||f^*(\boldsymbol{x}) - f_1(\boldsymbol{x}))|^2 - |f^*(\boldsymbol{x}) - f_2(\boldsymbol{x}))|^2\Big| \\
\leq&\Big||f^*(\boldsymbol{x}) - f_1(\boldsymbol{x}) - (f^*(\boldsymbol{x}) - f_2(\boldsymbol{x}))|^2\Big| \\
=&\Big||(f^*(\boldsymbol{x}) - f^*(\boldsymbol{x})) - (f_1(\boldsymbol{x})) - f_2(\boldsymbol{x})))|^2\Big| \\
=&|(f_1(\boldsymbol{x})) - f_2(\boldsymbol{x})))|^2\;.
\end{aligned}$$

Thus, the loss function ℓ_{l^2} is L-Lipschitz with the Lipschitz constant $L = 1$. Note that this is the Lipschitz constant of the loss function ℓ_{l^2}, which is not related to the Lipschitz constant of functions of the function space $L^2([0, 2\pi]^N)$.

Parallel to the proof of Thm. 11 in [73], we now define the set

$$\mathcal{G} = \Big\{[0, 2\pi]^N \times [0, 2\pi]^N \ni (\boldsymbol{x}, \boldsymbol{x}) \mapsto \ell_{h^k}(f^*(\boldsymbol{x}), f(\boldsymbol{x}))\Big| f^* \in H^k([0, 2\pi]^N) \\ \text{and } f \in \mathcal{F}_\Omega^B\Big\}.$$

We can now upper bound the Rademacher complexity $\hat{\mathcal{R}}_S(\mathcal{G})$ for a training set

S with I data points and a target function f^* as

$$\begin{aligned}
\hat{\mathcal{R}}_S(\mathcal{G}) &= \frac{1}{I}\mathbb{E}_\sigma\left[\sup_{f\in\mathcal{F}_\Omega^B}\sum_{i=1}^{I}\sigma_i \ell_{h^k}(f(\boldsymbol{x}_i), f^*(\boldsymbol{x}_i))\right] \\
&= \frac{1}{I}\mathbb{E}_\sigma\left[\sup_{f\in\mathcal{F}_\Omega^B}\sum_{i=1}^{I}\sigma_i \sum_{|\alpha|\le k}\ell_{l^2}(D^\alpha f(\boldsymbol{x}_i), D^\alpha f^*(\boldsymbol{x}_i))\right] \\
&\le \frac{1}{I}\mathbb{E}_\sigma\left[\sup_{D^\alpha f(\boldsymbol{x})\in\mathcal{F}_\Omega^B,|\alpha|\le k}\sum_{i=1}^{I}\sigma_i \sum_{|\alpha|\le k}\ell_{l^2}(D^\alpha f(\boldsymbol{x}_i), D^\alpha f^*(\boldsymbol{x}_i))\right] \\
&= \sum_{|\alpha|\le k}\frac{1}{I}\mathbb{E}_\sigma\left[\sup_{D^\alpha f(\boldsymbol{x})\in\mathcal{F}_\Omega^B}\sum_{i=1}^{I}\sigma_i \ell_{l^2}(D^\alpha f(\boldsymbol{x}_i), D^\alpha f^*(\boldsymbol{x}_i))\right] \\
&\le \xi \sup_{|\alpha|\le k}\frac{1}{I}\mathbb{E}_\sigma\left[\sup_{D^\alpha f(\boldsymbol{x})\in\mathcal{F}_\Omega^B}\sum_{i=1}^{I}\sigma_i \ell_{l^2}(D^\alpha f(\boldsymbol{x}_i), D^\alpha f^*(\boldsymbol{x}_i))\right] .
\end{aligned}$$

The i.i.d. random variables $\sigma_i \in \{-1, 1\}$ are the Rademacher random variables and ξ is the number of derivatives D^α with $|\alpha| \le k$. Here, we first used the relation between the loss functions ℓ_{l^2} and ℓ_{h^k}. Then, we used the fact that the supremum over functions and derivatives $D^\alpha f(\boldsymbol{x}) \in \mathcal{F}_\Omega^B, |\alpha| \le k$ which are independent from each other is larger than the supremum which is only taken over the functions $f \in \mathcal{F}_\Omega^B$, in which case the derivatives that are taken account in the loss functions have to be the derivatives of these functions. In the last inequality, we used that each of the ξ terms in the sum $\sum_{|\alpha|\le k}$ can be bounded above by its supremum.

We can now apply Talagrand's lemma on the quantity

$$\frac{1}{I}\mathbb{E}_\sigma\left[\sup_{D^\alpha f(\boldsymbol{x})\in\mathcal{F}_\Omega^B}\sum_{i=1}^{I}\sigma_i \ell_{l^2}(D^\alpha f(\boldsymbol{x}_i), D^\alpha f^*(\boldsymbol{x}_i))\right], \tag{2.57}$$

for a fixed $|\alpha| \le k$ in which way we obtain the upper bound

$$\hat{\mathcal{R}}_S(\mathcal{G}) \le \xi \hat{\mathcal{R}}_{S|_x}(\mathcal{F}_\Omega^B), \tag{2.58}$$

where we used that the loss function ℓ_{l^2} has the Lipschitz constant $L = 1$ and where $S|_x := \{\boldsymbol{x}_i\}_{i=0}^{I}$ is the set of the unlabeled training data points. The supremum $\sup_{|\alpha|\le k}$ can be omitted on the right hand side of the bound, since the subset $S|_x$ of the training set does not include the labels $D^\alpha f^*(\boldsymbol{x}_i)$ and since we assumed the function family $\mathcal{F}_\Omega^B$ to contain the relevant derivatives as well. This upper bound corresponds to Eq.(97) in the proof of Thm. 11 in [73], apart from the additional factor ξ.

Secondly, in the last step of the proof in [73], the authors use standard generalization bounds as stated in Thm. 1.15 in [105]. The formulation of this standard generalization bound theorem allows for the loss function ℓ_{h^k} as well. □

Definition 5 (Compact embedding). *[106, Definition 1.25] Let X and Y be normed spaces with the norms $\|\cdot\|_X$ and $\|\cdot\|_Y$, respectively, and X a subspace of Y. Let $I : X \to Y$, $Ix = x$ for all $x \in X$ be the embedding operator from X to Y. We say that X is continuously embedded in Y, and write $X \to Y$, if there exists a constant C, such that*

$$\|Ix\|_Y \leq C\|x\|_X, \forall x \in X \ . \tag{2.59}$$

We call the embedding compact, if X is continuosly embedded in V and the embedding operator I is compact.

Definition 6. *We write $C^0_B(U)$ for the space of bounded, continuous functions on U.*

Definition 7 (Finite cone and Cone condition). *[106, Definitions 4.4 and 4.6] Let $v, x \in \mathbb{R}^N$ be nonzero vectors, let $\angle(x, v)$ be the angle between vectors x and v. For given such v, a $\rho > 0$ and a κ such that $0 < \kappa \leq \pi$, the set*

$$C_{v,\rho,\kappa} = \{x \in \mathbb{R}^N : x = 0 \text{ or } 0 < \|x\| \leq \rho, \angle(x, v) \leq \kappa/2\} \tag{2.60}$$

is called a finite cone of height ρ, axis direction v and aperture angle κ with vertex at the origin.

We say that $U \subseteq \mathbb{R}^N$ satisfies the cone condition, if there exists a finite cone C such that every $x \in U$ is the vertex of a finite cone C_x contained in U and congruent to C.

Theorem 2.11 (Rellich-Kondrachov). *[106, Theorem 6.3, Part I and II] Let U be a domain in $\mathbb{R}^N$ satisfying the cone condition, let U_0 be a bounded subdomain of U, and let U_0^N be the intersection of U_0 with a N-dimensional plane in $\mathbb{R}^N$. Let $k \geq 1$ be integers. Let one of the following cases hold:*

1. $2k < N$ *and* $1 \leq p < 2N/(N - 2k)$
2. $2k = N$ *and* $1 \leq p < \infty$
3. $2k > N$ *and* $1 \leq p < \infty$

Then, the following embeddings are compact:

$$H^k(U) \to L^p(U_0^N) \ . \tag{2.61}$$

Additionally, in case 3, the following embedding is compact:

$$H^k(U) \to C^0_B(U_0^N) \ . \tag{2.62}$$

Remark. *The theorem relates to the Rellich-Kondrachov Theorem stated in [106] in the following way:*

- *Case 1 and Case 2 are the two cases stated in Part 1 of Thm. 6.3 in [106].*
- *Case 3 corresponds to the first and second case of Part 2 in Thm. 6.3 in [106].*
- *We use a different notation: The symbols Ω, j, p, q, k, n, m used in [106] are here equal to $U, 0, 2, p, N, N, k$, respectively.*
- *We formulate the theorem for the special cases $W^{k,2} = H^k$ and $W^{0,p} = L^p$ of the Sobolev spaces.*

With these preliminary results, we can prove Thms. 2.5, 2.6 and 2.7, which we restate here:

Theorem 2.5 (Generalization bound for H^k)**.** *Let $f^* \in \mathcal{F} \subseteq H^k([0, 2\pi]^N)$ be a target function, and let there be a $B > 0$ and a $\tilde{B} > 0$, such that $\mathcal{F}^B_\Omega \subseteq \mathcal{H}^{\tilde{B}}_\Omega$ is a suitable model family. Let us further assume that $\ell_{h^k}(f_1(\boldsymbol{x}), f_2(\boldsymbol{x})) \leq c$ for all $\mathbf{x} \in [0, 2\pi]^N$, and for all $f_1, f_2 \in \mathcal{F}^B_\Omega$ or $\mathcal{F}$. For any $\delta \in (0, 1)$ and the empirical risk $D_{h^k}(f^*, f)$ trained on an i.i.d. training data S with size I and containing data of ξ partial derivatives, the following holds for all functions $f \in \mathcal{F}^B_\Omega$ with probability at least $1 - \delta$:*

$$D_{H^k}(f^*, f) \leq D_{h^k}(f^*, f) + r(|\Omega|, \xi, B, \tilde{B}, c, I, \delta), \tag{2.63}$$

where $r(|\Omega|, \xi, B, \tilde{B}, c, I, \delta) \to 0$ as $I \to \infty$.

Proof. In the work [73], the authors developed generalization bounds for the function family defined in Eq. (2.10). We restated the theorem in Thm. 2.10. As we have shown in Corollary 1, the theorem also holds for the loss function ℓ_{h^k}.

According to the assumption, the function f^* is in $\mathcal{F}^B_\Omega$. The choice of the constant $\tilde{B}$ such that $\mathcal{F}^B_\Omega \subseteq \mathcal{H}^{\tilde{B}}_\Omega$ is satisfied depends on the encoding strategy. As written in [73], it can for example for integer valued frequencies be chosen as $\tilde{B} = 2B$. Thus, Lemma 1 can be applied and the following bound holds:

$$D_{H^k}(f^*, f_{h^k}) \leq D_{h^k}(f^*, f_{h^k}) + r(|\Omega|, B, \tilde{B}, c, I, \delta), \tag{2.64}$$

with a function $r(|\Omega|, B, \tilde{B}, c, I, \delta)$ which tends to 0 as $I \to \infty$. □

Theorem 2.6 (Generalization bound for L^p)**.** *Let $f^* \in \mathcal{F} \subseteq H^k([0,2\pi]^N)$ be a target function, and let there be a $B > 0$ and a $\tilde{B} > 0$, such that $\mathcal{F}_\Omega^B \subseteq \mathcal{H}_\Omega^{\tilde{B}}$ is a suitable model family. Let us further assume that $\ell_{h^k}(f_1(\boldsymbol{x}), f_2(\boldsymbol{x})) \leq c$ for all $\mathbf{x} \in [0,2\pi]^N$, and for all $f_1, f_2 \in \mathcal{F}_\Omega^B$ or $\mathcal{F}$. Assume that $k, p \in \mathbb{N}$ satisfy one of the two following cases:*

1. *$N\left(\frac{1}{2} - \frac{1}{p}\right) < k < N/2$ and $1 \leq p < N$.*
2. *$k \geq N/2$ and $1 \leq p < \infty$.*

For any $\delta \in (0,1)$ and the empirical risk $D_{h^k}(f^*, f)$ trained on an i.i.d. training data S with size I and containing data of ξ partial derivatives, the following holds for all functions $f \in \mathcal{F}_\Omega^B$ with probability at least $1-\delta$:

$$\frac{1}{C} D_{L^p}(f^*, f) \leq D_{h^k}(f^*, f) + r(|\Omega|, \xi, B, \tilde{B}, c, I, \delta), \tag{2.65}$$

where C is a constant and $r(|\Omega|, \xi, B, \tilde{B}, c, I, \delta) \to 0$ as $I \to \infty$.

Proof. We will prove the theorem by proving the following two inequalities:

$$\frac{1}{C} D_{L^p}(f^*, f) \leq D_{H^k}(f^*, f) \leq D_{h^k}(f^*, f) + r(|\Omega|, B, \tilde{B}, c, I, \delta)\ . \tag{2.66}$$

The right hand side inequality is following directly from Thm. 2.5, and the left hand side inequality is a consequence of Thm. 2.11. Let us look at case 1 in Thm. 2.11: We want to rewrite the bound $p < 2N/(N-2k)$ as an upper bound for k for a given p. Let us therefore firstly check, which values p is allowed to reach. Due to k being bound from above by $k < N/2$, the upper bound on p, $p < 2N/(N-2k)$ is maximal for $k = \frac{N}{2} - 1$, in which case the upper bound on p becomes $p < N$. That means that values for p chosen in $1 \leq p < N$ are valid values. With p such chosen, the bound $p < 2N/(N-2k)$ is equivalent to bounding k in the following way:

$$N\left(\frac{1}{2} - \frac{1}{p}\right) < k\ . \tag{2.67}$$

For case 2 in Thm. 2.11, we have the inequalities $k \geq N/2$ and $1 \leq p < \infty$.

Further, because of the assumptions $\ell_{h^k}(f^*(\mathbf{x}), f(\mathbf{x})) \leq c$ for all $\mathbf{x} \in [0,2\pi]^N$, the subdomain $U = [0,2\pi]^N$ is equal to U_0, and because an N-dimensional plane in $\mathbb{R}^N$ is $\mathbb{R}^N$ itself, U is also equal to U_0^N. Let C be a cone of height at most π, angle at most $\pi/2$. Then, for each $\boldsymbol{x}$ in $U = [0,2\pi]^N$, we can choose an appropriate axis direction such that C_x lies entirely in U, so it satisfies the cone condition.

To sum up, Thm. 2.11 states that for the cases

1. $N\left(\frac{1}{2}-\frac{1}{p}\right) < k < N/2$ and $1 \leq p < N$.
2. $k \geq N/2$ and $1 \leq p < \infty$,

the following embeddings are compact:

$$H^k([0,2\pi]^N) \to L^p([0,2\pi]^N) \ . \tag{2.68}$$

According to the definition of a compact embedding (Definition 5), there exists a constant C, such that

$$\|f^* - f\|_{L^p} \leq C\|f^* - f\|_{H^k} \ . \tag{2.69}$$

□

Theorem 2.7 (Generalization bound for C^0)**.** *Let $f^* \in \mathcal{F} \subseteq H^k([0,2\pi]^N)$ be a target function, and let there be a $B > 0$ and a $\tilde{B} > 0$, such that $\mathcal{F}_\Omega^B \subseteq \mathcal{H}_\Omega^{\tilde{B}}$ is a suitable model family. Let us further assume that $\ell_{h^k}(f_1(\boldsymbol{x}), f_2(\boldsymbol{x})) \leq c$ for all $\mathbf{x} \in [0,2\pi]^N$, and for all $f_1, f_2 \in \mathcal{F}_\Omega^B$ or $\mathcal{F}$ and that $\|f\|_\infty \leq B$ for all $f \in \mathcal{F}_\Omega^B$. Assume, that $k \in \mathbb{N}$ satisfies $k > N/2$. For any $\delta \in (0,1)$ and the empirical risk $D_{h^k}(f^*, f)$ trained on an i.i.d. training data S with size I and containing data of ξ partial derivatives, the following holds for all functions $f \in \mathcal{F}_\Omega^B$ with probability at least $1-\delta$:*

$$\frac{1}{C} D_{C^0}(f^*, f) \leq D_{h^k}(f^*, f) + r(|\Omega|, \xi, B, \tilde{B}, c, I, \delta), \tag{2.70}$$

where C is a constant and $r(|\Omega|, \xi, B, \tilde{B}, c, I, \delta) \to 0$ as $I \to \infty$.

Proof. The prove of this theorem is equivalent to the proof of Thm. 2.6 above. We will prove this theorem as well by proving the following two inequalities:

$$\frac{1}{C} D_{L^p}(f^*, f) \leq D_{H^k}(f^*, f) \leq D_{h^k}(f^*, f) + r(|\mathcal{M}|, I, \delta) \ . \tag{2.71}$$

The right hand side inequality is following directly from Thm. 2.5, and the left hand side inequality is a consequence of Thm. 2.11. As written in the proof of Thm. 2.6, the assumptions of Thm. 2.11 are satisfied, we can thus also apply it here.

The upper bound on the distance $D_{C^0}(f^*, f)$ in the supremum norm is a direct consequence of the third case in Thm. 2.11.

□

CHAPTER 3

Error and Resource Estimates of Variational Quantum Algorithms for Solving Differential Equations Based on Runge-Kutta Methods

3.1 Introduction

For more than four decades, quantum computing has captivated the minds of researchers, but significant experimental advancements have only been achieved in recent years. We are living in the era of Noisy Intermediate-Scale Quantum (NISQ) devices [15, 107–109], which, while capable of certain super classical computations in principle, are also susceptible to noise and errors, preventing them from providing substantial computational advantages over classical computers. Efforts to tackle these challenges have resulted in proliferate research lines dedicated to error correction and noise mitigation across various quantum architectures [110]. One approach to ameliorating some of the issues involves hybrid quantum-classical algorithms that use quantum computers to process information, while classical computers handle the correction and optimization processes, helping to reduce errors and improve efficiency. For example, in variational quantum algorithms, such as the Variational Quantum Eigensolver (VQE) or the Quantum Approximate Optimization Algorithm (QAOA), the

The contents of this chapter have been published in Ref. [46].

quantum part evaluates the cost function depending on a given set of parameters, while the classical part optimizes these parameters to minimize the cost function. In physical terms, the cost function is typically the energy of a Hamiltonian that encodes the optimization task. Extracting the final result from such a hybrid device is challenging, as errors arise both from quantum and classical sources. A notable source of error is the measurement shot noise arising out of the discrete nature of quantum measurements and the limited number of measurements taken [111].

Solving differential equations (DEs) is a critical task in various scientific and engineering fields, and several quantum computing-based proposals have been developed to tackle this problem. First ideas [35, 36, 112] were built around the quantum linear system algorithm [37], but they require fault-tolerant quantum computers. Later, approaches based on variational quantum algorithms were introduced, e.g., in Refs. [26, 113–115]. In one of the latter approaches, the DE is mapped to an imaginary-time Schrödinger equation [116]. It is solved by a variational algorithm, where the time evolution of a quantum state is approximated by a variational quantum circuit and mapped to the time evolution of the parameters of this circuit. After classically computing the parameters at the final time step, one reinserts them into the variational quantum circuit to prepare the evolved quantum state. This is an approach originally proposed in Ref. [117] for quantum simulation and has attracted a lot of interest since [118–124].

In the approach of Refs. [116, 117], the resulting time evolution of the quantum circuit parameters has the form of an ordinary differential equation (ODE). Classically, ODEs can be solved by time-discretization methods such as the Euler method or the more general Runge-Kutta methods [125] (RKMs) that can be categorized by their order p. The first proposed variational quantum algorithm for solving differential equations based on the Euler method is shown in Ref. [113]. Those methods approximate the time evolution by calculating a truncated Taylor expansion at each time step, and the higher the order of the RKM, the lower is the resulting truncation error, but the higher is the required number of function evaluations. There are also different generalizations of the Euler methods, like the linear multistep methods or the general linear methods [125], which can lead to similar accuracies as RKMs with different requirements of the number of function evaluations.

Previous works have proposed that choosing RKMs, such as the widely used classical RKM [126] with the order $p = 4$ instead of the Euler method, will also be favorable in solving the time evolution of the quantum circuit parameters [117, 127, 128]. However, each function evaluation incorporates evaluations of quantum circuits, making higher-order RKMs more demanding on the quantum resources. Moreover, quantum circuit evaluations are affected by the "shot noise bottleneck," where the error scales as $\mathcal{O}(1/N_{meas}^2)$ with the number of measurements N_{meas}, introducing an additional source of error. The number of measurements is a precious resource in quantum computing, as it is the most costly and time-consuming operation and as it is severely limited by the available runtime of the device before it requires recalibration [107, 129, 130]. That is why, for the practicality of the variational algorithm, it is crucial to minimize the total number of quantum circuit measurements.

In this chapter, we focus on solving differential equations based on the approach of Refs. [116, 117]. We investigate whether higher-order RKMs outperform the Euler method in solving the time evolution of the circuit parameters by analyzing the different sources of error and resource requirements. Specifically, we provide a detailed analysis of the total error of the variational quantum algorithm for solving differential equations, defined as the trace distance between the actual solution and the output of the variational algorithm. We explicitly consider the truncation error associated with the chosen RKM and the shot noise error. Other relevant sources of errors, such as circuit error (gate infidelity, bias, SPAM errors) and the representation error (the variational circuit being able to represent the solution at all time steps with its parameters), are assumed ideal as they depend among others on the chosen Ansatz and problem instance [131]. We establish rigorous error bounds and use them to estimate a sufficient number of circuit evaluations required by the algorithm for a given target error, based on the order of the RKM. Additionally, we perform an analysis of the RKMs under the assumption of no shot noise.

Further, we validate the resource estimates through benchmarking: the analysis without shot noise is demonstrated with a simple ODE, while the analysis of the variational algorithm is applied to option pricing, where the dynamics are described by the Black-Scholes equation [26, 81]. The latter is a partial differential equation that has attracted a lot of attention in the variational quantum computing community [26, 128, 132]. This shows that the application of our method is not restricted to ODEs but can also be applied to solving partial differential equations. We directly compare the total number of circuit evaluations required by the algorithm, depending on the order of the chosen RKM. In this chapter, we derive rigorous error bounds that are general, but may overestimate the true error in practice. Our resource estimates are based on optimizing resources with respect to these upper bounds and might therefore be overly conservative. Such an approach constitutes the optimal parameter selection strategy, providing a guaranteed success probability.

Similar error and resource estimates have been conducted in Refs. [116, 133–135]. However, our analysis is integrating the truncation errors of RKMs and shot noise errors, making it comprehensive, and offers direct comparisons of resource requirements between different RKMs and an a-priori analysis that leads to substantial savings in the cost of the algorithms.

Our work is relevant not only to quantum algorithms that use variational approaches but also to those that employ RKMs for solving DEs, such as in Ref. [112, 136]. Our analysis highlights the importance of studying the sensitivity of classical numerical methods for ODEs to perturbations in the input function, which has hardly been explored so far.

From our results, we conclude that depending on the parameters distinct to the problem at hand, higher-order RKMs are decreasing the resource requirements. In the use case of option pricing via the Black-Scholes equation, we showed that an RKM of order $p = 2$ is requiring the minimal number of total quantum circuit evaluations. For other applications, even higher-order methods might be favorable. With our thorough analysis of the involved parameters, we provide a straightforward framework that can be applied to other use cases that can be tackled by solving a DE in the form of Eq. (3.11) and decrease the resource requirements of the variational algorithms by suggesting the most efficient RKM.

In the interest of making this chapter self-contained, we build up the chapter in the following way: We begin with an introduction to the RKM and the variational quantum algorithm from [116] and with the problem statement. In Sec. 3.3-3.4, we show estimates of the errors and minimal resources required for ODE solving without and with shot noise. In Sec. 3.5, we analyze parameters of the variational algorithm that the error and resource estimates depend on. Afterwards, we are performing numerical analyzes of a simple ODE without shot noise and of an option pricing use case in Sec. 3.6. In Sec. 3.7, we provide a discussion of the results and conclusions.

3.2 Preliminaries

In this chapter, we investigate the impact of different types of errors on variational quantum algorithms for solving DEs based on Runge-Kutta methods. These algorithms are motivated by the variational quantum algorithm for imaginary time evolution introduced in Ref. [116]. In the following, we firstly present the Runge-Kutta methods, which is a family of classical methods for solving ODEs. Secondly, we present the variational quantum algorithm for solving DEs that are based on the Runge-Kutta methods. And thirdly, we introduce the errors and resources that we analyze in this chapter.

3.2.1 Runge-Kutta methods

Let us consider the initial value problem, which is an ODE together with an initial condition:

$$\begin{aligned} \frac{dy(\tau)}{d\tau} &= f\left(\tau, y(\tau)\right) \\ y(\tau_0) &= y_0 \ , \end{aligned} \tag{3.1}$$

where τ is the time and $y(\tau)$ is an element of the image of an unknown function in a scalar or vector form that we want to determine, and where $f\left(\tau, y(\tau)\right)$ fulfills the assumptions of the Picard-Lindelöf theorem, guaranteeing the existence of a unique solution to Eq. (3.1).

Since most of the time an analytically closed form is not viable, numerical methods are the only way to obtain an approximate solution. A common way of solving Eq. (3.1) is with the so-called Runge-Kutta methods (RKMs). The RKMs are a class of methods based on the Taylor expansion of y in order to approximate the numerical solution of the ODE at the future time step by using evaluations of $f\left(\tau, y(\tau)\right)$.

A general blueprint of RKMs with s stages can be outlined as follows: For simplicity, let us set $\tau_0 = 0$. We divide the time interval $\tau \in [0, T]$, $T > 0$ into N_τ time steps denoted by τ_n, $n \in \{1, ..., N_\tau\}$. We assume the step size $\Delta\tau = \tau_n - \tau_{n-1} = T/N_\tau$ to be constant and denote the computed solution at the n-th time step by $y(\tau_n)$. Using Eq. (3.1) and the solution $y(\tau_n)$ at the n-th time step, we can compute $y(\tau_{n+1})$ in the following way (see Ref. [137, p.907]):

$$y(\tau_{n+1}) = y(\tau_n) + \Delta\tau \sum_{i=1}^{s} b_i k_i \ , \tag{3.2}$$

where the calculation of the latter function is done in s stages

$$\begin{aligned} k_1 &= f(\tau_n, y(\tau_n)), \\ k_2 &= f(\tau_n + c_2\Delta\tau, y(\tau_n) + a_{21}k_1\Delta\tau), \\ k_3 &= f(\tau_n + c_3\Delta\tau, y(\tau_n) + (a_{31}k_1 + a_{32}k_2)\Delta\tau), \\ &\ \ \vdots \\ k_s &= f(\tau_n + c_s\Delta\tau, y(\tau_n) + (a_{s1}k_1 + a_{s2}k_2 + \cdots + a_{s,s-1}k_{s-1})\Delta\tau). \end{aligned} \tag{3.3}$$

The constants $a_{i,j}$ $(1 \leq j < i \leq s)$, b_i $(1 \leq i \leq s\ )$ and c_i $(2 \leq i \leq s\ )$ are specific for each RKM. In order to be consistent, the constants have to satisfy

$$\sum_{i=1}^{s} b_i = 1, \quad \text{and} \quad \sum_{j=1}^{i-1} a_{i,j} = c_i, \quad \text{for} \quad 2 \leq i \leq s. \tag{3.4}$$

For later analysis, we define the maxima of these parameters for a specific RKM in the following way:

$$b_{max} = \max_i |b_i|, \tag{3.5}$$

$$a_{max} = \max_{i,j} |a_{i,j}| \ . \tag{3.6}$$

The simplest RKM is the Euler method $(s = 1)$, which approximates the function in one stage iteratively as follows:

$$y(\tau_i + \Delta\tau) = y(\tau_i) + \Delta\tau f\left(\tau_i, y(\tau_i)\right). \tag{3.7}$$

The estimation error ℓ_i induced at each time step τ_i due to the truncation of the Taylor series is referred to as the local truncation error (LTE) of the method. RKMs are classified according to the error scaling of their LTE. A RKM is said to have an order p if the LTE is bounded by an error that scales as $\mathcal{O}(\Delta\tau^{p+1})$. Generally, the order and the number of stages of an RKM are related by $s = p$ for $1 \leq p \leq 4$, and $s > p$ for $p \geq 5$. This discrepancy is due to the fact that finding the coefficients $a_{i,j}, b_i$ and c_i becomes increasingly difficult for higher values of p as it involves solving a system of non-linear equations that becomes more complicated for higher p. For higher-order methods this can only be achieved with an increasingly higher number of stages s. To the best of our knowledge, there is no closed formula for calculating the minimum number of stages required for a specific order. The relations up to order $p = 10$ are provided in Table 3.1.

Order p	**Number of stages s**
5	6
6	7
7	9
8	11
9	13
10	16

Table 3.1: Relation between order and the minimum number of stages of RKMs [125, 138, 139]

The following theorem provides an upper bound on the LTE of a p-th order RKM:

Theorem 3.1. *(See Ref. [140, p.180]) The LTE ℓ_n of the p-th order RKM at the step $n \in [1, N_\tau]$ is bounded by*

$$\|\ell_n\| = \|y(\tau_n) - y_n\| \leq \Delta\tau^{p+1} K L_{f\tau}^p M + \mathcal{O}(\Delta\tau^{p+2}) \ , \tag{3.8}$$

where $\|\cdot\|$ denotes a norm, which can be any norm on the state space (e.g., Euclidean, maximum, or 1-norm), as the bound is independent of the specific choice. Here, y_n is the RKM approximation of $y(\tau_n)$ calculated at step n, assumed that the value $y(\tau_{n-1})$ is exact. Further, $K > 0$ is a constant depending on the chosen RKM and $f(\tau_n, y(\tau_n))$ can be upper bounded by $M > 0$ and is Lipschitz-continuous with respect to the first variable, such that the following *bounds hold:*

$$\|f(\tau_n, y(\tau_n))\| \leq M, \quad \left\|\frac{\partial f(\tau_n, y(\tau_n))}{\partial \tau}\right\| \leq L_{f\tau}, \ldots, \tag{3.9}$$

$$\left\|\frac{\partial^i f(\tau_n, y(\tau_n))}{\partial \tau^i}\right\| \leq \frac{L^i_{f\tau}}{M^{i-1}} \ , \tag{3.10}$$

for all $1 \leq i \leq p$*,* $0 < \tau_n < T$ *and* $f(\tau_n, y(\tau_n)) := \frac{\partial y(\tau)}{\partial \tau}\Big|_{\tau_n}$.

Unless stated otherwise, the same norm is used consistently throughout the analysis for states, errors, Lipschitz bounds, and the corresponding induced operator norms. Since all norms on finite-dimensional spaces are equivalent, the order of the error bounds remains unaffected by the specific choice of norm, with only the constants differing. In our setting, we use the trace norm for quantum states and the Euclidean (2-)norm, together with its induced operator norm, for all other quantities.

We are estimating the constants K, M and $L_{f\tau}$ in Secs. 3.6.1 and 3.6.2 for the specific examples of DEs that we cover numerically in Sec. 3.6.

3.2.2 Variational quantum algorithm for solving differential equations

In this section, we present the variational algorithm for solving linear differential equations of the following type:

$$\frac{dy(\tau)}{d\tau} = -\mathcal{H} \cdot y(\tau) \tag{3.11}$$

$$y(\tau_0) = y_0 \ , \tag{3.12}$$

where $y(\tau) \in \mathbb{C}^d$ is a vector and $\mathcal{H}$ a $d \times d$-dimensional matrix. We assume w.l.o.g. that d is a power of 2 and that $\mathcal{H}$ is Hermitian. This differential equation is a special case of the initial value problem as stated in Eq. (3.1).

It is possible to map a wide variety of DEs to Eq. (3.11), such as stochastic DEs (see Ref. [128]), the Black-Scholes partial DE (see our analysis in Sec. 3.6.2 and Ref. [26]), or other linear partial DEs (see Ref. [141]). Typically, this mapping involves a discretization of the underlying space and differential operators onto a grid [26, 128, 141], but it is also possible to encode the solution via spectral methods like Chebyshev polynomials or in Fourier basis (see for example Ref. [44], although they use a different quantum algorithm).

If d is not a power of 2, it is always possible to embed $\mathcal{H}$ and $y(\tau)$ into higher-dimensional spaces, such that the assumption holds. The resulting higher dimensional space into which $y(\tau)$ is embedded is less than a factor of 2 larger than the original space.

If $\mathcal{H}$ is not Hermitian, several strategies can be applied: For some differential equations, it is possible to apply changes of variables in order to transform $\mathcal{H}$ to a Hermitian matrix, such as the transformation done in Ref. [26]. Alternatively, it is possible to divide any matrix $\mathcal{H}$ into a Hermitian and an anti-Hermitian part, which effectively leads to a doubling in the number of circuit evaluations, as demonstrated in Ref. [142]. It is also possible to use the technique shown in Ref. [143] that gives a mapping from Eq. (3.11) with a non-Hermitian $\mathcal{H}$ to the real time Schrödinger equation, which one can solve with the variational algorithm introduced in Ref. [117]. Further, there exists a generalization of the variational algorithm for imaginary time evolution that we present here which can be applied to linear differential equations with non-Hermitian matrices $\mathcal{H}$ without the need to embed them first into higher-dimensional Hilbert spaces [144]. Note that our analysis can easily be adapted to this generalized variational algorithm, as well as to the variational algorithm solving the real time Schrödinger equation.

The variational quantum algorithm that solves Eq. (3.11), is based on the variational quantum algorithm for imaginary time evolution introduced in Ref. [116]. Therefore, we firstly bring Eq. (3.11) into the form of quantum imaginary time evolution.

For the rest of the chapter, we assume that $\mathcal{H}$ is Hermitian. The matrix $\mathcal{H}$ can thus be decomposed in the following way:

$$\mathcal{H} = \sum_{m=1}^{N} \lambda_m \sigma_m \ . \tag{3.13}$$

Here, $\{\lambda_m\}_{m=1}^{N} \in \mathbb{R}$ are the decomposition coefficients and $\{\sigma_m\}_{m=1}^{N}$ are the Pauli strings, which are tensor products of single qubit Pauli matrices and the identity. This possible decomposition always exists and is unique, since the collection of d^2 Pauli strings form an orthogonal basis for Hermitian operators acting on a d dimensional Hilbert space. The operator $\mathcal{H}$ plays the role of the Hamiltonian of a quantum system. Effectively replacing the real time parameter t from the Schrödinger equation with $-i\tau$, where τ is a real number, Eq. (3.11) can then be considered as a Schrödinger equation in imaginary time.

Let us further realize the function $y(\tau)$ as a quantum state $|\psi(\tau)\rangle \in \mathbb{C}^d$ and the initial condition $y(\tau_0) = y_0$ as a state $|\psi(0)\rangle \in \mathbb{C}^d$. That can be done by normalizing the entries of the vector $y(\tau)$ and by subsequently encoding the resulting entries as the amplitudes of $|\psi(\tau)\rangle$:

$$|\psi(\tau)\rangle = \sum_{i=0}^{d-1} \frac{y(\tau)_i}{\sqrt{\|y(\tau)\|_2^2}} |i\rangle \ , \tag{3.14}$$

where $\{|i\rangle\}_{i=0}^{d-1}$ is the computational basis.

The problem of solving Eq. (3.11) then comes down to simulating the corresponding non-unitary time evolution $V(\tau) = e^{-\mathcal{H}\tau}$ that solves the following equation:

$$\frac{d\,|\psi(\tau)\rangle}{d\tau} = -\mathcal{H} \cdot |\psi(\tau)\rangle \tag{3.15}$$

with the initial state $|\psi(0)\rangle$ at time $\tau = 0$. The goal is then to calculate the evolved state $|\psi(T)\rangle$ at time T, which can be calculated from the initial state as:

$$|\psi(T)\rangle = \gamma(T)V(T)\,|\psi(0)\rangle\,, \tag{3.16}$$

$$\text{with the normalization} \qquad \gamma(\tau) = \left(\langle\psi(0)|\,V(2T)\,|\psi(0)\rangle\right)^{-1/2}\;. \tag{3.17}$$

While reading out the amplitudes of $|\psi(T)\rangle$ itself will only give the ratio of each basis vector in the solution $y(T)$ of the DE in Eq. (3.11), keeping track of the initial normalization from the mapping of $y(\tau_0)$ to $|\psi(0)\rangle$ and the intermediate normalization factors $\gamma(\tau)$ will give the resulting renormalization factor. One can identify $\gamma(\tau)$ as one of the parameters that is updated at each step (see for example Refs. [113, 128]).

As proposed in Ref. [116], the evolution of the state $|\psi(\tau)\rangle$ can be simulated by using a parameterized quantum circuit to approximate the evolved state. Instead of $|\psi(\tau)\rangle$, the variational quantum "trial" state

$$|\phi(\boldsymbol{\theta}(\tau))\rangle := R(\boldsymbol{\theta}(\tau))\,|\bar{0}\rangle\,, \quad |\phi(\boldsymbol{\theta}(\tau))\rangle \in \mathbb{C}^d, \tag{3.18}$$

prepared by a variational circuit $R(\boldsymbol{\theta}(\tau))$ with a vector of time-dependent parameters $\boldsymbol{\theta}(\tau) = (\theta_1(\tau), \theta_2(\tau), \ldots, \theta_{N_V}(\tau))^T \in \mathbb{R}^{N_V}$ is taken. The trial state is often referred to as the *Ansatz*. The circuit is chosen in such a way that it consists of a sequence of N_V layers each depending on one variational parameter as follows:

$$R(\boldsymbol{\theta}(\tau)) = R_{N_V}(\theta_{N_V}(\tau))R_{N_V-1}(\theta_{N_V-1}(\tau))\ldots R_1(\theta_1(\tau))\;. \tag{3.19}$$

Each $R_k(\theta_k(\tau))$, $k \in (1, \ldots, N_V)$ is a unitary operator that can be written as

$$R_k(\theta_k(\tau)) = \exp\left\{\left(\theta_k(\tau)\sum_{i=1}^{N_d} f_{k,i}\sigma_{k,i}\right)\right\}\,, \tag{3.20}$$

with fixed complex parameters $f_{k,i}$ and Pauli strings $\sigma_{k,i}$. For simplicity, we keep N_d fixed. In general, only a small subspace of the Hilbert space can be reached with such an Ansatz, but as was shown in Ref. [145], this is sufficient for physically relevant states. Furthermore, this Ansatz captures a wide class of possible implementations, such as the coupled cluster Ansatz [146] or hardware efficient methods [147, 148].

The idea of the method is to map the time evolution of the state $|\psi(\tau)\rangle$ according to the Hamiltonian $\mathcal{H}$ to a time evolution of the parameters $\boldsymbol{\theta}(\tau)$ of the state $|\phi(\boldsymbol{\theta}(\tau))\rangle$. To this end, one first finds the vector of parameters $\boldsymbol{\theta}(\tau = 0)$ such that it minimizes the distance $\| \, |\phi(\boldsymbol{\theta}(0))\rangle - |\psi(0)\rangle \, \|$. The McLachlan's variational principle [149, 150] as given by

$$\delta \| (d/d\tau + \mathcal{H}) \, |\phi(\boldsymbol{\theta}(\tau))\rangle \, \| = 0 \tag{3.21}$$

is fulfilled if Eq. (3.15) is valid for the trial state $|\phi(\boldsymbol{\theta}(\tau))\rangle$. Here $\| \, |\phi\rangle \, \| = \sqrt{\langle \phi | \phi \rangle}$ is the Hilbert space norm of a vector $|\phi\rangle$ and δ denotes an infinitesimal variation. However, if the chosen Ansatz is not expressive enough or too biased, Eq. (3.21) will not hold. The resulting errors are hard to control in practice, but there exist ways to estimate them. See for example Ref. [131, 134].

Eq. (3.21) is used to translate the time evolution of the trial state $|\phi(\boldsymbol{\theta}(\tau))\rangle$ to a time evolution of the vector of parameters $\boldsymbol{\theta}(\tau)$, given as an ODE (see Appendix 3.D):

$$\sum_{l=1}^{N_V} A_{kl} \left(\boldsymbol{\theta}(\tau)\right) \frac{\partial \theta_l(\tau)}{\partial \tau} = C_k \left(\boldsymbol{\theta}(\tau)\right) \quad \forall \tau \; . \tag{3.22}$$

Note that this ODE is acting on the N_V-dimensional vector $\boldsymbol{\theta}(\tau)$, where N_V is independent of the dimension d of the time evolution in Eq. (3.15), and is entirely determined by the number of parameters of the chosen Ansatz. In particular, N_V is not bound to be a power of 2. In principle, lower N_V will reduce the total number of circuit evaluations for our algorithm (see Sec. 3.5.1), but will also lead to a lower expressivity of the corresponding Ansatz.

The elements of the matrix A and the vector C are:

$$A_{kl} \left(\boldsymbol{\theta}(\tau)\right) = \mathfrak{Re} \left(\frac{\partial \langle \phi(\boldsymbol{\theta}(\tau))|}{\partial \theta_k} \frac{\partial |\phi(\boldsymbol{\theta}(\tau))\rangle}{\partial \theta_l} \right) , \tag{3.23}$$

$$C_k \left(\boldsymbol{\theta}(\tau)\right) = \mathfrak{Re} \left(- \frac{\partial \langle \phi(\boldsymbol{\theta}(\tau))|}{\partial \theta_k} \mathcal{H} \, |\phi(\boldsymbol{\theta}(\tau))\rangle \right) . \tag{3.24}$$

Taking the derivatives of Eq. (3.20) with respect to the parameters, one can calculate the derivative of the trial state in Eq. (3.18):

$$\frac{\partial |\phi(\boldsymbol{\theta}(\tau))\rangle}{\partial \theta_k} = \sum_{i=1}^{N_d} f_{k,i} R_{k,i} \, |\bar{0}\rangle \, , \tag{3.25}$$

$$R_{k,i} = R_{N_V} R_{N_V - 1} ... R_{k+1} R_k \sigma_{k,i} R_{k-1} \dots R_2 R_1 , \tag{3.26}$$

where we omitted the dependencies of the R_ks on θ_ks for simplicity. With the chosen *Ansatz* in Eq. (3.26), the matrix elements in Eq. (3.23) and in Eq. (3.24) can be computed as:

$$A_{k,l}\left(\boldsymbol{\theta}(\tau)\right) = \sum_{i,j=1}^{N_d} \left(f_{k,i}^* f_{l,j} \langle \bar{0}| R_{k,i}^\dagger R_{l,j} |\bar{0}\rangle + h.c. \right), \tag{3.27}$$

$$C_k\left(\boldsymbol{\theta}(\tau)\right) = \sum_{i=1}^{N_d} \sum_{m=1}^{N} \left(f_{k,i}^* \lambda_m \langle \bar{0}| R_{k,i}^\dagger \sigma_m R |\bar{0}\rangle + h.c. \right), \tag{3.28}$$

by measuring circuits illustrated in Fig. 3.1. Alternatively, it is possible to calculate the matrix element with parameter-shift rules [92, 151], making the circuits shorter in depth and therefore potentially more suitable for NISQ applications. They are applied to similar algorithms in Ref. [135, 152].

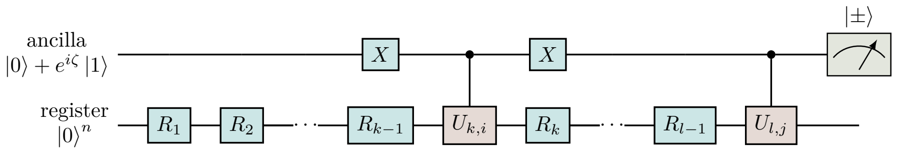


Figure 3.1: The quantum circuit evaluating the elements of A and C. The controlled unitary $U_{k,i}$ is one of the $\sigma_{k,i}$. Depending on if one is evaluating A or C, the controlled unitary $U_{l,j}$ is another $\sigma_{l,j}$ or one of the Pauli strings σ_m (in which case we take $l = N_V + 1$) that constitute the Hamiltonian, respectively.

If the matrix $A\left(\boldsymbol{\theta}(\tau)\right)$ is invertible for all $\boldsymbol{\theta}(\tau)$ in the relevant domain, then Eq. (3.22) can be written in the form of Eq. (3.1) by identifying $y(\tau)$ with $\boldsymbol{\theta}(\tau)$:

$$\frac{\partial \boldsymbol{\theta}(\tau)}{\partial \tau} = f\left(\boldsymbol{\theta}(\tau)\right) := A^{-1}\left(\boldsymbol{\theta}(\tau)\right) C\left(\boldsymbol{\theta}(\tau)\right), \tag{3.29}$$

and with the initial condition $\boldsymbol{\theta}(0)$ at time $\tau = 0$. The time evolution of the parameters $\boldsymbol{\theta}(\tau)$ is given as this ODE, and solving it until final time T gives the vector of parameters $\boldsymbol{\theta}(T)$ that yield the final state $|\phi(\boldsymbol{\theta}(T))\rangle$, which serves as an approximation of the evolved state $|\psi(T)\rangle$. If A is not invertible, regularization introduces an additional error (see Secs. 3.2.3 and 3.5.2). Similarly, the inversion itself will inevitably also include an additional error that has to be taken into account. As long as the errors stemming from regularization and inversion of A fulfill the error bounds in this chapter (such as Eq. (3.75)), our estimates are still applicable. We assume therefore for the remainder of this chapter that A is invertible.

We can thus solve a DE of the form in Eq. (3.11) with a hybrid quantum-classical algorithm by solving the ODE in Eq. (3.22) defining the parameters

$\boldsymbol{\theta}(\tau)$ of an Ansatz state.

In the next section, we introduce the errors and resources required for solving the DEs in Eqs. (3.1) and (3.11) with the Runge-Kutta methods and with the variational quantum algorithm, respectively.

3.2.3 Errors and resources

In this chapter, we analyze errors and resource requirements of the Runge-Kutta methods and the variational algorithm as described above. For this, we focus on analyzing the errors involved in solving the ODE using the RKM described in Sec. 3.2.1, and for the variational algorithm additionally the total error arising from preparing the trial state $|\phi(\boldsymbol{\theta}(T))\rangle$ defined in Eq. (3.18) using the method described in Sec. 3.2.2. In this chapter, we are not considering representation errors stemming from the chosen Ansatz state not being expressive enough, as it proves very challenging to estimate these errors in general [153]. However, they have an influence on both errors that we estimate, by playing a role in the approximation of the time evolution of $\boldsymbol{\theta}(\tau)$ and in the approximation of the final state $|\psi(T)\rangle$. They are specific to the task at hand, it is possible to derive a posteriori error bounds and they become less relevant for deep Ansätze, see for example Refs. [131, 134]. Also, we assume circuit error such as gate infidelity, bias and SPAM errors to be negligible. We further assume the matrices A as defined in Eq. (3.27) to be invertible (see our discussion in Sec. 3.5). In practice, it might be necessary to introduce matrix regularizations for the cases where A is not invertible (see techniques in Refs. [26, 116, 128, 152]), which would lead to additional errors.

We denote the total error arising while approximating the solution of the ODE (Eq. (3.1)) for noiseless evaluations of the functions $f(\tau, y(\tau))$:

$$\epsilon_{ODE}^{(0)} := \|y(\tau_{N_\tau}) - y_{N_\tau}\|_2 \ , \tag{3.30}$$

and analyze it in Sec. 3.3 (Thm. 3.2). For the variational quantum algorithm, the parameters calculated via an RKM from Eq. (3.29) additionally are influenced by shot noise in the evaluations of $f(\boldsymbol{\theta}(\tau))$, which we denote by the superscript δ:

$$\epsilon_{ODE}^{(\delta)} := \|\boldsymbol{\theta}(\tau_{N_\tau}) - \hat{\boldsymbol{\theta}}_{N_\tau}\|_2 \ . \tag{3.31}$$

We added the hat to the approximation of $\boldsymbol{\theta}_{N_\tau}$ of the parameters calculated via an RKM, in order to show that they may be perturbed (e.g., from shot noise). We analyze $\epsilon_{ODE}^{(\delta)}$ in Sec. 3.4.

Whenever the norm is not specified, we are using the 2-norm:

$$\|\boldsymbol{\theta}(\tau_{N_\tau}) - \hat{\boldsymbol{\theta}}_{N_\tau}\|_2 := \left(\sum_{i=1}^{N_V} |\boldsymbol{\theta}(\tau_{N_\tau})_i - \hat{\boldsymbol{\theta}}_{N_\tau,i}|^2 \right)^{1/2} , \tag{3.32}$$

where N_V is the length of the vectors $\boldsymbol{\theta}(\tau_{N_\tau})$ and $\hat{\boldsymbol{\theta}}_{N_\tau}$.

We define the total error arising from applying the variational algorithm to approximate the state $|\psi(T)\rangle$ with the trial state $|\phi(\hat{\boldsymbol{\theta}}_{N_\tau})\rangle$ as:

$$\epsilon_{total} := \|\psi(T) - \phi(\hat{\boldsymbol{\theta}}_{N_\tau})\|_1 , \tag{3.33}$$

where $\psi(T) = |\psi(T)\rangle \langle\psi(T)|$ and $\phi(\hat{\boldsymbol{\theta}}_{N_\tau}) = |\phi(\hat{\boldsymbol{\theta}}_{N_\tau})\rangle \langle\phi(\hat{\boldsymbol{\theta}}_{N_\tau})|$, where $|\psi(T)\rangle$ and $|\phi(\hat{\boldsymbol{\theta}}_{N_\tau})\rangle$ are as defined in Eqs. (3.14) and (3.18). For ϵ_{total}, we use the trace distance as the most convenient distance for quantum states, which for two pure states is defined as:

$$\|\psi(T) - \phi(\hat{\boldsymbol{\theta}}_{N_\tau})\|_1 := \sqrt{1 - |\langle\psi(T)|\phi(\hat{\boldsymbol{\theta}}_{N_\tau})\rangle|^2} . \tag{3.34}$$

We analyze the error ϵ_{total} in Sec. 3.5.1.

