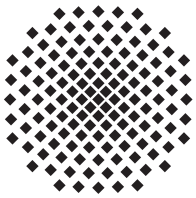


University of Stuttgart

Quantum Algorithms and Quantum Machine Learning for Differential Equations

Author: Niclas Schillo

Revised edition of the master's thesis of the same name at the Institute for Functional Matter and Quantum Technologies supervised by Jun.-Prof. Dr. Sungkun Hong and co-supervised by Dr. Andreas Sturm; submitted in October 2023.

Abstract

The fast and accurate solution of differential equations is a highly researched topic. Classical methods are able to solve very large systems, however, this can require high-performance computers and very long computational times. Since quantum computers promise significant advantages over classical computers, quantum algorithms for the solution of differential equations have received a lot of attention. Particularly interesting are algorithms that are relevant in the current Noisy-Intermediate-Scale-Quantum (NISQ) era, characterized by small and error-prone systems. In this context, promising candidates are variational quantum algorithms which are hybrid quantum-classical algorithms where only a part of the algorithm is executed on the quantum computer. Thus, they typically require fewer qubits and qubit gates and can tolerate the errors stemming from an imperfect quantum computer.
One important example of variational quantum algorithms is the so-called quantum circuit learning (QCL) algorithm, which can be used to approximate functions. Here, an ansatz function is formed with a data encoding layer, subsequently transformed by a shallow parameterized circuit and finally the measurement of an expectation value defines the function value.
In this thesis, this method is investigated in great detail, developing new and improved circuit designs and comparing their usefulness in approximating different functions. The method is also tested on a real quantum computer, which has not been reported in literature yet. For this purpose, the algorithm is executed on the superconducting quantum computer IBM Quantum System One in Ehningen to investigate its applicability in the NISQ era.
The concept of QCL can be combined with the parameter shift rule to determine derivatives. This enables the solution of nonlinear differential equations. This procedure is subjected to thorough testing across a multitude of differential equations while being compared to other quantum algorithms for solving differential equations. The strengths of this algorithm are shown but also the weaknesses are analyzed. Going beyond the current state of research, the method is extended to solve coupled differential equations with a single circuit, significantly reducing the computational effort. Lastly, a differential equation is successfully solved on the quantum computer IBM Quantum System One in Ehningen.

Contents

1 Introduction

Differential equations play a key role in science and engineering by providing the mathematical framework for countless natural phenomena [1]. Hence, their efficient and accurate solution is highly relevant in numerous scientific fields and also in various branches of industry. However, many differential equations are very hard to simulate e.g. due to their high degree of non-linearities or numerical instability. Thus, classical numerical methods for solving differential equations are a broad field of investigation. One commonly used approach is the finite difference method, where differential operators are replaced by discrete counterparts defined on a grid of the domain of the differential equation [2, 3]. In order to obtain an accurate solution a very fine grid is often required, which leads to a very high computational cost.
Another approach are spectral methods, where the solution of a differential equation is represented via a set of global basis functions. Here, for example, Fourier series or Chebyshev polynomials can be chosen as model functions. This ansatz transforms the problem into finding optimal free parameters to match the solution of the differential equation as closely as possible [4]. This method is also associated with great challenges: For complicated problems, a high number of basis functions must be selected and the process becomes very computationally expensive.
It is known that quantum algorithms can achieve a significant computational speedup for certain problems. One of two very well-known examples is Shor's algorithm [5], which can factorize numbers in polynomial time, while the best classical algorithms need subexponential time. The other notable example is Grover's search algorithm [6], where the quantum algorithm exhibits a quadratic speedup compared to the best-known classical algorithm.
Because of this potential advantage, a lot of research revolves around solving differential equations with the help of quantum algorithms in the hope of finding new techniques. For example, in the last few years, several quantum algorithms for solving differential equations have been proposed, which heavily rely on classical computation and only use quantum algorithms as a subroutine [7–11]. They share a commonality by utilizing quantum phase estimation [12] as the core element in their quantum component. Additionally, they employ oracle-based approaches for data access and rely on amplitude-encoded states. This comes with several caveats, such as the input and output problem [13], as well as significant computational overheads in constructing

quantum oracles. Especially for nonlinear differential equations quantum algorithms are sparse, with only a few notable examples [14].
It is important to note that the execution of all these algorithms requires a fault-tolerant quantum computer. However, current and near-future quantum computers have high error rates and few qubits. The intermediate phase is called the Noisy-Intermediate-Scale-Quantum (NISQ) era [15]. This makes the implementation of algorithms for large-scale fault-tolerant quantum computers unfeasible in the near future. Therefore, particularly intriguing are algorithms that are relevant in the context of the NISQ era.
Variational quantum algorithms use parameterized quantum circuits combined with a classical optimizer to tackle a multitude of problems. This makes variational quantum algorithms especially interesting in the context of the NISQ era since they typically need fewer qubits and qubit gates and work with higher error rates. Applications of this particular subfield of quantum algorithms have been explored in recent years for various problems. This development started in quantum chemistry with the rise of the variational quantum eigensolver (VQE) [16], which can be used to find the ground state of a given physical system. Promising applications of variational quantum algorithms now exist in many other areas such as optimization problems with the quantum approximate optimization algorithm (QAOA) [17] or generic machine learning tasks like image recognition [18, 19].
In this work, we consider a particular class of variational quantum algorithms, namely quantum circuit learning (QCL) [20]. This approach involves using variational quantum algorithms to approximate functions. Here, an ansatz function is formed by encoding the variable of a function into the qubit states. This step is called data encoding. Subsequently, a shallow parameterized circuit with entangling gates transforms these states. Finally, the value of the function is defined by a measurement of an expectation value.
Building upon this idea, it is possible to use QCL in combination with the parameter shift rule to solve differential equations [21]. The parameter shift rule is a mathematical approach to obtain gradients of a parameterized quantum circuit. This approach, which will be thoroughly investigated in this thesis, enables the solution of nonlinear differential equations using very shallow circuits with low qubit numbers making it suitable for the NISQ era.
This thesis is organized as follows: Chapter 2 introduces the basics of quantum computing and the parameter shift rule is derived in detail. Next, in Chapter 3, current quantum hardware is explained using the example of the real superconducting quantum computer IBM Quantum System One in Ehningen. In particular, the characteristics of NISQ computers are discussed with this example. Afterwards, variational quantum algorithms are explained in Chapter 4 and their relevance in the NISQ era

