Universiteit
Leiden
The Netherlands

Capturing dynamics with noisy quantum computers

Dechant, D.S.

Citation

Dechant, D. S. (2026, February 17). *Capturing dynamics with noisy quantum computers*. Retrieved from https://hdl.handle.net/1887/4290771

Version:	Publisher's Version
License:	Licence agreement concerning inclusion of doctoral thesis in the Institutional Repository of the University of Leiden
Downloaded from:	https://hdl.handle.net/1887/4290771

Note: To cite this publication please use the final published version (if applicable).

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

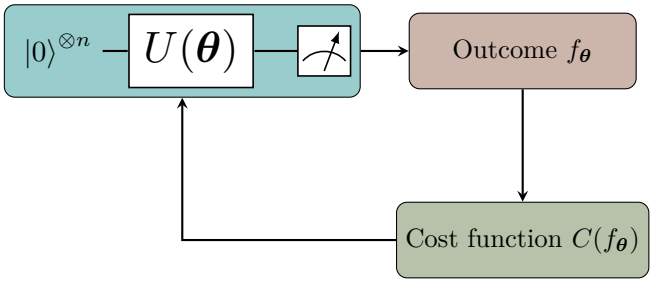

Figure 1.1: Sketch of a variational quantum algorithm. $U(\boldsymbol{\theta})$ represents a parameterized quantum circuit with variational parameters $\boldsymbol{\theta}$ that takes x as input, $f_{\boldsymbol{\theta}}$ is the expectation value of an observable and $C(f_{\boldsymbol{\theta}})$ is the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

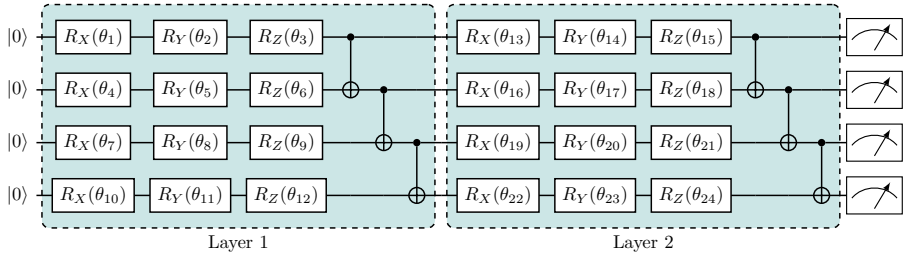

Figure 1.2: Example of a parameterized quantum circuit with 4 qubits and 2 layers, in the form of a hardware efficient Ansatz. Each layer consists of single-qubit Pauli rotations and CNOT gates. The single-qubit Pauli rotations are tunable with parameters θ_i.

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.

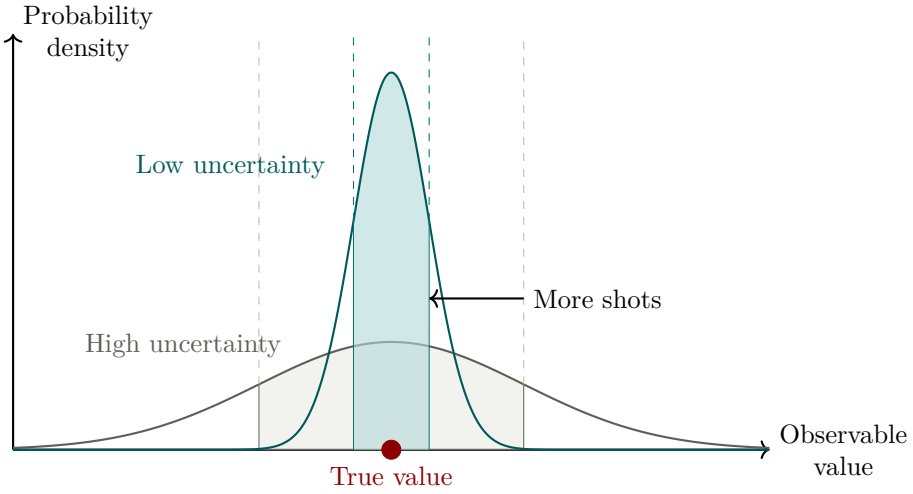

Figure 1.3: Increasing the number of shots in quantum measurements reduces the uncertainty of the measured observable. The shaded areas in this sketch show the probability density functions within standard deviation, indicating the area in which the true value lies with a probability of $\approx 68,27\%$.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