Based on the error estimates, we are estimating the resources needed in order for executing the RKMs and the variational algorithm. We define the cost of solving an ODE with an RKM as the total number of times that the function $f(\boldsymbol{\theta}(\tau))$ has to be evaluated for the whole RKM:

$$C(N_\tau, N_r, s, p) := sN_\tau(s,p)N_r(s,p) , \tag{3.35}$$

where N_r is the number of measurements of the function $f(\boldsymbol{\theta}(\tau))$ at a single stage of one RKM time step. In the absence of shot noise, the cost in Eq. (3.35) reduces to

$$C(N_\tau, s, p) := sN_\tau(s,p) , \tag{3.36}$$

as at each stage and time step, one needs only one evaluation of $f(\tau, y(\tau))$.

We are estimating the minima of the costs in Eq. (3.35) and in Eq. (3.36) required to reach a specified target error $\epsilon^{(\delta)}_{target}$ that upper bounds the error $\epsilon^{(\delta)}_{ODE} \leq \epsilon^{(\delta)}_{target}$ in Sec. 3.3 and 3.4.

For the variational algorithm described in Sec. 3.2.2, we also determine the total number of quantum circuit evaluations required. As we will demonstrate, calculating $f(\boldsymbol{\theta}(\tau))$ for a given input $\boldsymbol{\theta}(\tau)$ requires the preparation and measurement of several quantum circuits. This significantly increases the total number of quantum circuit evaluations beyond the cost in Eq. (3.35). We refer to this as N_{circ}, and we estimate it in Sec. 3.5.1.

3.3 Runge-Kutta methods without shot noise

For ease of exposition, in this section we analyze the total error arising from solving an ODE with an RKM without the presence of shot noise in the evaluations of $f(\tau, y(\tau))$ in Eq. (3.1). We also determine the minimal number $N_\tau^{(0)}$ of RKM steps required to achieve a prescribed accuracy.

In Thm. 3.2, we establish an upper bound on the total error, denoted $\epsilon_{ODE}^{(0)}$ in Eq. (3.30), by analyzing the error propagation due to the truncation error of the RKM (see Thm. 3.1).

We select the upper bound of the error $\epsilon_{ODE}^{(0)}$ as a target error, meaning the maximal error we can expect. This target error is the used to determine the minimal number of RKM steps needed to ensure that the resulting error remains within the target. The result is formalized in Thm. 3.3, and the corresponding minimal cost, as defined in Eq. (3.36), is derived in Corollary 1.

Theorem 3.2. *Let $y(\tau_{N_\tau})$ be the solution of Eq. (3.1). We assume that the assumptions of Thm. 3.1 hold. Let us further assume that there exists a Lipschitz constant L_{fy}, such that $\forall y_1(\tau_n), y_2(\tau_n)$ in the spaces $\{y(\tau_n) : \tau_n \in [0, T]\}$ and $\{y_n : n \in [1, N_\tau]\}$, the following holds:*

$$\|f(\tau_n, y_1(\tau_n)) - f(\tau_n, y_2(\tau_n))\| \leq L_{fy} \|y_1(\tau_n) - y_2(\tau_n)\| \ , \forall \tau_n \in [0, T]. \tag{3.37}$$

Then, the approximation y_{N_τ} of the solution $y(\tau_{N_\tau})$ at time τ_{N_τ} calculated via a p-th order RKM with s stages in N_τ time steps has the error defined in Eq. (3.30) bounded by:

$$\epsilon_{ODE}^{(0)} \leq \frac{(1 + F(N_\tau, s))^{N_\tau} - 1}{F(N_\tau, s)} \left(\frac{T}{N_\tau}\right)^{p+1} K L_{f\tau}^p M \ , \tag{3.38}$$

where we used the notation

$$F(N_\tau, s) := \frac{b_{max}}{a_{max}} \left(\left(1 + L_{fy} a_{max} \frac{T}{N_\tau}\right)^s - 1 \right). \tag{3.39}$$

For a definition of a_{max} and b_{max} see Eqs. (3.5) and (3.6). We define the upper bound in Eq. (3.38) as the target error, i.e., the theoretical maximum error that can occur when applying this method:

$$\epsilon_{target}^{(0)} := \frac{(1 + F(N_\tau, s))^{N_\tau} - 1}{F(N_\tau, s)} \left(\frac{T}{N_\tau}\right)^{p+1} K L_{f\tau}^p M \ . \tag{3.40}$$

Based on this result, we estimate the minimal number of time steps that are needed to guarantee a particular target error $\epsilon_{target}^{(0)}$ while solving Eq. (3.1):

Theorem 3.3. *Let the assumptions of Thms. 3.1-3.2 hold. Then, the minimal number of RKM steps $N_\tau^{(0)}$ required to solve Eq. (3.1) with a target error $\epsilon_{target}^{(0)}$ is*

$$N_\tau^{(0)} = L_{f\tau}T\left(\frac{KM\left(e^{b_{max}TL_{fy}s}-1\right)}{\epsilon_{target}^{(0)}b_{max}sL_{fy}}\right)^{1/p}. \tag{3.41}$$

Using the latter results, we get:

Corollary 1. *For a target error ϵ_{target}, the minimal value of the cost function (3.36) is*

$$C(N_\tau^{(0)}, s, p) = sL_{f\tau}T\left(\frac{KM\left(e^{b_{max}TL_{fy}s}-1\right)}{\epsilon_{target}^{(0)}b_{max}sL_{fy}}\right)^{1/p}, \tag{3.42}$$

where s is the number of the RKM stages, p is the RKMs order and $N_\tau^{(0)}$ is the minimal number of time steps of the chosen RKM.

The proofs of Thms. 3.2 and 3.3 are provided in Appendices 3.A and 3.B, respectively. In the following section, we analyze the RKM in the presence of the shot noise in the evaluations of the differential $f(\boldsymbol{\theta}(\tau))$.

3.4 Runge-Kutta methods under the presence of shot noise

In the variational algorithm that we presented in Sec. 3.2.2, we are solving the ODE given in Eq. (3.29) by using RKMs. The function $f(\boldsymbol{\theta}(\tau))$ from Eq. (3.29) is given by expectation values estimated via sampling quantum circuits. Therefore, we assume that instead of $f(\boldsymbol{\theta}(\tau))$, we have access to its approximation $\hat{f}(\boldsymbol{\theta}(\tau))$. We need to take into account the statistical errors arising while computing $\hat{f}(\boldsymbol{\theta}(\tau))$ based on the measurement results. The analysis of this section is valid for all differential equations as given in Eq. (3.1) that have a noisy $\hat{f}(\boldsymbol{\theta}(\tau))$.

By virtue of the central limit theorem, let us also assume that each measurement is drawn from a random Gaussian distribution with mean $f(\boldsymbol{\theta}(\tau))$ and standard deviation σ_{single}. Calculating the average of these measurements gives the estimate $\hat{f}(\boldsymbol{\theta}(\tau))$:

$$\hat{f}(\boldsymbol{\theta}(\tau)) = \frac{1}{N_r}\sum_{i=1}^{N_r}\hat{f}_i(\boldsymbol{\theta}(\tau)) , \tag{3.43}$$

where $\hat{f}_i(\boldsymbol{\theta}(\tau))$ is a single measurement result. By the central limit theorem as $N_r \to \infty$, the estimate $\hat{f}(\boldsymbol{\theta}(\tau))$ behaves as a Gaussian distribution:

$$\sqrt{N_r}(\hat{f}(\boldsymbol{\theta}(\tau)) - f(\boldsymbol{\theta}(\tau))) \to^d N(0, \sigma^2_{single}). \tag{3.44}$$

From Chebyshev's inequality we get the following bound for a $\delta > 0$:

$$P(\|\hat{f}(\boldsymbol{\theta}(\tau)) - f(\boldsymbol{\theta}(\tau))\| \geq \delta) \leq \frac{\sigma^2_{single}}{\delta^2 N_r}. \tag{3.45}$$

Alternatively, a Chernoff bound would be tighter, but more cumbersome to apply here and probably not lead to a different qualitative analysis.

If we take the probability $P(\|\hat{f}(\boldsymbol{\theta}(\tau)) - f(\boldsymbol{\theta}(\tau))\| \geq \delta)$ to be equal to $\eta \in (0,1)$, then with probability of $1-\eta$, the following bound holds:

$$\|\hat{f}(\boldsymbol{\theta}(\tau)) - f(\boldsymbol{\theta}(\tau))\| \leq \delta = \frac{\Sigma}{\sqrt{N_r}} \tag{3.46}$$

In the above, we defined $\Sigma = \frac{\sigma_{single}}{\sqrt{\eta}}$.

Assuming access to a noisy estimate of $f(\boldsymbol{\theta}(\tau))$, as described above, we derive error and resource estimates for solving ODEs using RKMs under the presence of shot noise.

In Thm. 3.4, we establish an upper bound on the error $\epsilon^{(\delta)}_{ODE}$ as defined in Eq. (3.31) by analyzing how both the truncation error of the RKM (see Thm. 3.1) and the shot noise in the estimation of $f(\boldsymbol{\theta}(\tau))$ (see Eq. (3.46)) contribute to error propagation.

We select the upper bound of the error $\epsilon^{(\delta)}_{ODE}$ as a target error, meaning the maximal error we can expect. This target error is the used to determine the minimal number of RKM steps and the minimal number of measurements of each $f(\boldsymbol{\theta}(\tau))$ required to keep the resulting error within the target. The result is formalized in Thm. 3.5, and the corresponding minimal cost, as defined in Eq. (3.35), is derived in Corollary 2.

Theorem 3.4. *Let us assume that the assumptions of Thm. 3.1 hold for Eq. (3.29). Under the conditions in Eq. (3.10), Eq. (3.37), the approximation $\hat{\boldsymbol{\theta}}_{N_\tau}$ at time τ_{N_τ} calculated via a p-th order RKM with s stages in N_τ time steps has the error defined in Eq. (3.31) upper bounded by:*

$$\epsilon^{(\delta)}_{ODE} \leq \frac{(1+F(N_\tau,s))^{N_\tau}-1}{F(N_\tau,s)}\left(\frac{3\delta}{L_{fy}}F(N_\tau,s) + \left(\frac{T}{N_\tau}\right)^{p+1} KL^p_{f\tau}M\right), \tag{3.47}$$

where we used the notation introduced in Eq. (3.39).

Let us further denote the latter upper bound as

$$\epsilon_{target}^{(\delta)} := \frac{(1+F(N_\tau,s))^{N_\tau}-1}{F(N_\tau,s)}\left(\frac{3\delta}{L_{fy}}F(N_\tau,s)+\left(\frac{T}{N_\tau}\right)^{p+1}KL_{f\tau}^{p}M\right). \tag{3.48}$$

Given this error estimate, we obtain the following resource minimization and the directly following minimal cost:

Theorem 3.5. *Let us assume that $\hat{f}(\boldsymbol{\theta}(\tau))$ is an approximation for $f(\boldsymbol{\theta}(\tau))$ calculated from N_r measurements, with the error scaling $\delta = \frac{\Sigma}{\sqrt{N_r}}$, where $\Sigma > 0$ is a constant. Then, calculating the approximated solution $\hat{\boldsymbol{\theta}}_{N_\tau}$ to the ODE in Eq. (3.29) by a p-th order RKM with s stages and with a target error $\epsilon_{target}^{(\delta)}$ defined in Eq. (3.48), the number of time steps required must be at least*

$$N_\tau^{(\delta)} = TL_{f\tau}\left(\frac{KM\left(e^{b_{max}sL_{fy}T}-1\right)(2p+1)}{\epsilon_{target}^{(\delta)}b_{max}sL_{fy}}\right)^{1/p}, \tag{3.49}$$

and at least

$$N_r^{(\delta)} = \frac{9\Sigma^2}{L_{fy}^2}\left(\frac{\epsilon_{target}^{(\delta)}}{\left(1+F(N_\tau^{(\delta)},s)\right)^{N_\tau^{(\delta)}}-1}-\left(\frac{T}{N_\tau^{(\delta)}}\right)^{p+1}\frac{KL_{ft}^{p}M}{F(N_\tau^{(\delta)},s)}\right)^{-2} \tag{3.50}$$

many measurements for the estimation of $f(\boldsymbol{\theta}(\tau))$ for each set of inputs $\boldsymbol{\theta}(\tau)$.

The proofs of Thms. 3.4 and 3.5 are given in Appendices 3.A and 3.C, respectively.

Corollary 2. *For a target error $\epsilon_{target}^{(\delta)}$, the minimal value of the cost function (3.35) is*

$$C(N_\tau^{(\delta)}, N_r^{(\delta)}, s, p) = sN_\tau^{(\delta)}(s,p)N_r^{(\delta)}(s,p), \tag{3.51}$$

where s is the number of stages, p is the order, $N_\tau^{(\delta)}$ and $N_r^{(\delta)}$ are the minimal number of time steps of the RKM; and the minimal number samples in calculating $\hat{f}(\boldsymbol{\theta}(\tau))$ for each set of inputs according to Thm. 3.5.

In the following section, we further analyze the latter error and resource estimates. By applying it to the variational quantum algorithm discussed in

Sec. 3.2.2, we show by numerical estimations in Sec. 3.6, how they evaluate for specific DEs such as given in Eq. (3.29). We shall see that with these results, we can choose a Runge-Kutta method such that the required total number of circuit evaluations will be minimal.

3.5 Analysis of the variational quantum algorithm

In this section, we analyze the total error as defined in Eq. (3.33) of the algorithm provided in Sec. 3.2.2. We show its relation to Eq. (3.31) and derive the resource requirements of the algorithm for a given target error, based on the estimates in Sec. 3.4. Afterwards, we continue with an estimation of the shot noise and further quantities based on a toy model we introduce. These further estimates are necessary for the numerical analysis of the following Sec. 3.6.

3.5.1 Error and resource estimate

We are interested in estimating the total error in Eq. (3.33). Using the triangle inequality, we get

$$\epsilon_{total} \leq \|\psi(T) - \phi(\boldsymbol{\theta}(T))\|_1 + \|\phi(\boldsymbol{\theta}(T)) - \phi(\hat{\boldsymbol{\theta}}_{N_\tau})\|_1 = \epsilon_{PQC} + \epsilon_{par} \ . \tag{3.52}$$

Here, we define the distance between the trial functions with the different input parameters as follows:

$$\epsilon_{par} := \|\phi(\boldsymbol{\theta}(T)) - \phi(\hat{\boldsymbol{\theta}}_{N_\tau})\|_1. \tag{3.53}$$

The representation error coming from approximating the function $\psi(T)$ with the trial function $\phi(\boldsymbol{\theta}(T))$ is defined as

$$\epsilon_{PQC} := \|\psi(T) - \phi(\boldsymbol{\theta}(T))\|_1. \tag{3.54}$$

Furthermore, using the multi-dimensional mean-value theorem [1], the second term of Eq. (3.52) can be bounded by

$$\epsilon_{par} \leq \sup_{\boldsymbol{\theta_0} \in \Xi} \|\nabla_{\boldsymbol{\theta}} \phi(\boldsymbol{\theta_0}(T))\|_{(1,2)} \epsilon_{ODE}^{(\delta)}, \tag{3.56}$$

[1]Multivariate Mean Value Theorem: for $x, y \in R^n$

$$\|f(x) - f(y)\|_q \leq \sup_{z \in [x,y]} \|f'(z)\|_{(q,p)} \|x - y\|_p, \tag{3.55}$$

Where $z \in [x, y]$ denotes a vector z contained in the set of points between $x, y \in R^n$ and $\|f'(z)\|_{(q,p)}$ is the $L_{(p,q)}$ norm of the derivative matrix of f evaluated at z.

where $\Xi = \{w\hat{\boldsymbol{\theta}}_{N_\tau} + (1-w)\boldsymbol{\theta}(T) | w \in [0,1]\}$. The norm of the Jacobian is defined as

$$\sup_{\boldsymbol{\theta_0}\in\Xi} \|\nabla_{\boldsymbol{\theta}}\phi(\boldsymbol{\theta_0}(T))\|_{(1,2)} := \sup_{\boldsymbol{\theta}_0\in\Xi} \sup_{\boldsymbol{\theta}^*(T)\in\Xi, \boldsymbol{\theta}^*(T)\neq 0} \frac{\|\nabla_{\boldsymbol{\theta}}\phi(\boldsymbol{\theta_0}(T)) \cdot \boldsymbol{\theta}^*(T)\|_1}{\|\boldsymbol{\theta}^*(T)\|_2} . \tag{3.57}$$

The following lemma shows us an upper bound to this expression.

Lemma 2. *The norm of the Jacobian with the chosen circuit (Eq.* (3.19) *and Eq.* (3.20)*) is bounded by*

$$\sup_{\boldsymbol{\theta}_0\in\Xi} \sup_{\boldsymbol{\theta}^*(T)\in\Xi, \boldsymbol{\theta}^*(T)\neq 0} \frac{\|\nabla_{\boldsymbol{\theta}}\phi(\boldsymbol{\theta_0}(T)) \cdot \boldsymbol{\theta}^*(T)\|_1}{\|\boldsymbol{\theta}^*(T)\|_2} \tag{3.58}$$

$$\leq \sup_{\boldsymbol{\theta}^*(T)\in\Xi, \boldsymbol{\theta}^*(T)\neq 0} \frac{\sum_k \left(\sum_j 2|f_{k,j}|\right) |\theta_k^*(T)|}{\|\boldsymbol{\theta}^*(T)\|_2} . \tag{3.59}$$

We provide the proof in Appendix 3.F.

According to this lemma, Eq. (3.57) is upper bounded by

$$S := \left(\sup_{\boldsymbol{\theta}^*(T)\in\Xi, \boldsymbol{\theta}^*(T)\neq 0} \frac{\sum_{k=1}^{N_V} \left(\sum_{j=1}^{N_d} 2|f_{k,j}|\right) |\theta_k^*(T)|}{\|\boldsymbol{\theta}^*(T)\|_1} \right) . \tag{3.60}$$

Thus, the total error is bounded by

$$\epsilon_{total} \leq \epsilon_{PQC} + S\epsilon_{ODE}^{(\delta)}. \tag{3.61}$$

As described in Sec. 3.2.3, there are errors which we assume to be negligible. In particular, we are disregarding the representation error ϵ_{PQC}. We are thus left with estimating S for the specific Ansatz at hand (see Sec. 3.6) and the error in calculating the parameters, $\epsilon_{ODE}^{(\delta)}$, which we analyzed in Sec. 3.4.

The resource that we would like to minimize for the application of the variational algorithm is the total number of circuit evaluations N_{circ} which is needed for running the algorithm. In comparison to the cost that was estimated in Sec. 3.4, we have to add an additional factor in order to get N_{circ}. The derivation of this factor follows.

Denote with N_A and N_C the numbers of circuits that are needed to calculate the matrix A and the vector C for specific input parameters. Each matrix A or vector C consists of N_V^2 or N_V elements, respectively where N_V is the number of parameters $\boldsymbol{\theta}$. Each element of A and C is calculated by the evaluation of N_d^2 or N_dN different circuits, respectively. Here N is the number of terms in the Hamiltonian and N_d is an upper bound on the number of Pauli strings in the unitary operator R_k of the variational circuits, as defined in Eq. (3.20). Thus, they are bounded above by $N_A \leq \overline{N_A} = N_V^2 N_d^2$ and $N_C \leq \overline{N_C} = N_V N_d N$. Let us assume that these upper bounds are always reached. We denote by $N_\tau^{(\delta)}$ the minimum number of time steps in the ODE solving algorithm and by $N_r^{(\delta)}$ the minimum number of times, each stage value $f\left(\boldsymbol{\theta}(\tau)\right)$ is measured; both have been estimated in Thm. 3.5. By s, we denote the number of stages that are calculated at each time step of the RKM. Therefore, the number of circuit evaluation N_{circ} needed to calculate $\vec{\theta}(T)$ is equal to

$$\begin{aligned} N_{circ} &= N_\tau^{(\delta)} s N_r^{(\delta)} (\overline{N_A} + \overline{N_C}) && (3.62)\\ &= N_\tau^{(\delta)} s N_r^{(\delta)} (N_V^2 N_d^2 + N_V N_d N) && (3.63)\\ &= N_\tau^{(\delta)} s N_r^{(\delta)} N_V N_d (N_V N_d + N)\ . && (3.64)\end{aligned}$$

One can see that the number of circuit evaluations in the whole circuit scales quadratically in the number of circuit parameters N_V.

Note that, if instead of counting the total number of circuit evaluations we care about the total run time of the algorithm, and allow parallel evaluations of $N_V^2 N_d^2 + N_V N_d N$ circuits, the cost comes down to $N_\tau^{(\delta)} s N_r^{(\delta)}$.

In the next subsection, we will further estimate the dependence of $N_r^{(\delta)}$ on the condition number of A, and give a comprehensive overview of the resource estimate in Eqs. (3.77)- (3.80).

3.5.2 Estimation of the shot noise

In Sec. 3.4, we presented error and resource estimates for RKMs with an error in the evaluations of $f(\tau, \boldsymbol{\theta(\tau)})$ stemming from shot noise. We derive our estimates on the basis of the bound in Eq. (3.46). Since we would like to incorporate the results of Sec. 3.4 into analyzes of the required resources for the variational quantum algorithm, we are studying the shot noise inherent to this algorithm in this section. In the end of this section, we will give an estimate of Σ as defined in Eq. (3.46).

Each term $\left(f_{k,i}^* f_{l,j} \langle\bar{0}| R_{k,i}^\dagger R_{l,j} |\bar{0}\rangle + h.c.\right)$ or $\left(f_{k,i}^* \lambda_m \langle\bar{0}| R_{k,i}^\dagger \sigma_m R |\bar{0}\rangle + h.c.\right)$ in the matrix A and the vector C as given in Eqs. (3.27) and (3.28) can be written in the following form:

$$q = a\mathrm{Re}\left(e^{i\zeta} \langle 0|^{\otimes n} U |0\rangle^{\otimes n}\right), \tag{3.65}$$

where the amplitude a and the phase ζ are determined by either $f_{k,i}^* f_{l,j}$ or

$f^*_{k,i}\lambda_m$. The unitary operator U is equal to either $R^\dagger_{k,i}R_{l,j}$ or $R^\dagger_{k,i}\sigma_m R$.

The terms q can be obtained by the parameterized quantum circuits shown in Fig. 3.1. Note that these circuits are different from the circuit which prepares the final state $|\phi(\hat{\boldsymbol{\theta}}_{N_\tau})\rangle$. An ancillary qubit of the circuit is initialized as $(|0\rangle + e^{i\zeta}|1\rangle)/\sqrt{2}$, while the remaining qubits are initialized in the state $|0\rangle^{\otimes n}$. A projective measurement of the ancillary qubit in the $\{|+\rangle, |-\rangle\}$ basis has following probability to find the qubit in the state $|+\rangle$:

$$\hat{P}_+ = \frac{\mathrm{Re}\big(e^{i\zeta}\langle 0|^{\otimes n} U |0\rangle^{\otimes n}\big) + 1}{2}. \tag{3.66}$$

To estimate the probability $\hat{P}_+$ and subsequently the terms q, we need to implement and measure the circuit N_r times. Let us assume that the measurements correspond to independent Bernoulli distributed random variables, which have the variance $\sigma^2(\hat{P}_+) = \hat{P}_+(1-\hat{P}_+)/N_r$.

Thus, the variances of the estimates of the elements of Eq. (3.27) are scaling as

$$\sigma^2\left(\mathrm{Re}\left(e^{i\zeta}\langle 0|^{\otimes n} U |0\rangle^{\otimes n}\right)\right) = 4\sigma^2(\hat{P}_+) = \mathcal{O}(\frac{1}{N_r}). \tag{3.67}$$

After estimating $\mathrm{Re}\big(e^{i\zeta}\langle 0|^{\otimes n} U |0\rangle^{\otimes n}\big)$, the amplitudes of $f^*_{k,i}f_{l,j}$ or $f^*_{k,i}\lambda_m$ have to be calculated classically and multiplied. Since they are not depending on each other, all of the circuits with the same input parameters $\boldsymbol{\theta}(\tau)$ can be run in parallel.

We denote the realization of the matrices A and C in Eq. (3.23) as $\hat{A}$ and $\hat{C}$, respectively.

The variances of the estimates $\hat{A}_{k,l}(\boldsymbol{\theta}(\tau))$ and $\hat{C}_k(\boldsymbol{\theta}(\tau))$ are scaling as

$$\sigma^2\left(\hat{A}_{k,l}(\tau)\right)\mathcal{O}\left(\frac{N_d^2}{N_r}\right), \quad \sigma^2\left(\hat{C}_k(\tau)\right)\mathcal{O}\left(\frac{N_d N}{N_r}\right). \tag{3.68}$$

Then by Chebyshev's inequality, we can conclude that the probability of estimation error is bounded by

$$P\left(|A_{k,l} - \hat{A}_{k,l}| \geq b\right) \leq \frac{N_d^2}{N_r b^2}, \quad P\left(|C_k - \hat{C}_k| \geq b\right) \leq \frac{N_d N}{N_r b^2}. \tag{3.69}$$

We can conclude that we face up with estimation errors arise due to shot noise and the representation errors from calculating the matrix elements on the quantum circuits. The following lemma holds:

Lemma 3. *With the probability of at least* $1-\eta$, $\eta \in [0,1]$, *the following bounds* *hold:*

$$\|A - \hat{A}\| \leq \frac{\|\{\sigma_{k,l}\}_{k,l=1}^{N_V}\|}{\sqrt{N_r \eta}}, \tag{3.70}$$
$$\|C - \hat{C}\| \leq \frac{\|\{\sigma_k\}_{k=1}^{N_V}\|}{\sqrt{N_r \eta}},$$

where we used the notations

$$\sigma_{k,l} = \sqrt{\sum_{i,j=1}^{N_d} |f_{k,i}^* f_{l,j}|^2}, \quad \sigma_k = \sqrt{\sum_{i=1}^{N_d} \sum_{m=1}^{N} |f_{i,k}^* \lambda_m|^2}. \tag{3.71}$$

For a detailed proof, see Appendix 3.E.

This lemma provides us with upper bounds on the errors on A and C that arise from shot noise. These errors translate into an error in f as defined in Eq. (3.29), which can be estimated with the following lemma:

Lemma 4. *Consider the linear equation*

$$Af = C \ , \tag{3.72}$$

where A is a a non-singular $N_V \times N_V$-dimensional matrix and C an N_V-dimensional vector. Let us introduce disturbances in A and C by $A \mapsto A + \xi R$ and $C \mapsto C + \xi r$, leading to the disturbed linear equation

$$(A + \xi R)\hat{f} = C + \xi r \ , \tag{3.73}$$

where $\hat{f}$ is the solution vector under the disturbance, $\xi \geq 0$ is a real scalar, R is an $N_V \times N_V$ dimensional matrix and r is an N_V dimensional vector. Then, we can write the following estimate:

$$\frac{\|\hat{f} - f\|}{\|f\|} \leq \xi\kappa(A) \left(\frac{\|r\|}{\|C\|} + \frac{\|R\|}{\|A\|} \right) + \mathcal{O}(\xi^2) \ , \tag{3.74}$$

where $\kappa(A) := \|A\|\|A^{-1}\|$ is the condition number of A.

We give a proof of this lemma in Appendix 3.F.

If A is invertible, we can identify $f = A^{-1}C$, and can apply this theorem to get:

$$\|f - \hat{f}\| \leq \|A^{-1}C\|\xi\kappa(A) \left(\frac{\|r\|}{\|C\|} + \frac{\|R\|}{\|A\|} \right) + \mathcal{O}(\xi^2) \ . \tag{3.75}$$

In order to apply the theorem to our analysis above, we are defining $R = \{\sigma_{k,l}\}_{k,l=1}^{N_V}$, $r = \{\sigma_k\}_{k=1}^{N_V}$ and

$$\xi = \frac{1}{\sqrt{N_r \eta}} \,. \tag{3.76}$$

If in a specific instance, A is singular, it is not possible to invert it and thus apply Lemma 4. However, one can apply a regularization procedure by deviating the matrix elements of A slightly in such a way that it becomes non-singular. This will be leading to an additional error which, as we described in Sec. 3.2.3, is one of the error sources we are not taking into consideration in our analysis. For a way of dealing with this issue, see techniques used in Ref. [26, 116, 128, 152].

Taken together, we arrive at the following number of total circuit evaluations, combining Eqs. (3.39), (3.46), (3.62), (3.75), and the results of Thm. 3.5:

$$N_{circ} = N_\tau^{(\delta)} s N_r^{(\delta)} N_V N_d (N_V N_d + N) \tag{3.77}$$

$$N_r^{(\delta)} = \frac{9\Sigma^2}{L_{fy}^2} \left(\frac{\epsilon_{target}^{(\delta)}}{\left(1 + F(N_\tau^{(\delta)}, s)\right)^{N_\tau^{(\delta)}} - 1} - \left(\frac{T}{N_\tau^{(\delta)}}\right)^{p+1} \frac{K L_{ft}^p M}{F(N_\tau^{(\delta)}, s)} \right)^{-2} \tag{3.78}$$

$$\Sigma = \|A^{-1}C\| \frac{\kappa(A)}{\sqrt{\eta}} \left(\frac{\|r\|}{\|C\|} + \frac{\|R\|}{\|A\|} \right) + \mathcal{O}(\xi^2) \tag{3.79}$$

$$F(N_\tau, s) = \frac{b_{max}}{a_{max}} \left(\left(1 + L_{fy} a_{max} \frac{T}{N_\tau}\right)^s - 1 \right) . \tag{3.80}$$

To provide further details and to make the analysis concrete, in Sec. 3.5.3, we select a specific toy model as a representative example. We estimate the quantities $\|A\|$, $\|C\|$, $\|A^{-1}C\|$, $\kappa(A)$ and L_{fy} based on this toy model and in Sec. 3.6, we will estimate $\|R\|$, $\|r\|$, a_{max}, b_{max}, K, M and $L_{f\tau}$ for general Runge-Kutta methods and a chosen Ansatz.

3.5.3 Estimations based on a toy model

In order to apply the error and resource estimates from Sec. 3.4 in practice (such as we do in Sec.3.6), we need to make estimates on several quantities. Especially challenging is the estimation of the condition number $\kappa(A)$, $\|A\|$, $\|C\|$ and $\|A^{-1}C\|$ that are needed for the estimates on shot noise as derived in Sec. 3.5.2 and the estimation of the Lipschitz constant L_{fy}, as defined in Eq. (3.37).

As those quantities depend on A and C, we are taking the following considerations: Both A and C depend on expectation values from quantum circuits as written in Eq. (3.65). The parameters $\boldsymbol{\theta}(\tau_k)$ at specific time steps τ_k vary a lot depending on the instance being solved. Since the expectation values that constitute the coefficients of A and C are depending on these parameters as well as the chosen Ansatz, it is non-viable to estimate them in general.

However, motivated by the fact that expectation values of parameterized quantum circuits defined as in our chosen Ansatz (see Eq. (3.18)) resemble truncated Fourier series (see Refs. [67, 154]), we simulate plausible coefficients of A and C according to the following toy model:

Let us assume that each of the elements $A_{k,l}$ and C_k take the form of functions $w_{k,l}(\tilde{\theta})$ and $w_k(\tilde{\theta})$ in one input parameter $\tilde{\theta}$, respectively:

$$w_{k,l}(\tilde{\theta}) := \alpha_{1,k,l}\cos\left(\alpha_{2,k,l}\tilde{\theta}+\alpha_{3,k,l}\right)+\alpha_{4,k,l}\sin\left(\alpha_{5,k,l}\tilde{\theta}\right), \tag{3.81}$$

$$w_k(\tilde{\theta}) := \alpha_{1,k}\cos\left(\alpha_{2,k}\tilde{\theta}+\alpha_{3,k}\right)+\alpha_{4,k}\sin\left(\alpha_{5,k}\tilde{\theta}\right) . \tag{3.82}$$

We are choosing the amplitudes $\alpha_{1,k,l}$, $\alpha_{1,k}$ and $\alpha_{4,k,l}$, $\alpha_{4,k}$ to be randomly drawn from the Gaussian distribution $\mathcal{N}(1, 0.1)$, and the phases $\alpha_{2,k,l}$, $\alpha_{2,k}$ and $\alpha_{5,k,l}$, $\alpha_{5,k}$ and phase differences $\alpha_{3,k,l}$, $\alpha_{3,k}$ to be randomly drawn from the Gaussian distribution $\mathcal{N}(0, 0.1)$.

Condition number and norms on A and C

In the literature, it is often assumed that the condition number of a matrix scales polynomially in its dimension [155]. If this assumption holds for the $N_V \times N_V$-dimensional matrix A, there exists a constant Γ, such that with high probability, the following bound holds:

$$\kappa(A) \leq N_V^{\Gamma} . \tag{3.83}$$

In order to find out if this bound also holds if the coefficients of A are sampled according to the toy model as described above, we plot such $\kappa(A)$ and the functions N_V^2, N_V^3 and N_V^4 for varying N_V in Fig. 3.2.

Most of the condition numbers lie above the threshold N_V^2, while for $N_V \geq 10$, most condition numbers lie below the threshold N_V^3. Typically, more than 10 parameters are chosen in applications of the variational algorithm (We are using $N_V = 25$ in our numerical analysis in Sec. 3.6). In rare cases, A constructed by the toy model is ill-conditioned, and $\kappa(A)$ significantly exceeds those bounds. As described earlier, those cases can be mitigated by a regularization of A with the drawback of introducing an additional error.

We therefore assume that the bound in Eq. (3.83) holds with high probability for our toy model, with the constant $\Gamma = 3$.

Analogously, we estimate the norms $\|A\|$, $\|C\|$ and $\|A^{-1}C\|$ based on A and C that follow the introduced toy model. In Fig. 3.3, we plot those norms and the functions $f_1(N_V) = N_V$ and $f_2(N_V) = \sqrt{N_V}$ for varying N_V.

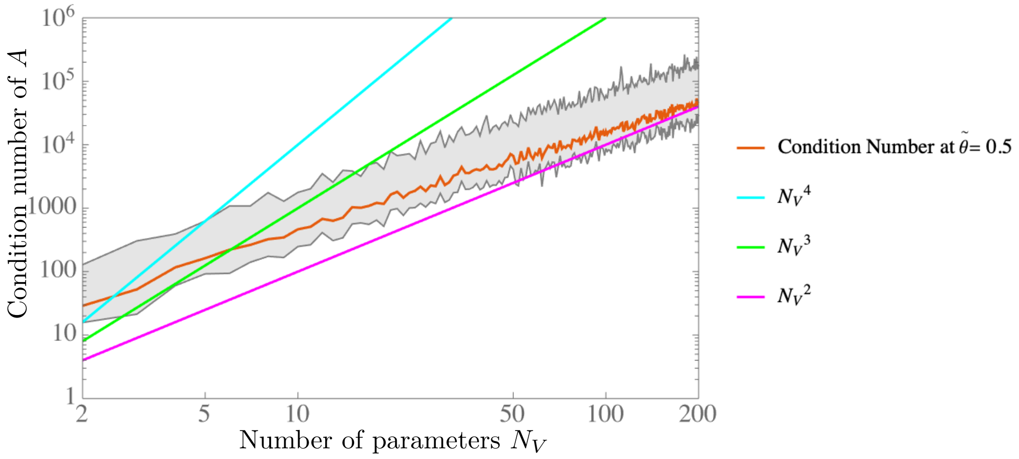


Figure 3.2: A log plot comparing the condition number obtained from the toy model with $\tilde{\theta} = 1/2$ and different upper bounds due to N_V^2, N_V^3 and N_V^4. We sampled 100 different matrices A according to the toy model and calculated the resulting condition number for each sample. The orange line shows the median and the gray shaded area shows the range between the 0.16 and 0.84 quantiles of the condition number estimates. We used the Frobenius norm.

We see that it is reasonable to assume the scaling $\|A\| = \Theta(N_V)$ and $\|C\| = \Theta(\sqrt{N_V})$. For $\|A^{-1}C\|$, we cannot make similarly reasonable estimates due to the high fluctuation. However, we can assume an upper bound $\|A^{-1}C\| \leq 60$, which holds with high probability in the number of parameter range $N_V \in [0, 100]$. Similar to managing the condition number, it is also possible to enforce this bound by a regularization of the matrix A.

The above analysis on estimates and bounds for $\kappa(A)$, $\|A\|$, $\|C\|$ and $\|A^{-1}C\|$ included into the estimate in Eq. (3.46) leads to the following inequality to hold with high probability:

$$\|f - \hat{f}\| < \delta \leq \frac{60}{\sqrt{N_r \eta}} N_V^3 \left(\frac{\|\{\sigma_k\}_{k=1}^{N_V}\|}{\sqrt{N_V}} + \frac{\|\{\sigma_{k,l}\}_{k,l=1}^{N_V}\|}{N_V} \right) . \tag{3.84}$$

Hence, we conclude the following upper bound for Σ (see Eq. (3.46)) with high probability:

$$\Sigma \leq \frac{60}{\sqrt{\eta}} N_V^3 \left(\frac{\|\{\sigma_k\}_{k=1}^{N_V}\|}{\sqrt{N_V}} + \frac{\|\{\sigma_{k,l}\}_{k,l=1}^{N_V}\|}{N_V} \right). \tag{3.85}$$

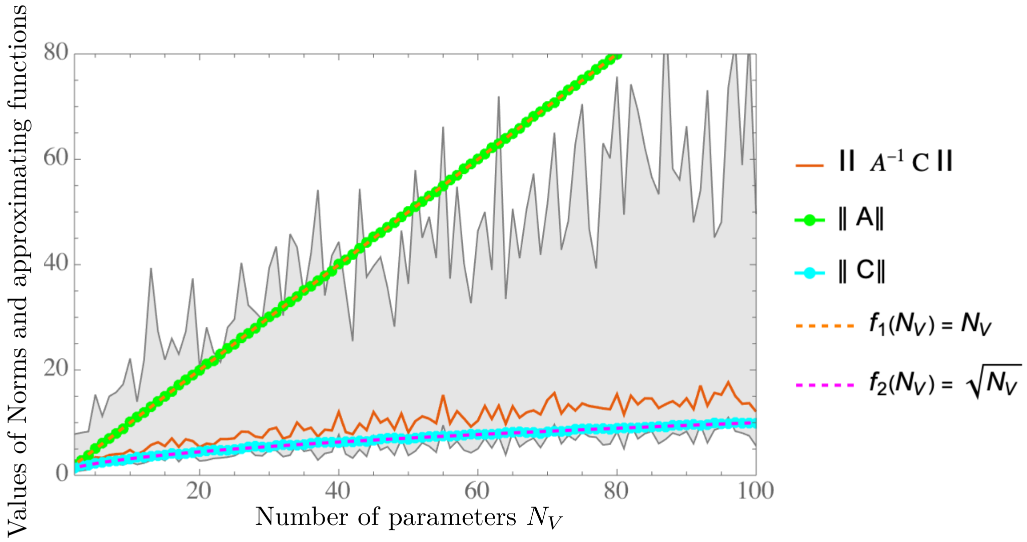


Figure 3.3: A plot comparing the norms $\|A\|$, $\|C\|$ and $\|A^{-1}C\|$ obtained from the toy model with the functions $f_1(N_V) = N_V$ and $f_2(N_V) = \sqrt{N_V}$. In order to estimate $\|A^{-1}C\|$, we did 100 samples for A and C and calculated the resulting norm for each sample. The orange line shows the median and the gray shaded area shows the range between the 0.16 and 0.84 quantiles of the estimates for $\|A^{-1}C\|$. For $\|A\|$, we used the Frobenius norm and for $\|C\|$ and $\|A^{-1}C\|$ the 2 norm.

Estimation of the Lipschitz constant

We have seen in Sec. 3.4, that the Lipschitz constant L_{fy} of $f(\boldsymbol{\theta}(\tau))$ with respect to the $\boldsymbol{\theta}(\tau)$ has a direct influence on the error and resource estimates. In fact, as we will see in Sec. 3.6, the sensitivity of the total cost is higher with L_{fy} than with most of the other parameters. Let us therefore analyze the Lipschitz constant for the variational algorithm described in Sec. 3.2.2. In this case, $f(\boldsymbol{\theta}(\tau))$ is defined as

$$f(\boldsymbol{\theta}(\tau)) = f\left(\boldsymbol{\theta}(\tau)\right) := A^{-1}\left(\boldsymbol{\theta}(\tau)\right) C\left(\boldsymbol{\theta}(\tau)\right) \ , \tag{3.86}$$

and the Lipschitz constant L_{fy} turns into the upper bound

$$L_{fy} \geq \frac{\|f\left(\boldsymbol{\theta}_1(\tau)\right) - f\left(\boldsymbol{\theta}_2(\tau)\right)\|}{\|\boldsymbol{\theta}_1(\tau) - \boldsymbol{\theta}_2(\tau)\|} \ , \quad \forall \boldsymbol{\theta}_1(\tau), \boldsymbol{\theta}_2(\tau) \ . \tag{3.87}$$

We are again modeling the matrix A and the vector C with the toy model described in Eqs. (3.81) and (3.82). This toy model has a single input variable $\tilde{\theta}$ instead of an N_V-dimensional vector $\boldsymbol{\theta}(\tau)$. Let us vary this parameter, while we keep the other (randomly drawn) parameters of the toy model fixed, and define the following function:

$$\mathrm{Lip}\left(\tilde{\theta}_1, \tilde{\theta}_2\right) := \frac{\left\| f\left(\tilde{\theta}_1\right) - f\left(\tilde{\theta}_2\right) \right\|}{\left\| \tilde{\theta}_1 - \tilde{\theta}_2 \right\|} \, . \tag{3.88}$$

We show two 3D plots of this function, with varying $\tilde{\theta}_1$ and $\tilde{\theta}_2$ and fixed dimensions of A and C with $N_V = 25$ in Fig. 3.4. For each of the two plots, the random toy model parameters are drawn independently.

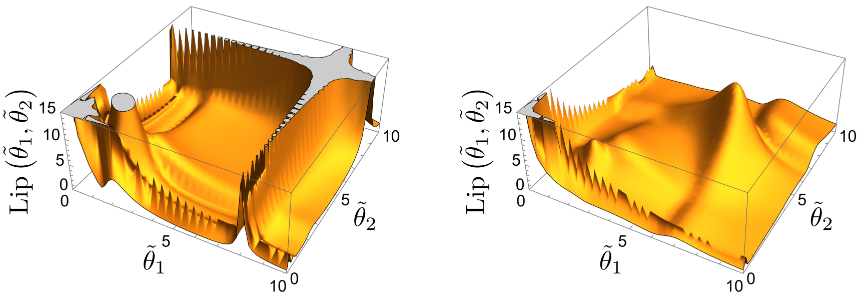


Figure 3.4: The function $\mathrm{Lip}\left(\tilde{\theta}_1, \tilde{\theta}_2\right)$ defined in Eq. (3.88) plotted for two different drawings of random matrix and vector parameters according to the toy model from Eqs. (3.81) and (3.82). We fixed $N_V = 25$ and varied the parameters $\tilde{\theta}_1$ and $\tilde{\theta}_2$ between 0 and 10.

At the gray areas, the function $\mathrm{Lip}\left(\tilde{\theta}_1, \tilde{\theta}_2\right)$ exceeds the value of 15. We see that in order to be able to bound $\mathrm{Lip}\left(\tilde{\theta}_1, \tilde{\theta}_2\right)$ from above by a Lipschitz constant L_{fy}, one has to guarantee that at each update step from $\tilde{\theta}_1$ to $\tilde{\theta}_2$, the function $\mathrm{Lip}\left(\tilde{\theta}_1, \tilde{\theta}_2\right)$ stays inside the region below the threshold $L_{fy} = 15$. It is possible to take higher estimates for L_{fy} and create possibly larger areas for valid $\tilde{\theta}_1$ and $\tilde{\theta}_2$. However, this comes to the account of less tight error and resource bounds in Thms. 3.4 and 3.5.

Interestingly enough, the function $\mathrm{Lip}\left(\tilde{\theta}_1, \tilde{\theta}_2\right)$ keeps similar patterns for changing the mean and the variance of the distributions from which the amplitudes of the toy model are drawn. However, it shows an increasingly more rugged landscapes for both increasing the mean and the variance of the random distributions of the phases of the toy model. As expected, the function $\mathrm{Lip}\left(\tilde{\theta}_1, \tilde{\theta}_2\right)$ behaves periodically with respect to the phase difference parameter of the toy model, with the period depending on the phases.

Under the assumption that the toy model we chose models the behavior of A and C sufficiently well, we use the estimates for $\kappa(A)$, $\|A\|$, $\|C\|$, $\|A^{-1}C\|$ and L_{fy} derived in this section in order to concretize the error and resource estimates from Sec. 3.4, and to numerically analyze them in the following section.

3.6 Numerical analysis of the error and resource estimates

In this section, we are numerically analyzing the error and resource estimates from Sec. 3.3 and Sec. 3.4 for an application of the RKMs to solving a simple ODE and for an application of the variational algorithm presented in Sec. 3.2.2 to solving a partial DE coming from finance.

3.6.1 Solving a classical ODE without shot noise

Let us analyze the cost function provided in Corollary 1 with a simple ODE:

$$\frac{\partial \boldsymbol{\theta}(\tau)}{\partial \tau} = \frac{\boldsymbol{\theta}(\tau)}{2}, \quad \boldsymbol{\theta}(0) = 1 \; . \tag{3.89}$$

This ODE has the exponential function $\boldsymbol{\theta}(\tau) = \exp\left(\frac{\tau}{2}\right)$ as a solution. We choose this simple ODE, because it is easy to analyze and to estimate the corresponding parameters that are used for the cost function.

Parameter estimation and sensitivity analysis for the classical ODE solver

The Lipschitz constant L_{fy} is equal to 0.5. If we pick a final time $T = 5$, then the function $f(\boldsymbol{\theta}(\tau)) = \frac{1}{2}\boldsymbol{\theta}(\tau)$ is upper bounded by $M = 13$. The derivative of $f(\boldsymbol{\theta}(\tau))$ with respect to τ is equal to $0.25 \exp\left(\frac{1}{2}\tau\right)$ (since we know $\boldsymbol{\theta}(\tau) = \exp\left(\frac{1}{2}\tau\right)$). Thus we can choose a Lipschitz constant $L_{f\tau} = 3.1$, which upper bounds this derivative for times $\tau \in [0, T = 5]$.

It can be easily checked that the other bounds required in Thm. 3.1 hold as well.

The approximation of $b_{max} := \max_i |b_i|$ is depending on the chosen Runge-Kutta method. While there exist Runge-Kutta methods that have coefficients $|b_i| > 1$, those are the exception as one can see by looking at several methods (for example, in [140]).

To estimate the constant K, let us look at its definition [140, Chapter 318]:

$$K := \sum_{|t|=p+1} \frac{1}{\sigma(t)} \left| \Phi(t) - \frac{1}{t!} \right|, \tag{3.90}$$

where t is a rooted tree of order $|t|$, the quantities $\sigma(t)$ and $t!$ are defined as products of factorials, and lie between 1 and $|t|!$. The quantity $\Phi(t)$ depends on the coefficients of the chosen method. For a rigorous definition of these terms, see [140]. As it is discussed in [140, Chapter 318], it is not possible to create a general rule for estimating the terms in K. However, what can be said is that the number of terms in the sum, the number of unlabeled rooted trees with $p+1$ nodes, is asymptotically equal to [156] $0.439 \cdot 2.956^{p+1} \cdot (p+1)^{-3/2}$. This series scales exponentially in p. Thus, if the summands $\frac{1}{\sigma(t)}\left|\Phi(t) - \frac{1}{t!}\right|$ were constant in p, K would scale exponentially in p as well. However, in [140] it is said that it is reasonable to assume that K is constant in $\Delta\tau$. In Ref. [140, Chapter 244], several Runge-Kutta (and related) methods were developed for which the corresponding error constants were estimated, all of which are upper-bounded in magnitude by 1. In the following analysis, we therefore assume that $K \leq 5$ holds with high probability. But, as we will see later, the total cost does not change as much in the parameter K as it does in other parameters, and we found that the implications of our work still hold qualitatively with a pessimistic upper bound of $K \leq 0.439 \cdot 2.956^{p+1} \cdot (p+1)^{-3/2}$. Let us further choose a target error of $\epsilon_{target} = 0.001$.

We collect the estimates for the analysis in this section in Table 3.2. Based

Parameter	**Estimate**
b_{max}	1
L_{fy}	0.5
T	5
K	5
$L_{f\tau}$	3.1
M	13
ϵ_{target}	0.001

Table 3.2: Estimates of various parameters that optimize the savings by using a higher-order Runge-Kutta method instead of the Euler method. Used for the analysis in Fig. 3.6.

on these default values, we are analyzing the sensitivity of the total cost on tuning the parameters in Fig. 3.5. We show how the total cost changes when a single default value is multiplied with a scaling factor, whereas the remaining parameters are kept at default. We see that the cost does go down for higher-order Runge-Kutta methods and that the parameter with the most effect is the total time T, the Lipschitz constant L_{fy}, the parameter b_{max} stemming from the Runge-Kutta method and the total target error ϵ_{target}. The spike in the cost at order $p = 6$ is due to the non-linear relationship between the order of the method and the number of stages (see Table 3.1).

Numerical analysis of the classical ODE solver

Using the parameters estimated above in Table 3.2, we estimate the total cost for Runge-Kutta methods of different orders. For the relation between the minimum number of stages s needed for a particular order p, we use Table 3.1.