is discussed. Moreover, a comprehensive explanation of the QCL method explored in this thesis is provided in this chapter. Following this, Chapter 5 delves deeper into the method of function approximation with QCL, exploring various circuit designs and comparing their efficiency in approximating different functions. The theory behind QCL is investigated in depth in Chapter 6 and illuminated to gain transferable knowledge for other algorithms. With this new knowledge, the complexity of these circuits is reduced to a necessary level while still maintaining the same functionality to ensure their executability on current and near-future devices. To investigate its applicability in the NISQ era, in Chapter 7, the method is tested on the real superconducting quantum computer IBM Quantum System One in Ehningen. In Chapter 8, the parameter shift rule is investigated in more detail and also tested on the IBM Quantum System One. Following this, in Chapter 9, the method of solving differential equations with the parameter shift rule is subjected to thorough testing across a multitude of differential equations, while also being compared to other quantum algorithms. The purpose of this comparison is to examine the strengths and weaknesses, particularly in the context of the NISQ era. Furthermore, the method is extended to solve coupled differential equations with a single circuit which significantly reduces the computational effort. Lastly, a differential equation is solved on the quantum computer IBM Quantum System One in Ehningen.

2 Introduction to Quantum Computing

In this chapter, we present the fundamental principles and techniques of quantum computing and introduce a universal language for describing quantum computations. For more information, we recommend [22], which this chapter is loosely based on.

2.1 Quantum States

In classical computing, information is represented by bits, which can assume the values 0 or 1. In quantum information, the fundamental unit is not the bit but the so-called quantum bit which is usually abbreviated as qubit. Analogous to a classical bit, a qubit consists of two distinguishable quantum states. Those states can be described with the two orthogonal vectors

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

However, unlike classical discrete states, a qubit exhibits a continuous nature, making it possible to be in any superposition of those two states. This means that every qubit state can be described as a linear combination

$$|\psi\rangle = a\,|0\rangle + b\,|1\rangle \ ,$$

where $a, b \in \mathbb{C}$ are complex numbers that satisfy the normalization condition

$$|a|^2 + |b|^2 = 1\,. \tag{2.1}$$

The inner product space of the quantum states is called Hilbert space.
Because of condition (2.1), the state of a qubit can be rewritten as

$$|\psi\rangle = e^{i\varphi_1} \cos\left(\frac{\theta}{2}\right) |0\rangle + e^{i\varphi_2} \sin\left(\frac{\theta}{2}\right) |1\rangle \ ,$$

with $\theta \in [0, \pi]$ and $\varphi_i \in [0, 2\pi)$. Since the global phase of a qubit state cannot be measured, it is irrelevant in the context of quantum computing. The state is therefore

simplified to only include a relative phase φ between $|0\rangle$ and $|1\rangle$. This results in the expression

$$|\psi\rangle = \cos\left(\frac{\theta}{2}\right)|0\rangle + e^{i\varphi}\sin\left(\frac{\theta}{2}\right)|1\rangle \ ,$$

with $\varphi \in [0, 2\pi)$. The state can now be displayed on a sphere when the parameters θ and φ are re-interpreted as spherical coordinates. This sphere is called the Bloch sphere [23]. The state vector of an arbitrary qubit state $|\psi\rangle$ on the Bloch sphere is shown in Figure 1.

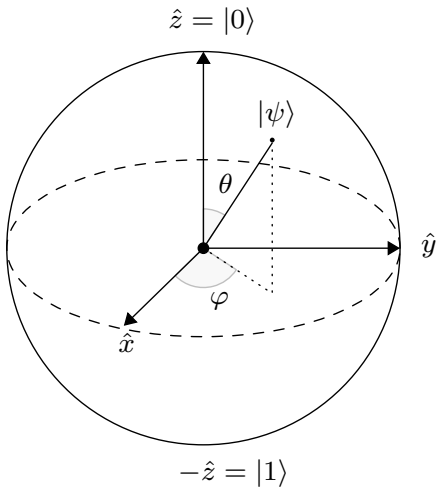


Figure 1: Bloch sphere with the state vector of an arbitrary state $|\psi\rangle$.

2.2 Measurement and Expectation Value

When we measure the state of a qubit in the standard computational basis $\{|0\rangle, |1\rangle\}$, the probability of the outcome $|0\rangle$ is $|a|^2$ and the probability of the outcome $|1\rangle$ is $|b|^2$. Because the absolute squares of the amplitudes are probabilities, condition (2.1) can be explained.

After a measurement, the system collapses into the measured state and thus the operation is irreversible. As a consequence, it is not possible to determine the exact state, because the qubit is always in the state $|0\rangle$ or $|1\rangle$ after a measurement in the computational basis. Only by doing many measurements of identical states, the probability amplitudes a and b can be estimated and the state can be reconstructed. These multiple measurements are called shots and the associated statistical error is called shot noise. The shot noise decreases with the square root of the number of shots.

A measurable property and its associated operator acting in Hilbert space is called observable. An important metric in quantum information theory is the expectation value with respect to an observable $\hat{A}$. The expectation value gives the average of

all the possible outcomes of a measurement depending on their probabilities. The expectation value for an arbitrary quantum state $|\psi\rangle$ can be expressed as

$$\langle \hat{A} \rangle = \langle \psi | \hat{A} | \psi \rangle \,,$$

where $\langle \psi | = |\psi\rangle^\dagger$ is the conjugate transpose (or Hermitian adjoint) of the vector $|\psi\rangle$.

2.3 Quantum Gates

Qubit states can be manipulated by applying quantum gates to them. Such transformations have to be described by a unitary matrix U. A unitary matrix is defined as

$$U^\dagger U = U U^\dagger = I \,,$$

where I is the identity matrix and $U^\dagger$ is the conjugate transpose of the matrix U. Three commonly used operators in quantum computing are the so-called Pauli operators. They are denoted as

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \text{ and } Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} .$$

The Pauli-X operator expresses a bit-flip, the Pauli-Z operator a phase-flip and the Pauli-Y operator a bit- and phase-flip. Because of this fundamental property, Pauli operators occupy a key role in quantum information.
The Pauli operators can be used to obtain three additional very useful unitary matrices when they are exponentiated. These are the rotational operators $R_X(\theta)$, $R_Y(\theta)$ and $R_Z(\theta)$ which rotate the state around the corresponding axes on the Bloch sphere. They are defined as

$$R_X(\theta) = e^{-i\theta X/2} = \begin{pmatrix} \cos\frac{\theta}{2} & -i\sin\frac{\theta}{2} \\ -i\sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{pmatrix} ,$$

$$R_Y(\theta) = e^{-i\theta Y/2} = \begin{pmatrix} \cos\frac{\theta}{2} & -\sin\frac{\theta}{2} \\ \sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{pmatrix} \text{ and}$$

$$R_Z(\theta) = e^{-i\theta Z/2} = \begin{pmatrix} e^{-i\theta/2} & 0 \\ 0 & e^{i\theta/2} \end{pmatrix} .$$

Another important quantum gate is the Hadamard gate which is defined by the matrix

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

This gate creates an equal superposition state if applied to the computational basis states

$$H\,|0\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}} \quad \text{and} \quad H\,|1\rangle = \frac{|0\rangle - |1\rangle}{\sqrt{2}} .$$

If we consider the Bloch sphere, the Hadamard gate performs a π rotation around the $(\hat{x} + \hat{z})/\sqrt{2}$ axis.