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

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

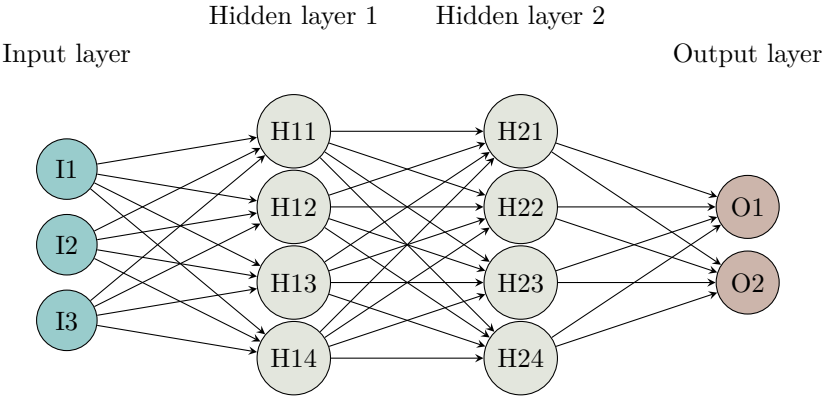

Figure 1.4: Sketch of a deep neural network with two hidden layers that each have four hidden units (H11-H14, H21-H24). The input layer (I1-I3) receives data and the output layer (O1, O2) produces the predictions of the network. The arrows show how information flows through the network, each corresponds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

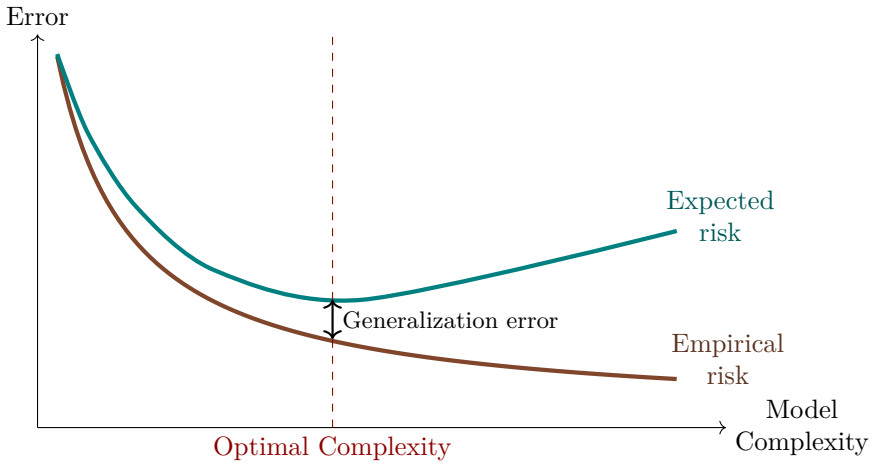

Figure 1.5: Illustration of empirical risk and expected risk as functions of model complexity. Here, *model complexity* refers to the expressiveness or capacity of the model class. This depends on factors such as the number of trainable parameters, circuit depth, or the number of qubits. As complexity 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].

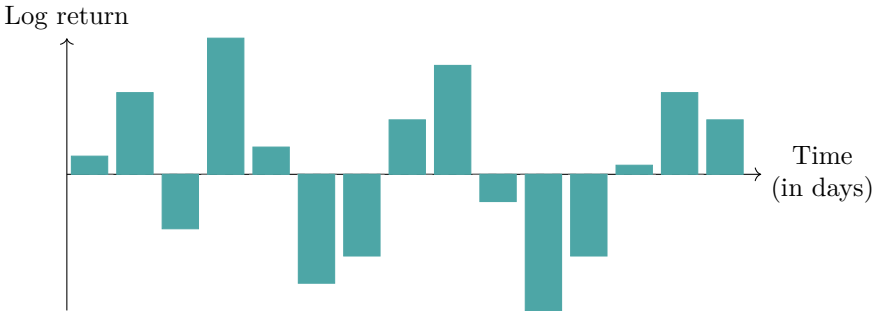

Figure 1.6: Sketch of a financial time series of log returns over several days, illustrating typical fluctuations around zero.

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

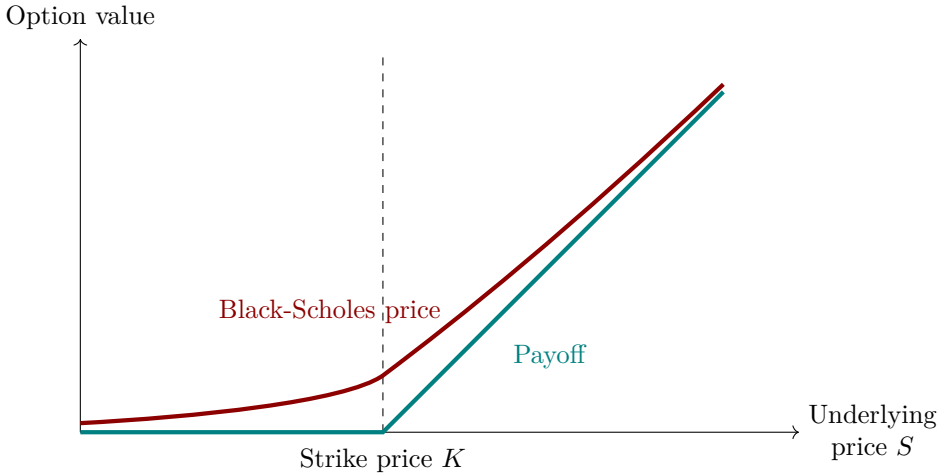

Figure 1.7: Sketch of the payoff (blue) and the Black-Scholes price (red) of a European call option at a specific time and with strike price K as a function of the underlying price S. The option price equals the payoff at maturity, and lies above it before maturity due to time value.

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.