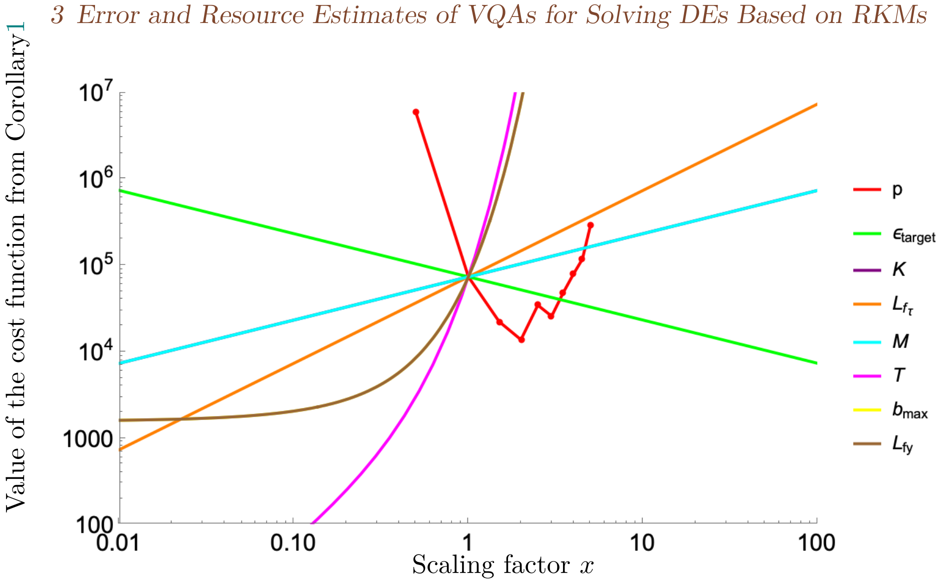


Figure 3.5: A plot comparing the sensitivity of the cost function with respect to different parameters. The intersecting point in the middle is the value of the cost function where all parameters are chosen default as given in Table 3.2 as well as $p = 2$ as a default. In each graph, we are changing one parameter, while keeping all the other parameters at default. We are changing the parameter by multiplying with the scaling factor x given as the abscissa. The graphs for M and K overlap as well as the graphs for L_{fy} and $b_{\max}$. The continuous red line for p is just for visualization purposes, as p is integer.

We show the results in Fig. 3.6. On the left hand side is a table with the cost depending on the order and the ratio of the Euler method compared to a Runge-Kutta method of order p. On the right hand side, we plotted this ratio depending on the Runge-Kutta order. We see that for our particular example, all methods of order $2 \leq p \leq 10$ are more cost efficient than the Euler method, where a Runge-Kutta method of order 4 is the most cost efficient with savings of a factor of ca. 10^3 compared to the cost of the Euler method. There is a second spike for the order $p = 6$, because of the non-linear relationship between the order of the method and the number of stages as shown in Table 3.1.

3.6.2 Solving a linear PDE with the variational quantum algorithm

We are numerically analyzing the estimates from Sec. 3.4 and Sec. 3.5 for an application of the variational algorithm presented in Sec. 3.2.2 to solving the Black Scholes model, which is a linear partial DE coming from finance. The motivation for choosing this model is twofold: On the one hand, we are using this example in order to specify the estimates done in previous sections and

p	cost(p)	cost(1)/cost(p)	$N_\tau^{(0)}$
1	2.25×10^7	1.00	2.25×10^7
2	9.60×10^4	2.35×10^2	4.80×10^4
3	1.99×10^4	1.13×10^3	6.63×10^3
4	1.01×10^4	2.22×10^3	2.54×10^3
5	1.38×10^4	1.64×10^3	2.29×10^3
6	1.03×10^4	2.18×10^3	1.47×10^3
7	1.36×10^4	1.65×10^3	1.52×10^3
8	1.71×10^4	1.32×10^3	1.56×10^3
9	2.07×10^4	1.09×10^3	1.60×10^3
10	3.33×10^4	6.76×10^2	2.08×10^3

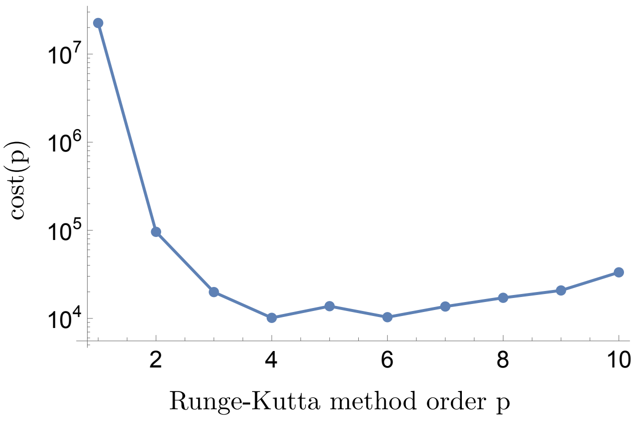


Figure 3.6: Comparison of the cost for different RKM orders, where the estimated parameters are fine tuned to maximize the savings by using a higher order as given in Table 3.2. In the first column, we have the Runge-Kutta method order p. In the second column is the total cost of an algorithm that uses a Runge-Kutta method of order p, in the third column the ratio of an algorithm that uses the Euler method ($p = 1$) with an algorithm that uses a Runge-Kutta method of order p, and in the fourth column is the minimal number of time steps $N_\tau^{(0)}$. In the plot, we show the cost, which is calculated according to Corollary 1, plotted against the order of the Runge-Kutta method.

in showing the consequences of the choice of RKMs on the total cost. On the other hand, we are using this example to stress the wide range of DEs on which the variational algorithm presented in Sec. 3.2.2 can be applied to.

Black Scholes model and error analysis

Let us begin with describing the problem of option pricing and the application of the variational algorithm to it. This application has been studied before in several ways [26, 127, 128, 132].

A European call option is used in practice in the following way: Initially, at time $t = 0$, an option is acquired, specifying a particular underlying asset, an expiration time t_{final}, and a strike price K. When time reaches $t = t_{final}$, the option buyer faces a decision: whether to exercise the option by buying the asset at the strike price K or to refrain from exercising it. The buyer's rational choice at time t_{final} is determined by the price $S(t_{final})$ of the underlying asset at that moment. If $S(t_{final}) > K$, indicating that the final asset price exceeds the strike price, the buyer can use the option to purchase the asset for the price K and immediately sell it at the higher market price of $S(t_{final})$ in an ideal market. This transaction results in a profit of $S(t_{final}) - K$ for the buyer. Conversely, if $S(t_{final}) \leq K$, the buyer would not choose to exercise the option by purchasing the asset since they would not be able to generate a profit from selling it in the market.

One can conclude that the payoff of the buyer is equal to

$$V(t_{final}, S) = \max\{S(t_{final}) - K, 0\} \ . \tag{3.91}$$

Since the stock price at time t_{final} is unknown, one models the stock price stochastically. A simple model is the so-called Black-Scholes model which characterizes the stock price $S(t)$ as a stochastic variable that follows a geometric Brownian motion:

$$dS(t) = \mu S(t)dt + \sigma S(t)dW_t \ , \tag{3.92}$$

where μ is the drift of the stock price, σ is its standard deviation (called 'volatility') and the random variable dW_t is a Wiener process. The arbitrage assumption states that it is impossible to build a portfolio which gives positive return without risk. By incorporating this assumption and using Itô calculus, the stochastic DE in $S(t)$ is transformed to a parabolic partial DE (PDE) that models the price of a call option $V(t, S)$:

$$\frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} + rS\frac{\partial V}{\partial S} = rV \ . \tag{3.93}$$

This PDE is called the Black Scholes equation. The parameters of this model are the volatility σ of the stock price and the risk-free interest rate r. Both are assumed to be independent of time. The Black Scholes can be mapped to the imaginary time Schrödinger equation with the following transformations:

Applying the transformations $\tau = (t_{final} - t)\sigma^2$, $x = \log(S)$, and subsequently

$u(\tau, x) = e^{-ax-b\tau}V(\tau, x)$ with the parameters $a = \frac{1}{2} - \frac{r}{\sigma^2}$ and $b = -\frac{1}{2}a^2 - \frac{r}{\sigma^2}$, where $u(\tau, x)$ is a modified price, one obtains [26]:

$$\frac{\partial}{\partial \tau}u(\tau, x) = \frac{1}{2}\frac{\partial^2}{\partial x^2}u(\tau, x) \ . \tag{3.94}$$

Note that $\tau \in [0, T]$, where we write $T = t_{final}\sigma^2$. This equation is equivalent to the imaginary time Schrödinger equation in Eq. (3.15), where $|\psi(\tau)\rangle$ is a quantum state representing the option price and $\mathcal{H} = -\frac{1}{2}\frac{\partial^2}{\partial x^2}$ is the Hamiltonian operator. Solving Eq. (3.15) is hence equivalent to solving the Black-Scholes equation in Eq. (3.93) after reversing the transformations.

We can interpret the state with the option price in the following way:

Consider a register of n qubits. On this chain, we define a set of 2^n pairwise orthogonal states $\{|x\rangle\}_x$. We select an interval of possible stock prices and discretize the interval into 2^n points. Consequently, we associate the basis states with stock prices, where the state $|0\rangle^{\otimes n}$ corresponds to the minimum stock price, and the state $|1\rangle^n$ corresponds to the maximum stock price of the chosen interval.

The quantum state from Eq. (3.16) encodes the option value corresponding to a particular stock price in the amplitudes of $|\psi(\tau)\rangle$ in the following way:

$$|\psi(\tau)\rangle = \sum_x \sqrt{p_x(\tau)}\,|x\rangle \ , \quad \sum_x p_x(\tau) = 1, \quad \forall\tau. \tag{3.95}$$

The boundary condition is $p_x(0) = \gamma(0)V(0, x)e^{-ax}$, where $\gamma(0)$ is the normalization constant and $V(0, x)$ is the option price at time $\tau = 0$. Thus, by obtaining the probability $p_x(\tau)$ at time $T = t_{final}\sigma^2$ for a particular stock price $x = \log(S)$, the corresponding option price can be calculated as:

$$V(t_{final}, x) = \gamma^{-1}(T)p_x(T)e^{ax+bt_{final}\sigma} \ . \tag{3.96}$$

In the following subsections, we are estimating the parameters based on the application of the variational algorithm to this option pricing use case and do a numerical analysis.

Parameter estimations and sensitivity analysis for the variational quantum algorithm

We are now making educated guesses for the parameters that are related to the use case of option pricing as described above and analyze the sensitivity of our error and resource analysis from Sec. 3.4 with respect to these parameters. In this and the following section, we use the symbol $\sim$ for approximations of a term with a scalar, including rounding errors.

In several papers [26, 127, 128, 132], the application of the variational algorithm as described has been applied to option pricing. We are basing our parameter estimates on Ref. [26, Sec. 5.1.], where the authors take an Ansatz with the parameters $N = 16$, $N_V = 25$ and $N_d = 1$. That means that the matrix A is a 25×25 matrix, the vectors C and $\boldsymbol{\theta}$ have 25 elements, as well as the matrix $\sigma_{k,l}$ and the vector σ_k that we defined in Lemma 3. We take the probability η from the same lemma equal to $\eta \sim 0.05$.

A typical total time t_{final} in option pricing is one year and a typical volatility is $\sigma = 0.2 \cdot 1/\text{year}^2$. Thus, $t_{final} \cdot \sigma^2 = T \sim 0.04$.

In Ref. [26, Chapter 3], the parameters $f_{k,j}$ are defined as $f = i/2$. With this, let us approximate the perturbations $\|\sigma_{k,l}\|$ and $\|\sigma_k\|$:

It makes sense to see them in terms of the maximal deviation of individual entries. Given $f = i/2$, they are for the matrix equal to $N_d^2 \cdot 1/2 \cdot 2$ and for the vector $N_d N \cdot 1/2 \cdot 2$. Thus, we assume the total norms to scale as $\|\sigma_{k,l}\| \leq N_V N_d^2 \cdot 1$ and $\|\sigma_k\| \leq N_V N_d N \cdot 1$.

For approximations of the parameters a_{max}, b_{max} and K, see our discussion in Sec. 3.6.1. We pick now the parameters $a_{max} = b_{max} = 1$ and the $K = 5$. But as above, we see later that N_{circ} does not depend strongly on K and our results are still qualitatively valid for a large variety of K.

In line with the discussion in Sec. 3.5.1, we estimate the Lipschitz constant L_{fy} to be equal to $L_{fy} = 15$, the condition number to be upper bounded as $\kappa(A) \leq N_V^3 = 15625$ and $M = 60$ with the caveat of having to assure the toy model is chosen adequate enough and the bounds can be guaranteed by regularization to satisfy these bound.

We are also estimating a probability coming from the shot noise of $\eta = 0.05$. Combining the estimates, we are therefore getting with high probability the upper bound

$$\Sigma \leq \frac{60}{\sqrt{\eta}} N_V^3 \left(\frac{N_V N_d N}{\sqrt{N_V}} + \frac{N_V N_d^2}{N_V} \right) \sim 3.4 \times 10^8 \ . \tag{3.97}$$

The parameter $L_{f\tau}$ is defined as an upper bound to the time derivative of $f(\boldsymbol{\theta}(\tau))$. Since the time dependence of f is fully carried via the function $\boldsymbol{\theta}(\tau)$, we can apply a reasoning similar to that for estimating L_{fy}. We assume that the same behavior and the same bounds hold and, therefore, estimate $L_{f\tau} = L_{fy} = 15$.

We cannot determine if the other bounds hold that are required in Thm. 3.1, since we do not have full knowledge of the function $f(\boldsymbol{\theta}(\tau))$.

For estimating the quantity S as defined in Eq. (3.61), we need to take the following into account: We estimate that for all $1 \leq k \leq N_V$, $|\theta_k^*(T)| \sim \frac{\|\boldsymbol{\theta}\|_2}{N_V}$. Together with taking $f_{k,j} = i/2$ as above, we get $S \sim 1$.

Lastly, we are choosing a target error of $\epsilon_{target} \sim 0.001$. Let us thus make the following estimates, collected in Table 3.3:

Let us now show a plot where we examine the scaling of N_{circ} with respect

Parameter	Estimate
a_{max}	1
b_{max}	1
L_{fy}	15
T	0.04
K	5
$L_{f\tau}$	15
M	60
S	1
Σ	3.4×10^8
ϵ_{target}	0.001

Table 3.3: Estimates of various parameters based on the option pricing application. Used for the analysis in Figs. 3.7 and 3.8.

to the input parameters. It is difficult to make general statements about the scaling of N_{circ} from Eq. (3.62), since it does not have a trivial form, thus a look at a plot helps with understanding its scaling. We take as default the parameters estimated above and analyze in Fig. 3.7 the deviations caused by changing a single parameter away from the default.

We conclude from this figure, that N_{circ} does not change much for the parameters a_{max}, M, $L_{f\tau}$, K and Σ. As it was proved impossible to estimate K in a general way, it is promising to see that our estimate of N_{circ} does not scale with it as much as with other parameters.

In addition, N_{circ} changes faster for the parameters p, T, b_{max}, ϵ_{target}, and L_{fy}. For p in particular, we see that a minimum of N_{circ} is reached for a value higher than for $p = 1$. We will examine the behavior with p with more detail in the next section.

Numerical analysis of the variational quantum algorithm

For the numerical analysis, we use the parameters given in Table 3.3 that are fit to the option pricing use case in order to calculate N_{circ}, which is the total number of circuit evaluations needed for the algorithm. As we mentioned in Sec. 3.6.2, the parameter K is the most challenging to estimate. For this analysis, we choose $K = 5$. A comparison of the costs of algorithms based on Runge-Kutta and Euler methods is given in Fig. 3.8. We see that the highest saving in N_{circ} compared to the Euler method can be done with a Runge-Kutta method of order $p = 2$.

To reach a target error of $\epsilon_{target} = 0.001$, we need an $N_r^{(\delta)} \sim 10^{22}$, which is equivalent to requiring classical machine precision for the entries of matrices A and vectors C ($\epsilon \sim 1/\sqrt{N_r^{(\delta)}} \sim 5 \times 10^{-12}$). The total number of circuit

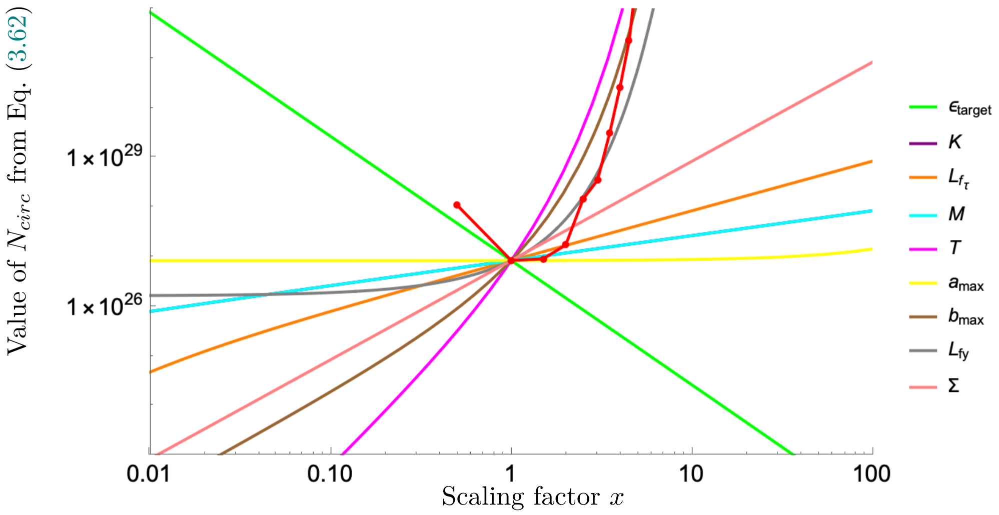


Figure 3.7: A plot comparing the sensitivity of N_{circ} with respect to different parameters. The intersecting point in the middle is the value of N_{circ} where all parameters are chosen default as given in Table 3.3 as well as $p = 2$ as a default. In each graph, we are changing one parameter, while keeping all the other parameters at default. We are changing the parameter by multiplying with the scaling factor x given as the abscissa. The following colors are corresponding to N_{circ} with one parameter changed: p, ϵ_{target}, $L_{f\tau}$, M and K, T, $a_{\max}$, $b_{\max}$, L_{fy}, Σ, where x is the scaling factor at the abscissa. The graphs for M and K overlap. The continuous red line for p is just for visualization purposes, as p is integer.

evaluations N_{circ} is equal to 1.62×10^{28}.

In Ref. [18, Supplementary information VIA], the time that the superconducting Sycamore chip can be used before having to be recalibrated is around 1 day. That implies that all calculations have to be done in at most 24 hours. The time for one readout of the Sycamore chip takes around $4\mu s$ [157]. That means that it is possible to evaluate around 2×10^{10} quantum circuits.

It is obvious that the high accuracy is needed because of the inversion of the matrix A and the resulting error propagation. The number $N_r^{(\delta)}$ given in Thm. 3.5 depends quadratically on the factor Σ estimated in (3.97) to be upper bounded with high probability by 3.4×10^8. If it was possible to reach a bound $\Sigma \leq 1$, the number $N_r^{(\delta)}$ would decrease to the order 10^7 which would be feasible for quantum hardware, .

In order to illustrate the potential resource savings that can be gained by choosing a higher-order RKM instead of the Euler method, we provide a second analysis based on a different choice of parameters provided in Table 3.4. This choice is inspired by the estimates in Table 3.3, changing some of the parameters within reasonable ranges in order to increase the resource savings. The resulting resource requirements are shown in Fig. 3.9. We conclude that by choosing a Runge-Kutta method of order $p = 4$, there have to be done a factor of $\sim 2.56 \times 10^3$ less circuit evaluations than when choosing the Euler method. However, the number of shots for each circuit ($N_r^{(\delta)} \sim 1.98 \times 10^{26}$) is still too high in order to be realized on any quantum hardware.

p	$N_{circ}(p)$	$N_{circ}(1)/N_{circ}(p)$	$N_r^{(\delta)}$	$N_\tau^{(\delta)}$	Circuits
1	2.13×10^{29}	1	7.03×10^{21}	2.96×10^{4}	3.03×10^{7}
2	1.62×10^{28}	13.18	3.87×10^{22}	2.04×10^{2}	4.19×10^{5}
3	1.75×10^{28}	12.21	1.53×10^{23}	37.06	1.14×10^{5}
4	3.31×10^{28}	6.45	5.19×10^{23}	15.55	6.38×10^{4}
5	3.38×10^{29}	6.31×10^{-1}	5.48×10^{24}	10.03	6.17×10^{4}
6	7.79×10^{29}	2.74×10^{-1}	1.56×10^{25}	6.96	4.99×10^{4}
7	7.49×10^{30}	2.85×10^{-2}	1.41×10^{26}	5.74	5.30×10^{4}
8	7.00×10^{31}	3.05×10^{-3}	1.25×10^{27}	4.98	5.62×10^{4}
9	6.45×10^{32}	3.31×10^{-4}	1.08×10^{28}	4.47	5.96×10^{4}
10	2.16×10^{34}	9.9×10^{-6}	3.03×10^{29}	4.33	7.11×10^{4}

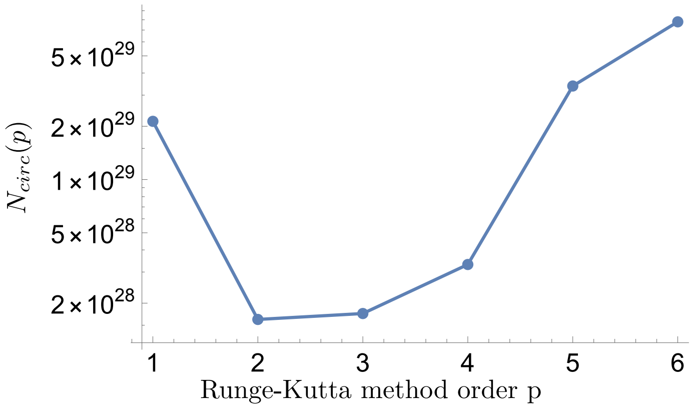


Figure 3.8: Comparisons of the resource requirements for different RKM orders, for the parameters in Table 3.3, applicable to the option pricing application. In the first column, we have the RKM order p. In the second column is N_{circ} of an algorithm that uses a RKM of order p and in the third column the ratio of N_{circ} of an algorithm that uses the Euler method ($p = 1$) with N_{circ} of an algorithm that uses a RKM of order p. In the third, fourth, and fifth columns we show $N_r^{(\delta)}$, the number of time steps $N_\tau^{(\delta)}$ and the number of different circuits. In the plot, we show $N_{circ}(p)$ plotted against p. The total number of circuit evaluations N_{circ} is calculated according to Eq. (3.62).

p	$N_{circ}(p)$	$N_{circ}(1)/N_{circ}(p)$	$N_r^{(\delta)}$	$N_\tau^{(\delta)}$	Circuits
1	1.12×10^{37}	1	1.15×10^{25}	9.56×10^{8}	9.80×10^{11}
2	2.63×10^{34}	4.28×10^{2}	3.93×10^{25}	3.26×10^{5}	6.68×10^{8}
3	6.33×10^{33}	1.78×10^{3}	9.57×10^{25}	2.15×10^{4}	6.61×10^{7}
4	4.39×10^{33}	2.56×10^{3}	1.98×10^{26}	5.41×10^{3}	2.22×10^{7}
5	1.00×10^{34}	1.12×10^{3}	6.78×10^{26}	2.40×10^{3}	1.48×10^{7}
6	1.11×10^{34}	1.01×10^{3}	1.14×10^{27}	1.36×10^{3}	9.76×10^{6}
7	2.60×10^{34}	4.33×10^{2}	3.06×10^{27}	9.22×10^{2}	8.50×10^{6}
8	5.90×10^{34}	1.91×10^{2}	7.61×10^{27}	6.88×10^{2}	7.75×10^{6}
9	1.33×10^{35}	84.69	1.82×10^{28}	5.47×10^{2}	7.29×10^{6}
10	4.92×10^{35}	22.87	6.48×10^{28}	4.63×10^{2}	7.59×10^{6}

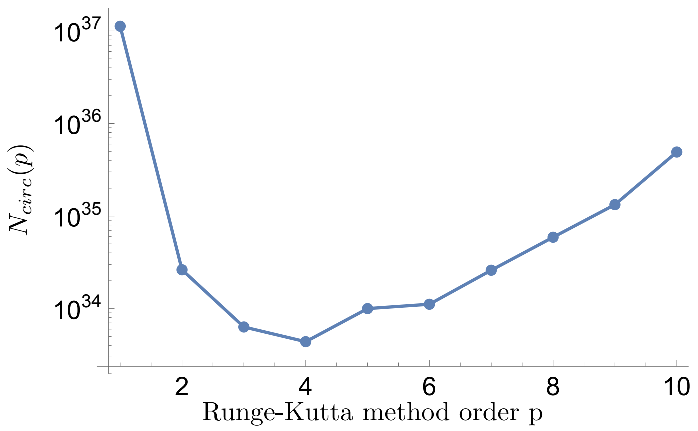


Figure 3.9: Comparison of N_{circ} for different RKM orders, where the estimated parameters are fine-tuned to maximize the savings by using a higher order p as given in Table 3.4. In the first column, we have the Runge-Kutta method order p. In the second column is N_{circ} of an algorithm that uses a Runge-Kutta method of order p and in the third column the ratio of N_{circ} of an algorithm that uses the Euler method ($p = 1$) with N_{circ} of an algorithm that uses a Runge-Kutta method of order p. In the third, fourth, and fifth columns, we show the number of shots per circuit $N_r^{(\delta)}$, the number of time steps $N_\tau^{(\delta)}$, and the number of circuits. In the plot, we show $N_{circ}(p)$ plotted against p. The total number of circuit evaluations N_{circ} is calculated according to Eq. (3.62).

3.7 Conclusions

In this chapter, we developed error and resource estimates of variational algorithms for solving differential equations based on Runge-Kutta methods. In particular, the estimates depend on different parameters that come from both the chosen method and the differential equation at hand, and depend on both

Parameter	**Estimate**
a_{max}	1
b_{max}	0.5
L_{fy}	0.1
T	4
K	20
$L_{f\tau}$	15
M	60
S	1
Σ	3.4×10^8
ϵ_{target}	0.001

Table 3.4: Estimates of various parameters that optimize the savings by using a higher-order Runge-Kutta method instead of the Euler method. Used for the analysis in Fig. 3.9.

the error from the Runge-Kutta method and the error that comes from shot noise in the noisy evaluations of the differential function. Additionally, we performed numerical simulations of the minimal resources required for both solving a simple ODE by using Runge-Kutta methods without shot noise and for solving a variational algorithm applied to a linear PDE from option pricing. This shows that our method is not restricted to ODEs but can also be applied to solving partial or stochastic differential equations. Furthermore, the algorithms we analyzed are based on the variational quantum algorithms for solving real and imaginary time evolution [116, 117]; our analysis can therefore be directly applied to them.

Our results suggest that depending on certain parameters, such as the difference between initial and final time, Lipschitz constants of the differential function, and target error, the resource requirements change drastically.

Moreover, we show that for the particular ODE that we solve in the case without shot noise and a chosen target error of at most $\epsilon_{target} = 0.001$, a fourth-order Runge-Kutta method is the most resource-efficient, while for the option pricing use case solved by a variational quantum algorithm and the same target error of $\epsilon_{target} = 0.001$, the most resource-efficient method is a second-order Runge-Kutta method. By choosing these methods, we can minimize the total cost by a factor of 2.22×10^3 for the first case and the total number of quantum circuit evaluations by a factor of 13.18 for the option pricing use case. For solving the option pricing PDE, even when using a Runge-Kutta method with the order of $p = 2$, we estimated that the algorithm needs at least 1.62×10^{28} evaluations of quantum circuits in order to compute the option price at final time with a maximum error of $\epsilon_{target} = 0.001$ in trace distance. In practical scenarios, the state has to be read out, and therefore the total resource requirements will be even higher.

However, our results are worst-case estimates that guarantee staying within the target error, while in practice, one might expect to achieve them with lower resource requirements for sufficiently well-behaved functions. In particular, Lemma 4 gives an upper bound for errors induced by solving linear systems. But even for ill-conditioned problems, a smaller error is possible in practice (see for example Ref. [158]). Also, the estimates in Secs. 3.5.3 and 3.6 are upper bounds that can in general be tightened for specific problems at hand.

The PDE chosen in our use case, the Black-Scholes equation, can be solved analytically, so only more complicated dynamics are of practical interest. Considering this and the fact that a requirement of at least 3.87×10^{22} evaluations for each of the 4.19×10^{5} quantum circuits (of which only 10^{3} can be run in parallel, see Sec. 3.5.1) is unrealistic on current quantum computing platforms, our results are rather pessimistic for practical implementations of this algorithm in option pricing. As discussed in the previous section, this high requirement of quantum circuit evaluations mainly stems from the error-propagation of inverting a matrix of estimated observables. This leads us to question the general feasibility of the underlying quantum algorithm. However, related approaches as proposed in Refs. [119] and [120] which do not require matrix inversion, might therefore require a much lower number of quantum circuit evaluations.

We show that by tuning the parameters in the variational algorithm and therefore reaching realms outside of this use case, we can get a saving factor of 2.56×10^{3} by choosing a Runge-Kutta method of order $p = 4$ instead of the Euler method. This suggests that after careful analysis of the parameters, one can choose a Runge-Kutta method that minimizes the resource requirements.

Our analysis had a number of limitations, which however likely do not make matters better. We do not analyze possible stability issues of the differential equations, because it proves very challenging to include them in the estimation of the variational algorithm and the option pricing use case. However, for a full picture of the error and resource analysis, the stability of methods and DEs has to be considered. We neglected possible representation errors that stem from the quantum circuits not being able to approximate a state that encodes the option price and which can capture the dynamics of the variational parameters satisfyingly. In practical scenarios, these have to be taken into account and can possibly be estimated depending on the chosen quantum computing platform (see for example Ref. [131]). Further, we assumed circuit error such as gate infidelity, bias and SPAM errors to be negligible, as well as errors introduced to potentially necessary matrix regularization of the matrix defined in Eq. (3.27). However, as long as when adding those sources of errors, our error bounds (for example in Eq. (3.84)) still hold, our results can be applied to these scenarios as well.

It might be possible to tighten our estimates in a few different ways. Recent works [135, 152] showed how it is possible to decrease the number of state preparations for estimations of the matrices A from scaling as $\Theta(N_V^2)$ to $\Theta(N_V)$ at the cost of accuracy of the matrix entries. Combining these results with our analysis might further decrease the total resource requirements. Several bounds in our analysis are not tight and could show to be overly pessimistic in real scenarios, like the local truncation error of the Runge-Kutta method (Thm. 3.1) and the bounds and estimates on the shot noise (Lemma 3 and Sec. 3.6.2). Further, the resource requirements can possibly be decreased by using linear multistep methods that only use one new stage per time step and reuse evaluations from previous time steps. Adapting our analysis to these methods can give new insights into a comparison of Runge-Kutta methods and linear multistep methods.

Since the shot noise error introduced for the analysis in Sec. 3.4 is Gaussian noise, it might give further insight to formulate the differential equations as stochastic differential equations and to analyze the error and resource requirements within this framework.

It might be promising to apply this framework to use cases that use Runge-Kutta methods and have similar error sources, such as quantum algorithms for solving other differential equations or to classical algorithms that are using noisy data, which so far has been barely examined. Also, it might be possible to apply this analysis to the training of neural networks [159].

3.A Proof of Theorems 3.2 and 3.4

Proof. Let us denote the RKM calculated $\boldsymbol{\theta}_n$ with an error-carrying $\hat{f}$ at time step n as $\hat{\boldsymbol{\theta}}_n$. Calculating the LTE in $\hat{\boldsymbol{\theta}}_n$, we assume that it is calculated from a noiseless $\boldsymbol{\theta}(\tau_{n-1})$. For an s-stage RKM $\hat{\boldsymbol{\theta}}_n$ is calculated by:

$$\hat{\boldsymbol{\theta}}_n = \boldsymbol{\theta}(\tau_{n-1}) + \Delta\tau \sum_{i=1}^{s} b_i \hat{k}_i \left(\tau_{n-1}; \boldsymbol{\theta}(\tau_{n-1}); \{\hat{k}_m\}_{m=1}^{i-1}\right) , \tag{3.98}$$

where we write

$$\hat{k}_1(\tau_{n-1}; \boldsymbol{\theta}(\tau_{n-1})) := \hat{f}\left(\tau_{n-1}; \boldsymbol{\theta}(\tau_{n-1})\right), \quad \text{and}$$

$$\hat{k}_i \left(\tau_{n-1}; \boldsymbol{\theta}(\tau_{n-1}); \{\hat{k}_m\}_{m=1}^{i-1}\right) := \hat{f}\left(\tau_{n-1} + c_i\Delta\tau; \boldsymbol{\theta}(\tau_{n-1}) + \Delta\tau \sum_{m=1}^{i-1} a_{i,m}\hat{k}_m\right),$$

for all $i \geq 2$,

and where we used the abbreviation $\hat{k}_m$ by dropping the arguments.

Thus, the LTE in the presence of noise is:

$$\hat{\ell}_n := \boldsymbol{\theta}(\tau_n) - \hat{\boldsymbol{\theta}}_n = \boldsymbol{\theta}(\tau_n) - \boldsymbol{\theta}(\tau_{n-1}) - \Delta\tau \sum_{i=1}^{s} b_i \hat{k}_i \left(\tau_{n-1}; \boldsymbol{\theta}(\tau_{n-1}); \{\hat{k}_m\}_{m=1}^{i-1}\right) \tag{3.99}$$

We can immediately write

$$\hat{\ell}_n \leq \boldsymbol{\theta}(\tau_n) - \boldsymbol{\theta}_n + \boldsymbol{\theta}_n - \hat{\boldsymbol{\theta}}_n \leq \ell_n + \boldsymbol{\theta}_n - \hat{\boldsymbol{\theta}}_n \ , \tag{3.100}$$

where the noiseless LTE ℓ_n can be upper bounded by the bound in Thm. 3.1.

Let us define the global truncation error of an RKM with s stages and n time steps in the presence of noise as:

$$\hat{e}_n := \boldsymbol{\theta}(\tau_n) - \hat{\boldsymbol{\theta}}_0 - \Delta\tau \sum_{r=1}^{n} \sum_{i=1}^{s} b_i \breve{k}_i \left(\tau_{r-1}; \hat{\boldsymbol{\theta}}_{r-1}; \{\breve{k}_m\}_{m=1}^{i-1}\right) \ , \tag{3.101}$$

where we write

$$\breve{k}_1 \left(\tau_{r-1}; \hat{\boldsymbol{\theta}}_{r-1}\right) := \breve{f}\left(\tau_{r-1}; \hat{\boldsymbol{\theta}}_{r-1}\right), \quad \text{and}$$

$$\breve{k}_i \left(\tau_{r-1}; \hat{\boldsymbol{\theta}}_{r-1}; \{\breve{k}_m\}_{m=1}^{i-1}\right) := \breve{f}\left(\tau_{r-1} + c_i\Delta\tau; \hat{\boldsymbol{\theta}}_{r-1} + \Delta\tau \sum_{m=1}^{i-1} a_{i,m}\breve{k}_m\right),$$

for all $i \geq 2$,

and again use the abbreviation $\breve{k}_m$ by dropping the arguments. We write the superscript $\breve{f}$ instead of $\hat{f}$ in order to distinguish the noisy evaluation with noisy inputs with the noisy evaluation with noiseless inputs. They both however carry the error δ compared to the noise-free case, as written in Eq. (3.46). Recursively, we get:

$$\hat{e}_{n+1} - \hat{e}_n = \hat{\ell}_{n+1} + \Delta\tau \sum_{i=1}^{s} b_i \left(\hat{k}_i \left(\tau_n; \boldsymbol{\theta}(\tau_n); \{\hat{k}_m\}_{m=1}^{i-1}\right) - \breve{k}_i \left(\tau_n; \hat{\boldsymbol{\theta}}_n; \{\breve{k}_m\}_{m=1}^{i-1}\right)\right) \ . \tag{3.102}$$

Taking the absolute values from the latter equation, we get

$$\|\hat{e}_{n+1}\| \leq \|\hat{e}_n\| + \|\hat{\ell}_{n+1}\| \tag{3.103}$$

$$+ \Delta\tau \sum_{i=1}^{s} |b_i| \left\| \hat{k}_i \left(\tau_n; \boldsymbol{\theta}(\tau_n); \{\hat{k}_m\}_{m=1}^{i-1}\right) - \breve{k}_i \left(\tau_n; \hat{\boldsymbol{\theta}}_n; \{\breve{k}_m\}_{m=1}^{i-1}\right) \right\| \ . \tag{3.104}$$

We assume that the function $f(\tau, \boldsymbol{\theta}(\tau))$ satisfies the Lipschitz condition:

$$\left\| f(\tau_n, \boldsymbol{\theta}(\tau_n)) - f(\tau_n, \hat{\boldsymbol{\theta}}_n) \right\| \leq L_{fy} \|\boldsymbol{\theta}(\tau_n) - \hat{\boldsymbol{\theta}}_n\| \ , \tag{3.105}$$

and that the noise evaluations are upper bounded by

$$\|\hat{k}_i - k_i\| \leq \delta \tag{3.106}$$

$$\|\breve{k}_i - k_i\| \leq \delta \ . \tag{3.107}$$

We introduce the notation

$$S_i := \left\| \hat{k}_i \left(\tau_n; \boldsymbol{\theta}(\tau_n); \{\hat{k}_m\}_{m=1}^{i-1} \right) - \breve{k}_i \left(\tau_n; \hat{\boldsymbol{\theta}}_n; \{\breve{k}_m\}_{m=1}^{i-1} \right) \right\| . \tag{3.108}$$

Using the triangle inequality, we get

$$S_1 = \left\| \hat{k}_1 \left(\tau_n; \boldsymbol{\theta}(\tau_n) \right)) - \breve{k}_1 \left(\tau_n; \hat{\boldsymbol{\theta}}_n \right) \right\| \tag{3.109}$$

$$\leq \left\| \hat{k}_1 \left(\tau_n; \boldsymbol{\theta}(\tau_n) \right)) - k_1 \left(\tau_n; \boldsymbol{\theta}(\tau_n) \right)) \right\| + \left\| k_1 \left(\tau_n; \hat{\boldsymbol{\theta}}_n \right) - \breve{k}_1 \left(\tau_n; \hat{\boldsymbol{\theta}}_n \right) \right\|$$

$$+ \left\| k_1 \left(\tau_n; \boldsymbol{\theta}(\tau_n) \right)) - k_1 \left(\tau_n; \hat{\boldsymbol{\theta}}_n \right) \right\| \tag{3.110}$$

$$\leq 2\delta + L_{fy} \|\hat{e}_n\|. \tag{3.111}$$

Again using the triangle inequality, we get for S_i, $i \geq 2$:

$$S_i = \left\| \hat{k}_i \left(\tau_n; \boldsymbol{\theta}(\tau_n); \{\hat{k}_m\}_{m=1}^{i-1} \right) - \breve{k}_i \left(\tau_n; \hat{\boldsymbol{\theta}}_n; \{\breve{k}_m\}_{m=1}^{i-1} \right) \right\| \tag{3.112}$$

$$\leq \left\| \hat{k}_i \left(\tau_n; \boldsymbol{\theta}(\tau_n); \{\hat{k}_m\}_{m=1}^{i-1} \right) - k_i \left(\tau_n; \boldsymbol{\theta}(\tau_n); \{\hat{k}_m\}_{m=1}^{i-1} \right) \right\| \tag{3.113}$$

$$+ \left\| k_i \left(\tau_n; \hat{\boldsymbol{\theta}}_n; \{\breve{k}_m\}_{m=1}^{i-1} \right) - \breve{k}_i \left(\tau_n; \hat{\boldsymbol{\theta}}_n; \{\breve{k}_m\}_{m=1}^{i-1} \right) \right\| \tag{3.114}$$

$$+ \left\| k_i \left(\tau_n; \boldsymbol{\theta}(\tau_n); \{\hat{k}_m\}_{m=1}^{i-1} \right) - k_i \left(\tau_n; \hat{\boldsymbol{\theta}}_n; \{\breve{k}_m\}_{m=1}^{i-1} \right) \right\| \tag{3.115}$$

$$\leq 2\delta + L_{fy} \left\| \boldsymbol{\theta}(\tau_n) - \hat{\boldsymbol{\theta}}_n \right\| + \Delta\tau \sum_{m=1}^{i-1} L_{fy} |a_{im}| \left\| \hat{k}_m - \breve{k}_m \right\| \tag{3.116}$$

$$\leq 2\delta + L_{fy} \|\hat{e}_n\| + \Delta\tau \sum_{m=1}^{i-1} L_{fy} |a_{im}| S_m \tag{3.117}$$

$$\leq S_1 + \Delta\tau \sum_{m=1}^{i-1} L_{fy} |a_{im}| S_m \ . \tag{3.118}$$

Then, we can write the upper bound of the error of the s order RKM as

$$\|\hat{e}_{n+1}\| \leq \|\hat{e}_n\| + \|\hat{\ell}_{n+1}\| + \Delta\tau \sum_{i=1}^{s} |b_i| S_i \ , \tag{3.119}$$

where from the above analysis, we obtained the following recursion:

$$S_1 \leq 2\delta + L_{fy}\|\hat{e}_n\| \tag{3.120}$$

$$S_i \leq S_1 + \Delta\tau \sum_{m=1}^{i-1} L_{fy}|a_{im}|S_m, \quad \forall i \geq 2. \tag{3.121}$$

Let us use that $|a_{ij}|$ is upper bounded with $a_{max} := \max_{i,j} |a_{i,j}|$. Then we can write:

$$S_1 \leq 2\delta + L_{fy}\|\hat{e}_n\| \tag{3.122}$$

$$S_i \leq S_1 + \Delta\tau L_{fy} a_{max} \sum_{m=1}^{i-1} S_m, \quad \forall i \geq 2. \tag{3.123}$$

A proof by induction shows that for all $i \geq 1$,

$$S_i \leq S_1 \left(1 + \Delta\tau L_{fy} a_{max}\right)^{i-1}. \tag{3.124}$$

This result together with Eq. (3.119) gives us the error estimate:

$$\begin{aligned}
\|\hat{e}_{n+1}\| &\leq \|\hat{e}_n\| + \|\hat{\ell}_{n+1}\| + \Delta\tau S_1 \sum_{i=1}^{s} |b_i| \left(1 + \Delta\tau L_{fy} a_{max}\right)^{i-1} &(3.125)\\
&\leq \|\hat{e}_n\| + \|\hat{\ell}_{n+1}\| + \Delta\tau S_1 b_{max} \frac{\left(1 + \Delta\tau L_{fy} a_{max}\right)^{s} - 1}{L_{fy}\Delta\tau a_{max}} &(3.126)\\
&\leq \|\hat{e}_n\| + \left(\max_n \|\hat{\ell}_n\| + \left(2\delta + L_{fy}\|\hat{e}_n\|\right) b_{max} \frac{\left(1 + \Delta\tau L_{fy} a_{max}\right)^{s} - 1}{L_{fy} a_{max}}\right)\\
&\leq \alpha\|\hat{e}_n\| + \beta, \quad \text{for all} \quad s \geq 1, &(3.127)
\end{aligned}$$

where we used the notation

$$\alpha = 1 + F(n+1, s), \quad \beta = \frac{2\delta}{L_{fy}} F(n+1, s) + \max_n \|\hat{\ell}_n\|, \tag{3.128}$$

$$F(n+1, s) := \frac{b_{max}}{a_{max}} \left(\left(1 + \frac{\Theta}{n+1}\right)^{s} - 1\right), \quad \Theta = L_{fy} a_{max} T, \tag{3.129}$$

an the fact that $\Delta\tau = T/(n+1)$ holds. . Because $\|\hat{e}_0\| = 0$, we get

$$\|\hat{e}_{n+1}\| \leq \frac{\alpha^{n+1}-1}{\alpha-1}\beta. \tag{3.130}$$

The error estimate of the LTE goes analogous to the error estimate of the global truncation error. We can write

$$\|\hat{\ell}_n\| \leq \|\ell_n\| + \Delta\tau \sum_{i=1}^{s} |b_i| T_i \tag{3.131}$$

with the recursive relation

$$T_1 := \delta, \quad T_i := T_1 + \Delta\tau \sum_{m=1}^{i-1} L_{fy} |a_{im}| T_m. \tag{3.132}$$

Thus, we get the bounds

$$\|\hat{\ell}_n\| \leq \|\ell_n\| + \frac{\delta}{L_{fy}} F(n+1, s), \quad \text{for } s \geq 1. \tag{3.133}$$

Using these upper bounds, we can rewrite Eq. (3.130) as follows

$$\|\hat{e}_{n+1}\| \leq \frac{\alpha^{n+1}-1}{\alpha-1}\beta^* \ , \tag{3.134}$$

where

$$\beta^* = \frac{3\delta}{L_{fy}} F(n+1, s) + \max_n \|\ell_n\| \tag{3.135}$$

According to Thm. 3.1, we can bound the LTE ℓ_n and get after $N_t := n+1$ steps the following error:

$$\|\hat{e}_{N_\tau}\| \leq \frac{(1+F(N_\tau, s))^{N_\tau}-1}{F(N_\tau, s)} \left(\frac{3\delta}{Lfy} F(N_\tau, s) + \left(\frac{T}{N_\tau}\right)^{p+1} K L_{f\tau}^p M \right). \tag{3.136}$$

That ends the proof for Thm. 3.4, for which $f(\tau, \boldsymbol{\theta}(\tau)) = f(\boldsymbol{\theta}(\tau))$. One can see that noise free case corresponds to $\delta = 0$, for which $y(\tau_n)$ corresponds to $\boldsymbol{\theta}(\tau_n)$ and y_n to $\boldsymbol{\theta}_n$, which proofs Thm. 3.2. $\square$

3.B Proof of Theorem 3.3

Proof. Let us denote the upper bound of the noise free $\epsilon^{(\delta)}_{ODE}$ as

$$\epsilon^{(0)}_{target} := \frac{(1+F(N_\tau,s))^{N_\tau}-1}{F(N_\tau,s)}\left(\frac{T}{N_\tau}\right)^{p+1} KL^p_{f\tau}M\ , \tag{3.137}$$

To find N_τ we can solve the latter equation numerically. However, under the reasonable assumption that $\Theta << N_\tau$, we can use the following approximations

$$\left(\frac{\Theta}{N_\tau}+1\right)^s = 1+\frac{s\Theta}{N_\tau}+O\left(\frac{\Theta^2}{N_\tau^2}\right), \tag{3.138}$$

$$\left(\frac{b_{max}(s\Theta)}{a_{max}N_\tau}+1\right)^{N_\tau} \approx \exp\left(\frac{b_{max}s\Theta}{a_{max}}\right), \tag{3.139}$$

to rewrite

$$F(N_\tau,s)\approx\frac{b_{max}}{a_{max}}\frac{s\Theta}{N_\tau},\quad (1+F(N_\tau,s))^{N_\tau}\approx\exp\left(\frac{b_{max}s\Theta}{a_{max}}\right) \tag{3.140}$$

Using this, we rewrite Eq. (3.137) in a way to obtain an approximate solution $N^{(0)}_\tau$ given by Eq. (3.41). □

3.C Proof of Theorem 3.5

Proof. We model the shot noise with single shot variance Σ and the number of shots N_r, so $\delta=\Sigma/\sqrt{N_r}$ holds. Solving the resulting Eq. (3.48) for N_r we get:

$$N^{(\delta)}_r := \frac{9\Sigma^2}{L^2_{fy}}\left(\frac{\epsilon^{(\delta)}_{target}}{(1+F(N_\tau,s))^{N_\tau}-1}-\left(\frac{T}{N_\tau}\right)^{p+1}\frac{KL^p_{f\tau}M}{F(N_\tau,s)}\right)^{-2}, \tag{3.141}$$

Substituting it in the cost function (3.35) we get:

$$\mathcal{C}(N_\tau,N^{(\delta)}_r,s,p) = \frac{9\Sigma^2 p}{L^2_{fy}}N_\tau^{2p+3} \tag{3.142}$$

$$\times\left(\frac{F(N_\tau,s)((1+F(N_\tau,s))^{N_\tau}-1)}{\epsilon^{(\delta)}_{target}F(N_\tau,s)N_\tau^{p+1}-T^{p+1}KL^p_{f\tau}M((1+F(N_\tau,s))^{N_\tau}-1)}\right)^2,$$

where

$$\epsilon^{(\delta)}_{target} \neq \frac{T^{p+1} K L^p_{f\tau} M((1+F(N_\tau,s))^{N_\tau} - 1)}{F(N_\tau,s) N_\tau^{p+1}} = \epsilon^{(0)}_{target}, \tag{3.143}$$

that is always true. We want to find the optimal value for N_τ that minimizes the latter cost. To this end, we set the derivative of the cost function with respect to N_τ to zero. The equation we get in nonlinear and can be solved numerically.