2.4 Multi Qubit Systems

For multiple qubits, the system can be described with the help of the tensor product. The tensor product for two vectors $\vec{v} = (v_1, v_2, ..., v_K)^T$ and $\vec{w} = (w_1, w_2, ..., w_L)^T$ results in a $K \times L$-matrix defined as

$$(\vec{v} \otimes \vec{w})_{ij} = v_i w_j ,$$

where $\otimes$ denotes the tensor product. It is a way of combining vector spaces of the individual qubits to form a larger Hilbert space. A system of N qubits is represented by a 2^N-dimensional Hilbert space whose basis states can be written as tensor products of the single qubit states. We first consider a system consisting of two qubits. The four computational basis states are

$$|0\rangle \otimes |0\rangle = |00\rangle , |0\rangle \otimes |1\rangle = |01\rangle , |1\rangle \otimes |0\rangle = |10\rangle \text{ and } |1\rangle \otimes |1\rangle = |11\rangle .$$

An arbitrary two-qubit state has the form

$$|\psi\rangle = a_{00}\,|00\rangle + a_{01}\,|01\rangle + a_{10}\,|10\rangle + a_{11}\,|11\rangle ,$$

with $a_{ij} \in \mathbb{C}$. Here, the probability of measuring a basis state $|ij\rangle$ is $|a_{ij}|^2$. Hence, similar to the single-qubit case, the condition

$$|a_{00}|^2 + |a_{01}|^2 + |a_{10}|^2 + |a_{11}|^2 = 1$$

has to be fulfilled. A fundamental characteristic of multi-qubit states is that entanglement can exist. Entanglement means that several qubits share a state without the possibility of assigning well-defined states to the individual qubits. An entangled

two-qubit state can not be expressed as a product state of the form

$$|\psi\rangle_{q1} |\phi\rangle_{q2}$$

where $|\psi\rangle_{q1}$ and $|\phi\rangle_{q2}$ are states of the first qubit and the second qubit, respectively. Hence, entangled qubit states can not be described independently of the state of the other qubit.

Next, we focus on multi-qubit gates. One of the most important multi-qubit gates is the controlled NOT gate or CNOT gate which is defined by the matrix

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

This gate has two input qubits: the control qubit and the target qubit. If the control qubit is set to $|0\rangle$, the target qubit remains unchanged. If the control qubit is set to $|1\rangle$, the target qubit is flipped. Applying the gate to the computational basis states, it maps the states

$$|00\rangle \rightarrow |00\rangle , \; |01\rangle \rightarrow |01\rangle , \; |10\rangle \rightarrow |11\rangle \text{ and } |11\rangle \rightarrow |10\rangle .$$

With the implementation of controlled NOT gates and single-qubit gates, all conceivable unitary operations can be performed. This is called a universal gate set.

Another very relevant gate is the SWAP gate. This gate swaps the state and therefore the information of the two qubits involved in the operation. It is defined by the matrix

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

This gate becomes especially important when we want to realize multi-qubit gates on hardware platforms where not all physical qubits are connected. Then separate qubits can be swapped, making multi-qubit operations possible between all qubits.

Collections of quantum gates interconnected by quantum wires are called quantum circuits. Quantum wires do not represent physical wires but indicate that an operation is applied to the same qubit at a later time. A simple example with two qubits is shown in Figure 2.

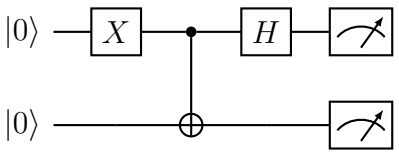


Figure 2: An example of a quantum circuit with an X gate followed by a controlled NOT gate and a Hadamard gate. In the end, both states are measured in the computational basis.

Quantum circuits can visualize complex algorithms in a manageable way and thus they will be heavily used in this work.

2.5 Density Matrices

Density matrices or density operators are a powerful generalization of the state vectors to describe quantum states. For a state $|\psi\rangle$, the density matrix is defined as

$$\rho = |\psi\rangle \langle\psi| \ .$$

Quantum states that can be expressed with a state vector $|\psi\rangle$ are called pure states. Those states lie on the surface of the Bloch sphere. Due to interactions with the environment, noise and decoherence, it is possible that a qubit is in a so-called mixed state. This is a statistical combination of different pure states. A density matrix can represent such states and indicate the probability of a system being in a particular pure state. For an ensemble of pure states, the density matrix is defined as

$$\rho = \sum_i p_i \, |\psi_i\rangle \langle\psi_i| \ ,$$

where p_i is the probability of measuring state $|\psi_i\rangle$. Mixed states can be represented by points inside the Bloch sphere. An expression that nicely shows this behavior is

$$\rho = \tfrac{1}{2}(I + xX + yY + zZ) \, ,$$

where X, Y and Z are the Pauli operators and $x, y, z \in \mathbb{R}$ are the coordinates of the Bloch sphere. For a pure state the condition

$$x^2 + y^2 + z^2 = 1$$

has to be true and for a mixed state

$$x^2 + y^2 + z^2 < 1\,.$$

All the expressions we introduced for state vectors can be reformulated for density matrices. For instance, if a unitary operator U acts on the system we obtain the new density matrix

$$\rho' = U\rho U^\dagger\,.$$

Measurements can also be written in the density matrix formalism. If we measure in a certain basis $\{|\psi\rangle\,, |\psi\rangle_\perp\}$, the probability of measuring the state $|\psi\rangle$ is defined by

$$\mathsf{Prob}(|\psi\rangle) = \langle\psi|\rho|\psi\rangle\ .$$

The expectation value of a given observable $\hat{A}$ can be calculated by

$$\langle\hat{A}\rangle = \mathsf{tr}(\rho\hat{A})\,,$$

where tr() indicates the trace of the matrix. We consider the special case where the observable $\hat{A}$ is a Pauli operator P. Then we obtain

$$\langle P\rangle = \mathsf{tr}(\rho P) = p \quad \text{with} \quad (p, P) \in \{(x, X), (y, Y), (z, Z)\}\,.$$

Due to the special formalism of density matrices, it can also be advantageous to represent pure states as density matrices, which will be applied in the course of this work.