However, we can use the approximation in Eq. (3.139) and the expression

$$N_\tau \log\left(\frac{b_{max} s\Theta}{a_{max} N_\tau} + 1\right) = \log\left(\frac{b_{max} s\Theta}{a_{max} N_\tau} + 1\right)^{N_\tau} \approx \left(\frac{b_{max} s\Theta}{a_{max}}\right). \tag{3.144}$$

Those two approximations give the following simplified equation

$$\frac{\epsilon^{(\delta)}_{target}\left(a_{max} N_\tau (N_\tau + \Theta) - b_{max} s\Theta\left(2N_\tau^2 e^{\frac{b_{max} s\Theta}{a_{max}}} + N_\tau\left(2s\Theta e^{\frac{b_{max} s\Theta}{a_{max}}} - 1\right) - \Theta\right)\right)}{(N_\tau + \Theta)(a_{max} N_\tau + b_{max} s\Theta)\left(e^{\frac{b_{max} s\Theta}{a_{max}}} - 1\right)}$$
$$+ \frac{2\epsilon^{(\delta)}_{target}(b_{max} s\Theta) e^{\frac{b_{max} s\Theta}{a_{max}}}}{a_{max}\left(e^{\frac{b_{max} s\Theta}{a_{max}}} - 1\right)} - \frac{a_{max} K M L^p_{f\tau}\left(\frac{T}{N_\tau}\right)^{p+1}(2N_\tau p + N_\tau - 2(s-1)\Theta)}{b_{max} s\Theta}$$
$$= 0 .$$

Neglecting some of the terms due to the fact that $\Theta << N_\tau$ we simplify the equation to the form that it can be solved analytically:

$$N^{(\delta)}_\tau = T L_{f\tau}\left(\frac{\left(e^{b_{max} s L_{fy} T} - 1\right) K M (2p+1)}{s b_{max} L_{fy} \epsilon^{(\delta)}_{target}}\right)^{1/p}. \tag{3.145}$$

That ends the proof. $\square$

3.D McLachlan's variational principle

Theorem 3.6. *McLachlan's variational principle [149] applied to the imaginary time evolution of a state $|\psi(\boldsymbol{\theta}(\tau))\rangle$, is given by:*

$$\delta \| (d/d\tau + \mathcal{H}) |\psi(\theta(\tau))\rangle \| = 0 . \tag{3.146}$$

Assuming θ to be real, it is solved by the following ordinary differential equations [150]:

$$\sum_j A_{ij}\dot{\theta}_j = C_i, \tag{3.147}$$

where the matrix elements are

$$A_{ij} = \mathfrak{Re}\left(\frac{\partial \langle\psi(\theta(\tau))|}{\partial\theta_i}\frac{\partial |\psi(\theta(\tau))\rangle}{\partial\theta_j}\right), \tag{3.148}$$

$$\textit{and} \quad C_i = \mathfrak{Re}\left(-\frac{\partial \langle\psi(\theta(\tau))|}{\partial\theta_i}\mathcal{H}|\psi(\theta(\tau))\rangle\right). \tag{3.149}$$

Proof. Note that the variation in Eq. (3.146) is equivalent to the variation of its square $\delta\|(d/d\tau+\mathcal{H})|\psi(\theta(\tau))\rangle\|^2 = 0$. The derivative can be written as follows

$$\frac{\partial|\psi(\theta(\tau))\rangle}{\partial\tau} = \frac{\partial|\psi(\theta(\tau))\rangle}{\partial\theta_i}\frac{\partial\theta_i}{\partial\tau} = \frac{\partial|\psi(\theta(\tau))\rangle}{\partial\theta_i}\dot{\theta}_i. \tag{3.150}$$

Next, we expand the following:

$$\begin{aligned}
\|(d/d\tau+\mathcal{H})|\psi(\theta(\tau))\rangle\|^2 &= \big((d/d\tau+\mathcal{H})|\psi(\theta(\tau))\rangle\big)^\dagger\big((d/d\tau+\mathcal{H})|\psi(\theta(\tau))\rangle\big) \\
&= \sum_{ij}\frac{\partial\langle\psi(\theta(\tau))|}{\partial\theta_i}\frac{\partial|\psi(\theta(\tau))\rangle}{\partial\theta_j}\dot{\theta}_i^*\dot{\theta}_j + \sum_i\frac{\partial\langle\psi(\theta(\tau))|}{\partial\theta_i}(\mathcal{H})|\psi(\theta(\tau))\rangle\dot{\theta}_i^* \\
&+ \sum_i\langle\psi(\theta(\tau))|\mathcal{H}\frac{\partial|\psi(\theta(\tau))\rangle}{\partial\theta_i}\dot{\theta}_i + \langle\psi(\theta(\tau))|\mathcal{H}^2|\psi(\theta(\tau))\rangle\ .
\end{aligned} \tag{3.151}$$

Thus, the variation of this term with respect to $\dot{\theta}_i$ yields:

$$\begin{aligned}
&\delta\|(d/d\tau+\mathcal{H})|\psi(\theta(\tau))\rangle\|^2 \\
&= \left(\sum_{ij}\frac{\partial\langle\psi(\theta(\tau))|}{\partial\theta_i}\frac{\partial|\psi(\theta(\tau))\rangle}{\partial\theta_j}\dot{\theta}_j + \frac{\partial\langle\psi(\theta(\tau))|}{\partial\theta_i}\mathcal{H}|\psi(\theta(\tau))\rangle\right)\delta\dot{\theta}_i^* \\
&+ \left(\sum_{ij}\frac{\partial\langle\psi(\theta(\tau))|}{\partial\theta_j}\frac{\partial|\psi(\theta(\tau))\rangle}{\partial\theta_i}\dot{\theta}_j^* + \langle\psi(\theta(\tau))|\mathcal{H}\frac{\partial|\psi(\theta(\tau))\rangle}{\partial\theta_i}\right)\delta\dot{\theta}_i\ .
\end{aligned} \tag{3.152}$$

The variational principle is satisfied if latter equation is equal to zero, hence when following expression holds:

$$\sum_j \frac{\partial \langle\psi(\theta(\tau))|}{\partial\theta_i}\frac{\partial|\psi(\theta(\tau))\rangle}{\partial\theta_j}\dot{\theta}_j = -\frac{\partial\langle\psi(\theta(\tau))|}{\partial\theta_i}\mathcal{H}|\psi(\theta(\tau))\rangle \ . \tag{3.153}$$

If $\dot{\theta}_i$ is real, the variation changes to

$$\begin{aligned}
&\delta\|(d/d\tau+\mathcal{H})|\psi(\theta(\tau))\rangle\|^2 \\
&=\sum_j\left(\frac{\partial \langle\psi(\theta(\tau))|}{\partial\theta_i}\frac{\partial|\psi(\theta(\tau))\rangle}{\partial\theta_j}+\frac{\partial \langle\psi(\theta(\tau))|}{\partial\theta_j}\frac{\partial|\psi(\theta(\tau))\rangle}{\partial\theta_i}\right)\dot{\theta}_j\delta\dot{\theta}_i \\
&\quad+\left(\frac{\partial\langle\psi(\theta(\tau))|}{\partial\theta_i}\mathcal{H}|\psi(\theta(\tau))\rangle+\langle\psi(\theta(\tau))|\mathcal{H}\frac{\partial|\psi(\theta(\tau))\rangle}{\partial\theta_i}\right)\delta\dot{\theta}_i \ .
\end{aligned} \tag{3.154}$$

This is equal to zero when the following holds:

$$\begin{aligned}
&\sum_j\left(\frac{\partial \langle\psi(\theta(\tau))|}{\partial\theta_i}\frac{\partial|\psi(\theta(\tau))\rangle}{\partial\theta_j}+\frac{\partial \langle\psi(\theta(\tau))|}{\partial\theta_j}\frac{\partial|\psi(\theta(\tau))\rangle}{\partial\theta_i}\right)\dot{\theta}_j \\
&=-\left(\frac{\partial\langle\psi(\theta(\tau))|}{\partial\theta_i}\mathcal{H}|\psi(\theta(\tau))\rangle+\langle\psi(\theta(\tau))|\mathcal{H}\frac{\partial|\psi(\theta(\tau))\rangle}{\partial\theta_i}\right) \ ,
\end{aligned} \tag{3.155}$$

which is equivalent to Eq. (3.147). □

3.E Shot noise estimates

Theorem 3.7. *(Shot noise error for evaluating A defined in Eq. (3.27))*
For N_r evaluations of each of the circuits that calculate the matrix elements of A, we get the following bound for the probability:

$$P\left(\|A-\hat{A}_S\| < \frac{\|\{\sigma_{k,l}\}_{k,l=1}^{N_V}\|}{\sqrt{N_r\eta}}\right) > 1-\eta \ , \tag{3.156}$$

where $0<\eta\leq 1$ and the elements of the standard deviation matrix are

$$\sigma_{k,l}=\sqrt{\sum_{i,j=1}^{N_d}|f^*_{k,i}f_{l,j}|^2} \ . \tag{3.157}$$

Proof. For a random matrix A with finite non-zero variance matrix σ^2 and expectation values matrix $\hat{A}_S$ the multi-dimensional Chebyshev's inequality

$$P(\|A - \hat{A}_S\| \geq k\|\sigma\|) \leq \frac{1}{k^2} \ , \tag{3.158}$$

holds, for any real number $k > 0$. If we measure each circuit N_r times, the mean value is calculated as

$$\hat{A}_S = \frac{1}{N_r}\sum_{m=1}^{N_r} \hat{A}_m \ , \tag{3.159}$$

where each $\hat{A}_m$ is the matrix calculated by evaluating each circuit one time. The $\sigma_{k,l}(\hat{A}_m)$ is the standard deviation for each matrix element of $\hat{A}_m$, and the total standard deviation for each element in $\hat{A}_S$ is given by

$$\sigma_{k,l}(\hat{A}_S) = \frac{\sigma_{k,l}(\hat{A}_m)}{\sqrt{N_r}} \ . \tag{3.160}$$

These are the elements that form the matrix $\sigma = \{\sigma_{k,l}(\hat{A}_m)\}_{k,l=1}^{N_V}$. Defining $\eta := 1/k^2$, we get therefore

$$P\left(\|A - \hat{A}_S\| < \frac{\|\sigma\|}{\sqrt{N_r \eta}}\right) > 1 - \eta \ . \tag{3.161}$$

We can further bound the elements of σ in the following way. Each matrix element $\sigma_{k,l}(\hat{A}_S)$ is in general evaluated by several circuits. Let us bound the standard deviation of each single circuit by the maximum standard deviation 1, since the eigenvalues of the Pauli-X matrix are $\{1, -1\}$. We obtain:

$$\sigma_{k,l}(\hat{A}_m) \leq \sigma_{k,l} = \sqrt{\sum_{i,j=1}^{N_d} |f_{k,i}^* f_{l,j}|^2}. \tag{3.162}$$

That ends the proof. $\square$

Theorem 3.8. *(Shot noise error for evaluating C defined in Eq. (3.27))*
For N_r evaluations of each of the circuits that calculate the elements of C, we get the following bound:

$$P\left(\|C - \hat{C}_S\| < \frac{\|\{\sigma_k\}_{k=1}^{N_V}\|}{\sqrt{N_r \eta}}\right) > 1 - \eta \ , \tag{3.163}$$

where $0 < \eta \leq 1$ and

$$\sigma_k = \sqrt{\sum_{i=1}^{N_d} \sum_{m=1}^{N} |f_{i,k}^* \lambda_m|^2} \ , \tag{3.164}$$

holds.

Proof. The proof is similar to the proof of Thm. 3.7, but the standard deviation of the matrix elements is bounded by Eq. (3.164). □

Corollary 3. *The result of the Thms. 3.7 and 3.8 are valid for all matrices A and vectors C that are calculated with the circuits in Fig. 3.1, in particular for all possible input parameter $\boldsymbol{\theta}$ of the Ansatz in Eq.* (3.18)*. Therefore, we can state that with probability of at least $1-\eta$ the following bounds hold for all $\boldsymbol{\theta}$:*

$$\|A(\boldsymbol{\theta}) - \hat{A}(\boldsymbol{\theta})\| \leq \frac{\|\{\sigma_{k,l}\}_{k,l=1}^{N_V}\|}{\sqrt{N_r \eta}} \ , \tag{3.165}$$

and

$$\|C(\boldsymbol{\theta}) - \hat{C}(\boldsymbol{\theta})\| \leq \frac{\|\{\sigma_k\}_{k=1}^{N_V}\|}{\sqrt{N_r \eta}} \ . \tag{3.166}$$

3.F Error bound theorems

Proof of Lemma4: Let us identify $\hat{f}$ with $f(\xi)$, the vector under disturbance ξ. Accordingly, $f(0) = f$. Assume that the derivative $\frac{\partial f(\xi)}{\partial \xi}$ exists and calculate the derivative of Eq. (3.73) with respect to ξ:

$$(A + \xi R)\frac{\partial f(\xi)}{\partial \xi} + R f(\xi) = r \tag{3.167}$$

At $\xi = 0$, we have:

$$\left.\frac{\partial f(\xi)}{\partial \xi}\right|_{\xi=0} = A^{-1}(r - R f(0)) \ . \tag{3.168}$$

The Taylor expansion of $f(\xi)$ around $\xi = 0$ reads

$$f(\xi) = f(0) + \xi \left.\frac{\partial f(\xi)}{\partial \xi}\right|_{\xi=0} + \mathcal{O}(\xi^2) \tag{3.169}$$

$$= f(0) + \xi A^{-1}(r - R f(0)) + \mathcal{O}(\xi^2) \tag{3.170}$$

Therefore, we can estimate

$$\begin{aligned}
\frac{\|f(\xi)-f(0)\|}{\|f(0)\|} &\leq \xi\frac{\|A^{-1}(r-Rf(0))\|}{\|f(0)\|}+\mathcal{O}(\xi^2) && (3.171)\\
&\leq \xi\frac{\|A^{-1}\|\|r-Rf(0)\|}{\|f(0)\|}+\mathcal{O}(\xi^2) && (3.172)\\
&\leq \xi\|A^{-1}\|\frac{\|r\|+\|R\|\|f(0)\|}{\|f(0)\|}+\mathcal{O}(\xi^2) && (3.173)\\
&\leq \xi\|A^{-1}\|\left(\frac{\|r\|}{\|f(0)\|}+\|R\|\right)+\mathcal{O}(\xi^2) && (3.174)\\
&\leq \xi\|A^{-1}\|\|A\|\left(\frac{\|r\|}{\|A\|\|f(0)\|}+\frac{\|R\|}{\|A\|}\right)+\mathcal{O}(\xi^2) && (3.175)\\
&\leq \xi\|A^{-1}\|\|A\|\left(\frac{\|r\|}{\|Af(0)\|}+\frac{\|R\|}{\|A\|}\right)+\mathcal{O}(\xi^2) && (3.176)\\
&\leq \xi\kappa(A)\left(\frac{\|r\|}{\|C\|}+\frac{\|R\|}{\|A\|}\right)+\mathcal{O}(\xi^2)\ . && (3.177)
\end{aligned}$$

□

Proof of Lemma 2: The gradient $\nabla_{\boldsymbol{\theta}}\phi(\boldsymbol{\theta}(\tau))$ evaluates in the chosen circuit as

$$\frac{\partial\phi(\theta(\tau))}{\partial\theta_k}=\sum_{j=1}^{N_d}\left(f_{k,j}R_{k,j}\left|0\right\rangle^{\otimes n}\left\langle\phi(\theta(\tau))\right|+\left|\phi(\theta(\tau))\right\rangle\left\langle 0\right|^{\otimes n}R_{k,j}^{\dagger}f_{k,j}^{\dagger}\right)\ ,$$

where we used Eq. (3.26). Then the trace norm of the dot product with the parameter vector $\boldsymbol{\theta}^*(T)$ evaluates as

$$\begin{aligned}
&\|\nabla_{\boldsymbol{\theta}}\phi(\boldsymbol{\theta}_{\mathbf{0}}(T))\cdot\boldsymbol{\theta}^*(T)\|_1 && (3.178)\\
&=\left\|\sum_{k=1}^{N_V}\left[\sum_{j=1}^{N_d}\left(f_{k,j}R_{k,j}\left|0\right\rangle^{\otimes n}\left\langle\phi(\boldsymbol{\theta}_{\mathbf{0}}(T))\right|+\left|\phi(\boldsymbol{\theta}_{\mathbf{0}}(T))\right\rangle\left\langle 0\right|^{\otimes n}R_{k,j}^{\dagger}f_{k,j}^{\dagger}\right)\right]\theta_k^*(T)\right\|_1\\
&\leq\sum_{k=1}^{N_V}\left\|\left[\sum_{j=1}^{N_d}\left(f_{k,j}R_{k,j}\left|0\right\rangle^{\otimes n}\left\langle\phi(\boldsymbol{\theta}_{\mathbf{0}}(T))\right|+\left|\phi(\boldsymbol{\theta}_{\mathbf{0}}(T))\right\rangle\left\langle 0\right|^{\otimes n}R_{k,j}^{\dagger}f_{k,j}^{\dagger}\right)\right]\theta_k^*(T)\right\|_1\\
&\leq\sum_{k=1}^{N_V}\left[\sum_{j=1}^{N_d}\left(|f_{k,j}|\left\|R_{k,j}\left|0\right\rangle^{\otimes n}\left\langle\phi(\boldsymbol{\theta}_{\mathbf{0}}(T))\right|\right\|_1\right.\right.\\
&\qquad\left.\left.+\left\|\left|\phi(\boldsymbol{\theta}_{\mathbf{0}}(T))\right\rangle\left\langle 0\right|^{\otimes n}R_{k,j}^{\dagger}\right\|_1|f_{k,j}^{\dagger}|\right)\right]|\theta_k^*(T)|
\end{aligned}$$

$$\leq \sum_{k=1}^{N_V} \left[\sum_{j=1}^{N_d} \left(|f_{k,j}| + |f_{k,j}^{\dagger}| \right) \right] |\theta_k^*(T)| = \sum_{k=1}^{N_V} \left[\sum_{j=1}^{N_d} 2|f_{k,j}| \right] |\theta_k^*(T)| \ ,$$

where in the third line, we used the fact that any quantum state has trace norm at most one. □

CHAPTER 4

Mitigating Shot Noise in Local Overlapping Quantum Tomography with Semidefinite Programming

4.1 Introduction

Accurately characterizing quantum states is a crucial subroutine for many tasks in quantum computing, and can be achieved by quantum state tomography methods [9, 160, 161]. It has applications ranging from certifying quantum devices [162] to the design and execution of variational quantum algorithms [21]. However, the full characterization of a quantum state becomes impractical for larger systems. This is due to an exponential growth in the number of required measurement settings, which correspond to distinct Pauli strings. To address this challenge, several alternative strategies have been proposed, such as classical shadow tomography [163], which significantly reduce the number of required measurement settings. However, even with these techniques, each measurement setting must typically be repeated for a number of shots, N_{meas}, to achieve the desired accuracy, with the statistical uncertainty scaling as $O(1/\sqrt{N_{meas}})$. Since the measurements collapse the quantum state, the state must be re-prepared for each shot.

Many applications, such as ground state optimization of local Hamiltonians [39, 164], rely on the tomography of k-qubit reduced density matrices

The contents of this chapter have been published in Ref. [90].

(RDMs), which describe a subsystem of the full quantum state [165]. The estimation of all k-qubit RDMs of an n-qubit system can be achieved by using $\binom{n}{k} \cdot 3^k$ distinct measurement settings [166–168]. However, even with exact estimates of RDMs, the full characterization of a general quantum state would entail resolving the quantum marginal problem [169], a task known to be QMA-complete [170, 171]. These extensive measurement requirements pose a bottleneck for applying variational quantum algorithms in scenarios requiring high-precision results, such as in fields like computational chemistry, where a precision of 10^{-3} Hartree [172] (chemical accuracy) is often the target.

Recent strategies like overlapping tomography [164, 168, 173] have emerged to minimize the total number of measurement settings while maintaining high precision. These methods achieve this by parallelizing measurements on non-overlapping subsystems and efficiently organizing information from measurements on overlapping subsystems. However, current methods, while promising, often overlook higher-order correlations and compatibility constraints among RDMs, potentially limiting their accuracy.

In this chapter, we propose a novel hierarchy of data-driven semidefinite programs (SDPs) to estimate a set of overlapping reduced density matrices (RDMs) from quantum measurements. Our approach focuses specifically on random Pauli string measurements of n-qubit states with fixed locality. We leverage the inherent higher-order correlations present in quantum state measurement data–information that would be lost when estimating overlapping RDMs independently. This enables us to tighten the uncertainty intervals of the RDM estimates for a given number of measurement shots, with particular advantages in low-shot regimes. Such SDP relaxations are built on re-imposing partial compatibility with the quantum marginal problem [171, 174, 175]. We further constrain each RDM to satisfy the physical requirements of a valid density matrix, namely unit trace and positive semidefiniteness. This comprehensive approach yields two key benefits: it resolves the compatibility issues between overlapping RDMs that arise in linear inversion [176], while simultaneously enhancing the global consistency of the entire set of RDMs.

Our method demonstrates superior performance compared to standard tomography methods across two numerical benchmarks. In the first evaluation, we assess the ability of our method to estimate ground-state RDMs and energies of the 1D XY model with open boundary conditions by using random Pauli string measurements. Under the same measurement budget, our approach achieves more accurate energy estimates compared to conventional tomography methods. We further validate our technique through application to algorithmic cooling [177–179], a practical use case in near-term quantum computing where RDMs inform quantum circuit design. In this application domain, our method again demonstrates measurable advantages over traditional approaches that rely on independent RDM reconstructions.

This chapter is organized as follows: Sec. 4.2 introduces quantum tomography, semidefinite programming, and related literature. Sec. 4.3 presents the proposed SDP-based reconstruction method and numerical results. In Sec. 4.4, we apply our method to algorithmic cooling. Finally, Sec. 4.5 provides conclusions and future directions.

4.2 Background and preliminaries

4.2.1 Notations

We consider an n-qubit system whose Hilbert space is $\mathcal{H} = (\mathbb{C}^2)^{\otimes n}$. For our mathematical representation, we outline the relevant definitions and notations below. Let $\sigma_{0,1,2,3}$ denote, respectively, the identity matrix $\mathbb{I}$ and the Pauli X, Y, Z matrices. For an n-qubit system, the identity matrix is denoted by $\mathbb{I}_n$. A vector $\mathbf{i} = (i_1, i_2, ..., i_n) \in \{0,1,2,3\}^n := \mathcal{I}_n$ is used to represent a specific Pauli basis $\sigma_{\mathbf{i}}$, which is a shorthand for the Pauli string $\sigma_{i_1} \otimes \sigma_{i_2} \otimes ... \otimes \sigma_{i_n}$, in $\{\mathbb{I}, X, Y, Z\}^n$. For simplicity, we use this notation throughout the text. Note that Pauli strings form an orthogonal basis of the $\mathbb{R}$-vector space of Hermitian matrices. We denote the set $\{1, ..., m\}$ by $[m]$. For a Hermitian matrix A, the notation $A \succeq 0$ indicates that A is positive semidefinite, i.e., all its eigenvalues are non-negative. We denote by ρ_{AB} the density matrix that describes a quantum state in the Hilbert space $\mathcal{H}_{AB} = \mathcal{H}_A \otimes \mathcal{H}_B$, where $A = \{a_1, \ldots, a_{|A|}\}$ and $B = \{b_1, \ldots, b_{|B|}\}$ label distinct sets of qubits in the respective subsystems. The partial trace over subsystem A is defined as:

$$\mathrm{Tr}_A\left[\rho_{AB}\right] := \sum_k (\langle k| \otimes \mathbb{I}_B)\rho_{AB}(|k\rangle \otimes \mathbb{I}_B), \tag{4.1}$$

where $\{|k\rangle\}$ forms an orthonormal basis of the Hilbert space $\mathcal{H}_A$.

4.2.2 Quantum state tomography

Quantum state tomography [9, 160, 161] is a cornerstone of quantum information science, enabling the reconstruction of quantum states through systematic measurements on an ensemble of identical quantum states. For an n-qubit quantum state, its density matrix, represented by ρ, can be fully characterized by the following relation:

$$\rho = \frac{1}{2^n}\left(\sum_{\mathbf{i}\in\mathcal{I}_n} C_{\mathbf{i}}\sigma_{\mathbf{i}}\right), \tag{4.2}$$

$$C_{\mathbf{i}} = \mathrm{Tr}[\rho\sigma_{\mathbf{i}}]. \tag{4.3}$$

Here, $C_{\mathbf{i}}$ constitutes an element of the corresponding n-qubit Bloch vector, with $C_{\mathbf{0}} = 1$. In practical settings, $C_{\mathbf{i}}$ is not directly accessible, instead an estimate $\hat{C}_{\mathbf{i}}$ is computed by averaging over a finite number of measurements $N_{\mathrm{meas},\mathbf{i}}$ performed in the basis $\sigma_{\mathbf{i}}$:

$$\hat{C}_{\mathbf{i}} = \frac{1}{N_{\mathrm{meas},\mathbf{i}}} \sum_{k=1}^{N_{\mathrm{meas},\mathbf{i}}} m_{\mathbf{i}}^{(k)}, \tag{4.4}$$

where $m_{\mathbf{i}}^{(k)} \in \{1, -1\}$ represents the k-th measurement outcome. In measuring a particular Pauli string $\sigma_{\mathbf{i}}$, one obtains a binary string $s_{\sigma_{\mathbf{i}}} = \{0,1\}^n$, from which the outcome

$$m_{\mathbf{i}}^{(k)} = (-1)^{|s_{\sigma_{\mathbf{i}}}|}, \tag{4.5}$$

is calculated, with $|s_{\sigma_{\mathbf{i}}}|$ denoting the string Hamming weight, defined as the total count of 1s in $s_{\sigma_{\mathbf{i}}}$. We denote the reconstructed state as $\hat{\rho}$, characterized by the $4^n - 1$ different $\{\hat{C}_{\mathbf{i}}\}_{\mathbf{i}\in\mathcal{I}_n\setminus\{\mathbf{0}\}}$. Due to redundancies [180], it is sufficient to simulate measurements in 3^n different bases, each corresponding to a specific Pauli string $\sigma_{\mathbf{i}}$ with $\mathbf{i} \in \{1,2,3\}^n$, to estimate all $\hat{C}_{\mathbf{i}}$. In essence, if a string contains a 0, then the corresponding qubits are not measured, which is equivalent to measuring the RDM of the complementary subsystem. The measurement data collected from the other measurements is sufficient for reconstructing this RDM. The reconstructed state $\hat{\rho}$ takes the form:

$$\hat{\rho} = \frac{1}{2^n}\left(\mathbb{I}_n + \sum_{\mathbf{i}\in\mathcal{I}_n\setminus\{\mathbf{0}\}} \frac{1}{N_{\mathrm{meas},\mathbf{i}}} \sum_{k=1}^{N_{\mathrm{meas},\mathbf{i}}} m_{\mathbf{i}}^{(k)} \sigma_{\mathbf{i}}\right). \tag{4.6}$$

We will refer to this method as the *standard quantum state tomography*. According to the Chernoff-Hoeffding bound, achieving an additive error ϵ in each $\hat{C}_{\mathbf{i}}$ w.r.t. the true values $C_{\mathbf{i}}$ requires on the order of $N_{\mathrm{meas}} \approx 4\log(2)n/\epsilon^2$ shots [168] for a constant failure probability. If each $\hat{C}_{\mathbf{i}}$ is estimated with N_{meas} shots, then $3^n\, N_{\mathrm{meas}}$ total measurements are required, because there are 3^n non-identity Pauli strings, rendering this standard quantum state tomography infeasible for larger n. For instance, for a 50-qubit system this would result in $3^{50} \approx 7.18 \times 10^{23}$ measurement settings, and even with the Google Sycamore chip's $4\mu s$ [157] readout time, performing only one shot per setting would take about 10^{10} years.

To ease this exponential scaling in the number of qubits n, various methods have been proposed. Most notably, classical shadow tomography [163] provides a way to extract certain few-body observables with a number of measurements that grows only logarithmically in the number of those observables. Other techniques, such as neural-network-based tomography [181], rely on sampling protocols and often presume certain structural features of the underlying system. Using global mutually unbiased bases on the full Hilbert space, rather than local Pauli product bases, the number of measurement settings required for quantum state tomography can be reduced to $2^n + 1$ [182, 183]. Furthermore, informationally complete measurements such as symmetric informationally complete positive operator-valued measures (SIC-POVMs) offer an alternative that can reduce the number of measurement settings, potentially down to a single setting, and can be combined with classical-shadow-based tomography schemes [184].

In Sec. 4.2.4, we will present the overlapping tomography method of [168], which can be applied for quantum state tomography of RDMs.

4.2.3 Semidefinite programming

A semidefinite programming (SDP) problem involves optimizing a linear function subject to constraints expressed as linear matrix equalities and inequalities. The feasible regions of an SDP are known as spectrahedra, and efficient algorithms such as interior point methods [185, 186] can be used to solve them. In its standard primal form, an SDP can be written as:

$$\begin{aligned} \min \quad & \langle C, X \rangle \\ \text{s.t.} \quad & \langle A_i, X \rangle = b_i, \quad i = 1, 2, ..., m, \\ & X \succeq 0. \end{aligned} \tag{4.7}$$

where the cost matrix C, the decision matrix X, and the constraint matrices A_i are Hermitian, b is a real vector, and $\langle \cdot, \cdot \rangle$ denotes the Hilbert-Schmidt inner product, defined as $\langle X, Y \rangle := \operatorname{Tr}\left[X^\dagger Y\right]$, with $X^\dagger$ being the Hermitian conjugate of X. SDPs are particularly well-suited for quantum information [174], as density matrices are positive semidefinite. For instance, the quantum marginal problem seeks to determine whether a global state on a full system exists that is consistent with specified marginal states on subsystems, represented as reduced density matrices (RDMs) [171, 174, 175]. This problem can also be formulated as an SDP, as illustrated by the following example.

Assume we have perfect knowledge of the RDMs ρ_{23}, ρ_{13}, and ρ_{12}, acting on subsystems $\mathcal{H}_{23}$, $\mathcal{H}_{13}$, and $\mathcal{H}_{12}$, respectively, and the goal is to find a global state ρ_{123} on $\mathcal{H}_{123}$ that is consistent with these RDMs:

$$\begin{aligned} \min_{\rho_{123}} & 0 \\ \text{s.t.} \quad & \rho_{123} \succeq 0, \quad \mathrm{Tr}\left[\rho_{123}\right] = 1, \\ & \mathrm{Tr}_1\left[\rho_{123}\right] = \rho_{23}, \\ & \mathrm{Tr}_2\left[\rho_{123}\right] = \rho_{13}, \\ & \mathrm{Tr}_3\left[\rho_{123}\right] = \rho_{12}. \end{aligned} \tag{4.8}$$

Here, the minimization of the constant 0 is purely a formalism to match the standard SDP form in Eq. (4.7). If the constraints cannot be satisfied, the SDP is deemed infeasible. Despite the apparent simplicity of compatibility SDPs like Eq. (4.8), the general problem of determining whether a state $\rho_{123\ldots n}$ exists is QMA-complete [170, 171], making it computationally intractable even for quantum computers.

To address this complexity, we relax the problem in our method by using standard quantum state tomography estimates and accounting for their error ranges. Additionally, reinforcement learning has been shown to improve the selection of constraints that enforce compatibility with the quantum marginal problem in reconstructed solutions [187].

4.2.4 Overlapping tomography and related literature

The standard quantum state tomography of a full quantum state requires a number of measurement settings that scales as 3^n, where n is the number of qubits [180]. However, if the goal is to estimate only the $\binom{n}{k}$ different k-qubit reduced density matrices (RDMs), each RDM requires only 3^k different measurement settings. The total number of measurements, however, includes an additional overhead due to the number of shots needed to estimate each measurement setting.

Several strategies have been proposed to optimize the selection of measurement settings and minimize the total number of measurements required. In [168], a method called *quantum overlapping tomography* was introduced for estimating k-qubit RDMs. The key idea is to measure each qubit individually in a chosen basis, allowing non-overlapping k-qubit subsystems to be measured in parallel. Overlapping k-qubit subsystems are then reconstructed through classical post-processing. This approach reduces the required number of measurement settings to at most $e^{\mathcal{O}(k)} \log^2(n)$. The measurement selection is determined by a family of hash functions (n, k), which partition the system into k subsystems, with all qubits in each subsystem measured in the same basis. All k-qubit RDMs can then be reconstructed from this carefully chosen dataset. In a similar way, partitioning-based methods [188] lead to a scaling of the number of measurement settings needed as $\mathcal{O}\left(3^k \log^{k-1} n\right)$.

However, in many practical scenarios, such as systems with k-local Hamiltonians and nearest-neighbor interactions, only k-geometrically-local RDMs—those involving neighboring qubits—are required. This further reduces the number of measurement settings needed to a constant scaling with respect to the number of qubits n [164]. Recent advances have shown, both theoretically and experimentally, that leveraging graph theory can optimize this method, allowing even k-qubit RDMs to be estimated with a number of measurement settings that remains constant in the number of qubits n [173].

Other approaches have focused on reducing the statistical uncertainty due to shot noise in various contexts, such as quantum state tomography [189], ground-state estimation [190], and calculations of fidelity and von Neumann entropy [191]. SDP has also been applied to quantum marginal problems for tasks like ground-state estimation [192, 193], incorporating tensor network methods [194], or imposing entropy constraints on RDMs [195].

Our method builds on the principle of measuring the entire quantum system via product measurements on single qubits in the Pauli bases and performing post-processing to estimate k-local RDMs. However, rather than focusing on optimizing the selection of measurement settings, we introduce physicality and consistency constraints on the estimated RDMs. This approach explicitly accounts for the shot noise arising from a limited number of shots per measurement setting, providing a clearer benchmark for practical applications.

4.3 SDP-assisted overlapping tomography

In this section, we introduce the SDP-assisted overlapping tomography framework for estimating local RDMs of n-qubit states described by local Hamiltonians. We begin by presenting the hypergraph representation of local Hamiltonians in Sec. 4.3.1. In Sec. 4.3.2, we detail the core methodology of SDP-assisted overlapping tomography. Finally, to illustrate its effectiveness, we apply the approach to the ground states of the 1D XY model and discuss the numerical results in Sec. 4.3.3.

4.3.1 Local Hamiltonian

The Hamiltonian of an n-qubit k-local system can be represented by a hypergraph $G = (V_G, E_G)$, where each vertex $v_i \in V_G$ corresponds to a qubit, and each hyperedge $e_j \in E_G$ (connecting up to k vertices) represents a local Hamiltonian term. Concretely, the Hamiltonian takes the form

$$H = \sum_{j=1}^{m} H_j. \tag{4.9}$$

where m is the number of local subsystems, growing polynomially with n, and each H_j acts non-trivially on at most k qubits (i.e., it is k-local). Fig. 4.1 illustrates the hypergraph representation of a k-local Hamiltonian in a many-body system.

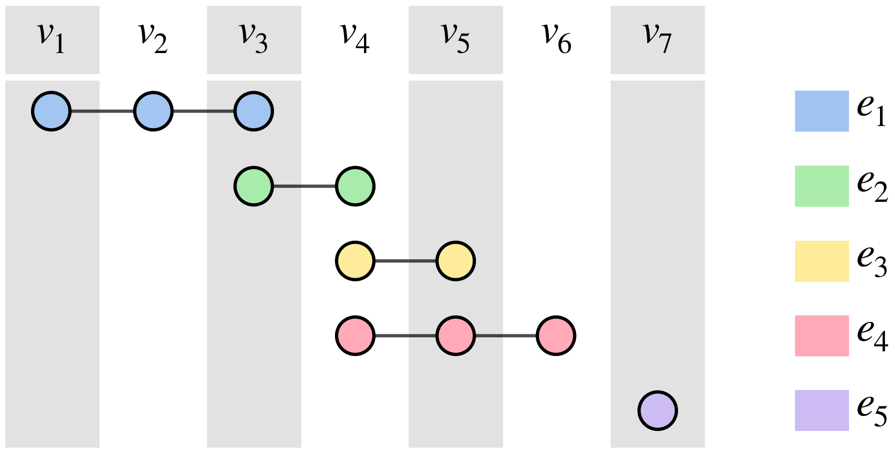


Figure 4.1: Hypergraph representation of the interaction structure of a Hamiltonian $H = \sum_{j=1}^{m} H_j$. The vertices $V_G = \{v_i\}_{i=1}^{n}$ correspond to $n = 7$ qubits. Each hyperedge $e_j \in E_G = \{e_j\}_{j=1}^{m}$ denotes a local Hamiltonian term H_j. In this example, E_G contains the sets $\{v_1, v_2, v_3\}$, $\{v_3, v_4\}$, $\{v_4, v_5\}$, $\{v_4, v_5, v_6\}$, $\{v_7\}$, hence $m = 5$. Hyperedges are shown as horizontal lines linking the relevant vertices, and the corresponding local Hamiltonian term is indicated in the legend on the right.

The hypergraph representation not only provides a visualization of the Hamiltonian structure but also gives a framework for reconstructing the relevant RDMs. Once measurement results are obtained from an ensemble of identical n-qubit states, one can reconstruct a collection of RDM estimates $\hat{\rho}_j$ for each local subsystem. This involves applying Eq. (4.6) to each of the m subsystems:

$$\left\{ \hat{\rho}_j := \frac{1}{2^{|e_j|}} \left(\mathbb{I}_{|e_j|} + \sum_{\mathbf{i} \in \mathcal{I}_{|e_j|} \setminus \{\mathbf{0}\}} \hat{C}_{\mathbf{i}}^{j} \sigma_{\mathbf{i}} \right) \right\}_{j=1}^{m} . \tag{4.10}$$

Here, $\hat{C}_{\mathbf{i}}^{j}$ is the estimator of $C_{\mathbf{i}}^{j} = \mathrm{Tr}\left[\mathrm{Tr}_{V_G \setminus e_j}[\rho] \sigma_{\mathbf{i}}\right] = \mathrm{Tr}[\rho_j \sigma_{\mathbf{i}}]$, where ρ is the true n-qubit global state and ρ_j is the true RDM of the j-th subsystem. Note that, each hyperedge $e_j = \{v_{j_1}, v_{j_2}, \ldots\}$ corresponds to the qubits on which H_j acts. These local RDMs are of particular interest because they are sufficient to estimate the energy expectation values of the local Hamiltonian:

$$\hat{E} = \sum_{j=1}^{m} \mathrm{Tr}[\hat{\rho}_j H_j]. \tag{4.11}$$

However, the reconstructed local RDMs $\{\hat{\rho}_j\}_{j=1}^m$ can be non-physical in practice; e.g., they may exhibit negative eigenvalues or fail to satisfy mutual compatibility (i.e., overlapping RDMs might not be consistent). While mutual compatibility can only be enforced approximately in real scenarios [187], imposing physicality constraints remains crucial for ensuring reliable energy estimates and other derived properties.

4.3.2 Methodology

In this section, we present the SDP-assisted overlapping tomography framework. Building on standard quantum tomography results, we formulate semidefinite programs (SDPs) that incorporate overlapping-compatibility (OC) and enhanced-compatibility (EC) constraints. These constraints aim to ensure the validity and consistency of the reconstructed states, allowing us to minimize or maximize the energy expectation within the feasible set. In essence, we determine the minimum and maximum energies compatible with the simulated data. By solving these SDP problems, we obtain a set of overlapping local RDMs that are more physically valid and mutually consistent.

Semidefinite relaxations

Relaxations of polynomial optimization problems based on semidefinite constraints play a central role in our method, in a similar spirit as Ref. [192, 196]. In this section, we describe the construction of these constraints.

Consider a local Hamiltonian associated with hypergraph $G = (V_G, E_G)$. For each hyperedge $e_j \in E_G$, we introduce the decision variables for the SDP problem as

$$\left\{ \tilde{\rho}_j := \frac{1}{2^{|e_j|}} \left(\mathbb{I}_{|e_j|} + \sum_{\mathbf{i} \in \mathcal{I}_{|e_j|} \setminus \{\mathbf{0}\}} \tilde{C}^j_{\mathbf{i}} \sigma_{\mathbf{i}} \right) \right\}_{j=1}^{m}, \tag{4.12}$$

where

$$\tilde{C}^j_{\mathbf{i}} \in \left[\hat{C}^j_{\mathbf{i}} - \epsilon^j_{\mathbf{i}}, \hat{C}^j_{\mathbf{i}} + \epsilon^j_{\mathbf{i}} \right], \quad \forall j \in [m],\ \forall \mathbf{i} \in \mathcal{I}_{|e_j|} \setminus \{\mathbf{0}\}. \tag{4.13}$$

Here, index j labels the local subsystem associated with the hyperedge e_j, and $\epsilon^j_{\mathbf{i}}$ is a relaxation variable associated to j-th local subsystem. It is proportional to the variance of $\hat{C}^j_{\mathbf{i}}$, scaled by a coefficient α, i.e., $\epsilon^j_{\mathbf{i}} = \alpha\,\mathrm{Var}(\hat{C}^j_{\mathbf{i}})$. The value of α relates to the probability that the unbiased estimator $\hat{C}^j_{\mathbf{i}}$ (of $\mathrm{Tr}[\rho_j \sigma_{\mathbf{i}}]$) lies within the chosen confidence interval, following Chernoff bound arguments. Throughout, tildes $(\tilde{\cdot})$ denote SDP decision variables, while hats $(\hat{\cdot})$ denote results obtained directly from the raw data during standard quantum state tomography.

Moreover, we introduce the semidefinite constraints

$$\tilde{\rho}_j \succeq 0, \quad \forall j \in [m], \tag{4.14}$$

and the overlapping-compatibility (OC) and enhanced-compatibility (EC) constraints

$$\mathscr{R} := \cup_{j,j' \in [m]} \mathcal{R}\left(\tilde{\rho}_j, \tilde{\rho}_{j'}\right), \tag{4.15}$$
$$\mathscr{G} := \cup_{j,j' \in [m]} \mathcal{G}\left(\tilde{\rho}_j, \tilde{\rho}_{j'}\right), \tag{4.16}$$

respectively. Specifically, $\mathcal{R}\big(\tilde{\rho}_j, \tilde{\rho}_{j'}\big)$ is the set of matrix equality constraints ensuring that the local RDMs are consistent on the intersection of their respective supports:

$$\mathcal{R}\left(\tilde{\rho}_j, \tilde{\rho}_{j'}\right) := \{\mathrm{Tr}_{e_j \setminus e_{j'}}\left[\tilde{\rho}_j\right] - \mathrm{Tr}_{e_{j'} \setminus e_j}\left[\tilde{\rho}_{j'}\right] = 0\}, \tag{4.17}$$

while $\mathcal{G}\big(\tilde{\rho}_j, \tilde{\rho}_{j'}\big)$ is the set of constraints that require the existence of a larger RDM $\tilde{\rho}$ whose marginals agree on the variables indexed by j and j':

$$\mathcal{G}\left(\tilde{\rho}_j, \tilde{\rho}_{j'}\right) := \ \{\tilde{\rho} \succeq 0, \mathrm{Tr}_{e_j}\left[\tilde{\rho}\right] = \tilde{\rho}_{j'}, \mathrm{Tr}_{e_{j'}}\left[\tilde{\rho}\right] = \tilde{\rho}_j\}. \tag{4.18}$$

Note that the dimension of $\tilde{\rho}$ can freely be chosen and that the definition of $\mathcal{G}$ in Eq. (4.18) is just one possible choice; in practice, there is a hierarchy of possible relaxations forming a partially ordered set. The optimal choice of $\mathcal{G}$ for a given computational budget is non-trivial, and one can achieve varying degrees of performance by judiciously selecting or refining these constraints [187, 197].

SDP problem formulation

The relaxations defined in the previous sub-section allow us to specify a feasible set of local RDMs. In this chapter, we focus on finding the minimum and maximum energies consistent with the simulated data used to estimate the tomographic local RDMs $\{\hat{\rho}_j\}$. Specifically, we consider the following optimization problem for determining the optimal state that minimizes (or maximizes) the energy under these constraints:

$$\begin{aligned} & \min(\max)_{\{\tilde{\rho}_j\}} \quad \sum_j \operatorname{Tr}\left[\tilde{\rho}_j H_j\right] \\ \text{s.t. } & \tilde{C}_{\mathbf{i}}^j \in \left[\hat{C}_{\mathbf{i}}^j - \epsilon_{\mathbf{i}}^j, \hat{C}_{\mathbf{i}}^j + \epsilon_{\mathbf{i}}^j\right], \ \forall j \in [m] \text{ and } \forall \mathbf{i} \in \mathcal{I}_{|e_j|} \setminus \{\mathbf{0}\}, \\ & \tilde{\rho}_j \succeq 0, \quad \forall j \in [m], \\ & \mathscr{R} \cup \mathscr{G}. \end{aligned} \tag{4.19}$$

Note that Eq. (4.19) does not yet incorporate all the SDP constraints derived from the quantum marginal problem needed to solve the above problem, as it only includes the initial-order EC constraints. Obtaining the exact solution requires higher-order EC constraints, but adding these leads to exponential growth in computational complexity [187, 197]. In large many-body systems, our formulation effectively discards higher-order constraints as n grows but k remains fixed, which loosens the energy bounds. Depending on the specific system, one may selectively include higher-level constraints that are most relevant for the problem at hand, here being the estimation of the minimum and maximum energy compatible with the measurement data [194]. Such strategies (especially the reinforcement-learning-based [187] and the renormalization-based [194] ones) can naturally be incorporated in our framework thereby allowing for even tighter estimates, but this would impair the fairness of the benchmarking of our method. Hence, in the rest of the chapter we proceed with the straightforward approach.

Because shot noise affects the data, setting $\epsilon_{\mathbf{i}}^j = 0$ (i.e., assuming no error) typically makes the SDP infeasible. Allowing a tolerance $\epsilon_{\mathbf{i}}^j$ around the estimated quantities provides a search region for solutions. As more data is collected, the ϵ-values shrink, thereby reducing the volume of the feasible set. Since the true state arises from a valid quantum system (satisfying the quantum marginal problem at all hierarchy levels), the resulting energy bounds become more accurate with increased data.

Studies on ground-state scenario

Semidefinite relaxations have long been used to investigate ground-state properties of many-body systems [192, 196]. We now examine the performance of SDP-assisted overlapping tomography in estimating ground states of local Hamiltonians.

Fig. 4.2 illustrates two feasible sets under the same search region but with or without EC constraints. These feasible sets are defined by all RDMs satisfying both the overlapping- ($\mathscr{R}$) and enhanced-compatibility ($\mathscr{G}$) constraints or only the overlapping ($\mathscr{R}$) constraints, respectively, while the search region refers to the intersection of positive semidefinite cones restricted by the confidence intervals in Eq. (4.13). Introducing EC constraints narrows the feasible set and

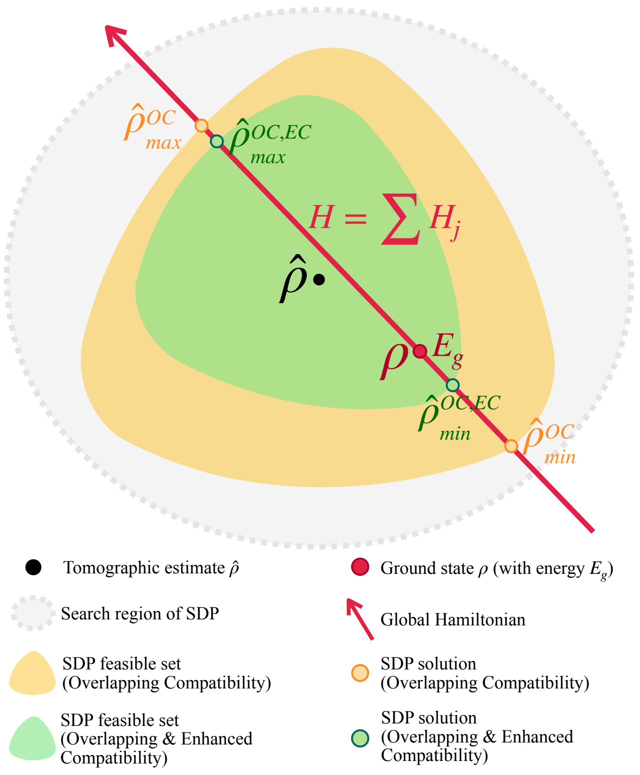

$\hat{\rho}^{OC}_{max}$
$\hat{\rho}^{OC,EC}_{max}$
$H = \sum H_j$
$\hat{\rho}$
ρ
E_g
$\hat{\rho}^{OC,EC}_{min}$
$\hat{\rho}^{OC}_{min}$
Tomographic estimate $\hat{\rho}$
Ground state ρ (with energy E_g)
Search region of SDP
Global Hamiltonian
SDP feasible set (Overlapping Compatibility)
SDP solution (Overlapping Compatibility)
SDP feasible set (Overlapping & Enhanced Compatibility)
SDP solution (Overlapping & Enhanced Compatibility)

Figure 4.2: Feasible set for the SDP problem. An n-qubit system with RDM ρ and energy E_g (shown in red) undergoes tomography, resulting in an approximate RDM $\hat{\rho}$ with energy $\hat{E}$ (black). The shaded gray region represents the (high-dimensional) search space. The yellow spectrahedron depicts the feasible set with only overlapping-compatibility (OC) constraints, whereas the green spectrahedron also includes enhanced-compatibility (EC).

yields tighter lower and upper bounds on ground-state energy estimate $\hat{E}_g$:

$$\mathrm{Tr}\left[H\hat{\rho}^{\mathrm{OC,EC}}_{\mathrm{min}}\right] \geq \mathrm{Tr}\left[H\hat{\rho}^{\mathrm{OC}}_{\mathrm{min}}\right], \tag{4.20}$$

$$\mathrm{Tr}\left[H\hat{\rho}^{\mathrm{OC,EC}}_{\mathrm{max}}\right] \leq \mathrm{Tr}\left[H\hat{\rho}^{\mathrm{OC}}_{\mathrm{max}}\right]. \tag{4.21}$$

The variable $\epsilon^j_{\mathbf{i}} = \alpha\,\mathrm{Var}(\hat{C}^j_{\mathbf{i}})$ in Eq. (4.13) defines the search region explored by the SDP, thereby controlling the confidence interval for the ground-state energy estimate. Different α-values yield different search regions, visualized in Fig. 4.3. The spectrahedra illustrates the feasible set after imposing positive semidefiniteness, OC, and EC constraints. The true state ρ (red) and its estimated counterpart $\hat{\rho}$ (black) are included for comparison. Crucially, when the search region is large enough within the feasible set to capture the true RDMs ρ, we obtain upper and lower bounds on the ground energy estimate $\hat{E}_g$:

$$\mathrm{Tr}\big[H\,\hat{\rho}^{\mathrm{SDP}}_{\min}(\alpha_0)\big] \;\le\; \hat{E}_g \;\le\; \mathrm{Tr}\big[H\,\hat{\rho}^{\mathrm{SDP}}_{\max}(\alpha_1)\big], \tag{4.22}$$

where $\alpha_0 > \alpha_1$.

Algorithm 4.1: SDP-assisted tomography on ground states

```
Input: $T$ identical ground-state preparations, Hamiltonian
       $H = \sum_{j=1}^{m} H_j$, hyperparameters $b > \Delta > 0$, where $\Delta$ is the
       tolerance.
Output: Local RDM estimates $\{\hat{\rho}_j^{\mathrm{SDP}}\}_{j=1}^{m}$ that minimize the energy,
        and the corresponding minimum energy $\hat{E}_g$.
Perform standard quantum state tomography to obtain estimated local
  RDMs $\{\hat{\rho}_j\}_{j=1}^{m}$ with the precision given by $T$ samples;
repeat
    Solve the SDP problem described by Eq.(4.19) with decision variables
      $\{\tilde{\rho}_j\}_{j=1}^{m}$ and coefficient $\alpha = b$;
    Denote the feasible set as $\mathcal{F}$;
    $b \leftarrow 2b$ ;
until $\mathcal{F} \neq \emptyset$;
Initialize $a = 0$;
while $|a - b| \geq \Delta$ do
    $\alpha_0 \leftarrow a + \frac{|a-b|}{2}$;
    Solve the SDP problem described by Eq.(4.19) with decision variables
      $\{\tilde{\rho}_j\}_{j=1}^{m}$ and coefficient $\alpha = \alpha_0$;
    Denote the feasible set as $\mathcal{F}$;
    if $\mathcal{F} \neq \emptyset$ then
        Denote the solution as $\mathcal{S} = \arg\min_{\{\tilde{\rho}_j\}} \sum_j \mathrm{Tr}\,[\tilde{\rho}_j H_j]$;
        $b \leftarrow \alpha_0$;
    else
        $a \leftarrow \alpha_0$;
$\{\hat{\rho}_j^{\mathrm{SDP}}\}_{j=1}^{m} \leftarrow \mathcal{S}$;
$\hat{E}_g \leftarrow \sum_{j=1}^{m} \mathrm{Tr}\big[\hat{\rho}_j^{\mathrm{SDP}} H_j\big]$;
```

We employ a bisection method (Alg. 4.1) to identify the smallest α_0 (within a tolerance Δ_0) for which the feasible set is non-empty, and analogously for α_1 using a maximization variant. This allows us to establish the bounds of Eq. (4.22) on $\hat{E}_g$.

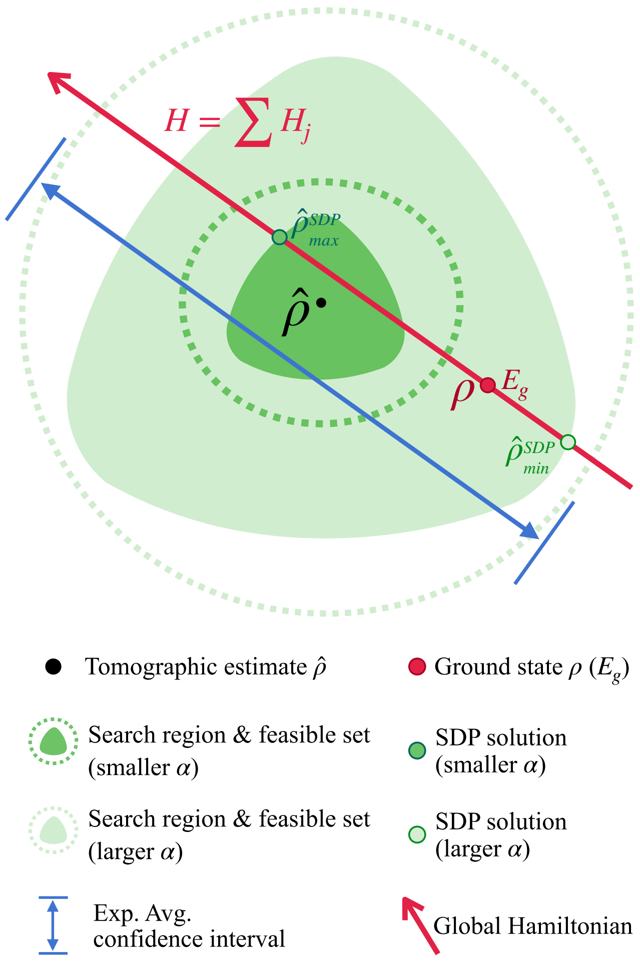

$H = \sum H_j$
$\hat{\rho}^{SDP}_{max}$
$\hat{\rho}$
ρ
E_g
$\hat{\rho}^{SDP}_{min}$
● Tomographic estimate $\hat{\rho}$
● Ground state ρ (E_g)
Search region & feasible set (smaller α)
SDP solution (smaller α)
Search region & feasible set (larger α)
SDP solution (larger α)
Exp. Avg. confidence interval
Global Hamiltonian

Figure 4.3: An illustrative depiction of the SDP feasibility region. The true RDM ρ with energy E_g of Hamiltonian H (red) and its tomographic estimate $\hat{\rho}$ (black) lie within a search region of adjustable size (dashed circle) determined by the coefficient α. The feasible set under overlapping-compatibility (OC) and enhanced-compatibility (EC) constraints appears as a shaded spectrahedron. The blue interval indicates the confidence interval for the average of the energy obtained by standard quantum state tomography from the simulated data.

4.3.3 Numerical simulations

In this section, we present numerical results for estimating the ground-state energy of the 1D-chain XY model [198]. We also compare the estimation accuracy achieved by standard tomography to that of the proposed SDP-assisted tomography.

Problem description

The Hamiltonian of the 1D-chain XY model is given by

$$H_C = \sum_{j=1}^{n-1} H_j = J \sum_{j=1}^{n-1} \left(X_j X_{j+1} + Y_j Y_{j+1} \right). \tag{4.23}$$

where H_j is the local two-qubit Hamiltonian term capturing the interaction between qubits j and $j+1$. Here, n denotes the total number of qubits, and J is the interaction strength. The subscript 'C' stands for 'chain'. This model is frustrated, meaning that the global ground state of H_C does not simultaneously minimize the energy of each local term H_j.

Although the accuracy of ground-state energy estimation through tomography is fundamentally limited by the number of ground-state samples, introducing additional constraints can significantly tighten the resulting confidence interval. In the context of the XY model H_C, the overlapping- (OC) and enhanced-compatibility (EC) constraints in Eq. (4.19) take the form

$$\mathscr{R}_C \coloneqq \cup_{j\in[n-1]} \mathcal{R}_C \left(\tilde{\rho}_j, \tilde{\rho}_{j+1} \right), \tag{4.24}$$
$$\mathscr{G}_C \coloneqq \cup_{j\in[n-2]} \mathcal{G}_C \left(\tilde{\rho}_j, \tilde{\rho}_{j+1} \right), \tag{4.25}$$

respectively, with

$$\mathcal{R}_C \left(\tilde{\rho}_j, \tilde{\rho}_{j+1} \right) \coloneqq \{ \mathrm{Tr}_{\{j\}} \left[\tilde{\rho}_j \right] - \mathrm{Tr}_{\{j+2\}} \left[\tilde{\rho}_{j+1} \right] = 0 \}, \tag{4.26}$$
$$\mathcal{G}_C \left(\tilde{\rho}_j, \tilde{\rho}_{j+1} \right) \coloneqq \{ \tilde{\rho} \succeq 0, \mathrm{Tr}_{\{j\}} \left[\tilde{\rho} \right] = \tilde{\rho}_{j+1}, \mathrm{Tr}_{\{j+2\}} \left[\tilde{\rho} \right] = \tilde{\rho}_j \}. \tag{4.27}$$

Here, $\tilde{\rho}_j$ is the SDP decision variable corresponding to the two-qubit subsystem of qubits $(j, j+1)$, and $\tilde{\rho}$ is a three-qubit SDP variable spanning qubits $j, j+1$, and $j+2$. The subscript in the partial trace indicates which qubit is being traced out.

Note that the energy function can be computed from the 2-qubit RDMs in a natural way, but also equivalently from the 3-qubit RDMs that stem from the ECs, either by taking their partial traces (which are compatible with the 2-qubit RDMs, by construction) or by extending the Hamiltonian terms to a larger Hilbert space (by appropriately tensoring them with identity operators).

Explanation on the results

For the SDP-assisted tomography, we are firstly given a number of identical copies of the XY model ground state, as in Eq. (4.23). We perform a series

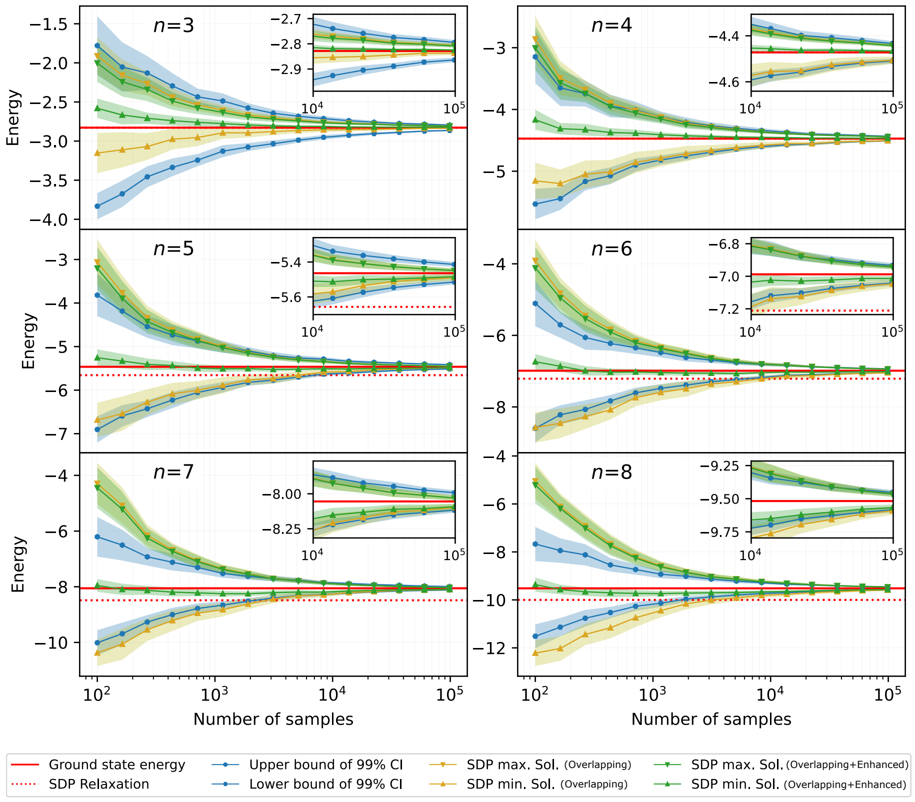


Figure 4.4: Tightening expectation value confidence interval via SDP-assisted quantum tomography. Ground-state energy estimates for the XY model ($J = 1$) with $n = 3, 4, 5, 6, 7, 8$ qubits against the number of state samples. The red solid lines indicate the exact ground-state energy; the red dashed lines are lower bounds from the relaxation method. The blue curves show the 99% confidence interval upper and lower bounds for standard quantum tomography. The yellow (green) curves give the results of SDP minimization with tolerance parameters $\Delta_0 = 0.1$, and maximization with tolerance parameters $\Delta_0 = 0.001$, under OC (OC+EC) constraints, respectively. In each plot, the two green curves can be interpreted as the upper and lower energy bounds using SDP-enhanced quantum tomography.

of measurements in randomly chosen Pauli basis $\sigma_{\mathbf{i}}$, with $\mathbf{i}$ uniformly sampled from $\{1, 2, 3\}^n$. From these measurement outcomes, we reconstruct the local RDMs via standard tomography and estimate the ground-state energy through Eq. (4.11). We then apply our SDP-based post-processing to refine these ground-state energy estimates.

The plots in Fig. 4.4 show the resulting confidence intervals for the ground-state energy as a function of the number of samples, for $n = 3, 4, 5, 6, 7, 8$. In each subplot, the red solid line indicates the exact ground-state energy from Eq. (4.23) obtained by exact diagonalization, while the blue region marks the 99% confidence interval from standard tomography. The green curves represent the solutions of Alg. 4.1 applied to SDP minimization and maximization (with OC+EC constraints, Eqs. (4.26) and (4.27)) and two different tolerances $\Delta_0 = 0.1$ and $\Delta_1 = 0.001$. This choice is motivated by the fact that there is a strong asymmetry between upper and lower bounds when tomographing the ground state. In contrast, the yellow curves illustrate the SDP results with only OC constraints (Eq. (4.26)).

Comparing the yellow and green curves marked with triangles shows that adding EC constraints substantially tightens the lower bound obtained by SDP minimization. When the sample size is small, the tomographic RDM $\hat{\rho}$ can significantly diverge from the true RDM ρ, creating bias in the SDP energy bound in the presence of large statistical noise. Together with the fact that the probability of failure in estimating the true ground state is higher in extremely-low-shot regimes, this fact makes it much more likely for the green curves marked with triangles to exceed the exact ground-state energy. However, as the number of samples grows, the SDP-assisted approach narrows the confidence interval more effectively than standard tomography. This improvement is especially clear in the zoomed-in subplots, where the green curve, representing SDP-assisted (OC+EC) bounds, spans a narrower range for $\langle H \rangle$ than the blue curve from standard quantum state tomography. Notably, SDP-assisted tomography can reduce the required number of samples by a factor of 10^1 to 10^2 compared to standard quantum state tomography, while achieving the same level of precision in lower-bounding the energy.

The red dashed lines represent lower bounds obtained via an existing relaxation method [196]. Such approaches optimize the ground-state energy under only a subset of the constraints that we use, thus guaranteeing a strict, albeit potentially conservative, bound. While such relaxations have successfully established lower bounds for the ground-state energies of local Hamiltonians in many-body settings [187, 194, 196, 199–202], the accuracy of these bounds tends to degrade for larger problem sizes. This shortfall arises from omitting higher-level EC constraints to keep the computational effort manageable.

All numerical simulations were performed using Qiskit [203] for quantum simulations and CVXPY [204, 205] to solve SDP formulations using the SCS optimizer.

4.4 Application

In this section, we demonstrate how to integrate our SDP-assisted tomography into a variational procedure for preparing and characterizing low-energy states of local Hamiltonians. Specifically, we embed SDP-assisted tomography within the *algorithmic cooling* (AC) method [177–179] to heuristically minimize the energy of a target quantum Hamiltonian. Sec. 4.4.1 introduces AC, and Sec. 4.4.2 explains how SDP-assisted tomography is incorporated into the AC workflow. Numerical results for the 1D-chain XY model are presented in Sec. 4.4.3, comparing the performance of AC both with and without semidefinite programming.

4.4.1 Algorithmic cooling

Algorithmic cooling (AC) aims to prepare a low-energy state $|\psi\rangle$ of an n-qubit quantum system governed by a local Hamiltonian H, using near-term quantum devices. The AC method is adapted to the practical capabilities of the experimental setup, which can differ considerably across platforms. Thus, we do not assume access to arbitrary unitaries; instead, we denote by $\mathbf{h}$ the set of Hermitian operators h for which e^{-iht} can be implemented natively on the device. The unitary e^{-iht} is implemented by turning on an interaction given by h for an amount of time t. We also assume, without loss of generality, that each $h \in \mathbf{h}$ does not commute with H (Hamiltonian of XY model). Fig. 4.5 outlines the AC workflow.

Optimization step

We now describe the optimization routine in AC, which incrementally builds a shallow circuit to prepare a low-energy state for local Hamiltonians. We begin by introducing the cooling principle.

Consider a Hamiltonian H of the form in Eq. (4.9) and an operator $h \in \mathbf{h}$ such that $[H, h] \neq 0$ and, for simplicity, $h^2 = \mathbb{I}$. The last assumption can be relaxed [151]. Given an initial quantum state $|\psi_0\rangle$,

$$\begin{aligned}\left|\psi_0^h(t)\right\rangle =& e^{-ith}\left|\psi_0\right\rangle \\ =& \left(\cos(t)\mathbb{I} - i\sin(t)h\right)\left|\psi_0\right\rangle,\end{aligned} \tag{4.28}$$

we define the energy function:

$$\begin{aligned}E_h(t) =& \left\langle\psi_0^h(t)\right| H \left|\psi_0^h(t)\right\rangle \\ =& \cos^2(t)\langle H\rangle + i\cos(t)\sin(t)\langle[h, H]\rangle + \sin^2(t)\langle hHh\rangle.\end{aligned} \tag{4.29}$$

By using trigonometric identities, one finds

$$E_h(t) = \langle H\rangle \; + \; A\,\sin^2(t) \; + \; \tfrac{B}{2}\,\sin(2t), \tag{4.30}$$

where $\langle\cdot\rangle$ indicates the expectation value in $|\psi_0\rangle$, $A := \langle hHh - H\rangle$, and $B :=$

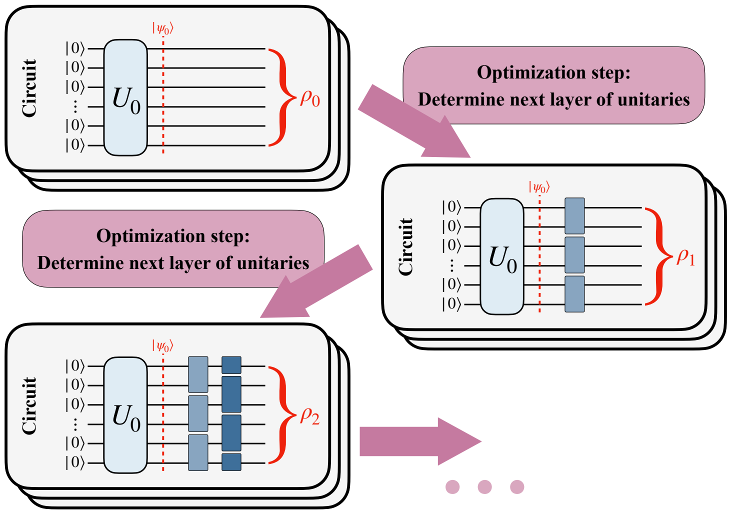


Figure 4.5: Conceptual diagram of algorithmic cooling (AC) Algorithmic cooling iteratively constructs a quantum circuit that prepares a low-energy state of a local Hamiltonian. Initially, the circuit is the identity, and the trial state is $|\psi_0\rangle = U_0 |0\rangle^{\otimes n}$. Each iteration generates an ensemble of identical states for tomography, yielding local RDMs that guide the choice of a new layer of gates (see Sec. 4.4.1), which is appended to form the next circuit.

$i\langle[h, H]\rangle$. The minimum of $E_h(t)$ occurs at

$$t^* = \frac{1}{2} \arctan \frac{-B}{A}, \tag{4.31}$$

yielding

$$E_h(t^*) = \langle H \rangle + \frac{1}{2}\left(A - \sqrt{A^2 + B^2}\right) . \tag{4.32}$$

Note that there is a slight ambiguity in the definition of t^* given in Eq. (4.31): the minimum is achieved at any t^* satisfying $\cos(2t^*) = A/\sqrt{A^2 + B^2}$ and $\sin(2t^*) = -B/\sqrt{A^2 + B^2}$. Choosing the opposite sign leads to a maximum at $t^* + \pi/2$,

$$E_h(t^* + \pi/2) = \langle H \rangle + \left(A + \sqrt{A^2 + B^2}\right)/2. \tag{4.33}$$

Therefore, if A and B can be accurately estimated, applying the interaction given by h (or $-h$) for a period of time $|t^*|$ will decrease the energy of $|\psi_0\rangle$ by an amount of $\frac{1}{2}\left(\sqrt{A^2 + B^2} - A\right)$. Since H is local, the value of A and B can be extracted from measurements on only a constant number of qubits:

$$A = \sum_{j:\mathrm{supp}(H_j \cap h) \neq \emptyset} \langle h H_j h - H_j \rangle \,, \tag{4.34}$$

$$B = \sum_{j:\mathrm{supp}(H_j \cap h) \neq \emptyset} i \left\langle [h, H_j] \right\rangle . \tag{4.35}$$

Here, $\mathrm{supp}(H_j \cap h) := \mathrm{supp}(H_j) \cap \mathrm{supp}(h)$, and $\mathrm{supp}(\cdot) \subseteq [n]$ denotes the qubits on which an operator acts non-trivially.

Because each update step is chosen to locally decrease energy, the energy function forms a monotonically decreasing sequence (assuming we have access to exact RDMs $\{\rho_j\}_{j=1}^{m}$) that is bounded by below, guaranteeing convergence (albeit not necessarily to the true ground state). The algorithmic cooling approach is thus a heuristic strategy that can fail for several reasons: $\mathbf{h}$ may be too small to reach the global ground state from the initial state; choosing unitaries in certain orders could lead to local minima; or noisy RDM estimates might produce suboptimal energy updates.

Nonetheless, AC provides a flexible, greedy technique for variationally reducing energy. Integrating SDP-assisted tomography can enhance the accuracy of local RDM estimates, thereby improving the reliability and efficiency of each iteration.

Algorithmic cooling circuit compilation

This subsection explains how we construct the layout of the variational circuit by integrating the optimization step from Sec. 4.4.1 into the algorithmic cooling (AC) method. Note that there are many ways to construct the heuristic, each with a different performance depending on the problem. Here, we outline a specific one for illustrative purposes.

We begin with an empty circuit, $\mathcal{U}_C = \mathbb{I}$, and an initial trial state $|\psi_0\rangle$. At each step, we consider all available operators $h \in \mathbf{h}$, selecting the one that yields the greatest energy reduction. We then append the corresponding unitary $U_k = e^{-i h_k t_k^*}$ to the circuit, ensuring a strictly decreasing energy at each iteration. Although this approach guarantees monotonic improvement, it may not fully exploit device capabilities.

To make better use of circuit depth, we group gates into layers $V_1, \ldots, V_L$. Each layer $V_l = U_l^1 \cdots U_l^k \cdots U_l^{K_l}$ consists of K_l parallel unitaries, with disjoint supports,

$$\mathrm{supp}\left(U_l^k\right) \cap \mathrm{supp}\left(U_l^{k'}\right) = \emptyset, \quad \forall\, 1 \leq k, k' \leq K_l. \tag{4.36}$$

The overall circuit is then $\mathcal{U}_C = V_L \cdots V_1$, allowing for a more efficient arrangement while still achieving a steady decrease in energy.

To optimize a layer V_l, one could iteratively:

1. Select an operator h, compute t^*, and update $V_l \leftarrow V_l\, e^{-iht^*}$.
2. Repeat for the next operator h', finding t'^*, and so on.

However, each new operator choice requires a full state preparation and measurement cycle. Since H is local, it can become more resource-efficient to collect a suitable set of RDMs for the current layout of V_l and then perform all unitary updates for that layer classically, rather than measuring after every single addition of e^{-iht^*}.

In Alg. 4.2, we present a more refined approach that avoids predefining the circuit layout. Instead, at each iteration, the algorithm greedily selects the operator h^* and corresponding t^* that produce the largest energy decrease. The selection is restricted to operators whose supports are disjoint from any non-trivial gates already in the layer. This ensures the circuit grows incrementally while maintaining a clear and consistent reduction in energy at each step.

Algorithm 4.2: Algorithmic cooling

```
Input: Initial state $|\psi_0\rangle$, Hamiltonian $H = \sum_{j=1}^m H_j$, operator set $\mathbf{h}_0$,
       maximum number of iterations $L$.
Output: Quantum circuit $\mathcal{U}_C$.
Initialize the system to $|\psi_0\rangle$ and set $\mathcal{U}_C = \mathbb{I}$;
$l \leftarrow 0$;
repeat
    Prepare an ensemble of identical states $\mathcal{U}_C\,|\psi_0\rangle$ and perform standard
      quantum state tomography to obtain local RDMs $\{\hat{\rho}_j\}$;
    $\mathbf{h} \leftarrow \mathbf{h}_0$, $V_l \leftarrow \mathbb{I}$;
    while $\mathbf{h} \neq \emptyset$ do
        Estimate the parameters $A$ (Eq. (4.34)) and $B$ (Eq. (4.35)) for
          every $h \in \mathbf{h}$ with local RDMs $\{\hat{\rho}_j\}$, respectively;
        Select the $h^*$ and $t^*$ that maximize the energy decrease
          $\frac{1}{2}\left(\sqrt{A^2 + B^2} - A\right)$, c.f. Eq. (4.32);
        $V_l \leftarrow e^{-ih^*t^*} V_l$;
        $\hat{\rho}_j \leftarrow e^{-ih^*t^*}\hat{\rho}_j e^{i(h^*)^\dagger t^*}$, $\forall \hat{\rho}_j \in \{\hat{\rho}_j\}$;
        $\mathbf{h} \leftarrow \mathbf{h} \setminus \{h \in \mathbf{h} \mid \mathrm{supp}(h) \cap \mathrm{supp}(h^*) \neq \emptyset\}$;
    $\mathcal{U}_C \leftarrow V_l \mathcal{U}_C$;
    $l \leftarrow l + 1$;
until $l = L$;
```

4.4.2 Embedding SDP-assisted tomography

In practice, the set of possible operators $\mathbf{h}$ can be extremely large. Rather than testing each $h \in \mathbf{h}$ individually, one can instead perform tomography on the reduced density matrices (RDMs) of relevant qubit subsets. Although tomography itself is expensive, if both the Hamiltonian H and the unitaries to be optimized are local, then the quantities $\langle hH_jh - H_j\rangle$ and $\langle [h, H_j]\rangle$ needed to identify the optimal unitary e^{-iht^*} depend only on the RDMs of $\mathrm{supp}(H_j) \cup \mathrm{supp}(h)$, which is constant for every j. These expectation values from Eqs. (4.34)–(4.35) can be computed via

$$\langle hH_jh - H_j\rangle = \mathrm{Tr}\big[\rho_r \big(hH_jh - H_j\big)\big], \tag{4.37}$$

$$i\langle [h, H_j]\rangle = i\,\mathrm{Tr}\big[\rho_r\,[h, H_j]\big], \tag{4.38}$$

where ρ_r is a collection of relevant RDMs of the current global state $|\psi\rangle$,

$$\rho_r = \mathrm{Tr}_{[n]\setminus(\mathrm{supp}(H_j)\cup\mathrm{supp}(h))}(|\psi\rangle\langle\psi|). \tag{4.39}$$

To obtain the relevant RDMs, one first performs state tomography. However, due to shot noise, the raw tomographic local RDMs may become non-physical. To address this, one can formulate an SDP (Eq. (4.19)) to recover locally consistent RDMs that minimize the energy while preserving physicality. These SDP-refined RDMs are then provided to the algorithmic cooling (AC) method (Alg. 4.2), guiding the optimization parameters for each cooling step.

4.4.3 Numerical simulations

We now present numerical simulations of the AC procedure for approximating the ground state of the 1D-chain XY model (a frustrated Hamiltonian), specified by Eq. (4.23). We consider two initial states: (a) the uniform superposition $|+\rangle^{\otimes n}$ and (b) a Hartree–Fock (mean-field) product state, which corresponds to the product state of minimal energy, which can be obtained efficiently in one dimension [206], thereby providing an in principle better initialization. We employ Alg. 4.2 that iteratively prepares the ground state and we restrict the set of available operators $\mathbf{h}$ to geometrically local Pauli operators.

Figs. 4.5a and 4.5b illustrates the results for different pairs (n, n_s), where n (number of qubits) ranges from 3 to 8 and n_s (number of samples per iteration) takes values $10^1, 10^2, 10^3, 10^4$. Each iteration uses n_s samples per iteration to estimate the necessary parameters for cooling either using standard tomography or SDP-assisted tomography. Within each subplot, the red line marks the exact ground-state energy. Two variants of AC method are simulated: (i) the blue curve corresponds to AC with standard tomography, using tomographically reconstructed local RDMs to compute energy, and (ii) the orange curve depicts

AC with SDP-refined RDMs, still using their reconstructed expectation value for energy. The green curve shows the SDP lower-bound on the energy at each iteration, representing the minimal energy compatible with the measured data under the chosen constraints. Each method is repeated 25 times, and the shaded areas around the curves indicate one-sided standard deviation.

Figs. 4.5a and 4.5b also showcase three noteworthy regimes: For a low number of measurements ($\sim 10^2$), the monotonicity property, especially of the AC with tomography, is broken. This is due to the shot noise being too dominant and confusing the heuristic of the AC. For the $|+\rangle^{\otimes n}$ as an initial state, and for $n_s = 10^2$, we observe a sudden dip in energy for the AC with tomography, followed by a steady increase in energy, as iterations progress. This is due to the AC adding gates that are not sufficiently close to optimal, because of the limitations in precision. In contrast, the behavior of AC with SDP is qualitatively closer to monotonicity, especially in the first few iterations, although ultimately converging to a similar value as the AC with tomography. For the $|HF\rangle$ as initial state, both AC with tomography and AC with SDP have the same phenomena, but the energy scales are much lower as we begin from the lowest-energy product state.

For a higher number of measurements ($\sim 10^3$), the AC with SDP performs better compared to the AC with tomography. This showcases a sweet-spot regime where our approach proves more advantageous, as this behavior is consistent across different system sizes and both initializations considered. However, for an even larger number of measurements ($\sim 10^4$), the performance of the two methods becomes similar. This is likely because, with more measurement data, the estimates provided by standard tomography are increasingly physical, leaving less room for the SDP to correct non-physicality. In the absence of shot noise, the performance of both methods is identical by construction.

It is worth noting that the number of samples used in our numerical simulations is significantly lower than in typical physical experiments, which highlights the efficiency of the approach, though it introduces some limitations. It should be noted that there is no guaranteed advantage of the orange curve (AC with SDP-optimized local RDMs) over the blue curve (AC with local RDMs obtained via tomography) in any single simulation. This is because AC is a heuristic method, and its performance depends on various factors, such as the initial state, measurement settings, and other parameters. However, in some settings, the blue and orange curve have non-overlapping shaded areas. Additionally, by construction, the green curve (AC with SDP-optimized local RDMs and minimized energy) provides a lower bound on the orange curve.

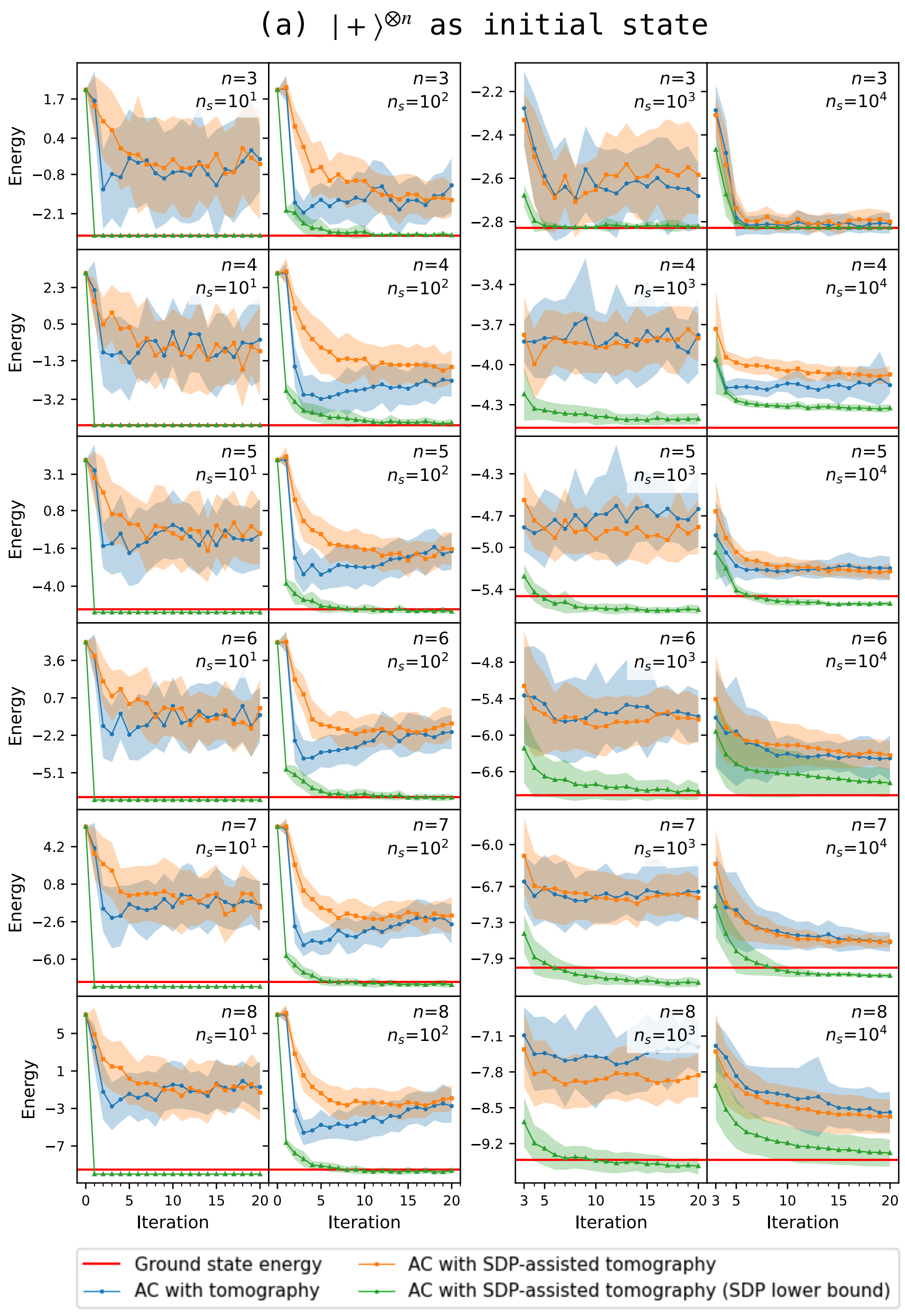


Figure 4.5a: Algorithmic cooling results for a 1D-chain XY model (frustrated Hamiltonian), initialized in $|+\rangle^{\otimes n}$. Energy expectation values are shown as a function of cooling iterations using standard tomography and SDP methods.

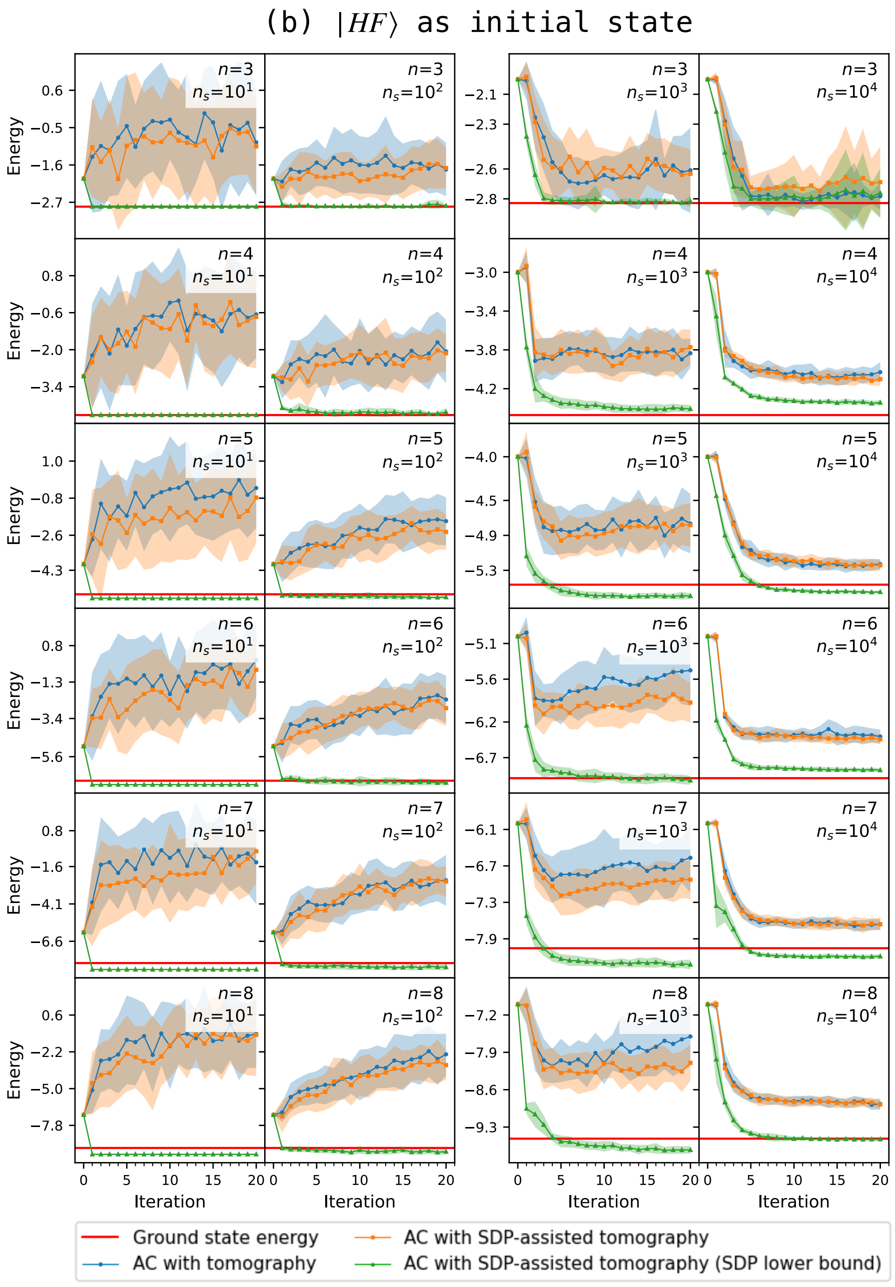


Figure 4.5b: Algorithmic cooling results for a 1D-chain XY model (frustrated Hamiltonian), initialized in $|HF\rangle$. Energy expectation values are shown as a function of cooling iterations using standard tomography and SDP methods.

4.5 Conclusion and outlook

In this chapter, we introduced a semidefinite programming (SDP)-assisted technique for reconstructing overlapping reduced density matrices (RDMs) from experimental measurement data, effectively addressing the challenges posed by shot noise in near-term quantum computing. By enforcing overlapping-compatibility (OC) and enhanced-compatibility (EC) constraints on local RDMs through a polynomial-sized SDP problem, our method alleviates the impact of limited measurements and ensures the consistency of the reconstructed local RDMs. As a result, our approach not only enhances the accuracy of local observable estimates but also provides tighter confidence intervals compared to standard tomographic procedures.

To illustrate its effectiveness, we applied SDP-assisted tomography within a variational quantum algorithm, algorithmic cooling (AC), aimed at heuristically preparing low-energy states of local Hamiltonians. Numerical simulations of a 1D-chain XY model demonstrated the advantages of our approach in terms of both accuracy and resource efficiency. These findings indicate that SDP-assisted tomography can serve as a valuable asset for boosting the performance of variational quantum algorithms and other quantum information processing tasks in the near term.

Looking ahead, a promising direction for future research is the incorporation of additional constraints, such as entropy constraints, into the SDP formulation to further refine the reconstruction process [195]. It would also be interesting to extend the SDP-assisted tomography framework to more complex many-body systems and investigate its behavior on frustration-free Hamiltonians. Applying it to specific quantum chemistry calculations might be promising, as done in a similar way in Refs. [207, 208]. In addition, exploiting the framework to optimize other local observables, such as correlation functions, is another potential area of exploration. Another open question is the effect of systematic errors on the application of our method, and how strategies such as those presented in Ref. [209] can help to reduce them.

Finally, examining how SDP-based methods can be combined with noise mitigation strategies on quantum hardware could offer valuable insights for practical enhancements in near-term quantum computing.

CHAPTER 5

Quantum Generative Modeling for Financial Time Series with Temporal Correlations

5.1 Introduction

In recent years, the use of machine learning - in particular neural-network based approaches - has expanded across many domains [48]. A particular approach are the generative adversarial networks (GANs), in which successively a generator and a discriminator are trained [210, 211]. The generator learns the underlying distribution and samples from it, while the discriminator learns to distinguish real data from generated samples. Typically, GANs are applied in image generation [212–214], but also to other data [215].

The ability of machine learning models to generalize well relies on the availability of large datasets [48]. Data augmentation methods are techniques which increase the training data set in order to alleviate these limitations [216]. Those methods typically involve slightly modifying the training data, but synthetic data generation by GANs is used as an approach for data augmentation [216, 217].

This is particularly relevant for finance, a computationally-heavy, yet difficult-to-model field [77]. Unlike domains where there is an abundance of high-quality data to train on, finance faces a fundamental challenge: the inherent non-repetitive nature of financial events. For example, the time-series of a specific

The contents of this chapter have been published in Ref. [91].

asset's price can only be observed once. A machine learning model that aims to learn properties based on the time-series of a specific asset therefore is heavily limited, as their ability to generalize well relies on large datasets. By learning the underlying distribution of financial time series as well as its desired temporal properties, one can generate new data that enables the creation of richer training sets [218, 219]. In particular, temporal correlations such as volatility clustering (periods of large variation are followed by periods of large variation, as do periods of low variation) are important for single time series.

Parallel to the development of machine learning, research in quantum computing and its potential application has increased. Motivated by the progress in quantum hardware, quantum algorithms research has been focusing on variational quantum algorithms in the last decade [20]. These hybrid algorithms consist of succinct calculations on a parameterized quantum circuit (PQC) and a classical optimizer [21]. The PQC contains tunable and fixed gates, and the classical optimizer calculates updated parameters based on measurements conducted on the final quantum state of the PQC at each step, until a certain precision or goal is reached.

Previous work proposed replacing the classical generator of a GAN with a parameterized quantum circuit [220, 221]. Quantum circuits have been proven to enable sampling from distributions which are intractable for classical circuits, and so the set of distributions they access is in general different than what classical models access. Consequently, it is expected they may be more effective with some classes of distributions that classical models struggle with [55, 56, 79, 222]. Subsequent studies expanded on this idea and examined if this property can be harnessed in the context of learning financial distributions. In particular, in [223], the idea to use QGANs for synthetic data generation of financial time series has been proposed and tested, using a quantum circuit Born machine, which performed better than a classical restricted Boltzmann machine on learning the distribution of correlated asset pairs with respect to the Wasserstein distance. However, generating financial time series which do not only follow the same distribution as real-world data, but also show their temporal correlations is challenging [78].

In this chapter, we develop a quantum GAN with a PQC as an expectation value sampler-based generator and a classical neural network as a discriminator for synthetic data generation and examine its ability in generating time series replicating the S&P 500 index, including its distribution and temporal correlations. The discrete time series spans from 20 to 40 points in time, where the expectation value of single-qubit Pauli-X and Pauli-Z operators is interpreted as the log return of the value of the index at each time step. The choice of these observables will be motivated in Sec. 5.3.2. We simulate the quantum circuits both with full-state simulations and with matrix product state (MPS) simulations (also known as the tensor train) [224, 225]. While full-state simulations are limited to short time intervals due to their exponential scaling, MPS simulations enable us to model longer time series by exploiting their ability to efficiently replicate linear structures such as financial time series. In the MPS simulations, we vary the bond dimension (also referred to as the tensor train rank) to balance computational costs and simulation accuracy.

We show that our model generates time series whose distribution closely matches that of the real time series. Furthermore, the generated samples show temporal correlations, which are qualitatively similar to those observed in real-world data. Our work shows the potential use of QGANs in learning time series with specific temporal correlations.

This chapter is organized as follows. In Sec. 5.2, we introduce the concepts of financial time series and its properties, as well as the concept of QGANs. Further, we cover related work. In Sec. 5.3, we present the simulations we used. We detail the data pre-processing as well as the quantum generator and the matrix product state simulation. We show the results of our simulations in Sec. 5.4, discuss them in Sec. 5.5, and conclude in Sec. 5.6.

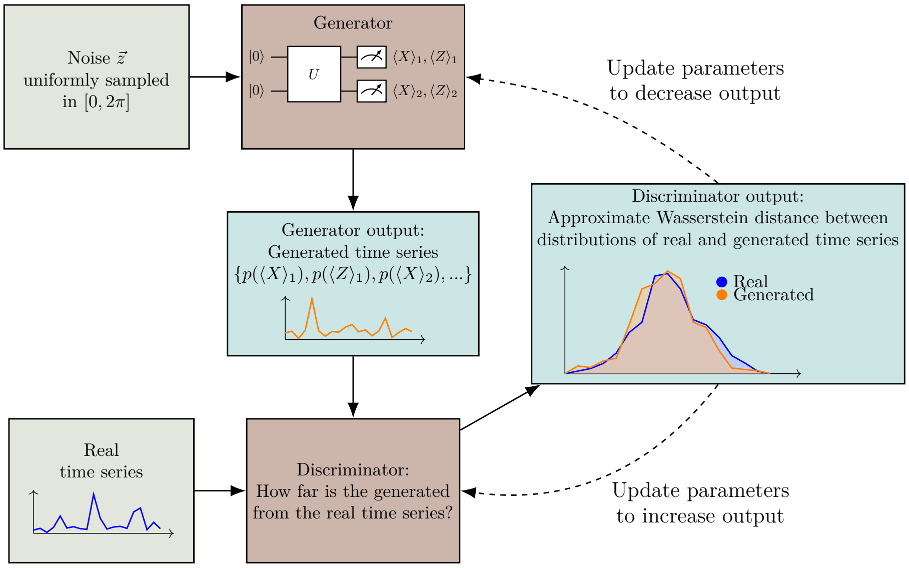


Figure 5.1: Structure of a generative adversarial network (GAN) used for time series generation. The discriminator takes both the generated and real time series as input and outputs its estimate for the Wasserstein distance $W_1(\mathcal{P}_r, \mathcal{P}_g)$ (see Eqs. (5.7) and (5.8)). The generator is trained in order to bring the distribution of the generated time series closer to the one of the real time series, the discriminator is trained to approximate the Wasserstein distance between them. Both the generator and discriminator are trained using different loss functions derived from the discriminator's output.

5.2 Background

In this section, we will introduce both financial time series and its temporal correlations as well as generative adversarial networks, and their adaptations based on the Wasserstein distance and with a quantum generator. Furthermore, we will present related work.

5.2.1 Financial time series

A time series is a set of data points ordered over a given time frame, typically at equally spaced time intervals. A financial time series is the set of financial variables such as prices, returns and volatility of assets, indices or other financial instruments.

An example of a financial time series is the S&P 500 index, which includes 500 of most valuable companies that are listed on US stock exchanges [75]. The price of many instruments, such as the S&P 500 index, deviates around a mean value that grows in time. Instead, examine the time series of the log return r_t, which is not dependent on this general market growth:

$$r_t = \log\left(\frac{S_t}{S_{t-1}}\right), \tag{5.1}$$

where S_t is the price of the index at time t. Simple models such as the Black-Scholes model [81] assume a normal distribution of these log returns. However, the distributions observed in the market have more complicated properties. The returns of assets do not typically follow a normal distribution, their distributions are more narrowly and spiked around the mean and have heavier tails (extreme events are more likely), which is in contrast to the normal distribution of the Black-Scholes model. Therefore, a main concern of research in finance is about creating models that can mimic time series with higher accuracy, and machine learning approaches have been increasingly explored for this case [77]. Many observed log returns share common properties, also called stylized facts, that originate mostly from behavior of parties that interact with the market [76]. These properties can be used in order to assess the quality of models of financial time series. In practice, it is considerably more difficult to generate time series that observe all of the stylized facts, than to only match the target distribution. However, for finance practitioners it is often vital to use models which show the stylized facts that are relevant for their use case [78] .

In this chapter, we focus on four stylized facts: non-Gaussianity, the absence of linear autocorrelation, volatility clustering and the leverage effect. The first of them is describing the behavior of the time-aggregated distribution. As written above, it is not shown in the Black-Scholes model, but is generally seen in real-world data. The latter three stylized facts are all temporal correlations between different values of the same time series. They can be analyzed with the help of the correlation function $corr(X, Y)$, which for the random variables X and Y is defined as:

$$corr(X, Y) = \frac{cov(X, Y)}{\sigma_X \sigma_Y} = \frac{\langle (X - \mu_X)(Y - \mu_Y) \rangle}{\sigma_X \sigma_Y}, \tag{5.2}$$

where $cov(X, Y)$ is the covariance, σ_X and σ_Y the standard deviations of the random variables and μ_X and μ_Y denote their respective means. For a sequence of independent random variables, the autocorrelation function vanishes at all nonzero time lags. For instance, the increments of Brownian motion are independent and therefore exhibit zero autocorrelation. The stylized facts of the absence of linear autocorrelation, volatility clustering and the leverage effect each have an intuitive reason for their emergence and can be observed by analyzing the autocorrelaiton of the absolute and identical values of the time series.

Firstly, the current and past values of financial time series are typically not linearly autocorrelated. At time t, this means that for all time differences $\tau > 0$, the expectation value $\mathbb{E}\left[corr(r_t, r_{t+\tau})\right]$, taken over different realizations of the time series, is close to zero. Intuitively this comes from the fact that any trend in the return is exploited by traders, which in turn weaken the effect. This exploitation of traders is a corollary of the so-called efficient market hypothesis.

Secondly, the absolute returns typically do exhibit correlation that slowly decays in time. This effect is also called volatility clustering, and can be examined by calculating the quantity $corr(|r_t|, |r_{t+\tau}|)$. It quantifies the observation that large changes in the price are followed by large changes, and equivalently small changes are followed by small changes.