2.6 Parameter Shift Rule

The parameter shift rule is a method to determine the derivative of a parameterized circuit [20, 24]. We consider a circuit that depends on the parameter x. The expectation value $y(x)$ regarding an observable $\hat{B}$ can be expressed as

$$y(x) = \langle\psi|U_P^\dagger(x)\hat{B}U_P(x)|\psi\rangle\ ,$$

where $U_P(x) = e^{-i\frac{x}{2}P}$ with $P \in \{X, Y, Z\}$. The gradient of $U_P(x)$ is given by

$$\frac{d}{dx}U_P(x) = -\frac{i}{2}Pe^{-i\frac{x}{2}P} = -\frac{i}{2}PU_P(x)\,. \tag{2.2}$$

Using the product rule and (2.2), the derivative of $y(x)$ can be written as

$$y'(x) = \frac{i}{2} \langle\psi|U_P^\dagger(P\hat{B} - \hat{B}P)U_P|\psi\rangle = \frac{i}{2} \langle\psi|U_P^\dagger[P,\hat{B}]U_P|\psi\rangle \ , \tag{2.3}$$

where $[P,\hat{B}] = P\hat{B} - \hat{B}P$ is the commutator. For the unitary operator $U_P(x)$ we get the expression

$$U_P\left(x = \pm\frac{\pi}{2}\right) = \cos\left(\frac{\pi}{4}\right) I \mp i \sin\left(\frac{\pi}{4}\right) P = \frac{1}{\sqrt{2}}(I \mp iP) \, .$$

Together with

$$U_P^\dagger(x) = U_P(-x)$$

the mathematical identity

$$[P,B] = -i\left(U_P^\dagger\left(\frac{\pi}{2}\right)\hat{B}U_P\left(\frac{\pi}{2}\right) - U_P^\dagger\left(-\frac{\pi}{2}\right)\hat{B}U_P\left(-\frac{\pi}{2}\right)\right)$$

can be derived. We combine this identity with Equation (2.3) to obtain

$$\begin{aligned} y'(x) =& \frac{1}{2}\left(\langle\psi|U_P^\dagger\left(x+\frac{\pi}{2}\right)\hat{B}U_P\left(x+\frac{\pi}{2}\right)|\psi\rangle\right. \\ & \left. - \langle\psi|U_P^\dagger\left(x-\frac{\pi}{2}\right)\hat{B}U_P\left(x-\frac{\pi}{2}\right)|\psi\rangle\right) \\ =& \frac{1}{2}\left(y\left(x+\frac{\pi}{2}\right) - y\left(x-\frac{\pi}{2}\right)\right) \, . \end{aligned}$$

Here, we have found a method of determining the derivative of the expectation value with respect to the parameter x. Only a few generalizations need to be made at this point. So far, we have considered the simple case where x occurs only at one place in the quantum circuit. However, the parameter x can also occur several times in different gates in the same circuit. If we want to determine the total derivative of x, we have to include a sum according to the product rule. The total derivative is described by the sum

$$y'(x) = \frac{1}{2}\sum_j \left(y_j\left(x+\frac{\pi}{2}\right) - y_j\left(x-\frac{\pi}{2}\right)\right) \, ,$$

where j indicates the different gates where the parameter x is included.
This parameter can additionally be embedded in an inner function $\varphi(x)$. This inner function must be considered according to the chain rule. The resulting derivative can be expressed as

$$y'(x) = \frac{1}{2}\sum_j \left(y_j\left(\varphi(x)+\frac{\pi}{2}\right) - y_j\left(\varphi(x)-\frac{\pi}{2}\right)\right) \cdot \varphi'(x) \, .$$

3 Quantum Hardware

Before discussing how qubits can be constructed, we review the general requirements necessary to build a qubit. Most importantly, we need a physical system with two distinguishable configurations corresponding to the computational states $|0\rangle$ and $|1\rangle$. In the case of an atom, this is usually the ground state and the first excited state of a valence electron. We can reliably distinguish these states because the ground state and the excited state have two different energy values and therefore can be identified with a measurement. Secondly, these states must be quantum mechanical in nature, i.e., they must be quantum states expressing phenomena such as superposition and entanglement. Finally, we need to consider that most quantum systems have more states than just the two previously mentioned. Atoms, for example, have an infinite number of energy states. However, we do not want our system to be in any of the other states. To avoid this, we must have a system with non-uniform spacing between energy levels. Thus, if we restrict ourselves to only interacting with the system with the energy ΔE between our two chosen states, we will not go beyond the states that define our qubit. In atoms, the non-uniform energy spacing is already inherent. In other systems, such as superconducting qubits, this must be actively considered.

3.1 Introduction to Superconducting Qubits

One of the most widely used and well-researched approaches for building quantum computers is the construction of superconducting qubits. This technology has progressed to the point where it has already entered the industry and many companies have begun building superconducting quantum computers, as we will see later in the example of IBM.
The fundamental effect that enables the construction of superconducting qubits is, as the name suggests, superconductivity. This is a quantum phenomenon where below a certain critical temperature the material transitions into a thermodynamic phase where the electrical resistance disappears completely. Some electrical circuits with superconducting wires cooled to temperatures of about 10 mK exhibit discrete energy

levels, similar to the energy levels in atoms. The simplest way to construct an electrical circuit that exhibits discrete energy levels is to combine a capacitor of capacitance C and an inductor of inductance L, as shown in Figure 3.

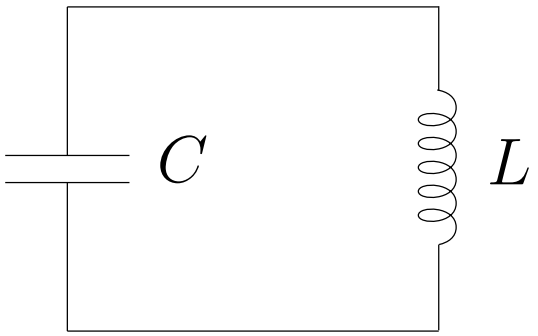


Figure 3: Circuit diagram of an LC circuit with a capacitor of capacitance C and an inductor of inductance L.