Thirdly, the leverage effect describes the rise in volatility when the price of an asset sinks. It can be observed by measuring the quantity $corr(|r^2_{t+\tau}|, r_t)$.

The reader can find more details of these properties in [76].

Synthetic data generation of financial time series concerns the generation of artificial time series that observe these stylized facts. These properties and their importance for practitioners differ depending on the time series and the application, and they are difficult to compare in general. Furthermore, different models generate time series with greatly varying quality in reproducing the stylized facts. Therefore, the resulting synthetic time series are typically assessed qualitatively if they are able to capture those properties [78].

However, in this chapter, we provide several quantitative metrics. In order to quantify how closely the generated time series reproduce the stylized facts of the S&P 500 index, we define the following metrics:

$$EMD(\theta) = \frac{1}{\tau_{max} + 1} \sum_{\tau=0}^{\tau_{max}} \left| r^{(SP500)}_{t+\tau} - r^{(\theta)}_{t+\tau} \right| \tag{5.3}$$

$$E_{id}^{ACF}(\theta) = \left(\frac{1}{\tau_{max}} \sum_{\tau=1}^{\tau_{max}} corr\left(r_t^{(\theta)}, r_{t+\tau}^{(\theta)}\right)^2 \right)^{1/2} \tag{5.4}$$

$$E_{abs}^{ACF}(\theta) = \Bigg(\frac{1}{\tau_{max}} \sum_{\tau=1}^{\tau_{max}} \Big[corr\left(|r_t^{(SP500)}|, |r_{t+\tau}^{(SP500)}|\right) - corr\left(|r_t^{(\theta)}|, |r_{t+\tau}^{(\theta)}|\right) \Big]^2 \Bigg)^{1/2} \tag{5.5}$$

$$E_{Lev}(\theta) = \Bigg(\frac{1}{\tau_{max}} \sum_{\tau=1}^{\tau_{max}} \Big[corr\left(|r_t^{(SP500)}|^2, r_{t+\tau}^{(SP500)}\right) - corr\left(|r_t^{(\theta)}|^2, r_{t+\tau}^{(\theta)}\right) \Big]^2 \Bigg)^{1/2} . \tag{5.6}$$

They are quantifying non-Gaussianity, absence of linear autocorrelation, volatility clustering and leverage effect, respectively. The first is based on the earth-movers distance, which is the discretized form of the Wasserstein distance, and the latter quantities are derived from the correlation functions describing these properties, as explained above. Here, the log returns of the generated time series are written as $r_t^{(\theta)}$, where θ stand for the parameters and hyperparameters of the simulated QGAN, and the log returns of the S&P 500 index are written as $r_t^{(SP500)}$. The lower these metrics are, the closer the stylized facts of the generated time series $r_t^{(\theta)}$ are resembling those of the time series $r_t^{(SP500)}$.

5.2.2 Wasserstein QGAN

Generative adversarial networks (GANs) [210, 211] are unsupervised machine learning-based methods that are powerful in generating images and have also been successfully applied to the generation of financial time series [226]. They consist of two neural network that compete in a game-like setup. The generator takes random noise as input and aims to create artificial data that is indistinguishable from real data. Both real data from the training set and artificial data from the generator is then fed to a discriminator which is being trained to detect the generated data, outputting a probability of the input data being real or fake. They are trained in an alternating fashion until the generator is able to create data indistinguishable from real data. GANs face challenges in training instability (one neural network overpowers the other) and mode collapse (The GAN focuses on creating data with limited variety) [227, 228].

Those challenges can be mitigated by replacing the discriminator with a critic that learns the Wasserstein distance between the real and generated data distributions, in the so-called Wasserstein GAN [228]. The Wasserstein distance between the real and generated probability measures $\mathcal{P}_r$ and $\mathcal{P}_g$, respectively, is defined as:

$$W_1(\mathcal{P}_r, \mathcal{P}_g) := \inf_{\pi \in \Gamma(\mathcal{P}_r, \mathcal{P}_g)} \mathbb{E}_{(x,y)\sim\pi} \left(\|x - y\|\right). \tag{5.7}$$

Here, $\Gamma(\mathcal{P}_r, \mathcal{P}_g)$ denotes the set of all couplings of $\mathcal{P}_r$ and $\mathcal{P}_g$, i.e., all joint probability measures whose marginals are $\mathcal{P}_r$ and $\mathcal{P}_g$. And by $(x, y) \sim \pi$, we denote that the random pair (x, y) is distributed according to the coupling π. Calculating this infimum is not feasible in practice, but the Kantorovich-Rubinstein duality delivers a quantity that can be used in a machine learning context [228, 229]:

$$W_1(\mathcal{P}_r, \mathcal{P}_g) = \sup_{\|f\|_L \leq 1} \left(\mathbb{E}_{x\sim\mathcal{P}_r} \left(f(x)\right) - \mathbb{E}_{\tilde{x}\sim\mathcal{P}_g} \left(f(\tilde{x})\right)\right), \tag{5.8}$$

where $\sup_{\|f\|_L \leq 1}$ is the supremum over all 1-Lipschitz functions. The role of the critic D is to maximise the loss function

$$\begin{aligned} L_D(D, \mathcal{P}_g) =& \mathbb{E}_{x\sim\mathcal{P}_r} \left(D(x)\right) - \mathbb{E}_{\tilde{x}\sim\mathcal{P}_g} \left(D(\tilde{x})\right) && (5.9)\\ &+ \lambda \mathbb{E}_{\hat{x}\sim\mathcal{P}_{\hat{x}}} \left(\left(\|\nabla_{\hat{x}} D(\hat{x})\|_2 - 1\right)^2 \right), && (5.10) \end{aligned}$$

where $D(x)$ is trained to approximate $f(x)$ in Eq. (5.8). The latter term is a gradient penalty regularization [230] that enforces the 1-Lipschitz condition by a scaling parameter λ and where $\hat{x} = \epsilon x + (1 - \epsilon)\tilde{x}$ with the random parameter $\epsilon \sim U[0, 1]$. This loss function will train the critic to approximate the Wasserstein distance between the probability distributions of real and generated data. In contrast, the role of the generator is to maximize

$$L_G(D, \mathcal{P}_g) = \mathbb{E}_{\tilde{x}\sim\mathcal{P}_g} \left(D(\tilde{x})\right). \tag{5.11}$$

See Fig. 5.1 for a sketch of a Wasserstein GAN.

Quantum generative adversarial networks (QGANs) are GANs in which the classical generator and/or the classical discriminator are replaced by a quantum circuit [220, 221].

They are motivated by the fact that quantum circuits can learn distributions efficiently that are hard to model by classical means [55, 56, 79, 222]. The proofs of advantage showcase that learning can not be achieved when the distribution is hard (generated by a quantum process). Market data is manifestly not so. However, in other models it was shown that one can have learning separations even if the generation of the data is classically tractable [231]. It remains an open question if such separations can also hold for estimation value sampler as we use here. However, even without separations it may be the case that quantum models simply have more convenient inductive biases than classical models and understanding those is valuable, and the interests of this work go

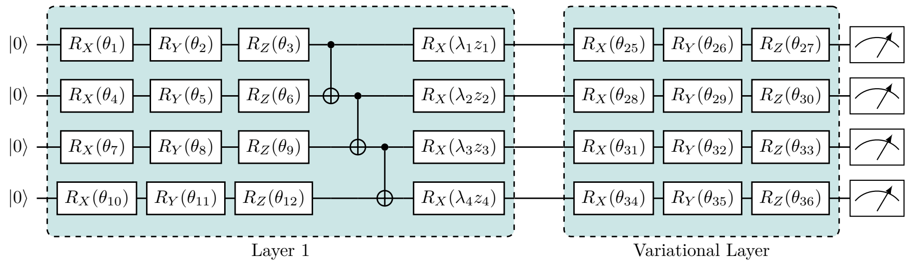

$|0\rangle$
$|0\rangle$
$|0\rangle$
$|0\rangle$
$R_X(\theta_1)$ $R_Y(\theta_2)$ $R_Z(\theta_3)$ $R_X(\lambda_1 z_1)$
$R_X(\theta_4)$ $R_Y(\theta_5)$ $R_Z(\theta_6)$ $R_X(\lambda_2 z_2)$
$R_X(\theta_7)$ $R_Y(\theta_8)$ $R_Z(\theta_9)$ $R_X(\lambda_3 z_3)$
$R_X(\theta_{10})$ $R_Y(\theta_{11})$ $R_Z(\theta_{12})$ $R_X(\lambda_4 z_4)$
$R_X(\theta_{25})$ $R_Y(\theta_{26})$ $R_Z(\theta_{27})$
$R_X(\theta_{28})$ $R_Y(\theta_{29})$ $R_Z(\theta_{30})$
$R_X(\theta_{31})$ $R_Y(\theta_{32})$ $R_Z(\theta_{33})$
$R_X(\theta_{34})$ $R_Y(\theta_{35})$ $R_Z(\theta_{36})$
Layer 1
Variational Layer

Figure 5.2: Example of a parameterized quantum circuit with 4 qubits and 1 layer used as the generator in the QGAN. Each layer consists of single-qubit Pauli rotations, CNOT gates and data uploading gates, which upload particular realizations of the random noise $\vec{z}$. Each element of $\vec{z}$ is uniformly sampled in $[0, 2\pi]$. For parameterized quantum circuits with more layers, each additional layer follows layer 1 and has the same layout. Before the measurement, there is a variational layer consisting of single-qubit Pauli rotations. The measurements are conducted single-qubit Pauli-X and Pauli-Z bases. The single-qubit Pauli rotations are tunable with parameters θ_i and the data-uploading gates are tunable with parameters λ_i. This Ansatz is a hardware-efficient Ansatz, which is commonly chosen in the field [21, 238].

into this direction.

A parameterized quantum circuit (PQC) consists of tunable and fixed gates, and measurements at the end. The parameters of the tunable gates are updated based on the loss function, which consists of the measurement results and the output of the discriminator.

The PQC can be used in different ways, for example as a quantum circuit Born machine [223, 232, 233] or as an expectation value sampler [234–237]. The latter approach is the one which we use for our QGANs. In the former case, a quantum circuit is used to learn an underlying distribution and every single sample forms a bit string corresponding to the learned distribution. The probability of each sample depends on the amplitudes of the final quantum state. However, the precision of the generated values is limited by the discrete nature of this approach. In contrast, the expectation value sampler identifies expectation values of quantum circuit measurement outcomes with samples of a distribution. The underlying randomness comes from classical noise uploaded to the quantum circuit.

A sketch of the PQC architecture used in this chapter is given in Fig. 5.2.

5.2.3 Related work

In recent years, classical algorithms for generating synthetic financial data have been proposed and explored, in GAN settings [78, 226, 239–241], and with other approaches [78, 242]. A common challenge is the generation of time series that exhibit all stylized facts sufficiently well [78].

QGANs were introduced in [220] and [221], substituting generator and discriminator with quantum circuits. Other approaches for QGANs, such as combining a classical discriminator with a quantum generator, and their applications have been explored as well [243, 244]. In [245] the generation of certain probability distributions and in [246], the generation of correlated stocks has been examined.

Our work was motivated by [223], which compared the performance of generating synthetic financial data of correlated asset pairs by the two models of restricted Boltzmann machines and quantum circuit Born machines, observing an advantage of the quantum circuit Born machine for comparable model sizes. Here we go beyond learning time-aggregated distributions of financial time series as in [223], by additionally examining the temporal correlations of generated time series. In contrast to the models used in [223], our method is based on the expectation value sampler, which was introduced in [234], proven to be universal in [236] and further generalized in [237]. An expectation value sampler outputs Pauli string expectation values in the range $[-1, 1]$, producing continuous variables. This is fundamentally different from quantum circuit Born machines, which generate discrete bit strings according to the Born rule, and Boltzmann machines, which also produce discrete outputs [223]. In Appendix 5.B, we adapted our approach for learning the distribution of correlated pairs of foreign exchanges and compare our results with the results of [223].

QGANs have also been used for other applications, such as image generation [247–250] and other discrete distributions [251], in generative chemistry [252], fraud detection [253], option pricing [254], and high-energy physics [255, 256]. Furthermore, other quantum machine learning strategies have been used in learning financial time series [257].

The Wasserstein QGAN, proposed in [258, 259] by substituting both generator and discriminator with a quantum circuit, shows improvement in training stability and efficiency compared to QGANs based on other metrics. Our simulations are using Wasserstein QGANs in which the generator is a PQC, whereas the discriminator is a classical neural network. The full-state simulation of such a Wasserstein-QGAN with gradient penalty as an application to generate financial time series has been explored in [260] and compared to classical GANs.

5.3 Implementation

For this chapter, we aim to generate time series whose distribution approximates the empirical distribution of real financial data. We train a Wasserstein QGAN for generating time series based on training data originating in the time series of the daily closing prices of the S&P 500 index [75], collected from 03.01.1950 until 29.09.2021. We use a hybrid approach, with a classical neural network as a discriminator and a parameterized quantum circuit (PQC) as a generator.

We use a convolutional neural network as a discriminator, motivated by [239], where it was used as the discriminator of a GAN that generates financial time series. All its activation functions are rectified linear units, except the last single neuron in the critic, which uses a linear activation function. The architecture of the discriminator is detailed in Appendix 5.A.

We simulated the QGAN based on a full simulation of the PQC by using the Tensorflow software library [261], and the QGAN based on the MPS approximation of the PQC with the JAX [262] and Quimb [263] software libraries. The gradients are calculated via automatic differentiation. As an optimizer for the training of the QGAN, we chose the Adam optimizer with a learning rate of 10^{-3}. All simulations were conducted on the Alice and XMARIS computing clusters at Leiden University.

In the following, we outline the data pre-and post-processing, the setup of the quantum generator, the full-state simulation and the MPS simulation applied in this chapter.

5.3.1 Data pre-and post-processing

The outputs of expectation value samplers are the expectation values of Pauli strings which lie in the range $[-1, 1]$. As this is a key difference to the raw time-series data in the form of log returns, which have unbounded support (see Eq. (5.1)), we perform data pre-and post-processing. For that, we follow the same approach as taken in Refs. [240, 264].

We first describe the approach for the pre-processing, which transforms the raw time series data of the S&P 500 index into training data used in our numerical simulations. This process consists of six steps: (i) data normalization, (ii) the inverse Lambert-W transform, (iii) data normalization, (iv) data clipping, (v) data rescaling and (vi) a rolling window.

(i) We normalize the time series data to have a mean of 0 and a variance of 1:

$$r_{t,(i)} := \frac{r_t^{(SP500)} - \mu_r}{\sigma_r}, \tag{5.12}$$

where $r_t^{(SP500)}$ is the original log return, calculated from the daily closing prices S_t of the S&P 500 index by $r_t^{(SP500)} = \log\left(\frac{S_t}{S_{t-1}}\right)$. By μ_r and σ_r, we denote the estimates of the mean and standard deviation of the log returns over the whole period of collected time series data (03.01.1950-29.09.2021).

(ii) As learning a heavy-tailed distribution can be challenging due to a limited number of samples in the tails, we implement the inverse Lambert-W transform on the normalized log returns. This transformation will bring the heavy-tailed distributed data closer to a Gaussian distribution. Given Lambert's W function, which is the inverse of $z = u \exp(u)$ with $z : \mathbb{R} \to \mathbb{R}$, we can define the following transform on the normalized heavy-tailed data set $V = \{r_{t,(i)}\}_t$:

$$W_\delta(r_{t,(i)}) := \operatorname{sgn}(r_{t,(i)}) \left(\frac{W(r_{t,(i)}^2 \delta)}{\delta} \right)^{1/2} \tag{5.13}$$

with $\delta \geq 0$ a tunable parameter, $\operatorname{sgn}(r_{t,(i)})$ the sign of $r_{t,(i)}$ and W the Lambert's W function. The inverse of this function is given by $r_{t,(i)} = W_\delta(r_{t,(i)}) \exp\left(\frac{\delta}{2} W_\delta(r_{t,(i)})^2\right)$ [265]. Throughout this chapter, we pick $\delta = 0.5$.

(iii) We normalize the transformed time series again such that it obtains a mean of 0 and a variance of 1:

$$r_{t,(iii)} := \frac{W_\delta(r_{t,(i)}) - \mu'_r}{\sigma'_r}, \tag{5.14}$$

where by μ'_r and σ'_r, we denote the estimates of the mean and standard deviation of the transformed time series $\{W_\delta(r_{t,(i)}\}_t$.

(iv) As the inverse Lambert-W transform can be ill-behaved at specific data points, we discard outliers with large deviations outside the 0.05% tails.

(v) Afterwards, we linearly map the data to the interval $[-1, 1]$. Let min and max denote the minimum and maximum values of the set $\{r_{t,(iv)}\}_t$, respectively. The transformation is given by

$$r_{t,(v)} = 2\, \frac{r_{t,(iv)} - \min}{\max - \min} - 1\,, \tag{5.15}$$

where $\{r_{t,(iv)}\}_t$ is the time series obtained after step (iv).

(vi) After these transformations of the log returns of the S&P 500 index, we divide the time series into smaller batches. We achieve this by applying a rolling window of window length m and stride s to the time series, which divides it into multiple subsequences. This creates subsequences with length m, which overlap and consequently correlate if the stride is shorter than the length of the window, $s < m$. For each of these subsequences, we then compute its probability distribution, which constitutes a sample of the training data set. Although the correlation between training samples is not ideal, the more extensive set of training samples can be beneficial for model performance. Throughout all simulations shown in this chapter, we used a stride of $s = 5$ and a window length of $m = 20$ and $m = 40$.

One sample of the resulting training data set is thus a subsequence of length 20 or 40 of the transformed time series of daily log returns of the S&P 500 index.

Each generated sample consists of the expectation values of $2n$ Pauli operators, $\{r_{t,PQC}\}_{t=1}^{2n} = \{\langle X \rangle_1, \langle Z \rangle_1, \langle X \rangle_2, \langle Z \rangle_2, ...\}$ from a PQC with n qubits (see Sec. 5.3.2), which are then post-processed by taking the following steps: (i)* data rescaling, (ii)* data renormalization, (iii)* forward Lambert transform, (iv)* data renormalization.

(i)* The data is rescaled by reversing step (v) in the pre-processing: The inverse mapping is given by

$$r_{t,(i)^*} = \frac{r_{t,PQC} + 1}{2} (\max - \min) + \min \,, \tag{5.16}$$

where $\{r_{t,PQC}\}_{t=1}^{2n}$ are the measured expectation values.

(ii)* The resulting set is normalized by reverting step (iii):

$$r_{t,(ii)^*} = \sigma'_r \, r_{t,(i)^*} + \mu'_r \,, \tag{5.17}$$

where μ'_r and σ'_r are calculated in the pre-processing step (iii).

(iii)* The inverse Lambert-W transformation is reversed by applying

$$r_{t,(iii)^*} = r_{t,(ii)^*} \exp\left(\frac{\delta \, r^2_{t,(ii)^*}}{2}\right) \,, \tag{5.18}$$

where δ is the parameter that we fix to 1/2 throughout the chapter.

(iv)* Finally, we also reverse the first normalization:

$$r_{t,gen} = r_{t,(iv)^*} = \sigma_r \, r_{t,(iii)^*} + \mu_r \,, \tag{5.19}$$

where μ_r and σ_r are the mean and standard deviation of the original time series.

At the end of the post-processing, each sample is a time series of length $2n$, which we write as:

$$\begin{aligned} r_{\text{gen}} &= \{r_1, r_2, r_3, r_4, ..., r_{2n}\} \\ &= \{p(\langle X \rangle_1), p(\langle Z \rangle_1), ..., p(\langle X \rangle_{2n}), p(\langle Z \rangle_{2n})\} \,, \end{aligned} \tag{5.20}$$

where by $p(\cdot)$ we denote the post-processing.

5.3.2 Quantum generator and full-state simulation

As a generator of the QGAN, we chose a PQC with an architecture that is sketched in Fig. 5.2, based on the hardware efficient Ansatz [21, 238]. The qubits are initialized in the $|0\rangle$ state. Each layer consists of single-qubit Pauli rotations, CNOT gates connecting nearest neighbors and noise encoding gates. The latter encode each a uniformly distributed noise sample with single-qubit rotations with trainable parameters. Such circuit architectures suffer from barren plateaus when scaled up in the number of qubits and layers [266]. Therefore, we do not consider them to be scalable in their current form. Instead, our investigation should be understood as establishing lower bounds on what can be achieved with quantum circuits: if these perform well at small scales, it motivates efforts to refine them for improved trainability at larger scales. Conversely, if they fail to perform even at small scales, this indicates that the application may be less promising than one might have hoped.

In Sec. 5.4, we present results from training circuits with 10 and 20 qubits and between 1 and 18 layers. This choice of the number of qubits and layers makes the QGANs classically simulatable. After the n-th layer, we apply single-qubit Pauli rotations and we measure each qubit in two bases: the Pauli-Z basis and the Pauli-X basis. We chose these measurements in order to enable the simulation of longer time series using fewer qubits. The exact consequences in terms of expressivity and potential greater sampling costs was discussed in [236, 267].

For the training of the QGAN and the analysis of the generated data at the end of the training, which we present in Sec. 5.4, we create samples of generated time series, each with a different random noise input. First, we collect the expectation values $\{\langle X\rangle_1, \langle Z\rangle_1, \langle X\rangle_2, \langle Z\rangle_2, ...\}$ at the end of the PQC, where $\langle X\rangle_i, \langle Z\rangle_i \in [-1, 1]$ are the expectation values of the measurement in the Pauli-X and Pauli-Z basis on the i-th qubit, respectively. Subsequently, we post-process the data as described in Sec. 5.3.1. After the post-processing, the obtained set $\{p(\langle X\rangle_1), p(\langle Z\rangle_1), p(\langle X\rangle_2), p(\langle Z\rangle_2), ...\}$, where $p(\cdot)$ stands as the post-processing map, constitutes one sample of a generated time series r_{gen} (see also Eq. (5.20)). A circuit of n qubits thus generates a time series of length $2n$. By generating multiple such time series samples from different random noise inputs, the QGAN approximates the empirical distribution of time series of the same window length in the training data.

The first part of our simulations is based on the full-state simulation of PQC. Let $|\psi_{\text{full}}\rangle$ be the quantum state that describes the state at the end of the parameterized quantum circuit (PQC) used in our quantum generative adversarial network (QGAN). In the computational basis, it takes the form

$$|\psi_{\text{full}}\rangle \approx \sum_{i_1, i_2, ..., i_n} c_{i_1, i_2, ..., i_n} |i_1, i_2, ..., i_n\rangle \ , \tag{5.21}$$

where $i_j \in \{0, 1\}$. The number of coefficients $c_{i_1, i_2, ..., i_n}$ of this state, and thus the memory requirement, scales exponentially with the number of qubits n. Moreover, the time cost of the classical full-state simulation scales linearly with the number of layers. This makes the full-state simulation quickly infeasible.

5.3.3 Matrix product state simulation

For being able to simulate PQC with a higher number of layers and qubits, we use matrix product states (MPS) as efficient approximation methods under some circumstances [224, 225, 268], which in the context of machine learning is also called the tensor-train decomposition [269]. They provide a compact representation of quantum states with limited entanglement and have been extensively used in physics [268]. This makes them well suited for simulating quantum states that are prepared by the PQC used in our QGAN.

An MPS represents an n-qubit quantum state $|\psi_{\text{full}}\rangle$ as a product of local tensors [268]:

$$|\psi_{\text{full}}\rangle = \sum_{i_1,\ldots,i_n} A^{[1]}_{i_1} A^{[2]}_{i_2} \cdots A^{[n]}_{i_n} |i_1 i_2 \cdots i_n\rangle\,, \tag{5.22}$$

where each $A^{[k]}_{i_k}$ is a $\chi_{k-1} \times \chi_k$-dimensional tensor. We call χ_k the bond dimensions of the MPS that controls the amount of entanglement the MPS can represent. The contraction of these tensors yields the amplitude corresponding to each computational basis state. For simplicity, we choose $A^{[1]}_{i_1}$ as $1 \times \chi$-dimensional, $A^{[n]}_{i_n}$ $\chi \times 1$-dimensional, and each remaining tensor $A^{[k]}_{i_k}$ with the dimensions $\chi \times \chi$. Such an MPS is described by $(2n-1)\chi^2 + 2\chi$ coefficients, which for constant χ scales linearly in the number of qubits, making it more efficient than the full-state simulation with 2^n coefficients. Fig. 5.3 provides a sketch of an MPS representation.

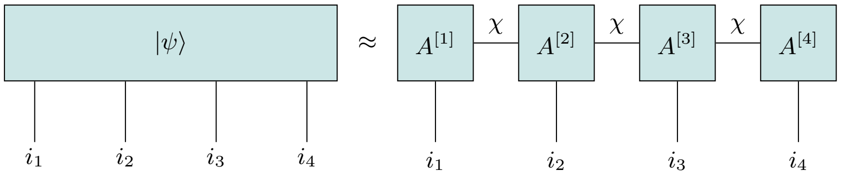


Figure 5.3: A matrix product state (MPS) consists of a chain of local tensors $A^{[j]}$ connected by virtual bonds of dimension χ, each with a physical leg representing a qubit index. A virtual bond represents a sum over a particular index of the tensors. The left term represents the state ψ, which lives in a 4-qubit space, and the right term shows its MPS approximation.

We simulate the quantum circuit using MPS in the following way: We start with the trivial tensor corresponding to the initial state of the circuit $|0\rangle^n$. Then, we apply each layer sequentially to the MPS, on which single-qubit Pauli rotations are trivially applied. After each CNOT gate, we recalculate the tensors by singular value decompositions (SVD) [270]. We partition the system into left and right parts and apply SVD, truncate the number of singular values to the bond dimension χ_k, and the left unitary multiplied with the truncated singular value matrix forms the tensor $A^{[k]}_{i_k}$. After applying every layer in this way, we get an MPS approximation of the output state $|\psi_{\text{full}}\rangle$ of the PQC.

To assess the quality of the MPS approximation for systems that can efficiently be simulated with the full-state, we compute the fidelity between the full quantum state $|\psi_{\text{full}}\rangle$, obtained from an exact state vector simulation, and the MPS-approximated state $|\psi_{\text{MPS}}\rangle$:

$$F = |\,\langle\psi_{\text{full}}|\,|\psi_{\text{MPS}}\rangle\,|^2. \tag{5.23}$$

We evaluate this fidelity for various values of the maximum bond dimension χ. As shown in Fig. 5.4, the fidelity increases with χ, indicating improved approximation accuracy. For a higher number of layers, the PQC increases the entanglement across the qubits, which decreases the fidelity for fixed bond dimension. For sufficiently large χ ($\chi = 32$ for 10 qubits), the MPS becomes numerically indistinguishable from the full-state [271].

In general, for a higher number of layers and qubits, it is not possible anymore to calculate this fidelity as it is not feasible to simulate the full-state. Instead, one can calculate the fidelity between MPS simulations of different bond dimensions, and choose the lowest bond dimension at which this fidelity does not increase anymore.

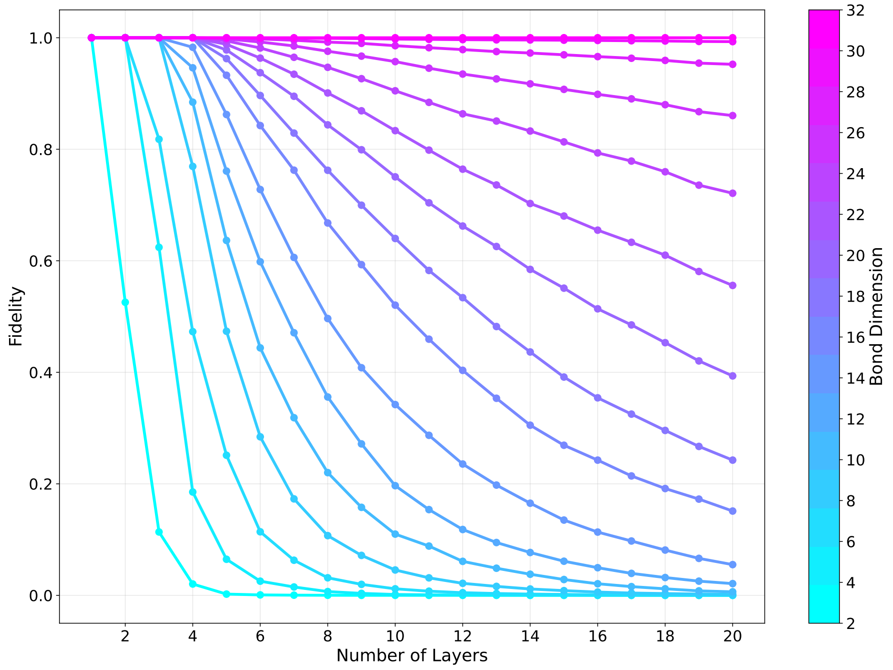


Figure 5.4: Fidelity between the exact quantum state prepared by a PQC consisting of 10 qubits, in the architecture as sketched in Fig. 5.2 and the MPS approximation as a function of the depth of the PQC, for different bond dimensions χ.

This tunability of the bond dimension provides a practical trade-off between simulation efficiency and accuracy. In the context of training the QGAN, where the PQC must be evaluated many times, moderate bond dimensions already yield sufficiently high fidelity while significantly reducing computational cost.

5.4 Results

In this section, we describe the results of our simulations. We simulated the parameterized quantum circuit (PQC) with the architecture shown in Fig. 5.2 in two ways, as described in Sec. 5.3. First, we performed full-state simulations for systems with up to 10 qubits and up to 6 layers. Second, we used matrix product state (MPS) simulations for systems with 10 and 20 qubits and between 1 and 18 layers. We performed 5 runs for each of these simulations, and describe their results in separate subsections. The results are discussed in Sec. 5.5.

5.4.1 Full-state simulation

For the full-state simulation, we chose a PQC consisting of 10 qubits and 8 layers. Therefore, for the generation of the training set from the historical S&P 500 time series, we set the window size to 20. In 5 runs, we trained a QGAN for 7900 epochs, and plotted the metrics of the generated time series of the best out of these 5 runs in Fig. 5.5. For this figure (and also for Figs. 5.8, 5.9, 5.11 and 5.12), we generated 1500 samples (each with different random noise inputs) of the time series, and calculate the correlations as the average of all samples and of all pairs of lags in each time series. The confidence intervals are calculated as in [272, p.51].

In **(a)**, we plot the probability density functions and in **(b)** the quantile-quantile plot of both the S&P 500 index and the generated time series. In **(c)**-**(h)**, we plot the metrics absolute autocorrelation, linear autocorrelation and the leverage effect, as an indication of the stylized facts as described in Sec. 5.2.1. The Subfigures **(c)**, **(e)** and **(g)** show the metrics of the S&P 500 index and the Subfigures **(d)**, **(f)** and **(h)** the metrics of the generated time series, respectively. Confidence intervals are calculated as in [272].

The generated time series closely resembles the distribution of the S&P 500 index, as shown in Subfigures **(a)** and **(b)**. Similar to the S&P 500 index, the generated time series shows a weaker, but decaying absolute autocorrelation (Subfigures **(c)** and **(d)**) and does not show linear autocorrelation (Subfigures **(e)** and **(f)**). The leverage effect, which is negative and increasing in the S&P 500 index (Subfigure **(g)**, is also reproduced in a weaker way in the generated time series (Subfigure **(h)**). As can be seen in Fig. 5.6, both the loss function and the temporal metrics decrease with the number of epochs, indicating stable training of the QGAN.

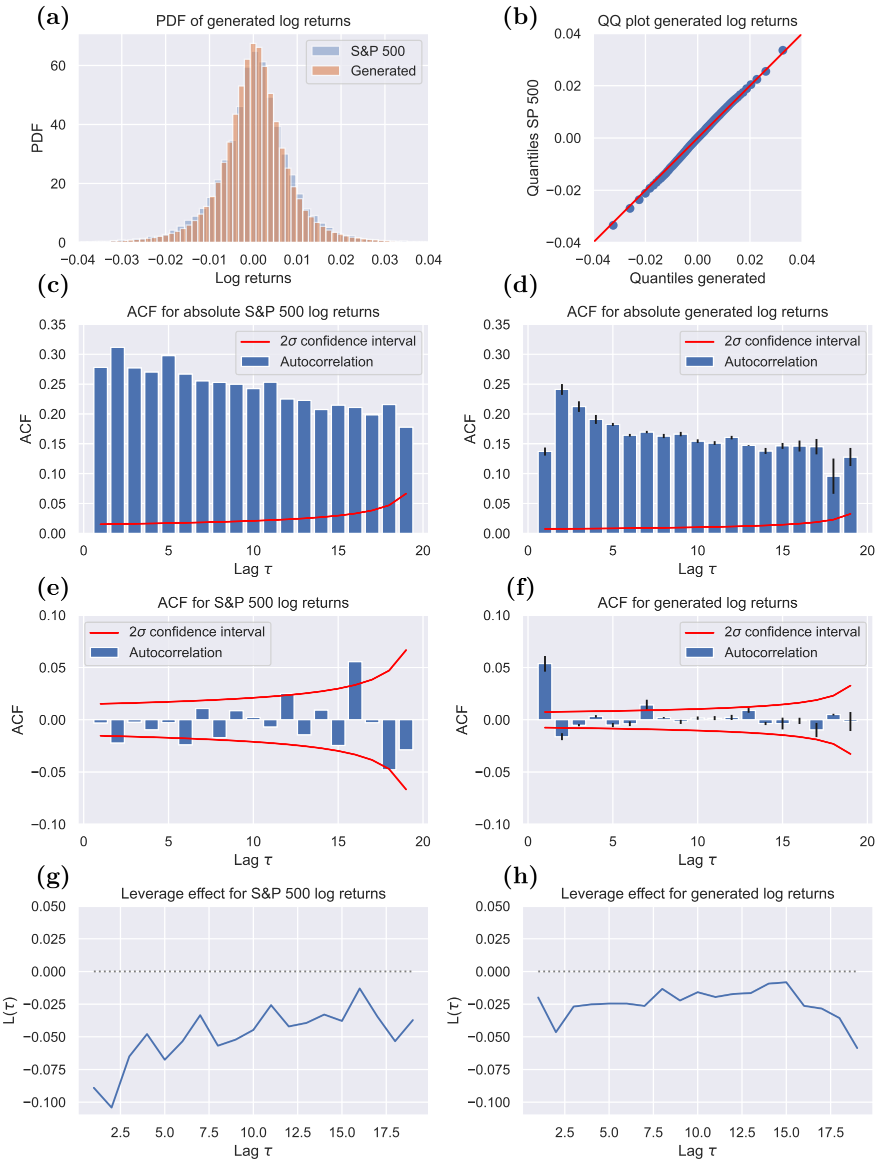


Figure 5.5: Metrics of the stylized facts for a synthetic time series of window size 20 generated by a QGAN, compared to the metrics of the S&P 500 index. The generator of the QGAN is a PQC based on the architecture shown in Fig. 5.2 consisting of 10 qubits and 8 layers, simulated with the full-state approach.

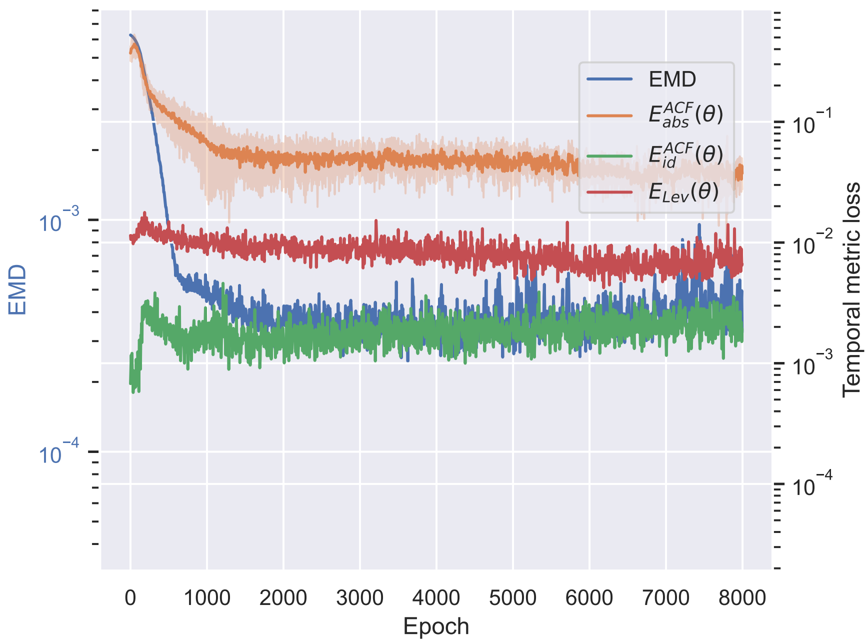


Figure 5.6: Wasserstein loss as defined in Eq.(5.9) (here called the EMD) and metrics corresponding to the temporal correlations as described in Sec. 5.2.1 in the training of the QGAN in the full-state simulation with 10 qubits and 8 layers (see Fig. 5.5 for the final metrics), depending on the number of epochs. We show the mean and standard deviation of the 5 training runs.

We show the metrics of the best out of 5 runs that correspond to the time series as shown in Fig. 5.5 in column **(a)** of Table 5.1.

Metrics	**(a)**	**(b)**	**(c)**	**(d)**	**(e)**
EMD	$3.3 \cdot 10^{-4}$	$6.1 \cdot 10^{-4}$	$2.7 \cdot 10^{-4}$	$2.9 \cdot 10^{-4}$	$4.2 \cdot 10^{-3}$
$E_{id}^{ACF}(\theta)$	$1.9 \cdot 10^{-3}$	$2.4 \cdot 10^{-3}$	$1.5 \cdot 10^{-3}$	$4.5 \cdot 10^{-4}$	$1.1 \cdot 10^{-3}$
$E_{abs}^{ACF}(\theta)$	$3.7 \cdot 10^{-2}$	$5.5 \cdot 10^{-2}$	0.15	0.31	0.99
$E_{Lev}(\theta)$	$6.6 \cdot 10^{-3}$	$5.8 \cdot 10^{-3}$	$4.7 \cdot 10^{-3}$	$2.2 \cdot 10^{-2}$	$4.4 \cdot 10^{-2}$

Table 5.1: Comparison of different metrics defined in Eqs. (5.3)-(5.6) for the best out of the 5 runs of **(a)** the full-state simulation with 10 qubits and 8 layers as in Figs. 5.5 and 5.6, **(b)** the full-state simulation with 10 qubits and 8 layers with a different circuit architecture as described in Appendix 5.C, **(c)** the full-state simulation with 10 qubits and 8 layers based on a circuit with CZ gates instead of CNOT gates as described in Appendix 5.D, **(d)** the MPS simulation with 10 qubits, 18 layers and a bond dimension 32 as in Figs. 5.8 and 5.7, **(e)** the MPS simulation with 20 qubits, 6 layers and a bond dimension 70 as in Fig. 5.9.

The Wasserstein QGAN does not explicitly account for temporal effects, so any such structure in the generated time series must result from other aspects of the model. To investigate the influence of the PQC architecture on these temporal effects, we trained a QGAN with a different PQC and present the results in Appendix 5.C. We indeed see that the absolute autocorrelation of the generated time series increases at larger time lags, and the leverage effect is less pronounced in comparison to the time series generated in Fig. 5.5. In contrast, no substantial difference in the quantile-quantile plots and the absolute autocorrelation can be observed. See Table 5.1 for a comparison of the metrics, which are higher than for the simulation shown in Fig. 5.5 apart from the leverage effect.

We also trained a QGAN with the full-state simulation based on a circuit that uses control-Z (CZ) gates instead of CNOT gates, and show the results in Appendix 5.D. The absolute autocorrelation decreases faster, and the leverage effect is more closely pronounced compared with the results shown in Fig. 5.5.

Additionally, to compare with the results of a GAN based on a quantum circuit Born machine [223], we trained the QGAN (based on the circuit with CZ gates instead of CNOT gates) on generating currency pairs; the results are shown in Appendix 5.B.

We analyze these results in Sec. 5.5.

As explained in Sec. 5.3, full-state simulation of PQCs quickly becomes infeasible as the number of layers and qubits increases. In the following, we describe MPS-based simulations, which make it feasible to simulate PQCs with larger numbers of layers and qubits.

5.4.2 MPS simulation

For the MPS simulation, we first chose a PQC of 10 qubits, with a varying number of layers (1, 5, 10 and 18) and different bond dimensions (1, 8, 16, 24 and 32) of the MPS. The MPS simulation of PQCs with 10 qubits generates time series with the same window size as the full-state simulation, making the results directly comparable. For higher numbers of layers and bond dimensions below $\chi = 32$, the MPS simulation is also faster than the full-state simulation.

For higher bond dimensions and number of layers, the training time in the MPS simulation increases, and the stylized facts of the generated time series vary considerably for each choice of the number of layers and bond dimension. See in Appendix 5.E for a comparison of the Wasserstein distance and metrics for the temporal effects for simulations of different numbers of layers and bond dimensions. In Fig. 5.8, we show the metrics of a generated time series from a well-performing QGAN that is trained for 7032 epochs, whose PQC consists of 18 layers and is simulated as an MPS with bond dimension 32. The metrics of this generated time series are shown in column **(c)** of Table 5.1. We chose to show the results for this particular model, as they match the stylized facts of the time series of the S&P 500 index qualitatively well, and as it proves that it is possible to train a QGAN for which the PQCs in the MPS simulation has more layers than what would be feasible with the full-state simulation.

The quantile-quantile plot shows that the generated time series matches the distribution of the S&P 500 index closely. In contrast to the time series generated with the full-state simulation shown in Fig. 5.5, the absolute autocorrelation (Subfigure **(d)**) that indicates volatility clustering is lower, but also decreasing for all time lags. Also the leverage effect is weaker than in the time series generated by the full-state simulation. The quantitative metrics decrease more slowly during training compared to the full-state simulation, as can be seen in Fig. 5.7.

Across the QGANs trained with different numbers of layers and bond dimensions in the MPS simulation, we generally observe that the generated time series reproduces the distribution, absence of linear autocorrelation, and volatility clustering, while the leverage effect is less pronounced.

In order to show that MPS can also be used for simulating QGANs that can generate time series with a larger window, we trained a QGAN with the MPS simulation of a PQC that consists of 20 qubits. Such a simulation would be infeasible with full-state simulation. We show the results of this simulation in Fig. 5.9 and in column **(d)** of Table 5.1. Since increasing the number of qubits and the bond dimension raises the time required to train each epoch, the QGAN is trained for only 650 epochs.

In the following section, we will analyze and compare the results of the different simulations shown here.

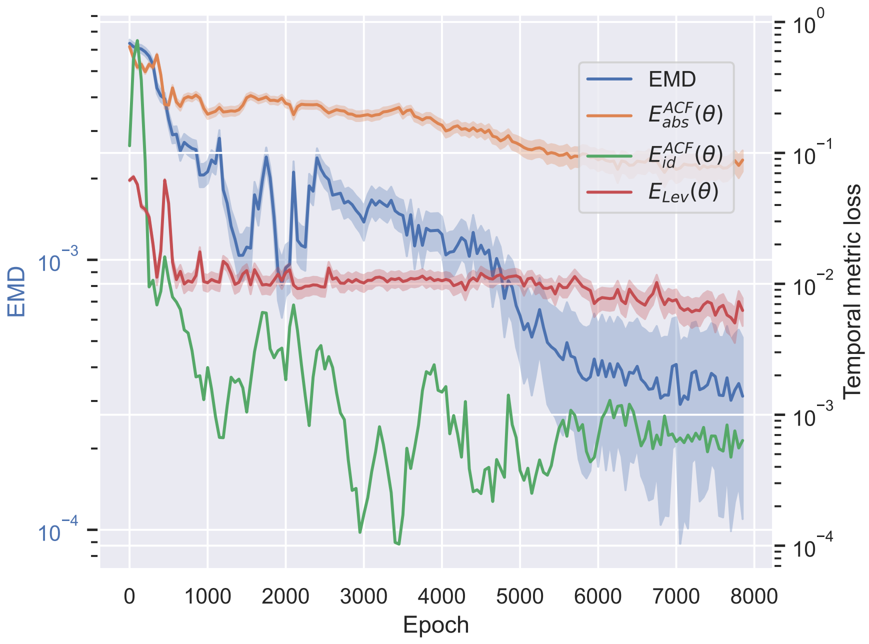


Figure 5.7: Wasserstein loss as defined in Eq. (5.9) (here called the EMD) and metrics corresponding to the temporal correlations as described in Sec. 5.2.1 in the training of the QGAN in the MPS simulation with 10 qubits, 18 layers and a bond dimension of 32 (see Fig. 5.8 for the final metrics), depending on the number of epochs. We show the mean and standard deviation of the 5 training runs.

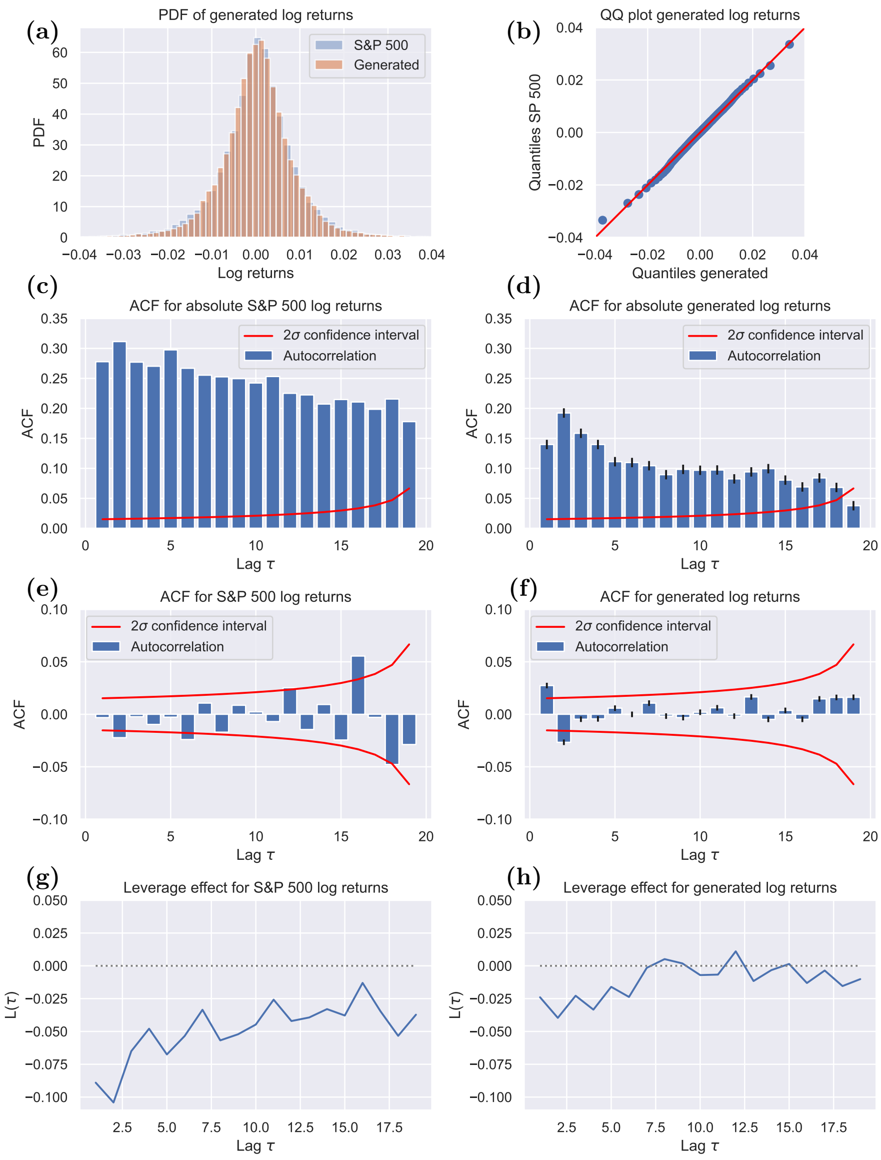


Figure 5.8: Metrics of the stylized facts for a synthetic time series of window size 20 generated by a QGAN, compared to the metrics of the S&P 500 index. The generator of the QGAN is a PQC consisting of 10 qubits and 18 layers, simulated with the MPS approach with bond dimension 32.

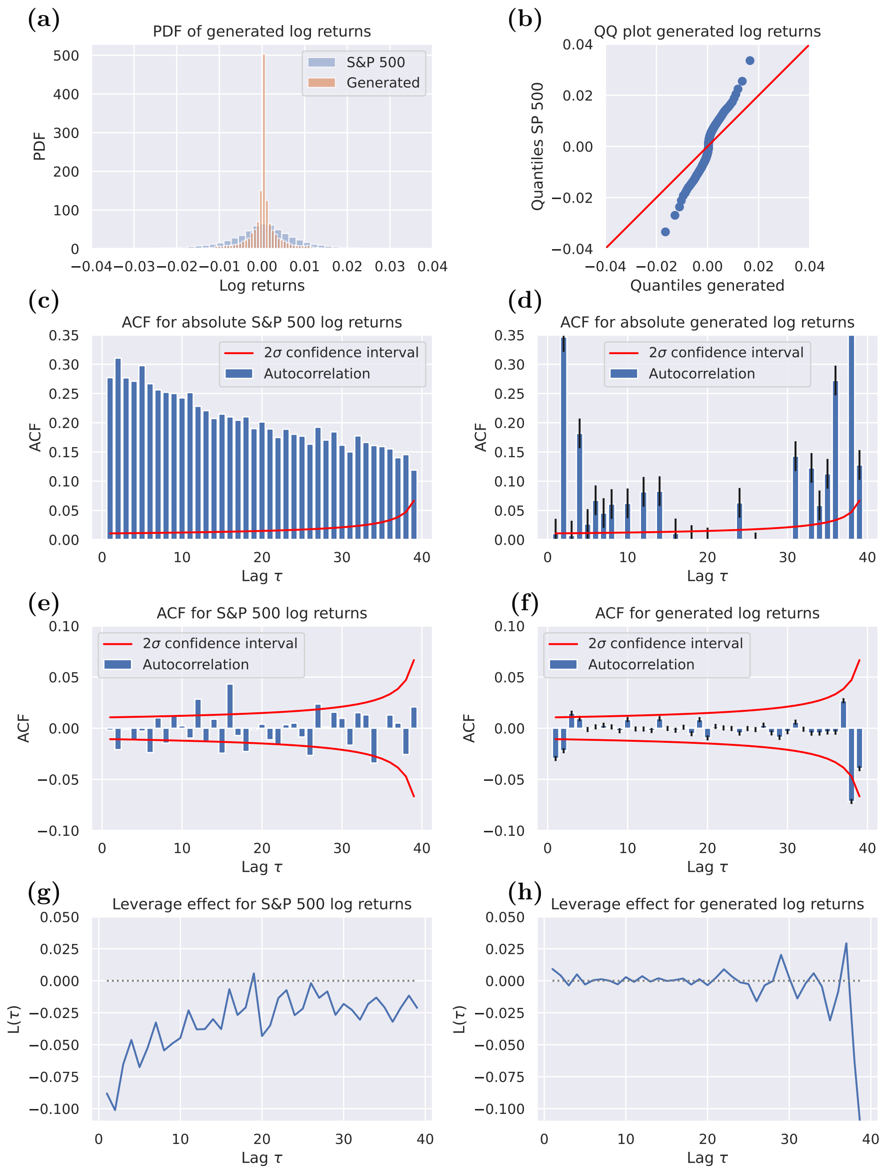


Figure 5.9: Metrics of the stylized facts for a synthetic time series of window size 40 generated by a QGAN, compared to the metrics of the S&P 500 index. The generator of the QGAN is a PQC consisting of 20 qubits and 7 layers, simulated with then MPS approach with bond dimension 70.

5.5 Analysis of the results

In all simulations, the probability distributions of the generated time series closely resembles the distribution of the S&P 500 index. The temporal correlations show significant differences between the simulations. While the absence of linear autocorrelations is visible in all simulated time series, their absolute autocorrelation (indicating volatility clustering) and the leverage are of different quality.

The full-state simulation (see Fig. 5.5) shows both effects, even though the absolute autocorrelation and the leverage effect are weaker than in the S&P 500 index.

The MPS simulation with 10 qubits, 18 layers, and bond dimension 32 (see Fig. 5.8) shows even weaker absolute autocorrelation and leverage effect.

Adding a CNOT gate between the first and the last qubits in each layer and performing the full-state simulation, leads to an increase in the qubit correlation (see Appendix 5.C). This might be a reason for the observation that the absolute autocorrelation of the generated time series increases at larger time lags. However, the leverage effect is pronounced weaker, and the other stylized facts do not differ substantially compared to the simulation shown in Fig. 5.5. Furthermore, in particular the Wasserstein loss is higher, as shown in Table 5.1, showing that the model generates time series that are further away from the real probability distribution. This proves that the architecture of the circuit indeed plays an important role on the quality of the generated time series.

The QGAN simulated with the full-state simulation which is based on a circuit that uses control-Z (CZ) gates instead of CNOT gates in Appendix 5.D, shows a faster decreasing absolute autocorrelation but more clearly pronounced leverage effect.

We benchmarked the QGAN with a full-state simulation against a quantum circuit Born machine in modeling the time-aggregated distribution of foreign exchange pairs yielding a better approximation of those distributions (see Appendix 5.B).

Using the MPS simulation, we also trained a QGAN with 20 qubits, 5 layers, and a bond dimension of 70 (see Fig. 5.9). This demonstrates that MPS can handle QGANs of greater complexity than those feasible with full-state simulation. However, the training of QGANs with PQCs of a higher number of layers and qubits and MPS of higher bond dimensions increases the number of epochs needed in the training. Additionally, each training epoch takes a longer time for these more complex models. For an equal computational cost, the generated time series therefore does not resemble the distributions and temporal effects of the target time series as closely as in the simulations with 10 qubits. But, by using a PQC with 20 qubits, it is possible to simulate time series with a larger window size of 40.

We remark that the loss landscapes differ significantly between full-state and

MPS simulations due to their different approximation and simulation structure. The difference in the quality of the generated time series can be partially attributed to the different features of the loss landscape.

Compared to the classical GAN experiments in [260], which use multi-layer dense neural networks as generators and either a multi-layer dense or convolutional neural network as discriminator (with the same specifications as described in Appendix 5.A), both of our quantum simulation methods yield qualitatively improved results, particularly with respect to the Wasserstein distance and volatility clustering, as observed in the plots of the stylized facts. Note that the window size used in the classical experiments differs from ours, which may influence the comparison.

5.6 Conclusions

We constructed a Wasserstein quantum generative adversarial network (QGAN) with a classical convolutional network as a discriminator and an expectation value sampler based on a parameterized quantum circuit (PQC) as a generator, in order to assess whether these quantum architectures have suitable inductive biases for generating synthetic financial time series known to be problematic for classical models. This approach leverages the PQC architecture to intrinsically capture temporal correlations in the time series, while the QGANs are trained solely on matching the aggregated distribution of the time series, by using a discriminator which learns the Wasserstein distance between the distributions of the generated time series and of the training data. We simulated a PQC with 10 qubits and 8 layers with a full-state simulation and a PQC with 10 and 20 qubits and with up to 18 layers as an approximation by a matrix product state (MPS) simulations with bond dimensions of up to 70. The latter approach allowed us to simulate PQCs with a higher number of layers and qubits, which makes it possible to train the generation of longer time series.

We compare the generated time series qualitatively with the S&P 500 index by their distributions and their temporal correlations, also called the stylized facts. These stylized facts are typically assessed qualitatively rather than quantitatively [78].

In this chapter, we showed that our trained QGANs generate time series that match the desired distributions and exhibit some of the temporal correlations seen in financial time series, such as in the S&P 500 index. Simulating the PQC with full-state simulations and MPS simulations yield different results, with circuit depth and the MPS bond dimension further influencing the performance. The three simulations performed with the full-state simulation show different behavior in particular of the absolute autocorrelation of the generated time series, indicating different qualities in capturing volatility clustering. The QGAN using the PQC given in Fig. 5.2 shows the closest match of this property (see

Fig. 5.5), whereas PQC architectures where an additional CNOT gate is added at the end of each layer leads to an increase in the absolute autocorrelation for higher time lags (see Fig. 5.11). Using CZ gates instead of CNOT gates in the PQC causes a quicker decrease of this effect (see Fig. 5.12). The MPS approach leads to weaker absolute autocorrelation and leverage effect compared to the full-state simulation (compare Fig. 5.5 and 5.8), but is able to simulate QGANs with a longer time window (see Fig. 5.9). Both simulation methods motivate the study of quantum hardware in their ability to generate financial time series with stylized facts. Our work has already motivated studies in which the effect of such generated data on the training of neural networks has been explored [273, 274].

The application of these QGANs as subroutines for applications such as option pricing [275] and risk analysis [276] can be explored as well. Furthermore, a possible extension of our method is to train the model to replicate correlated stocks of the S&P 500 index, motivated by research in community detection [277]. This could possibly be achieved by either learning the underlying distributions (in a similar way as done in Appendix 5.B), or by learning the individual time series similar to the ones in Sec. 5.4. As the number of qubits restricts the number of time steps and the number of stocks that can be generated, one could examine if quantum generators consisting of circuits on qudits can be successful, as that enables more independent measurements on each qudit. Specifically for qudits, not only superconducting qubits form a suitable experimental platform, but also trapped ions, neutral atoms and integrated photonics are excellent candidates for manipulating higher-dimensional quantum information [278, 279].

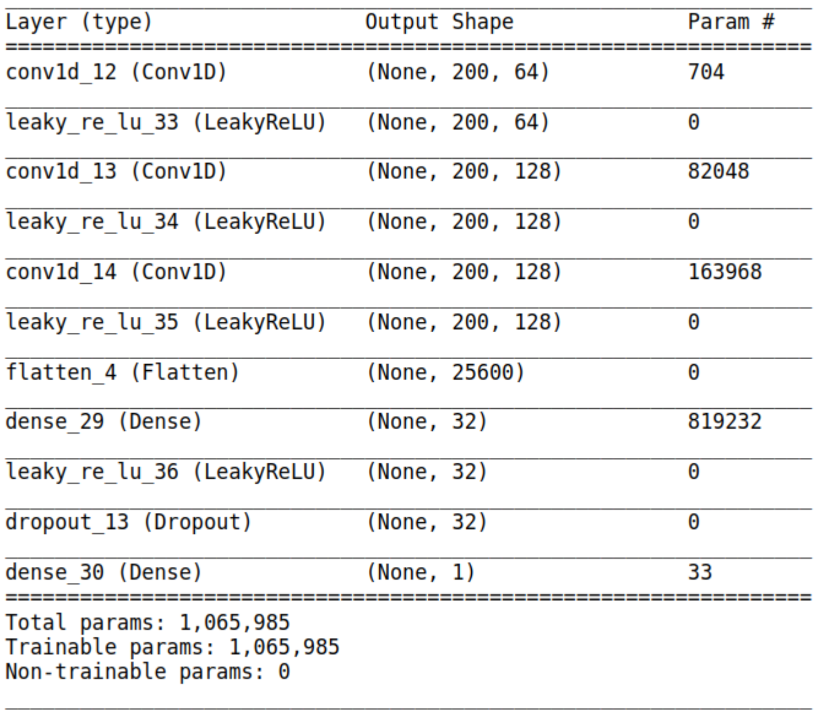
An improvement of the training of the QGANs could be achieved in several ways. Firstly, the effects of shot noise [237] in the training of the quantum generator could be explored. Secondly, different design choices, like choosing a different classical or quantum discriminator in the QGAN, diffusion model [242], or quantum long-short time memory models [280] might lead to different results. Thirdly, as the QGAN is trained with Wasserstein loss functions (see Eqs. (5.9) and (5.11)) that are taking the distribution of the time series into account, but not the temporal effects, an adaption of the training to consider them as well might lead to a better recovery of those temporal effects. In particular, it might be possible to not only gain a better match in the absolute autocorrelation and leverage effect, but also in the exact reproduction of the autocorrelation. Lastly, one could try different definitions of the quantum Wasserstein distance [281] that give theoretical improvements over the qualitative accuracy.

5.7 Code availability

The code supporting this chapter is available at the following repository: `https://github.com/LucasAugustusvd/Quantum-Finance`

5.A Architecture of the discriminator

We trained the classical discriminator in our QGAN simulations with a convolutional neural network. Table 5.2 summarizes its properties and hyperparameters. This choice is motivated by [239], where it successfully was applied as a discriminator of a GAN that generates financial time series.