This so-called LC circuit is an electrical circuit with an oscillating current at the frequency

$$\omega^2 = \frac{1}{LC}.$$

This circuit has great importance in the classical world and is found in a great number of technical devices from antennas to induction heaters. When this circuit is cooled to a very low temperature with superconducting wires, this oscillation has discrete energy levels, similar to the states in an atom. Unlike atoms, however, the energy gaps between the different states are constant, which means that it is not possible to address only two states. Fortunately, we can change this by integrating a Josephson junction. A Josephson junction consists of a very thin piece of insulating material between two superconducting metals. Electrons can tunnel through this junction which reduces the current. If we replace the inductor with such a junction, the energy levels of the superconducting circuit become unevenly spaced. Now, we have a behavior similar to atomic states, which is ideal for the construction of a qubit.
Another requirement for a quantum computer is a controlled interaction with the qubits. This can be realized by adding an additional gate capacitor with capacitance C_g to the circuit, so that it can receive external signals. The circuit is shown in Figure 4.

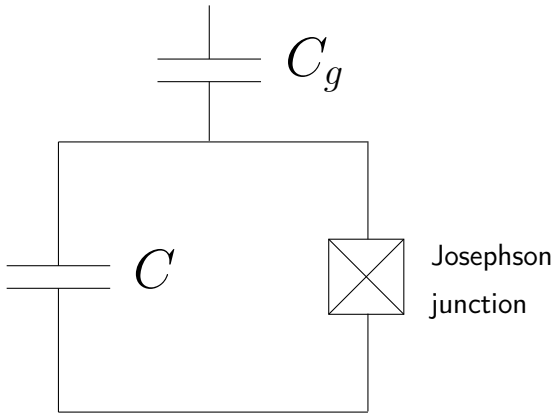


Figure 4: Circuit diagram of a superconducting qubit with an additional gate capacitor with capacitance C_g which is used to interact with the qubit.

To avoid disturbing the uneven energy spacing the capacitances C and C_g must be fine-tuned. The regime that has been found to be ideal is called the transmon regime and qubits in this regime are called transmon qubits or transmon in short. For these qubits, we can use wavelengths in the microwave range to interact with the qubits and implement single-qubit gates using finely adjusted microwave pulses.
Another challenge is the implementation of two-qubit gates. To obtain two two-qubit gates, a coupling capacitor with capacitance C_c connects two transmons. A schematic illustration is shown in Figure 5.

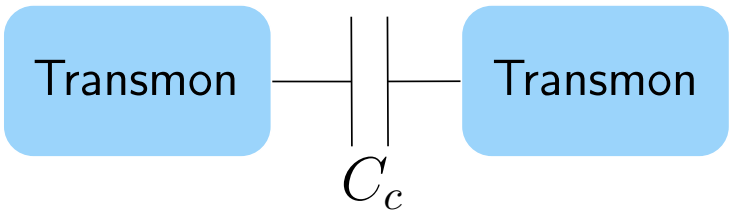


Figure 5: Illustration of a coupling capacitor with capacitance C_c between two transmon qubits.

With this approach, we can implement two-qubit gates by applying microwave pulses depending on the capacitance of the coupling capacitor.

3.2 IBM Quantum System One Ehningen

The Ehningen-based IBM quantum computer is a System One device. Those devices are the world's first commercial, universal quantum computers. The quantum chip in Ehningen is based on transmon qubits and is part of the chip-series called "Falcon". The quantum device in Ehningen is reserved exclusively for utilization by the Fraunhofer Society and its collaborating partners. Its design is characterized by a modular, compact structure optimized for stability and automatic calibration with the goal of providing uninterrupted service.
The quantum computer is located in a 2.7 m by 2.7 m airtight enclosure crafted from 1.27-centimeter thick borosilicate glass. Independent aluminum and steel frames are used to decouple the cryostat, control electronics and outer casing, which enhances the system's performance by reducing noise. The technical specifications for the quantum system in Ehningen are shown in Table 1.

IBM Quantum System One Ehningen	
Processor type	Falcon r5.11
Basis Gates	CX, ID, RZ, SX, X
Qubit number	27
Coherence time	$\approx$ 150 us
Single qubit gate error	$\approx$ 0.025%
Two qubit gate error	$\approx$ 0.8%
Readout error	$\approx$ 1%
CNOT gate time	$\approx$ 300 ns
Quantum Volume	64

Table 1: Technical specifications of IBM Quantum System One Ehningen.

The times and error rates can vary and depend on the exact qubits on the chip. It should also be noted that this chip, like all others in this series, does not have all-to-all connectivity but only certain qubits are connected to each other and form a ring-like structure. The exact structure of the connections is called a coupling map. The coupling map for the IBM Quantum System One Ehningen is shown in Figure 6.

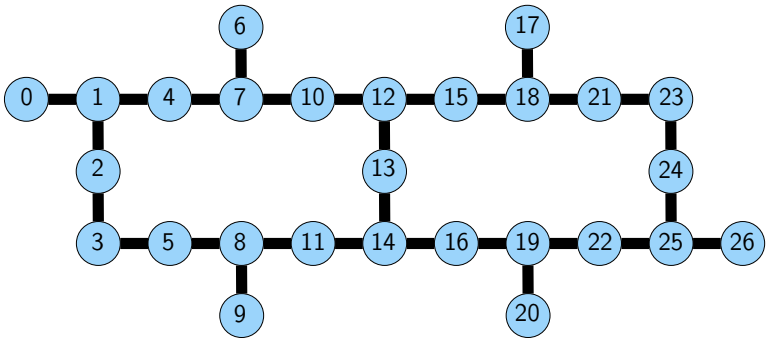


Figure 6: Coupling map of IBM Quantum System One Ehningen.

However, all qubits can still be entangled with each other by applying additional SWAP gates. Because these lead to additional errors, the exact choice of the qubits is very important.

3.3 Noisy-Intermediate-Scale-Quantum Era

As can be seen in the previous example of the IBM Quantum System One, the current state of quantum computing is defined by error-prone quantum processors of under 1000 qubits. This is the case not only for superconducting quantum computers but also for all other existing technologies such as trapped-ion quantum computers [25]. To realize fault-tolerant quantum computers, error correction algorithms have to be implemented. In these, several physical qubits are used to build a so-called logical qubit. Those algorithms protect the logical qubits from any errors that occur in the physical qubits. In order to implement qubit error correction, the errors of the physical qubits have to be below a certain threshold. This property is described by the threshold theorem [26].
However, the current processors are susceptible to environmental effects, have low coherence times and high gate errors and therefore lack the ability to apply the required quantum error correction algorithms. Those types of quantum computers that have not yet reached a level of development suitable for fault tolerance can be evaluated by their so-called quantum volume. It gives an overview of the number of qubits and the fidelity of the gates [27]. John Preskill introduced the term Noisy-Intermediate-Scale-Quantum (NISQ) era in 2018 to describe this transitional period until fault-tolerant quantum computers can be constructed [15]. In recent years, there has therefore been an increased search for algorithms that bring a quantum advantage despite these limitations. There are strong indications that NISQ computers can become useful [28]. However, much research is still needed in this area. Some promising algorithms are described in the following chapter.

4 Quantum Circuit Learning

In this chapter, we will explain quantum circuit learning which is the main algorithm for this work. This algorithm belongs to the class of variational quantum algorithms which will be introduced first.

4.1 Introduction to Variational Quantum Algorithms

In recent years, much effort has been put into the development of suitable algorithms for NISQ systems, with prominent candidates like the variational quantum eigensolver (VQE) [16] and the quantum approximate optimization algorithm (QAOA) [17]. These algorithms and many more are so-called variational quantum algorithms which are hybrid quantum-classical algorithms. In those algorithms, a cost function of a parameterized quantum circuit is evaluated and the parameters of this circuit are updated using classical optimizers to minimize the cost function. This process is illustrated in Figure 7.

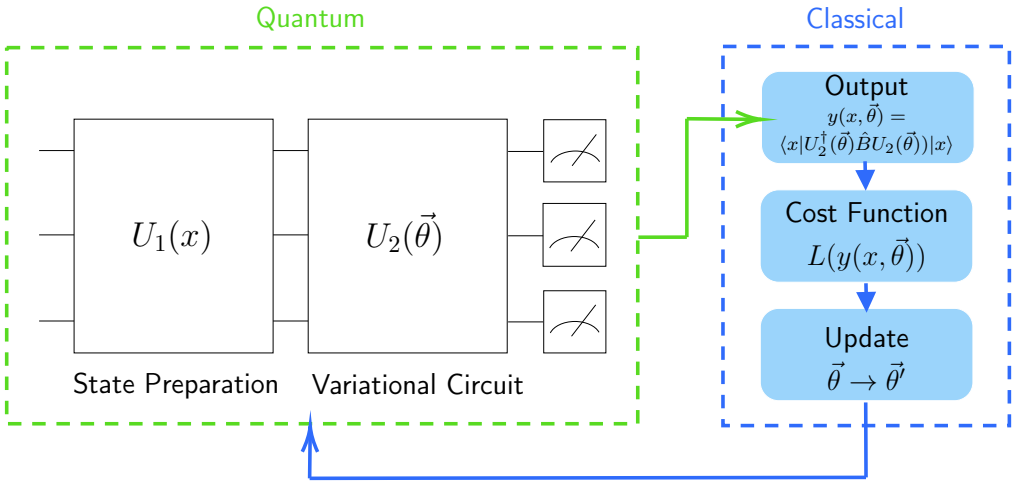


Figure 7: Example of a quantum variational algorithm consisting of state preparation $U_1(x)$ to initialize the state $|x\rangle$, the parameterized quantum circuit $U_2(\vec{\theta})$, measurement regarding some observable $\hat{B}$, classical processing and the updating of the variational parameters. This cycle is repeated until a desired accuracy is achieved.

As shown in Figure 7, a state preparation $U_1(x)$ is performed to initialize a desired state $|x\rangle$. This state can be problem-specific or it is the same for a specific algorithm. Next, a parameterized quantum circuit $U_2(\vec{\theta})$ is applied to the state. This can have a variety of different architectures. Finally, an expectation value of this new state $\langle x|U_2^\dagger(\vec{\theta})\hat{B}U_2(\vec{\theta})|x\rangle = y(x,\vec{\theta})$ is measured and processed classically. The state is evaluated using a problem-specific cost function $L(y(x,\vec{\theta}))$ and the variational parameters are adjusted accordingly to minimize its value.
As we have discussed in Chapter 3.3, NISQ computers are very prone to errors and noise, which heavily affects the computations. Variational quantum algorithms can adapt to these perturbations by optimizing parameters to achieve the most accurate results possible, thus accounting for systematic errors. In addition, NISQ computers have very limited qubit numbers and limited coherence times. As described previously, variational quantum algorithms are mostly based on hybrid approaches, combining classical and quantum-based computations. Thus only smaller problems are computed on the quantum computer which typically require fewer qubits and qubit gates and can tolerate errors stemming from NISQ computers. Hence, variational quantum algorithms are suited to efficiently use the limited resources to perform useful computations. They are capable of solving complex problems with compact circuits.

4.2 Quantum Circuit Learning

The quantum variational algorithm that will be investigated in this work is quantum circuit learning (QCL) [20]. This algorithm can be applied to approximate functions. The underlying idea is to encode a variable, which we call x, into the circuit. This step is called data encoding and can take different forms. After the information about the x-value is included in the circuit, a parameterized part is added as a next step and then a measurement of an expectation value is performed. The expectation value defines the function value $y(x)$.
To understand how specific functions $f(x)$ can be approximated so that the expectation value $y(x) \approx f(x)$, we focus on the example of a polynomial function. To start with, we look at a simple example of a density matrix. As presented in Chapter 2.5, the density matrix for the state $|0\rangle$ can be expressed as

$$\rho = |0\rangle\langle 0| = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} = \frac{1}{2}\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} + \frac{1}{2}\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} = \frac{1}{2}(I + Z)\,.$$

For our consideration, it makes sense to represent the density matrices with Pauli operators. Now, we add a rotation around the y-axis

$$R_Y(\theta) = \begin{pmatrix} \cos\frac{\theta}{2} & -\sin\frac{\theta}{2} \\ \sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{pmatrix} = \cos\left(\frac{\theta}{2}\right) I - i \cdot \sin\left(\frac{\theta}{2}\right) Y \,.$$

The density matrix of the resulting state can be described as

$$\begin{aligned} \rho'(\theta) &= R_Y(\theta) \cdot \rho \cdot R_Y(\theta)^\dagger \\ &= \frac{1}{2}\left(I + \sin(\theta)X + \cos(\theta)Z\right) \\ &= \frac{1}{2}\left(I + \sin(\theta)X + \sqrt{1-\sin^2(\theta)}Z\right) \,. \end{aligned}$$

We choose $\theta = \arcsin(x)$ to eliminate the trigonometric parts of the function and obtain

$$\tilde{\rho} = \rho'(\arcsin(x)) = \frac{1}{2}\left(I + xX + \sqrt{1-x^2}Z\right) \,. \tag{4.1}$$

This results in the variable x occurring in polynomial terms with an additional $\sqrt{1-x^2}$-term. With this, we have already completed a simple form of data encoding and the x-value is included in the quantum state. Up until now, we only regarded a single qubit. However, it is possible to use multiple qubits to achieve functions with higher orders.