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv1d_12 (Conv1D)           (None, 200, 64)           704
_________________________________________________________________
leaky_re_lu_33 (LeakyReLU)   (None, 200, 64)           0
_________________________________________________________________
conv1d_13 (Conv1D)           (None, 200, 128)          82048
_________________________________________________________________
leaky_re_lu_34 (LeakyReLU)   (None, 200, 128)          0
_________________________________________________________________
conv1d_14 (Conv1D)           (None, 200, 128)          163968
_________________________________________________________________
leaky_re_lu_35 (LeakyReLU)   (None, 200, 128)          0
_________________________________________________________________
flatten_4 (Flatten)          (None, 25600)             0
_________________________________________________________________
dense_29 (Dense)             (None, 32)                819232
_________________________________________________________________
leaky_re_lu_36 (LeakyReLU)   (None, 32)                0
_________________________________________________________________
dropout_13 (Dropout)         (None, 32)                0
_________________________________________________________________
dense_30 (Dense)             (None, 1)                 33
=================================================================
Total params: 1,065,985
Trainable params: 1,065,985
Non-trainable params: 0
_________________________________________________________________
```

Table 5.2: Hyperparameters and properties of the convolutional neural network used as the discriminator in the QGANs.

5.B Comparison with quantum circuit Born machine

In [223], a QGAN is constructed where the quantum generator was used as a quantum circuit Born machine. It was trained to generate distributions of foreign exchange pairs, producing samples that better matched the true distributions than those from a classical restricted Boltzmann machine with a comparable model size.

We also trained our QGAN, where the quantum circuit consisting of 4 qubits and 4 layers, is simulated with the full-state approach with CZ gates (instead of CNOT gates compared to Fig. 5.2), in reproducing the same pairs of foreign exchanges as in [223]. We trained the single-qubit Pauli-X and Pauli-Z observables on the distributions of the EUR/USD and the GBP/USD foreign exchange log returns, respectively. Fig. 5.10 shows the quantile-quantile plot comparing samples from our trained model with the target distribution. Our trained QGAN samples match the target distribution more closely than

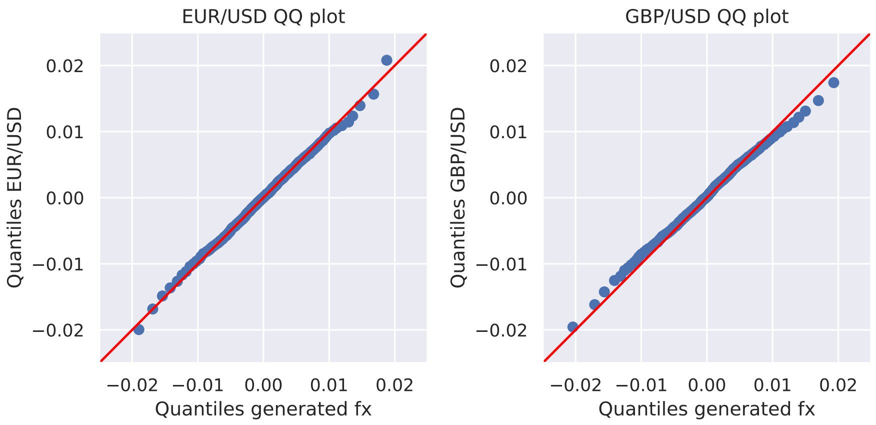


Figure 5.10: Quantile-quantile plot comparing samples from the trained QGAN model with a PQC of 4 qubits and 4 layers to the target distribution of EUR/USD and GBP/USD log returns.

the results for the quantum circuit Born machine and the classical restricted Boltzmann machine shown in Fig. 10 of [223], while using fewer qubits than used for the quantum circuit Born machine. This difference to the results from the quantum circuit Born machine comes from to the discrete nature of that model, which has naturally a higher imprecision of generated samples compared to the expectation value sampler used in our model.

5.C Full-state simulation: alternative circuit architecture

In addition to the PQC shown in Fig. 5.2, we trained a QGAN using a modified PQC architecture simulated with the full-state approach. In order to increase long-range qubit correlations, we added a CNOT gate between the first and 10th qubit in each layer of the PQC (results in Fig. 5.11). Subfigure **(d)** shows that this architectural change increases the absolute autocorrelation at larger time lags. The metrics of the generated time series are shown in column **(b)** of Table 5.1.

5.D Full-state simulation: CZ gates instead of CNOT gates

In Fig. 5.12 , we show the results of a full-state simulation using a circuit architecture in which the CNOT gates were substituted with control-Z (CZ) gates. Compare with the architecture sketched in Fig. 5.2 and the corresponding simulations shown in Figs. 5.5 and 5.6. Subfigure **(d)** shows that this architectural change leads to a faster decrease in the absolute autocorrelation. The metrics of the generated time series are shown in column **(c)** of Table 5.1.

5.E MPS simulations for different numbers of layers and bond dimensions

In Fig. 5.13, we show the quantitative metrics of training a QGAN where the PQC consisting of 10 qubits are simulated with the MPS approach for $1, 5, 10$ and 18 layers and bond dimensions of $1, 8, 16, 24$ and 32. See Sec. 5.4.2. Note that a bond dimension of 32 is giving an exact MPS approximation of the 10-qubit state.

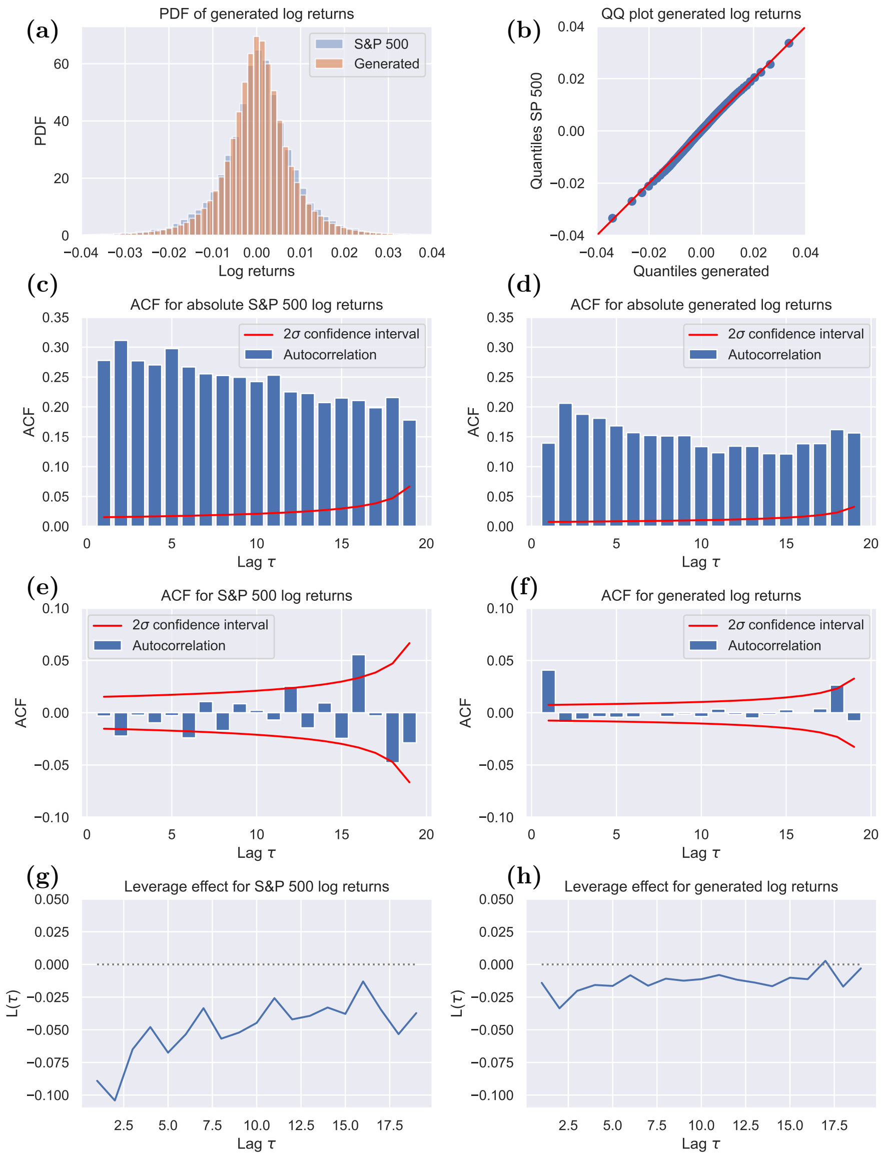


Figure 5.11: Metrics of the stylized facts for a synthetic time series of window size 20 generated by a QGAN, compared to the metrics of the S&P 500 index. The generator of the QGAN is a PQC consisting of 10 qubits and 8 layers, simulated with the full-state approach. Contrary to the PQC used in Fig. 5.5, we added an additional CNOT gate between the first and the 10th qubit in each layer.

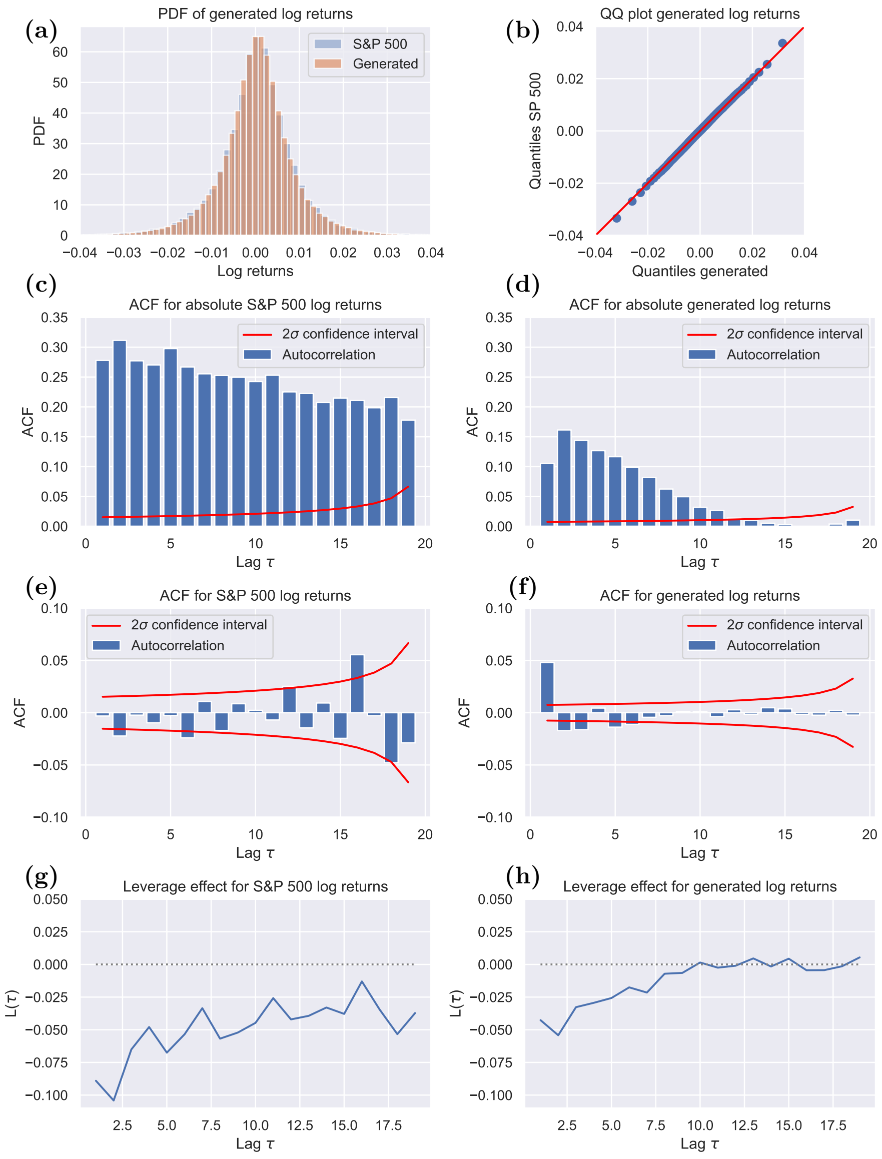


Figure 5.12: Metrics of the stylized facts for a synthetic time series of window size 20 generated by a QGAN, compared to the metrics of the S&P 500 index. The generator of the QGAN is a PQC based on the architecture shown in Fig. 5.2 consisting of 10 qubits and 8 layers and CZ gates instead of CNOT gates, simulated with the full-state approach.

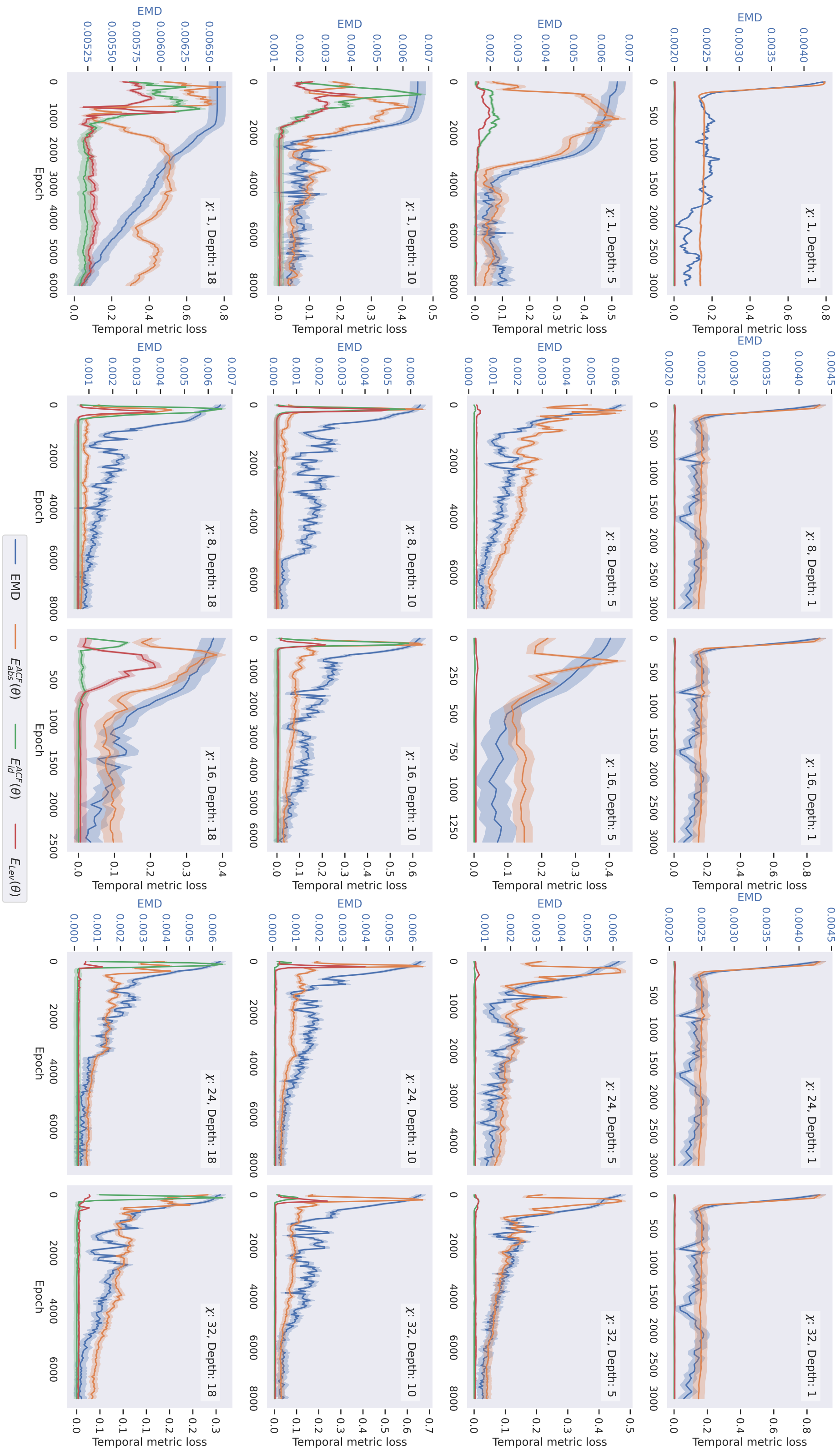


Figure 5.13: Wasserstein loss as defined in Eq. (5.9) (here called the EMD) and metrics corresponding to the temporal correlations as described in Sec. 5.2.1 in the training of the QGAN in the MPS simulation with 10 qubits, 1, 5, 10 and 18 layers and bond dimensions χ of $1, 8, 16, 24$ and 32, depending on the number of epochs.

CHAPTER 6

Conclusions

With this chapter, we conclude the thesis. We restate and conclude the research questions outlined in Sec. 1.4 in the beginning of this thesis. Finally, we re-outline promising directions of future work.

6.1 Research overview

This thesis dealt with several aspects of capturing dynamics with noisy quantum computers. We introduced several concepts as well as the research questions in Chapter 1, which we explored in Chapters 2, 3, 4, and 5. While we drew conclusions at the end of each of Chapters 2, 3, 4, and 5, we now come back to the specific research questions stated in the beginning of the thesis and reflect on them based on the results derived in the earlier chapters.

The first two research questions are concerned with theoretical properties of quantum machine learning:

Research Question 1: *Can parameterized quantum circuits approximate functions as well as their derivatives?*

In Chapter 2, which is based on the previously published work in Ref. [88], we answered this question. We showed that parameterized quantum circuits can not only approximate square integrable functions arbitrarily close in the L^2 distance, but also other function classes and with respect to other distances. In particular, by answering this research question, we proved that parameterized quantum circuits can approximate functions in the Sobolev space H^k with arbitrary precision under the Sobolev distance. Those spaces contain square-integrable functions whose partial derivatives up to order k are also square-integrable. The corresponding Sobolev distance is defined as the sum of the L^2 distances of the function and its partial derivatives up to order k. However, parameterized quantum circuits can only do so if certain conditions on the data input are met, for which we also provide a trivial solution.

Research Question 2: *How does an augmentation of the training data with derivatives of the target function influence generalization in quantum machine learning with parameterized quantum circuits?*

We also answered this question in Chapter 2, which is based on the results published in Ref. [88]. We proved a generalization bound that shows that a generalization of the approximation of both the function and its derivatives is possible if the training data includes not only labels of the target function, but also of its derivatives. Furthermore, we proved that including data of the derivatives of the target function also guarantees generalization bounds for the supremum and L^p distances. We found that the higher the dimension of the function, the more higher-order derivatives are required in order to achieve these bounds. These results give a theoretical explanation of earlier numerical findings that suggested improved generalization with classical neural networks [89], and thus also impact classical machine learning.

In the following research question, we focused on variational quantum algorithms for differential equations solving:

Research Question 3: *What is the total error arising in variational quantum algorithms for solving differential equations based on Runge-Kutta methods and which Runge-Kutta order minimizes the number of circuit evaluations needed?*

In Chapter 3, we provided an extensive error analysis and determined the resource requirements needed to achieve specific target errors, based on results published in Ref. [46]. In particular, we derived analytical error and resource estimates for scenarios with and without shot noise, examining shot noise in quantum measurements and truncation errors in Runge-Kutta methods. Our analysis did not take into account representation errors and hardware noise, as these are specific to the instance and the used device. We evaluated the implications of our results by applying them to two scenarios: classically solving a 1D ordinary differential equation and solving an option pricing linear partial differential equation with the variational algorithm, showing that the most resource-efficient methods are of order 4 and 2, respectively. We showed that even those most resource-efficient methods require a number of shots multiple orders of magnitude higher than what seems to be feasible on near-term quantum computers. However, those resource estimates might be lower in practice, as several error bounds on which these estimates are based on are not tight. Although our study minimizes resources for the upper bound we derive, we hope that the resulting prescription is a good heuristic for allocating resources in practice. The results may also be of interest to the numerical analysis community as they involve the accumulation of errors in the function description, a topic that has hardly been explored even in the context of classical differential equation solvers.

In many variational quantum algorithms, estimating reduced density matrices (RDMs) of the quantum system forms an important subroutine. However, this task is challenging, among others due to the effects of shot noise stemming from a limited number of measurements. Our following research questions therefore asked:

Research Question 4: *Is it possible to mitigate the effects of shot noise in the quantum state tomography of reduced density matrices by enforcing physicality conditions organized as semidefinite programs?*

In Chapter 4, we answered this question, which is addressed in Ref. [90]. We proposed a method to mitigate shot noise by reinforcing certain physicality constraints on RDMs. The first kind of these constraints, which we called the enhanced-compatibility, require RDMs to be compatible with higher-dimensional RDMs. Secondly, we included constraints that we called overlapping-compatibility which enforce that overlapping RDMs are consistent on those subsystems on which they overlap. We organized these compatibility constraints in semidefinite programs to reconstruct RDMs from simulated data. Our approach yields, on average, tighter bounds for the same number of measurements compared to tomography without compatibility constraints. We further demonstrated the versatility and efficacy of our method by integrating it into an algorithmic cooling procedure to prepare low-energy states of local Hamiltonians.

In the last research question of this thesis, we explored quantum approaches for generative modeling of dynamical systems of the financial market. In particular, we stated:

Research Question 5: *Can quantum generative adversarial networks capture the distribution and stylized facts of financial time series on a qualitative level?*

In Chapter 5, we presented our answer to this question, based on the results shown in Ref. [91]. We trained quantum generative adversarial networks

(QGANs), composed of a quantum generator and a classical discriminator, and used two classical simulation approaches for the quantum generator: a full simulation of the quantum circuits, and an approximate simulation of the latter using matrix product states. We tested the effect of the choice of circuit depths and bond dimensions of the matrix product state simulation on the generated time series. Overall, the QGANs were generally successful in capturing most of the temporal correlations observed in real financial data. But depending on the hyperparameters of the model, the generated synthetic financial time series differed in how well they reproduced the properties of financial time series. These differences allow selecting a model that best reproduces the desired properties of financial time series for specific applications.

6.2 Future work

Throughout this thesis, we suggested several directions in which the results of the thesis can be extended, at the end of each of Chapters 2, 3, 4, and 5. Let us summarize here the most important ones.

The exploration of the expressivity and generalization of parameterized quantum circuits in Chapter 2 motivates further study of their impact on applications such as differential equation solving and option pricing. Further, it might be promising to examine the role of other distances for both the expressivity and generalization of parameterized quantum circuits.

In Chapter 3, we did not take into account representation errors and hardware noise, although we gave several ideas and explanations in how to incorporate them in practice. It is therefore natural to ask how to categorize these error sources in a more systematic manner, by using techniques such as in [131]. We determined the act of classically inverting an (in general ill-conditioned) matrix with noisy elements as the key difficulty in this set of algorithms. Exploring alternative ways for this step might therefore significantly improve the chances of applying these and related algorithms for real-world use cases. It might also be interesting to apply the ideas of Chapter 3 to classical differential solvers, as many of those are dealing with noisy data.

The method presented in Chapter 4 can be extended by including additional constraints, such as entropy constraints, into the SDP formulation to further refine the reconstruction process [195]. An interesting question is to explore its effect with other systems, such as frustration-free Hamiltonians or for quantum chemistry calculations, as in [207, 208]. Furthermore, it might be promising to combine our method with noise mitigation strategies and to apply it in the optimization of other local observables, such as correlation functions.

The quantum generative adversarial networks of Chapter 5 can possibly be applied as subroutines for applications such as option pricing [275] and risk analysis [276]. Moreover, a possible extension of our method is to train the model to replicate correlated stocks of the S&P 500 index, motivated by research in community detection [277]. We gave several ideas for this extension in Chapter 5. We lastly suggest to explore possible improvement of the training of our model by several ways. In particular, as the model is trained with Wasserstein loss functions (see Eqs. (5.9) and (5.11)) that are taking the distribution of the time series into account, but not the temporal effects, an adaptation of the training to consider them as well might lead to a better recovery of those temporal effects.

Bibliography

[1] Y. Manin. Computable and uncomputable. *Sovetskoye Radio, Moscow*, **128**, 28 (1980).

[2] R. P. Feynman. Simulating physics with computers. *International Journal of Theoretical Physics*, **21**, 467–488 (1982).

[3] D. Deutsch. Quantum theory as a universal physical theory. *International Journal of Theoretical Physics*, **24**, 1–41 (1985).

[4] P. W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing*, **26**, 1484–1509 (1997).

[5] L. K. Grover. Quantum mechanics helps in searching for a needle in a haystack. *Physical Review Letters*, **79**, 325 (1997).

[6] A. M. Dalzell, S. McArdle, M. Berta, P. Bienias, C.-F. Chen, A. Gilyén et al. Quantum algorithms: A survey of applications and end-to-end complexities, arXiv:2310.03011 (2023).

[7] C. H. Bennett & G. Brassard. Quantum cryptography: Public key distribution and coin tossing. *Theoretical Computer Science*, **560**, 7–11 (2014).

[8] C. H. Bennett, G. Brassard, C. Crépeau, R. Jozsa, A. Peres & W. K. Wootters. Teleporting an unknown quantum state via dual classical and Einstein-Podolsky-Rosen channels. *Physical Review Letters*, **70**, 1895 (1993).

[9] M. A. Nielsen & I. L. Chuang. *Quantum computation and quantum information.* Cambridge University Press (2010).

[10] T. D. Ladd, F. Jelezko, R. Laflamme, Y. Nakamura, C. Monroe & J. L. O'Brien. Quantum computers. *Nature*, **464**, 45–53 (2010).

[11] J. Clarke & F. K. Wilhelm. Superconducting quantum bits. *Nature*, **453**, 1031–1042 (2008).

[12] H.-L. Huang, D. Wu, D. Fan & X. Zhu. Superconducting quantum computing: a review. *Science China Information Sciences*, **63**, 1–32 (2020).

[13] J. I. Cirac & P. Zoller. Quantum computations with cold trapped ions. *Physical Review Letters*, **74**, 4091 (1995).

[14] C. D. Bruzewicz, J. Chiaverini, R. McConnell & J. M. Sage. Trapped-ion quantum computing: Progress and challenges. *Applied Physics Reviews*, **6**, 021314 (2019).

[15] L. Henriet, L. Beguin, A. Signoles, T. Lahaye, A. Browaeys, G.-O. Reymond & C. Jurczak. Quantum computing with neutral atoms. *Quantum*, **4**, 327 (2020).

[16] A. Imamog, D. D. Awschalom, G. Burkard, D. P. DiVincenzo, D. Loss, M. Sherwin, A. Small et al. Quantum information processing using quantum dot spins and cavity QED. *Physical Review Letters*, **83**, 4204 (1999).

[17] E. Knill, R. Laflamme & G. J. Milburn. A scheme for efficient quantum computation with linear optics. *Nature*, **409**, 46–52 (2001).

[18] F. Arute, K. Arya, R. Babbush, D. Bacon, J. C. Bardin, R. Barends et al. Quantum supremacy using a programmable superconducting processor. *Nature*, **574**, 505–510 (2019).

[19] R. Acharya, L. Aghababaie-Beni, I. Aleiner, T. I. Andersen, M. Ansmann, F. Arute et al. Quantum error correction below the surface code threshold. *Nature*, **638**, 920–926 (2025).

[20] J. Preskill. Quantum computing in the NISQ era and beyond. *Quantum*, **2**, 79 (2018).

[21] M. Cerezo, A. Arrasmith, R. Babbush, S. C. Benjamin, S. Endo, K. Fujii et al. Variational quantum algorithms. *Nature Reviews Physics*, **3**, 625–644 (2021).

[22] E. Farhi, J. Goldstone & S. Gutmann. A quantum approximate optimization algorithm, arXiv:1411.4028 (2014).

[23] S. McArdle, S. Endo, A. Aspuru-Guzik, S. C. Benjamin & X. Yuan. Quantum computational chemistry. *Reviews of Modern Physics*, **92**, 015003 (2020).

[24] D. Herman, C. Googin, X. Liu, Y. Sun, A. Galda, I. Safro, M. Pistoia & Y. Alexeev. Quantum computing for finance. *Nature Reviews Physics*, **5**, 450–465 (2023).

[25] M. Cerezo, G. Verdon, H.-Y. Huang, L. Cincio & P. J. Coles. Challenges and opportunities in quantum machine learning. *Nature Computational Science*, **2**, 567–576 (2022).

[26] F. Fontanela, A. Jacquier & M. Oumgari. A quantum algorithm for linear PDEs arising in finance. *SIAM Journal on Financial Mathematics*, **12**, SC98–SC114 (2021).

[27] A. Barenco, C. H. Bennett, R. Cleve, D. P. DiVincenzo, N. Margolus, P. Shor, T. Sleator, J. A. Smolin & H. Weinfurter. Elementary gates for quantum computation. *Physical Review A*, **52**, 3457 (1995).

[28] S. Aaronson. Read the fine print. *Nature Physics*, **11**, 291–293 (2015).

[29] A. Kandala, A. Mezzacapo, K. Temme, M. Takita, M. Brink, J. M. Chow & J. M. Gambetta. Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets. *Nature*, **549**, 242–246 (2017).

[30] L. Leone, S. F. Oliviero, L. Cincio & M. Cerezo. On the practical usefulness of the hardware efficient ansatz. *Quantum*, **8**, 1395 (2024).

[31] S. V. Ershkov, E. Y. Prosviryakov, N. V. Burmasheva & V. Christianto. Towards understanding the algorithms for solving the Navier–Stokes equations. *Fluid Dynamics Research*, **53**, 044501 (2021).

[32] D. G. Zill. *A first course in differential equations with modeling applications*. Cengage Learning (2009).

[33] X. Mao. *Stochastic differential equations and applications*. Elsevier (2007).

[34] L. C. Evans. *Partial differential equations*, volume 19. American Mathematical Society (2022).

[35] D. W. Berry. High-order quantum algorithm for solving linear differential equations. *Journal of Physics A: Mathematical and Theoretical*, **47**, 105301 (2014).

[36] D. W. Berry, A. M. Childs, A. Ostrander & G. Wang. Quantum algorithm for linear differential equations with exponentially improved dependence on precision. *Communications in Mathematical Physics*, **356**, 1057–1081 (2017).

[37] A. W. Harrow, A. Hassidim & S. Lloyd. Quantum algorithm for linear systems of equations. *Physical Review Letters*, **103**, 150502 (2009).

[38] S. Jin, N. Liu & Y. Yu. Time complexity analysis of quantum algorithms via linear representations for nonlinear ordinary and partial differential equations. *Journal of Computational Physics*, **487**, 112149 (2023).

[39] J.-P. Liu, H. Ø. Kolden, H. K. Krovi, N. F. Loureiro, K. Trivisa & A. M. Childs. Efficient quantum algorithm for dissipative nonlinear differential equations. *Proceedings of the National Academy of Sciences*, **118**, e2026805118 (2021).

[40] C. Bravo-Prieto, R. LaRose, M. Cerezo, Y. Subasi, L. Cincio & P. J. Coles. Variational quantum linear solver. *Quantum*, **7**, 1188 (2023).

[41] H.-L. Liu, Y.-S. Wu, L.-C. Wan, S.-J. Pan, S.-J. Qin, F. Gao & Q.-Y. Wen. Variational quantum algorithm for the Poisson equation. *Physical Review A*, **104**, 022418 (2021).

[42] C. J. Trahan, M. Loveland, N. Davis & E. Ellison. A variational quantum linear solver application to discrete finite-element methods. *Entropy*, **25**, 580 (2023).

[43] A. Abbas, D. Sutter, C. Zoufal, A. Lucchi, A. Figalli & S. Woerner. The power of quantum neural networks. *Nature Computational Science*, **1**, 403–409 (2021).

[44] A. E. Paine, V. E. Elfving & O. Kyriienko. Physics-informed quantum machine learning: Solving nonlinear differential equations in latent spaces without costly grid evaluations, arXiv:2308.01827 (2023).

[45] B. E. Baaquie. *Quantum finance: Path integrals and Hamiltonians for options and interest rates.* Cambridge University Press (2007).

[46] D. Dechant, L. Markovich, V. Dunjko & J. Tura. Error and resource estimates of variational quantum algorithms for solving differential equations based on Runge-Kutta methods. *Journal of Mathematical Physics*, **67**, 012205 (2026).

[47] D. An, N. Linden, J.-P. Liu, A. Montanaro, C. Shao & J. Wang. Quantum-accelerated multilevel Monte Carlo methods for stochastic differential equations in mathematical finance. *Quantum*, **5**, 481 (2021).

[48] S. J. Prince. *Understanding deep learning*. MIT Press (2023).

[49] OpenAI. Chatgpt: Optimizing language models for dialogue (2022). URL https://openai.com/blog/chatgpt. Accessed: 2025-07-04.

[50] J. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, O. Ronneberger et al. Highly accurate protein structure prediction with AlphaFold. *Nature*, **596**, 583–589 (2021).

[51] A. Ramesh, M. Pavlov, G. Goh, S. Gray, C. Voss, A. Radford, M. Chen & I. Sutskever. Zero-shot text-to-image generation. In *International Conference on Machine Learning*, pages 8821–8831. PMLR (2021).

[52] V. Havlíček, A. D. Córcoles, K. Temme, A. W. Harrow, A. Kandala, J. M. Chow & J. M. Gambetta. Supervised learning with quantum-enhanced feature spaces. *Nature*, **567**, 209–212 (2019).

[53] S. Jerbi, C. Gyurik, S. Marshall, H. Briegel & V. Dunjko. Parametrized quantum policies for reinforcement learning. *Advances in Neural Information Processing Systems*, **34**, 28362–28375 (2021).

[54] Y. Liu, S. Arunachalam & K. Temme. A rigorous and robust quantum speed-up in supervised machine learning. *Nature Physics*, **17**, 1013–1017 (2021).

[55] A. Wilms, L. Ohff, A. Skolik, J. Eisert, S. Khatri & D. A. Reiss. Quantum reinforcement learning of classical rare dynamics: Enhancement by intrinsic Fourier features, arXiv:2504.16258 (2025).

[56] R. Molteni, S. C. Marshall & V. Dunjko. Quantum machine learning advantages beyond hardness of evaluation, arXiv:2504.15964 (2025).

[57] M. Schuld & N. Killoran. Quantum machine learning in feature Hilbert spaces. *Physical Review Letters*, **122**, 040504 (2019).

[58] S. Lloyd, M. Mohseni & P. Rebentrost. Quantum algorithms for supervised and unsupervised machine learning, arXiv:1307.0411 (2013).

[59] P. Rebentrost, M. Mohseni & S. Lloyd. Quantum support vector machine for big data classification. *Physical Review Letters*, **113**, 130503 (2014).

[60] S. Lloyd, M. Mohseni & P. Rebentrost. Quantum principal component analysis. *Nature Physics*, **10**, 631–633 (2014).

[61] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe & S. Lloyd. Quantum machine learning. *Nature*, **549**, 195–202 (2017).

[62] A. W. Harrow, A. Hassidim & S. Lloyd. Quantum algorithm for linear systems of equations. *Physical Review Letters*, **103**, 150502 (2009).

[63] E. Tang. A quantum-inspired classical algorithm for recommendation systems. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, pages 217–228 (2019).

[64] E. Tang. Dequantizing algorithms to understand quantum advantage in machine learning. *Nature Reviews Physics*, **4**, 692–693 (2022).

[65] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems*, **2**, 303–314 (1989).

[66] Z. Lu, H. Pu, F. Wang, Z. Hu & L. Wang. The expressive power of neural networks: A view from the width. *Advances in Neural Information Processing Systems*, **30** (2017).

[67] M. Schuld, R. Sweke & J. J. Meyer. Effect of data encoding on the expressive power of variational quantum-machine-learning models. *Physical Review A*, **103**, 032430 (2021).

[68] A. Pérez-Salinas, D. López-Núñez, A. García-Sáez & J. I. Latorre. One qubit as a universal approximant. *Physical Review A*, **104**, 012405 (2021).

[69] Z. Yu, H. Yao, M. Li & X. Wang. Power and limitations of single-qubit native quantum neural networks. *Advances in Neural Information Processing Systems*, **35**, 27810–27823 (2022).

[70] B. Casas & A. Cervera-Lierta. Multidimensional Fourier series with quantum circuits. *Physical Review A*, **107**, 062612 (2023).

[71] T. Goto, Q. H. Tran & K. Nakajima. Universal approximation property of quantum machine learning models in quantum-enhanced feature spaces. *Physical Review Letters*, **127**, 090506 (2021).

[72] J. R. McClean, S. Boixo, V. N. Smelyanskiy, R. Babbush & H. Neven. Barren plateaus in quantum neural network training landscapes. *Nature Communications*, **9**, 4812 (2018).

[73] M. C. Caro, E. Gil-Fuster, J. J. Meyer, J. Eisert & R. Sweke. Encoding-dependent generalization bounds for parametrized quantum circuits. *Quantum*, **5**, 582 (2021).

[74] C. W. Oosterlee & L. A. Grzelak. *Mathematical modeling and computation in finance: with exercises and Python and MATLAB computer codes*. World Scientific (2019).

[75] S&P Dow Jones Indices. S&P 500®. https://www.spglobal.com/spdji/en/indices/equity/sp-500/#overview (2025). Accessed: 2025-06-19.

[76] R. Cont. Empirical properties of asset returns: Stylized facts and statistical issues. *Quantitative Finance*, **1**, 223–236 (2001).

[77] M. F. Dixon, I. Halperin & P. Bilokon. *Machine Learning in Finance: From Theory to Practice*. Springer International Publishing, Cham (2020).

[78] M. Dogariu, L.-D. Ştefan, B. A. Boteanu, C. Lamba, B. Kim & B. Ionescu. Generation of realistic synthetic financial time-series. *ACM Transactions on Multimedia Computing, Communications, and Applications*, **18**, 1–27 (2022).

[79] S. Aaronson & A. Arkhipov. The computational complexity of linear optics. *Theory of Computing*, **9**, 143–252 (2013).

[80] S. Natenberg. *Option volatility and pricing*. McGraw-Hill New York (1994).

[81] F. Black & M. Scholes. The pricing of options and corporate liabilities. *Journal of Political Economy*, **81**, 637–654 (1973).

[82] The Royal Swedish Academy of Sciences. Myron S. Scholes – facts (1997). URL https://www.nobelprize.org/prizes/economic-sciences/1997/press-release/. Accessed: 2025-07-04.

[83] P. G. Zhang. *Exotic options: a guide to second generation options*. World Scientific (1997).

[84] J. C. Hull & S. Basu. *Options, futures, and other derivatives*. Pearson Education India (2016).

[85] P. Carr & D. Madan. Option valuation using the fast Fourier transform. *Journal of Computational Finance*, **2**, 61–73 (1999).

[86] J. Ruf & W. Wang. Neural networks for option pricing and hedging: a literature review. *Journal of Computational Finance*, **24**, 1–46 (2020).

[87] P. Rebentrost, B. Gupt & T. R. Bromley. Quantum computational finance: Monte Carlo pricing of financial derivatives. *Physical Review A*, **98**, 022321 (2018).

[88] A. Manzano, D. Dechant, J. Tura & V. Dunjko. Approximation and generalization capacities of parametrized quantum circuits for functions in Sobolev spaces. *Quantum*, **9**, 1658 (2025).

[89] B. Huge & A. Savine. Differential machine learning, arXiv:2005.02347 (2020).

[90] Z. J. Wang, D. Dechant, Y. J. Patel & J. Tura. Mitigating shot noise in local overlapping quantum tomography with semidefinite programming. *Physical Review A*, **111**, 052444 (2025).

[91] D. Dechant, E. Schwander, L. Van Drooge, C. Moussa, D. Garlaschelli, V. Dunjko & J. Tura Brugués. Quantum generative modeling for financial time series with temporal correlations. *Machine Learning: Science and Technology* (2026).

[92] K. Mitarai, M. Negoro, M. Kitagawa & K. Fujii. Quantum circuit learning. *Physical Review A*, **98**, 032309 (2018).

[93] Y. Du, M.-H. Hsieh, T. Liu & D. Tao. Expressive power of parametrized quantum circuits. *Physical Review Research*, **2**, 033125 (2020).

[94] M. Raissi, P. Perdikaris & G. E. Karniadakis. Physics informed deep learning (part i): Data-driven solutions of nonlinear partial differential equations, arXiv:1711.10561 (2017).

[95] L. Gonon & A. Jacquier. Universal approximation theorem and error bounds for quantum neural networks and quantum reservoirs. *IEEE transactions on neural networks and learning systems*, **36**, 11355–11368 (2025).

[96] F. J. Gil Vidal & D. O. Theis. Input redundancy for parameterized quantum circuits. *Frontiers in Physics*, **8**, 297 (2020).

[97] A. Pérez-Salinas, A. Cervera-Lierta, E. Gil-Fuster & J. I. Latorre. Data re-uploading for a universal quantum classifier. *Quantum*, **4**, 226 (2020).

[98] L. Fejér. Untersuchungen über Fouriersche Reihen. *Mathematische Annalen*, **58**, 51–69 (1903).

[99] D. Gottlieb & C.-W. Shu. On the Gibbs phenomenon and its resolution. *SIAM Review*, **39**, 644–668 (1997).

[100] M. Mohri, A. Rostamizadeh & A. Talwalkar. *Foundations of machine learning*. MIT Press (2018).

[101] H. L. Royden & P. Fitzpatrick. *Real analysis*, volume 2. Macmillan New York (1968).

[102] V. Burenkov. Extension theorems for Sobolev spaces. In *The Maz'ya Anniversary Collection: Volume 1: On Maz'ya's work in functional analysis, partial differential equations and applications*, pages 187–200. Springer (1999).

[103] F. Weisz. ℓ_1-summability of higher-dimensional Fourier series. *Journal of Approximation Theory*, **163**, 99–116 (2011).

[104] C. Canuto & A. Quarteroni. Approximation results for orthogonal polynomials in Sobolev spaces. *Mathematics of Computation*, **38**, 67–86 (1982).

[105] M. M. Wolf. Mathematical foundations of supervised learning (2023). Lecture Notes.

[106] R. A. Adams & J. J. Fournier. *Sobolev spaces*. Elsevier (2003).

[107] P. Jurcevic, A. Javadi-Abhari, L. Bishop, I. Lauer, D. Bogorin, M. Brink et al. Demonstration of quantum volume 64 on a superconducting quantum computing system. *Quantum Science and Technology*, **6**, 025020 (2021).

[108] H. Levine, A. Keesling, G. Semeghini, A. Omran, T. Wang, S. Ebadi et al. Parallel implementation of high-fidelity multiqubit gates with neutral atoms. *Physical Review Letters*, **123**, 170503 (2019).

[109] A. Risinger, D. Lobser, A. Bell, C. Noel, L. Egan, D. Zhu, D. Biswas, M. Cetina & C. Monroe. Characterization and control of large-scale ion-trap quantum computers. *Bull. Am. Phys. Soc.*, **66** (2021).

[110] A. Fowler, M. Mariantoni, J. Martinis & A. Cleland. Surface codes: Towards practical large-scale quantum computation. *Physical Review A*, **86**, 032324 (2012).

[111] G. Scriva, N. Astrakhantsev, S. Pilati & G. Mazzola. Challenges of variational quantum optimization with measurement shot noise. *Physical Review A*, **109**, 032408 (2024).

[112] S. K. Leyton & T. J. Osborne. A quantum algorithm to solve nonlinear differential equations, arXiv:0812.4423 (2008).

[113] M. Lubasch, J. Joo, P. Moinier, M. Kiffner & D. Jaksch. Variational quantum algorithms for nonlinear problems. *Physical Review A*, **101**, 010301 (2020).

[114] F. Y. Leong, W.-B. Ewe & D. E. Koh. Variational quantum evolution equation solver. *Scientific Reports*, **12**, 10817 (2022).

[115] O. Kyriienko, A. E. Paine & V. E. Elfving. Solving nonlinear differential equations with differentiable quantum circuits. *Physical Review A*, **103**, 052416 (2021).

[116] S. McArdle, T. Jones, S. Endo, Y. Li, S. Benjamin & X. Yuan. Variational ansatz-based quantum simulation of imaginary time evolution. *npj Quantum Information*, **5**, 75 (2019).

[117] Y. Li & S. C. Benjamin. Efficient variational quantum simulator incorporating active error minimization. *Physical Review X*, **7**, 021050 (2017).

[118] L. Nagano, A. Bapat & C. W. Bauer. Quench dynamics of the Schwinger model via variational quantum algorithms. *Physical Review D*, **108**, 034501 (2023).

[119] S. Barison, F. Vicentini & G. Carleo. An efficient quantum algorithm for the time evolution of parameterized circuits. *Quantum*, **5**, 512 (2021).

[120] M. Benedetti, M. Fiorentini & M. Lubasch. Hardware-efficient variational quantum algorithms for time evolution. *Physical Review Research*, **3**, 033083 (2021).

[121] Y.-X. Yao, N. Gomes, F. Zhang, C.-Z. Wang, K.-M. Ho, T. Iadecola & P. P. Orth. Adaptive variational quantum dynamics simulations. *PRX Quantum*, **2**, 030307 (2021).

[122] C. Cirstoiu, Z. Holmes, J. Iosue, L. Cincio, P. J. Coles & A. Sornborger. Variational fast forwarding for quantum simulation beyond the coherence time. *npj Quantum Information*, **6**, 82 (2020).

[123] K. Heya, K. M. Nakanishi, K. Mitarai, Z. Yan, K. Zuo, Y. Suzuki et al. Subspace variational quantum simulator. *Physical Review Research*, **5**, 023078 (2023).

[124] A. J. Pool, A. D. Somoza, C. Mc Keever, M. Lubasch & B. Horstmann. Nonlinear dynamics as a ground-state solution on quantum computers. *Physical Review Research*, **6**, 033257 (2024).

[125] J. C. Butcher. General linear methods. *Acta Numerica*, **15**, 157–256 (2006).

[126] W. Kutta. *Beitrag zur näherungsweisen Integration totaler Differentialgleichungen.* Teubner (1901).

[127] S. K. Radha. Quantum option pricing using Wick rotated imaginary time evolution, arXiv:2101.04280 (2021).

[128] H. Alghassi, A. Deshmukh, N. Ibrahim, N. Robles, S. Woerner & C. Zoufal. A variational quantum algorithm for the Feynman-Kac formula. *Quantum*, **6**, 730 (2022).

[129] K. Brown, J. Kim & C. Monroe. Co-designing a scalable quantum computer with trapped atomic ions. *npj Quantum Inf.*, **2**, 1–10 (2016).

[130] V. Schäfer, C. Ballance, K. Thirumalai, L. Stephenson, T. Ballance, A. Steane & D. Lucas. Fast quantum logic gates with trapped-ion qubits. *Nature*, **555**, 75–78 (2018).

[131] J. Gacon, J. Nys, R. Rossi, S. Woerner & G. Carleo. Variational quantum time evolution without the quantum geometric tensor. *Physical Review Research*, **6**, 013143 (2024).

[132] K. Kubo, K. Miyamoto, K. Mitarai & K. Fujii. Pricing multiasset derivatives by variational quantum algorithms. *IEEE Transactions on Quantum Engineering*, **4**, 1–17 (2023).

[133] A. Miessen, P. J. Ollitrault & I. Tavernelli. Quantum algorithms for quantum dynamics: A performance study on the spin-boson model. *Physical Review Research*, **3**, 043212 (2021).

[134] C. Zoufal, D. Sutter & S. Woerner. Error bounds for variational quantum time evolution. *Physical Review Applied*, **20**, 044059 (2023).

[135] I. Kolotouros, D. Joseph & A. Narayanan. Accelerating quantum imaginary-time evolution with random measurements, arXiv:2407.03123 (2024).

[136] B. Zanger, C. B. Mendl, M. Schulz & M. Schreiber. Quantum algorithms for solving ordinary differential equations via classical integration methods. *Quantum*, **5**, 502 (2021).

[137] W. H. Press, S. A. Teukolsky, W. T. Vetterling & B. P. Flannery. *Numerical recipes 3rd edition: The art of scientific computing.* Cambridge University Press (2007).

[138] S. Khashin. A symbolic-numeric approach to the solution of the Butcher equations. *Canadian Applied Mathematics Quarterly*, **17**, 555–569 (2009).

[139] D. K. Zhang. An explicit 16-stage Runge-Kutta method of order 10 discovered by numerical search. *Numerical Algorithms*, pages 1–25 (2024).

[140] J. C. Butcher. *Numerical methods for ordinary differential equations.* John Wiley & Sons (2016).

[141] S. Jin, N. Liu & Y. Yu. Quantum simulation of partial differential equations via Schrödingerisation: technical details, arXiv:2212.14703 (2022).

[142] I. O. Sokolov, W. Dobrautz, H. Luo, A. Alavi & I. Tavernelli. Orders of magnitude increased accuracy for quantum many-body problems on quantum computers via an exact transcorrelated method. *Physical Review Research*, **5**, 023174 (2023).

[143] S. Jin, N. Liu & Y. Yu. Quantum simulation of partial differential equations via Schrödingerization. *Physical Review Letters*, **133**, 230602 (2024).

[144] S. Endo, J. Sun, Y. Li, S. C. Benjamin & X. Yuan. Variational quantum simulation of general processes. *Physical Review Letters*, **125**, 010501 (2020).

[145] D. Poulin, A. Qarry, R. Somma & F. Verstraete. Quantum simulation of time-dependent Hamiltonians and the convenient illusion of Hilbert space. *Physical Review Letters*, **106**, 170501 (2011).

[146] A. G. Taube & R. J. Bartlett. New perspectives on unitary coupled-cluster theory. *International Journal of Quantum Chemistry*, **106**, 3393–3401 (2006).

[147] P.-L. Dallaire-Demers, J. Romero, L. Veis, S. Sim & A. Aspuru-Guzik. Low-depth circuit ansatz for preparing correlated fermionic states on a quantum computer. *Quantum Science and Technology*, **4**, 045005 (2019).

[148] A. Peruzzo, J. McClean, P. Shadbolt, M.-H. Yung, X.-Q. Zhou, P. J. Love, A. Aspuru-Guzik & J. L. O'brien. A variational eigenvalue solver on a photonic quantum processor. *Nature Communications*, **5**, 4213 (2014).

[149] A. McLachlan. A variational solution of the time-dependent Schrödinger equation. *Molecular Physics*, **8**, 39–44 (1964).

[150] X. Yuan, S. Endo, Q. Zhao, Y. Li & S. C. Benjamin. Theory of variational quantum simulation. *Quantum*, **3**, 191 (2019).

[151] L. Markovich, S. Malikis, S. Polla & J. Tura. Parameter shift rule with optimal phase selection. *Physical Review A*, **109**, 062429 (2024).

[152] A. A. Anuar, F. Jamet, F. Gironella, F. S. I. au2 & R. Rossi. Operator-projected variational quantum imaginary time evolution, arXiv:2409.12018 (2024).

[153] L. Bittel, S. Gharibian & M. Kliesch. The optimal depth of variational quantum algorithms is QCMA-hard to approximate. In *38th Computational Complexity Conference (CCC 2023)*, volume 264, page 34. Schloss Dagstuhl–Leibniz-Zentrum für Informatik (2023).

[154] J. Landman, S. Thabet, C. Dalyac, H. Mhiri & E. Kashefi. Classically approximating variational quantum machine learning with random Fourier features, arXiv:2210.13200 (2022).

[155] V. H. Vu & T. Tao. The condition number of a randomly perturbed matrix. In *Proceedings of the Thirty-Ninth Annual ACM Symposium on Theory of Computing*, pages 248–255 (2007).

[156] N. J. A. Sloane. The on-line encyclopedia of integer sequences (2024). URL `https://oeis.org/A000081`. Accessed: 2024-07-03.

[157] Google. Sycamore data sheet (2024). URL `https://quantumai.google/hardware/datasheet/weber.pdf`. Accessed: 2024-07-22.

[158] B. Datta. *Numerical methods for linear control systems*. Elsevier (2004).

[159] Y. Xie, X. Wu & R. Ward. Linear convergence of adaptive stochastic gradient descent. In *International Conference on Artificial Intelligence and Statistics*, pages 1475–1485. PMLR (2020).

[160] A. I. Lvovsky & M. G. Raymer. Continuous-variable optical quantum-state tomography. *Reviews of Modern Physics*, **81**, 299–332 (2009).

[161] M. Cramer, M. B. Plenio, S. T. Flammia, R. Somma, D. Gross, S. D. Bartlett, O. Landon-Cardinal, D. Poulin & Y.-K. Liu. Efficient quantum state tomography. *Nature Communications*, **1**, 149 (2010).

[162] M. Kliesch & I. Roth. Theory of quantum system certification. *PRX Quantum*, **2**, 010201 (2021).

[163] H.-Y. Huang, R. Kueng & J. Preskill. Predicting many properties of a quantum system from very few measurements. *Nature Physics*, **16**, 1050–1057 (2020).

[164] B. G. Araújo, M. M. Taddei, D. Cavalcanti & A. Acín. Local quantum overlapping tomography. *Physical Review A*, **106**, 062441 (2022).

[165] E. Davidson. *Reduced density matrices in quantum chemistry*, volume 6. Elsevier (2012).

[166] N. Linden, S. Popescu & W. Wootters. Almost every pure state of three qubits is completely determined by its two-particle reduced density matrices. *Physical Review Letters*, **89**, 207901 (2002).

[167] N. Linden & W. Wootters. The parts determine the whole in a generic pure quantum state. *Physical Review Letters*, **89**, 277906 (2002).

[168] J. Cotler & F. Wilczek. Quantum overlapping tomography. *Physical Review Letters*, **124**, 100401 (2020).

[169] A. A. Klyachko. Quantum marginal problem and N-representability. *Journal of Physics: Conference Series*, **36**, 72 (2006).

[170] A. Broadbent & A. B. Grilo. QMA-hardness of consistency of local density matrices with applications to quantum zero-knowledge. *SIAM Journal on Computing*, **51**, 1400–1450 (2022).

[171] J. Kamminga & D. Rudolph. On the complexity of pure-state consistency of local density matrices, arXiv:2411.03096 (2024).

[172] A. J. McCaskey, Z. P. Parks, J. Jakowski, S. V. Moore, T. D. Morris, T. S. Humble & R. C. Pooser. Quantum chemistry as a benchmark for near-term quantum computers. *npj Quantum Information*, **5**, 99 (2019).

[173] K. Hansenne, R. Qu, L. T. Weinbrenner, C. de Gois, H. Wang, Y. Ming, Z. Yang, P. Horodecki, W. Gao & O. Gühne. Optimal overlapping tomography. *Physical Review Letters*, **135**, 060801 (2025).

[174] P. Skrzypczyk & D. Cavalcanti. *Semidefinite programming in quantum information science.* IOP Publishing (2023).

[175] A. Aloy, M. Fadel & J. Tura. The quantum marginal problem for symmetric states: applications to variational optimization, nonlocality and self-testing. *New Journal of Physics*, **23**, 033026 (2021).

[176] C. Schwemmer, L. Knips, D. Richart, H. Weinfurter, T. Moroder, M. Kleinmann & O. Gühne. Systematic errors in current quantum state tomography tools. *Physical Review Letters*, **114**, 080403 (2015).

[177] P. O. Boykin, T. Mor, V. Roychowdhury, F. Vatan & R. Vrijen. Algorithmic cooling and scalable NMR quantum computers. *Proceedings of the National Academy of Sciences*, **99**, 3388–3393 (2002).

[178] S. Polla, Y. Herasymenko & T. E. O'Brien. Quantum digital cooling. *Physical Review A*, **104**, 012414 (2021).

[179] H. R. Grimsley, S. E. Economou, E. Barnes & N. J. Mayhall. An adaptive variational algorithm for exact molecular simulations on a quantum computer. *Nature Communications*, **10**, 3007 (2019).

[180] J. B. Altepeter, D. F. James & P. G. Kwiat. 4 qubit quantum state tomography. In *Quantum state estimation*, pages 113–145. Springer (2004).

[181] G. Torlai, G. Mazzola, J. Carrasquilla, M. Troyer, R. Melko & G. Carleo. Neural-network quantum state tomography. *Nature Physics*, **14**, 447–450 (2018).

[182] W. K. Wootters & B. D. Fields. Optimal state-determination by mutually unbiased measurements. *Annals of Physics*, **191**, 363–381 (1989).

[183] R. Adamson & A. M. Steinberg. Improving quantum state estimation with mutually unbiased bases. *Physical Review Letters*, **105**, 030406 (2010).

[184] R. Stricker, M. Meth, L. Postler, C. Edmunds, C. Ferrie, R. Blatt, P. Schindler, T. Monz, R. Kueng & M. Ringbauer. Experimental single-setting quantum state tomography. *PRX Quantum*, **3**, 040310 (2022).

[185] N. Karmarkar. A new polynomial-time algorithm for linear programming. In *Proceedings of the Sixteenth Annual ACM Symposium on Theory of Computing*, pages 302–311 (1984).

[186] H. Jiang, T. Kathuria, Y. T. Lee, S. Padmanabhan & Z. Song. A faster interior point method for semidefinite programming. In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 910–918. IEEE (2020).

[187] B. Requena, G. Muñoz-Gil, M. Lewenstein, V. Dunjko & J. Tura. Certificates of quantum many-body properties assisted by machine learning. *Physical Review Research*, **5**, 013097 (2023).

[188] X. Bonet-Monroig, R. Babbush & T. E. O'Brien. Nearly optimal measurement scheduling for partial tomography of quantum states. *Physical Review X*, **10**, 031064 (2020).

[189] J. O. de Almeida, M. Kleinmann & G. Sentís. Comparison of confidence regions for quantum state tomography. *New Journal of Physics*, **25**, 113018 (2023).

[190] H. Westerheim, J. Chen, Z. Holmes, I. Luo, T. Nuradha, D. Patel, S. Rethinasamy, K. Wang & M. M. Wilde. Dual-VQE: A quantum algorithm to lower bound the ground-state energy, arXiv:2312.03083 (2023).

[191] L. Zambrano, D. Farina, E. Pagliaro, M. M. Taddei & A. Acin. Certification of quantum state functions under partial information. *Quantum*, **8**, 1442 (2024).

[192] J. Wang, J. Surace, I. Frérot, B. Legat, M.-O. Renou, V. Magron & A. Acín. Certifying ground-state properties of many-body systems. *Physical Review X*, **14**, 031006 (2024).

[193] A. Gresch & M. Kliesch. Guaranteed efficient energy estimation of quantum many-body Hamiltonians using ShadowGrouping. *Nature Communications*, **16**, 689 (2025).

[194] I. Kull, N. Schuch, B. Dive & M. Navascués. Lower bounds on ground-state energies of local Hamiltonians through the renormalization group. *Physical Review X*, **14**, 021008 (2024).

[195] H. Fawzi, O. Fawzi & S. O. Scalet. Entropy constraints for ground energy optimization. *Journal of Mathematical Physics*, **65** (2024).

[196] T. Barthel & R. Hübener. Solving condensed-matter ground-state problems by semidefinite relaxations. *Physical Review Letters*, **108**, 200404 (2012).

[197] Y.-K. Liu, M. Christandl & F. Verstraete. Quantum computational complexity of the N-representability problem: QMA complete. *Physical Review Letters*, **98**, 110503 (2007).

[198] E. Lieb, T. Schultz & D. Mattis. Two soluble models of an antiferromagnetic chain. *Annals of Physics*, **16**, 407–466 (1961).

[199] D. A. Mazziotti. Variational reduced-density-matrix method using three-particle N-representability conditions with application to many-electron molecules. *Physical Review A*, **74**, 032501 (2006).

[200] D. A. Mazziotti. Realization of quantum chemistry without wave functions through first-order semidefinite programming. *Physical Review Letters*, **93**, 213001 (2004).

[201] M. Navascués, S. Pironio & A. Acín. Bounding the set of quantum correlations. *Physical Review Letters*, **98**, 010401 (2007).

[202] K. S. Rai, I. Kull, P. Emonts, J. Tura, N. Schuch & F. Baccari. A hierarchy of spectral gap certificates for frustration-free spin systems, arXiv:2411.03680 (2024).

[203] A. Javadi-Abhari, M. Treinish, K. Krsulich, C. J. Wood, J. Lishman, J. Gacon et al. Quantum computing with Qiskit, arXiv:2405.08810 (2024).

[204] A. Agrawal, R. Verschueren, S. Diamond & S. Boyd. A rewriting system for convex optimization problems. *Journal of Control and Decision*, **5**, 42–60 (2018).

[205] S. Diamond & S. Boyd. CVXPY: A Python-embedded modeling language for convex optimization. *Journal of Machine Learning Research*, **17**, 1–5 (2016).

[206] N. Schuch & J. I. Cirac. Matrix product state and mean-field solutions for one-dimensional systems can be found efficiently. *Physical Review A*, **82**, 012314 (2010).

[207] G.-L. R. Anselmetti, M. Degroote, N. Moll, R. Santagati & M. Streif. Classical optimisation of reduced density matrix estimations with classical shadows using N-representability conditions under shot noise considerations, arXiv:2411.18430 (2024).

[208] I. Avdic & D. A. Mazziotti. Fewer measurements from shadow tomography with N-representability conditions. *Physical Review Letters*, **132**, 220802 (2024).

[209] D. Rosset, R. Ferretti-Schöbitz, J.-D. Bancal, N. Gisin & Y.-C. Liang. Imperfect measurement settings: Implications for quantum state tomography and entanglement witnesses. *Physical Review A*, **86**, 062325 (2012).

[210] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville & Y. Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc. (2014).

[211] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville & Y. Bengio. Generative adversarial networks. *Communications of the ACM*, **63**, 139–144 (2020).

[212] A. Radford, L. Metz & S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks, arXiv:1511.06434 (2016).

[213] T. Karras, T. Aila, S. Laine & J. Lehtinen. Progressive growing of GANs for improved quality, stability, and variation, arXiv:1710.10196 (2018).

[214] T. Karras, S. Laine & T. Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4401–4410 (2019).

[215] J. Gui, Z. Sun, Y. Wen, D. Tao & J. Ye. A review on generative adversarial networks: Algorithms, theory, and applications. *IEEE Transactions on Knowledge and Data Engineering*, **35**, 3313–3332 (2021).

[216] C. Shorten & T. M. Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of Big Data*, **6**, 60 (2019).

[217] F. H. K. dos Santos Tanaka & C. Aranha. Data augmentation using GANs. *Proceedings of Machine Learning Research XXX*, **1**, 16 (2019).

[218] V. K. Potluru, D. Borrajo, A. Coletta, N. Dalmasso, Y. El-Laham, E. Fons et al. Synthetic data applications in finance, arXiv:2401.00081 (2024).

[219] S. Assefa. Generating synthetic data in finance: Opportunities, challenges and pitfalls. *SSRN Electronic Journal* (2020).

[220] S. Lloyd & C. Weedbrook. Quantum generative adversarial learning. *Physical Review Letters*, **121**, 040502 (2018).

[221] P.-L. Dallaire-Demers & N. Killoran. Quantum generative adversarial networks. *Physical Review A*, **98**, 012324 (2018).

[222] A. Abbas, D. Sutter, C. Zoufal, A. Lucchi, A. Figalli & S. Woerner. The power of quantum neural networks. *Nature Computational Science*, **1**, 403–409 (2021).

[223] B. Coyle, M. Henderson, J. Chan Jin Le, N. Kumar, M. Paini & E. Kashefi. Quantum versus classical generative modelling in finance. *Quantum Science and Technology*, **6**, 024013 (2021).

[224] S. Östlund & S. Rommer. Thermodynamic limit of density matrix renormalization. *Physical Review Letters*, **75**, 3537–3540 (1995).

[225] F. Verstraete, M. , V. & J. and Cirac. Matrix product states, projected entangled pair states, and variational renormalization group methods for quantum spin systems. *Advances in Physics*, **57**, 143–224 (2008).

[226] F. Eckerli & J. Osterrieder. Generative adversarial networks in finance: An overview, arXiv:2106.06364 (2021).

[227] D. Saxena & J. Cao. Generative adversarial networks (GANs): Challenges, solutions, and future directions. *ACM Computing Surveys*, **54**, 1–42 (2022).

[228] M. Arjovsky, S. Chintala & L. Bottou. Wasserstein generative adversarial networks. In *Proceedings of the 34th International Conference on Machine Learning*, pages 214–223. PMLR (2017).

[229] C. Villani. The Wasserstein distances. In C. Villani, editor, *Optimal Transport: Old and New*, pages 93–111. Springer, Berlin, Heidelberg (2009).

[230] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin & A. C. Courville. Improved training of Wasserstein GANs. *Advances in Neural Information Processing Systems*, **30** (2017).

[231] R. Sweke, J.-P. Seifert, D. Hangleiter & J. Eisert. On the quantum versus classical learnability of discrete distributions. *Quantum*, **5**, 417 (2021).

[232] M. Benedetti, D. Garcia-Pintos, O. Perdomo, V. Leyton-Ortega, Y. Nam & A. Perdomo-Ortiz. A generative modeling approach for benchmarking and training shallow quantum circuits. *npj Quantum Information*, **5**, 45 (2019).

[233] J.-G. Liu & L. Wang. Differentiable learning of quantum circuit Born machines. *Physical Review A*, **98**, 062324 (2018).

[234] J. Romero & A. Aspuru-Guzik. Variational quantum generators: Generative adversarial quantum machine learning for continuous distributions. *Advanced Quantum Technologies*, **4**, 2000003 (2021).

[235] A. Anand, J. Romero, M. Degroote & A. Aspuru-Guzik. Noise robustness and experimental demonstration of a quantum generative adversarial network for continuous distributions. *Advanced Quantum Technologies*, **4**, 2000069 (2021).

[236] A. Barthe, M. Grossi, S. Vallecorsa, J. Tura & V. Dunjko. Parameterized quantum circuits as universal generative models for continuous multivariate distributions. *npj Quantum Information*, **11**, 121 (2025).

[237] K. Shen, A. Kurkin, A. P. Salinas, E. Shishenina, V. Dunjko & H. Wang. Shadow-frugal expectation-value-sampling variational quantum generative model, arXiv:2412.17039 (2024).

[238] A. Kandala, A. Mezzacapo, K. Temme, M. Takita, M. Brink, J. M. Chow & J. M. Gambetta. Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets. *Nature*, **549**, 242–246 (2017).

[239] S. Takahashi, Y. Chen & K. Tanaka-Ishii. Modeling financial time-series with generative adversarial networks. *Physica A: Statistical Mechanics and its Applications*, **527**, 121261 (2019).

[240] M. Wiese, K. , Robert, K. , Ralf & P. and Kretschmer. Quant GANs: Deep generation of financial time series. *Quantitative Finance*, **20**, 1419–1440 (2020).

[241] K. Zhang, G. Zhong, J. Dong, S. Wang & Y. Wang. Stock market prediction based on generative adversarial network. *Procedia Computer Science*, **147**, 400–406 (2019).

[242] T. Takahashi & T. Mizuno. Generation of synthetic financial time series by diffusion models. *Quantitative Finance*, **25**, 1507–1516 (2025).

[243] J. Tian, X. Sun, Y. Du, S. Zhao, Q. Liu, K. Zhang et al. Recent advances for quantum neural networks in generative learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **45**, 12321–12340 (2023).

[244] T. A. Ngo, T. Nguyen & T. C. Thang. A survey of recent advances in quantum generative adversarial networks. *Electronics*, **12**, 856 (2023).

[245] C. Zoufal, A. Lucchi & S. Woerner. Quantum generative adversarial networks for learning and loading random distributions. *npj Quantum Information*, **5**, 1–9 (2019).

[246] S. Mourya, H. Leipold & B. Adhikari. Contextual quantum neural networks for stock price prediction. *Scientific Reports* (2026).

[247] H.-L. Huang, Y. Du, M. Gong, Y. Zhao, Y. Wu, C. Wang et al. Experimental quantum generative adversarial networks for image generation. *Physical Review Applied*, **16**, 024051 (2021).

[248] N.-R. Zhou, T.-F. Zhang, X.-W. Xie & J.-Y. Wu. Hybrid quantum–classical generative adversarial networks for image generation via learning discrete distribution. *Signal Processing: Image Communication*, **110**, 116891 (2023).

[249] D. Silver, T. Patel, W. Cutler, A. Ranjan, H. Gandhi & D. Tiwari. MosaiQ: Quantum generative adversarial networks for image generation on NISQ computers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7030–7039 (2023).

[250] S. L. Tsang, M. T. West, S. M. Erfani & M. Usman. Hybrid quantum–classical generative adversarial network for high-resolution image generation. *IEEE Transactions on Quantum Engineering*, **4**, 1–19 (2023).

[251] H. Situ, Z. He, Y. Wang, L. Li & S. Zheng. Quantum generative adversarial network for generating discrete distribution. *Information Sciences*, **538**, 193–208 (2020).

[252] P.-Y. Kao, Y.-C. Yang, W.-Y. Chiang, J.-Y. Hsiao, Y. Cao, A. Aliper et al. Exploring the advantages of quantum generative adversarial networks in generative chemistry. *Journal of Chemical Information and Modeling*, **63**, 3307–3318 (2023).

[253] D. Herr, B. Obert & M. Rosenkranz. Anomaly detection with variational quantum generative adversarial networks. *Quantum Science and Technology*, **6**, 045004 (2021).

[254] F. Fuchs & B. Horvath. A hybrid quantum Wasserstein GAN with applications to option pricing. *Available at SSRN 4514510* (2023).

[255] J. Baglio. Data augmentation experiments with style-based quantum generative adversarial networks on trapped-ion and superconducting-qubit technologies, arXiv:2405.04401 (2024).

[256] A. Di Meglio, K. Jansen, I. Tavernelli, C. Alexandrou, S. Arunachalam, C. W. Bauer et al. Quantum computing for high-energy physics: State of the art and challenges. *PRX Quantum*, **5**, 037001 (2024).

[257] E. Paquet & F. Soleymani. QuantumLeap: Hybrid quantum neural network for financial predictions. *Expert Systems with Applications*, **195**, 116583 (2022).

[258] B. T. Kiani, G. De Palma, M. Marvian, Z.-W. Liu & S. Lloyd. Learning quantum data with the quantum earth mover's distance. *Quantum Science and Technology*, **7**, 045002 (2022).

[259] S. Chakrabarti, H. Yiming, T. Li, S. Feizi & X. Wu. Quantum Wasserstein generative adversarial networks. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc. (2019).

[260] E. Schwander. *Quantum generative modelling for financial time series.* Master's thesis, Leiden University (2022).

[261] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean et al. TensorFlow: A system for large-scale machine learning. In *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation*, OSDI'16, pages 265–283. USENIX Association, USA (2016).

[262] J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin et al. JAX: composable transformations of Python+NumPy programs (2018). URL http://github.com/jax-ml/jax.

[263] J. Gray. Quimb: A Python package for quantum information and many-body calculations. *Journal of Open Source Software*, **3**, 819 (2018).

[264] C. Hogenboom. *WGAN financial time-series.* Master's thesis, University Maastricht (2025).

[265] G. M. Goerg. The Lambert way to Gaussianize heavy-tailed data with the inverse of Tukey's h transformation as a special case. *The Scientific World Journal*, **2015**, 909231 (2015).

[266] M. Larocca, S. Thanasilp, S. Wang, K. Sharma, J. Biamonte, P. J. Coles, L. Cincio, J. R. McClean, Z. Holmes & M. Cerezo. Barren plateaus in variational quantum computing. *Nature Reviews Physics*, **7**, 174–189 (2025).

[267] C. C. Shi. *Effects of observable choices in the performance of variational quantum generative models.* Master's thesis, Leiden University (2024).

[268] J. I. Cirac, D. Pérez-García, N. Schuch & F. Verstraete. Matrix product states and projected entangled pair states: Concepts, symmetries, theorems. *Reviews of Modern Physics*, **93**, 045003 (2021).

[269] I. V. Oseledets. Tensor-train decomposition. *SIAM Journal on Scientific Computing*, **33**, 2295–2317 (2011).

[270] A. Berezutskii, M. Liu, A. Acharya, R. Ellerbrock, J. Gray, R. Haghshenas et al. Tensor networks for quantum computing, arXiv:2503.08626 (2025).

[271] F. Verstraete, D. Porras & J. I. Cirac. Density matrix renormalization group and periodic boundary conditions: A quantum information perspective. *Physical Review Letters*, **93**, 227205 (2004).

[272] C. Chatfield. *Time series.* Forecasting, Boca Raton, Chapman & HaIl/CRC (1975).

[273] F. Orlandi, E. Barbierato & A. Gatti. Enhancing financial time series prediction with quantum-enhanced synthetic data generation: A case study on the S&P 500 using a quantum Wasserstein generative adversarial network approach with a gradient penalty. *Electronics*, **13**, 2158 (2024).

[274] D. Komninos. *Quantum computing for generative modeling and applications.* Master's thesis, Technical University of Crete (2023).

[275] P. Rebentrost, B. Gupt & T. R. Bromley. Quantum computational finance: Monte Carlo pricing of financial derivatives. *Physical Review A*, **98**, 022321 (2018).

[276] S. Woerner & D. J. Egger. Quantum risk analysis. *npj Quantum Information*, **5**, 15 (2019).

[277] M. MacMahon & D. Garlaschelli. Community detection for correlation matrices. *Physical Review X*, **5**, 021006 (2015).

[278] Y. Wang, Z. Hu, B. C. Sanders & S. Kais. Qudits and high-dimensional quantum computing. *Frontiers in Physics*, **8**, 589504 (2020).

[279] M. Ringbauer, M. Meth, L. Postler, R. Stricker, R. Blatt, P. Schindler & T. Monz. A universal qudit quantum processor with trapped ions. *Nature Physics*, **18**, 1053–1057 (2022).

[280] S. Y.-C. Chen, S. Yoo & Y.-L. L. Fang. Quantum long short-term memory. In *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8622–8626. IEEE, Singapore (2022).

[281] E. Beatty & D. Stilck França. Order p quantum Wasserstein distances from couplings. *Annales Henri Poincaré* (2025).

Acknowledgments

It is often said that choosing the right supervisors matters even more for a PhD than choosing the right topic. I was fortunate to have had Jordi and Vedran as supervisors. They were caring, flexible, generous with their time, and greatly complement each other. Jordi, thank you for being my first supervisor, for leading me through these years of becoming a researcher. I deeply value your dedication, enthusiasm and kindness in your supervision. Vedran, thank you for being my second supervisor. From you I learned so much about asking the right questions and communicating concisely and correctly. I would always choose you again as my supervisors.

In every project, I had the pleasure of working with different collaborators. Every one of them taught me something valuable, for which I am greatly thankful. In particular, I would like to thank Alberto for showing me the power of intuition, Liubov for teaching me clear mathematical notation and Lucas for the rewarding experience of co-supervising you.

During my PhD, I was part of the aQa group and was based at the Lorentz Institute. Being one of the first PhD students of aQa meant that I was able to experience its impressive and inspiring growth over the past years. It was a privilege to be part of such an open, friendly, and well-connected community, whose intimate and social atmosphere made most days enjoyable. There are too many members of the aQa group and the Lorentz Institute to mention here, but in particular I would like to thank my office mates Álvaro, Charles, Yash, Riccardo and Alicja for motivating discussions and laughter, Stefano for his mentorship and friendship throughout my PhD, and the aQa PIs for their efforts in creating this special environment. I would also like to thank Fran and Nina for their kindness, efficiency, and for keeping the aQa group and the Institute running so smoothly.

The defense and celebrations would not be possible without my paranymphs, Stef and Eloïc. Thank you for your friendship, for helping in organising this day and for assisting me throughout the defense.

This thesis would have been much harder without the support of all of my friends, and without the countless shared dinners, sailing days, museum visits or choir rehearsals. Thank you for being part of this journey. To my flat mates Luka, Jacek, Benoit, Patrick, Juanita and Stef, thank you for making me feel welcome at home, becoming good friends and providing me with lovely distraction in the evenings and weekends. A special thanks goes to Adrián, Marshall, Emiel, my parents, Stef, Felix and Stefano for offering me their hospitality and a roof over my head when I urgently needed a place to stay.

Finally, I am deeply grateful to my family for nurturing my curiosity and love of learning and for standing by me through all stages of my studies and life.

Samenvatting

Quantum computing vertegenwoordigt een fundamenteel nieuw paradigma van berekening en maakt nieuwe aanpakken mogelijk voor problemen die onoplosbaar zijn voor klassieke computers. Aangezien de momenteel beschikbare quantum apparaten klein, ruisgevoelig, onbetrouwbaar en duur zijn, is veel van de huidige vooruitgang gebaseerd op theoretische analyse en klassieke simulaties om hun mogelijkheden en beperkingen te begrijpen.

Dynamica beschrijft hoe de toestand van een systeem in de tijd verandert en kan wiskundig worden weergegeven door modellen zoals differentiaalvergelijkingen of tijdreeksen. In dit proefschrift onderzoeken we de toepassing van quantum computing om dergelijke dynamica vast te leggen vanuit verschillende complementaire perspectieven.

In het inleidende Hoofdstuk 1 presenteren we de fundamentele concepten. We introduceren quantum computing, variationele quantum algoritmen en ruis door sampling. Als toepassingen van variationele quantum computing bespreken we het oplossen van differentiaalvergelijkingen, quantum machine learning en toepassingen in de financiële sector. Ten slotte schetsen we verschillende onderzoeksvragen en beschrijven we de algemene structuur van het proefschrift.

In Hoofdstuk 2 onderzoeken we de rol van afgeleiden in quantum machine learning. We tonen aan dat geparametriseerde quantum circuits zowel functies als hun afgeleiden willekeurig goed kunnen benaderen, mits de invoergegevens op de juiste manier worden herschaald. Bovendien laten we zien dat het opnemen van zowel functiewaarden als afgeleidewaarden in de trainingsdataset de gegarandeerde benadering van de getrainde quantum modellen verbetert, waardoor benadering in sterkere normen mogelijk wordt die anders onbereikbaar zouden zijn. Aangezien de dynamica van een functie wordt bepaald door haar afgeleiden, verduidelijken deze inzichten hoe quantum machine learning dynamisch gedrag effectief kan vastleggen.

In Hoofdstuk 3 analyseren we een klasse van quantum algoritmen die zijn ontworpen om differentiaalvergelijkingen op te lossen. We voeren een foutenanalyse en een schatting van de benodigde middelen uit, met de nadruk op fouten die voortkomen uit de klassieke Runge-Kutta-subroutines evenals uit ruis door sampling bij het evalueren van quantum circuits. We passen deze schattingen toe op een differentiaalvergelijking uit de financiële optieprijzing en vergelijken hoe verschillende Runge-Kutta-methoden de totale benodigde rekenmiddelen beïnvloeden.

Quantum toestandstomografie is het proces waarbij een quantum toestand wordt gereconstrueerd uit meetgegevens en vormt een belangrijke subroutine in veel quantum algoritmen. In Hoofdstuk 4 presenteren we een methode om ruis door sampling in quantum toestandstomografie te verminderen door zowel meetgegevens als fysische beperkingen te formuleren als een semidefiniet programma. We tonen aan dat, afhankelijk van de onderliggende quantum toestand, er ruisregimes bestaan waarin onze methode beter presteert dan andere geavanceerde tomografietechnieken.

Ten slotte demonstreren we in Hoofdstuk 5 de toepassing van quantum generatieve adversariële netwerken voor het genereren van synthetische financiële tijdreeksen. We simuleren de quantum circuits met behulp van zowel volledige toestandsimulaties als tensor-netwerk gebaseerde simulaties. Voor klassieke modellen blijft het een uitdaging om synthetische financiële tijdreeksen te genereren die zowel de doelverdelingen volgen als realistische temporele correlaties vertonen. We tonen aan dat onze quantum modellen deze statistische en temporele eigenschappen kwalitatief goed kunnen vastleggen.

Summary

Quantum computing represents a fundamentally new paradigm of computation, enabling approaches to problems that are intractable for classical computers. Since currently available quantum devices are small, noisy, unreliable, and expensive, much of today's progress relies on theoretical analysis and classical simulations to understand their capabilities and limitations.

Dynamics describes how the state of a system changes over time, and can be mathematically represented by models such as differential equations or time series. In this thesis, we investigate the application of quantum computing to capturing such dynamics from several complementary perspectives.

In the introductory Chapter 1, we present the foundational concepts. We introduce quantum computing, variational quantum algorithms, and shot noise. As applications of variational quantum computing, we discuss solving differential equations, quantum machine learning, and applications to finance. Finally, we outline several research questions and describe the overall structure of the thesis.

In Chapter 2, we explore the role of derivatives in quantum machine learning. We demonstrate that parameterized quantum circuits can approximate both functions and their derivatives arbitrarily well, provided that the input data are appropriately rescaled. Furthermore, we show that incorporating both function values and derivative values in the training data set enhances the guaranteed approximation of the trained quantum models, allowing approximation in stronger norms that would otherwise be unattainable. As the dynamics of a function are governed by its derivatives, these insights clarify how quantum machine learning can effectively capture dynamical behavior.

In Chapter 3, we analyze a class of quantum algorithms designed to solve differential equations. We conduct an error analysis and resource estimation, focusing on errors arising from the classical Runge-Kutta subroutines as well as from shot noise in evaluating quantum circuits. We apply these estimates to a differential equation from financial option pricing and compare how different Runge-Kutta methods affect the total computational resources required.

Quantum state tomography is the process of reconstructing a quantum state from measurement data and constitutes an important subroutine in many quantum algorithms. In Chapter 4, we present a method for mitigating shot noise in quantum state tomography by formulating both measurement data and physical constraints as a semidefinite program. We show that, depending on the underlying quantum state, there exist noise regimes in which our method outperforms other state-of-the-art tomography techniques.

Finally, in Chapter 5, we demonstrate the application of quantum generative adversarial networks to generating synthetic financial time series. We simulate the quantum circuits using both full-state and tensor network-based simulations. For classical models, generating synthetic financial time series that both follow the target distributions and exhibit realistic temporal correlations remains challenging. We show that our quantum models can qualitatively capture these statistical and temporal properties well.

Zusammenfassung

Quantencomputing stellt ein grundlegend neues Paradigma der Berechnung dar und ermöglicht Ansätze für Probleme, die für klassische Computer unlösbar sind. Da die derzeit verfügbaren Quantencomputer klein, verrauscht, unzuverlässig und teuer sind, beruht ein Großteil des heutigen Fortschritts auf theoretischer Analyse und klassischen Simulationen, um ihre Fähigkeiten und Einschränkungen zu verstehen.

Dynamik beschreibt, wie sich der Zustand eines Systems im Laufe der Zeit verändert, und kann mathematisch durch Modelle wie Differentialgleichungen oder Zeitreihen dargestellt werden. In dieser Arbeit untersuchen wir die Anwendung von Quantencomputing zur Erfassung solcher Dynamiken aus mehreren komplementären Perspektiven.

Im einleitenden Kapitel 1 stellen wir die grundlegenden Konzepte vor. Wir führen in Quantencomputing, variationelle Quantenalgorithmen und Schrotrauschen ein. Als Anwendungen des variationellen Quantencomputings diskutieren wir das Lösen von Differentialgleichungen, Quanten-Maschinelles Lernen und Anwendungen in der Finanzwissenschaft. Abschließend skizzieren wir mehrere Forschungsfragen und beschreiben den allgemeinen Aufbau der Arbeit.

In Kapitel 2 untersuchen wir die Rolle von Ableitungen im Quanten-Maschinellen Lernen. Wir zeigen, dass parametrisierte Quantenschaltkreise sowohl Funktionen als auch deren Ableitungen beliebig gut approximieren können, sofern die Inputdaten entsprechend skaliert werden. Darüber hinaus zeigen wir, dass die Einbeziehung von sowohl Funktionswerten als auch Ableitungswerten in den Trainingsdatensatz die garantierte Approximation der trainierten Quantenmodelle verbessert und Approximationen in stärkeren Normen ermöglicht, die sonst unerreichbar wären. Da die Dynamik einer Funktion durch ihre Ableitungen bestimmt wird, verdeutlichen diese Erkenntnisse, wie Quanten-Maschinelles

Lernen dynamisches Verhalten effektiv erfassen kann.

In Kapitel 3 analysieren wir eine Klasse von Quantenalgorithmen, die zur Lösung von Differentialgleichungen entwickelt wurden. Wir führen eine Fehleranalyse und Ressourcenabschätzung durch und konzentrieren uns dabei auf Fehler, die sowohl aus den klassischen Runge-Kutta-Subroutinen als auch aus dem Schrotrauschen bei der Auswertung von Quantenschaltkreisen entstehen. Wir wenden diese Abschätzungen auf eine Differentialgleichung aus der Finanzoptionsbewertung an und vergleichen, wie verschiedene Runge-Kutta-Methoden die insgesamt benötigten Rechenressourcen beeinflussen.

Die Quanten-Zustandstomographie ist der Prozess der Rekonstruktion eines Quantenzustands aus Messdaten und stellt eine wichtige Subroutine in vielen Quantenalgorithmen dar. In Kapitel 4 stellen wir eine Methode zur Minderung von Schrotrauschen in der Quanten-Zustandstomographie vor, indem sowohl Messdaten als auch physikalische Nebenbedingungen als semidefinites Programm formuliert werden. Wir zeigen, dass, abhängig vom zugrunde liegenden Quantenzustand, Rauschregime existieren, in denen unsere Methode andere aktuelle Tomographietechniken übertrifft.

Abschließend demonstrieren wir in Kapitel 5 die Anwendung von Quanten-Generativen Adversarial Networks zur Erzeugung synthetischer finanzieller Zeitreihen. Wir simulieren die Quantenschaltkreise sowohl mit Vollzustands- als auch mit Tensornetzwerk-basierten Simulationen. Für klassische Modelle bleibt es eine Herausforderung, synthetische finanzielle Zeitreihen zu erzeugen, die sowohl den Zielverteilungen folgen als auch realistische zeitliche Korrelationen aufweisen. Wir zeigen, dass unsere Quantenmodelle diese statistischen und zeitlichen Eigenschaften qualitativ gut erfassen können.

Curriculum Vitæ

I was born on 28 December 1995 in Fulda, Germany, where I grew up on a farm in a nearby village. In Fulda, I attended the Rabanus-Maurus-Schule (Domgymnasium Fulda), from which I graduated in 2014.

In the same year, I began my studies in mathematics with a minor in theoretical physics at Georg-August-Universität Göttingen. Fascinated by the theoretical foundations of physics, I also enrolled in the bachelor's program in physics and completed both degrees in 2018. During an exchange semester at the National University of Singapore in fall 2017, I took a course on quantum information and computation, which sparked my lasting interest in this field. As there was no research group in quantum information in Göttingen at that time, I conducted my bachelor's thesis in condensed matter theory under the supervision of Fabian Heidrich-Meisner and Thomas Schick.

To deepen my understanding of quantum physics and quantum information, I continued with a master's degree in physics at the University of Copenhagen from 2018 to 2020, specializing in quantum physics. I wrote my master's thesis in the QMath group under the supervision of Matthias Christandl.

For my PhD studies, I joined the Applied Quantum Algorithms (aQa) group and the Lorentz Institute at Leiden University, where I was supervised by Jordi Tura and Vedran Dunjko. My research focused on variational quantum algorithms and their applications. I collaborated with various researchers on all of my projects and had the opportunity to co-supervise several bachelor's and master's students. During my PhD, I participated in several international schools and conferences, presenting my work and broadening my knowledge and network in the Netherlands, Germany, France, Spain, the United Kingdom, and Belgium.

Following the completion of my PhD, I will continue my research as a postdoctoral researcher at the Technical University of Hamburg in the group of Martin Kliesch, working on fault-tolerant quantum algorithms.

List of publications

[88] A. Manzano, **D. Dechant**, J. Tura & V. Dunjko. Approximation and generalization capacities of parametrized quantum circuits for functions in Sobolev spaces. *Quantum*, **9**, 1658 (2025).
[Chapter 2 is based on this publication.]

[46] **D. Dechant**, L. Markovich, V. Dunjko & J. Tura. Error and resource estimates of variational quantum algorithms for solving differential equations based on Runge-Kutta methods. *Journal of Mathematical Physics* **67**, 012205 (2026).
[Chapter 3 is based on this publication.]

[90] Z. J. Wang, **D. Dechant**, Y. J. Patel & J. Tura. Mitigating shot noise in local overlapping quantum tomography with semidefinite programming. *Physical Review A*, **111**, 052444 (2025).
[Chapter 4 is based on this publication.]

[91] **D. Dechant**, E. Schwander, L. van Drooge, C. Moussa, D. Garlaschelli, V. Dunjko & J. Tura. Quantum generative modeling for financial time series with temporal correlations. *Machine Learning: Science and Technology* (2026).
[Chapter 5 is based on this publication.]