Let us consider a system with N qubits with the data encoding in (4.1) on every qubit. With this, we obtain a tensor product of density matrices of the form

$$\tilde{\rho}(x) = \frac{1}{2^N}\bigotimes_{i=1}^{N}\left(I + xX + \sqrt{1-x^2}Z\right) \,. \tag{4.2}$$

To understand how we can use this expression to reach higher-order polynomials, we write the density matrix for the case of $N = 2$

$$\begin{aligned} \tilde{\rho}(x) =& \frac{1}{4}\bigotimes_{i=1}^{2}\left(I + xX + \sqrt{1-x^2}Z\right) \\ =& \frac{1}{4}\Big((I\otimes I) + \sqrt{1-x^2}(I\otimes Z) + x(I\otimes X) \\ &+ \sqrt{1-x^2}(Z\otimes I) + (1-x^2)(Z\otimes Z) + x\sqrt{1-x^2}(Z\otimes X) \\ &+ x(X\otimes I) + x\sqrt{1-x^2}(X\otimes Z) + x^2(X\otimes X)\Big) \\ =& \frac{1}{4}\Big((I\otimes X + X\otimes I)\,x + (I\otimes Z + Z\otimes I)\,\sqrt{1-x^2} \\ &+ (X\otimes X - Z\otimes Z)\,x^2 + (Z\otimes X + X\otimes Z)\,x\sqrt{1-x^2} \\ &+ (I\otimes I + Z\otimes Z)\Big) \,. \end{aligned}$$

We see that the tensor product creates a polynomial function with additional $\sqrt{1-x^2}$-terms up to the order of x^N. To make use of the higher orders, we want to be able to also measure them. For this purpose, we perform a measurement of the expectation value

$$\langle \hat{A} \rangle = \mathrm{tr}(\tilde{\rho}(x)\hat{A}) = y(x)\,,$$

with respect to an observable $\hat{A}$. In the case of $N = 2$, all functions $y(x)$ can be described by the function

$$\xi_\alpha(x) = \alpha_0 \cdot x + \alpha_1 \cdot \sqrt{1-x^2} + \alpha_2 \cdot x^2 + \alpha_3 \cdot x \cdot \sqrt{1-x^2} + \alpha_4\,, \tag{4.3}$$

with $\xi_\alpha(x) \in [-1, 1]$. This means that the expectation value $y(x)$ is a subset of $\xi_\alpha(x)$. The function $\xi_\alpha(x)$ will be called the "ansatz function".
In the end, we want the higher orders to be obtainable from the Z expectation value of a single qubit. To make this possible, we need to perform additional operations before the measurement. For this purpose, we implement multi-qubit gates. Those multi-qubit gates introduce entanglement, which is necessary for the higher-order terms to be transferred into a single-qubit observable. This means that if we entangle all qubits, the higher orders of Equation (4.2) can be transferred to a single qubit state.
For this reason, entangling gates in the form of controlled NOT gates are introduced. The two-qubit case is shown in Figure 8.

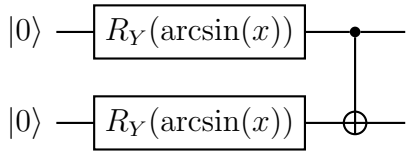


Figure 8: Two-qubit circuit with $R_Y(\arcsin(x))$-encoding followed by a controlled NOT gate.

In this work, circular entanglement is applied for more than two qubits, which can be achieved with a linear chain of entanglement gates and an additional entanglement gate between the first and the last qubit. In general, a variety of different entanglement methods are possible. For example, the time evolution of an Ising Hamiltonian can be used to create a highly entangled state [20]. However, in a large number of experiments we have found that controlled NOT gates give good results. Additionally, controlled NOT gates are part of the physical gate set (basis gates) of the IBM Quantum System One Ehningen, where this algorithm will be tested on. Therefore, this work does not focus on the analysis of different entanglement methods.

Next, we want to approximate specific functions $f(x)$. For this purpose, parameterized rotational gates are introduced. This changes the states in such a way that the expectation value $y(x)$, measured in the end, matches the desired function $f(x)$. The structure of this parameterized part and how the parameters are determined will be described in the following.
The parameterized part consists of a series of $\vec{\theta}$-parameterized rotational gates, with $\vec{\theta} = (\theta_0, \theta_1, ..., \theta_{3N-1})^T$. The variational parameters are introduced to reach as much of the functional space of Equation (4.3) as possible. The resulting circuit for two qubits can be seen in Figure 9.

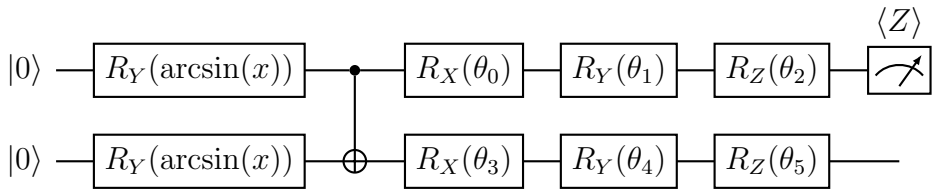


Figure 9: Two-qubit circuit with $R_Y(\arcsin(x))$-encoding followed by a controlled NOT gate and $\vec{\theta}$-parameterized x-, y- and z-rotations.

On each qubit, the $\vec{\theta}$-parameterized part consists of a parameterized x-, y- and z-rotation with $\theta_i \in [0, 2\pi)$, which is sufficient to achieve any unitary single-qubit operation apart from a global phase [22]. The specific choice of the rotational gates allows multiple possibilities with the same effect. It is also possible to use an x-, z- and additional x-rotation [20]. However, the arrangement described above has produced the best results.
The first qubit's Z expectation value is measured, defining the function value $y(x)$. The selection of the qubit is arbitrary and does not affect the performance of the algorithm. The goal is to obtain different expectation values $y(x) \in [-1, 1]$ for $x \in (-1, 1)$ of the form of (4.3) by changing $\vec{\theta}$:

$$y(x) = \alpha_0(\vec{\theta}) \cdot x + \alpha_1(\vec{\theta}) \cdot \sqrt{1 - x^2} + \alpha_2(\vec{\theta}) \cdot x^2 + \alpha_3(\vec{\theta}) \cdot x \cdot \sqrt{1 - x^2} + \alpha_4(\vec{\theta})$$

However, for the shallow circuit in Figure 9, it is not possible to achieve many variations of (4.3). This will be discussed in depth in Chapter 6.
To improve the ability to reach arbitrary functions, the entangling gates and the $\vec{\theta}$-parameterized gates are repeated multiple times up to a certain depth D. This is shown in Figure 10.

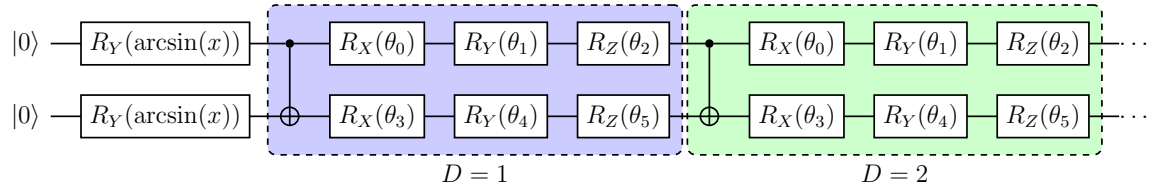


Figure 10: Two-qubit circuit with $R_Y(\arcsin(x))$-encoding followed by blocks of a controlled NOT gate and θ-parameterized x-, y- and z-rotations. The number of those blocks is defined as the depth D.

The same parameters $\vec{\theta}$ are repeated with every block, therefore the number of parameters stays the same and depends only on the qubit number. The effect of the depth is thoroughly discussed in Chapter 6.

To approximate a given function $f(x)$, the parameters $\vec{\theta}$ have to be chosen accordingly. However, the circuits are much too complicated to define the parameters analytically. Therefore, they are determined with the help of a classical optimizer. For this purpose, a cost function is defined, which is then minimized with the classical optimizer. This process is called training. The cost function is evaluated at several points of the function $f(x)$. These points are called training points. Up to a certain number, more training points lead to a more accurate result but also to more computational effort. The cost function that has shown to be the best suited for approximating functions is

$$L = \sum_i ||f(x_i) - y(x_i)||^2 , \tag{4.4}$$

where $y(x)$ is the Z expectation value of the first qubit and $\sum_i$ sums over a number of training points. In such optimizations, random training points are often chosen to prevent unwanted periodicity in the result. Aside from that, especially for polynomials, the so-called Chebyshev nodes are a suitable choice for the position of the training points because the resulting approximation minimizes the effect of Runge's phenomenon [29].

We have now described the process by which arbitrary functions can be approximated. In the following chapter, function approximation with QCL will be tested with the help of simulators.

5 Function Approximation: Simulations

The QCL method explained in Chapter 4.2 is the basis for approximating functions with a polynomial ansatz. In the following, we consider simple examples, to test the algorithm. Unwanted periodicity does not play a significant role here. Hence, for the sake of simplicity, 20 equidistant training points are chosen. The starting parameters for the classical optimizer are chosen randomly. Three simple examples with the three-qubit circuit in Figure 10 with a depth $D = 3$ and the cost function (4.4) on an error-free simulator without shot noise (exact simulator) are shown in Figure 11.

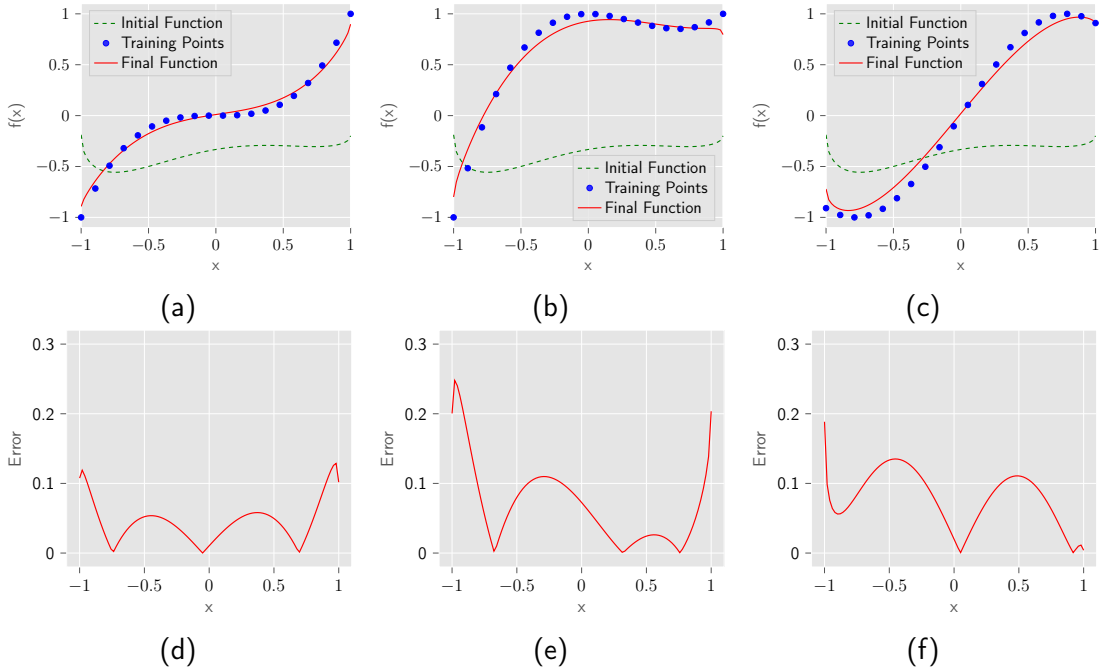


Figure 11: Function approximation on an exact simulator using $R_Y(\arcsin(x))$ data encoding with a qubit number $N = 3$ and a depth $D = 3$ with $f(x) = x^3$ (a), $f(x) = x^3 - x^2 + 1$ (b) and $f(x) = \sin(2x)$ (c). The cost function is evaluated on 20 equidistant training points and is minimized using SLSQP. The initial function shows the circuit output with the randomly chosen starting parameters. Additionally, in (d)-(f) the absolute values of the respective errors are plotted.

The cost function is classically minimized using Sequential Least Squares Programming (SLSQP) [30]. It can be seen that it is possible to approximate functions with

the QCL method. However, for more complex functions, there are strong deviations, which become evident in the error plots. To improve the approximation, a classical parameter σ_0 is introduced

$$L = \sum_i ||f(x_i) - y(x_i) \cdot \sigma_0||^2 . \tag{5.1}$$

As an additional classical post-processing step, this parameter is multiplied by the z expectation value $y(x)$. In addition to improving the results, the classical parameter changes the value range to $y(x) \cdot \sigma_0 \in \mathbb{R}$ and the approximations are no longer limited to functions with $f(x) \in [-1, 1]$. The same functions from Figure 11 are approximated with the new classical parameter σ_0. The resulting plots are shown in Figure 12.

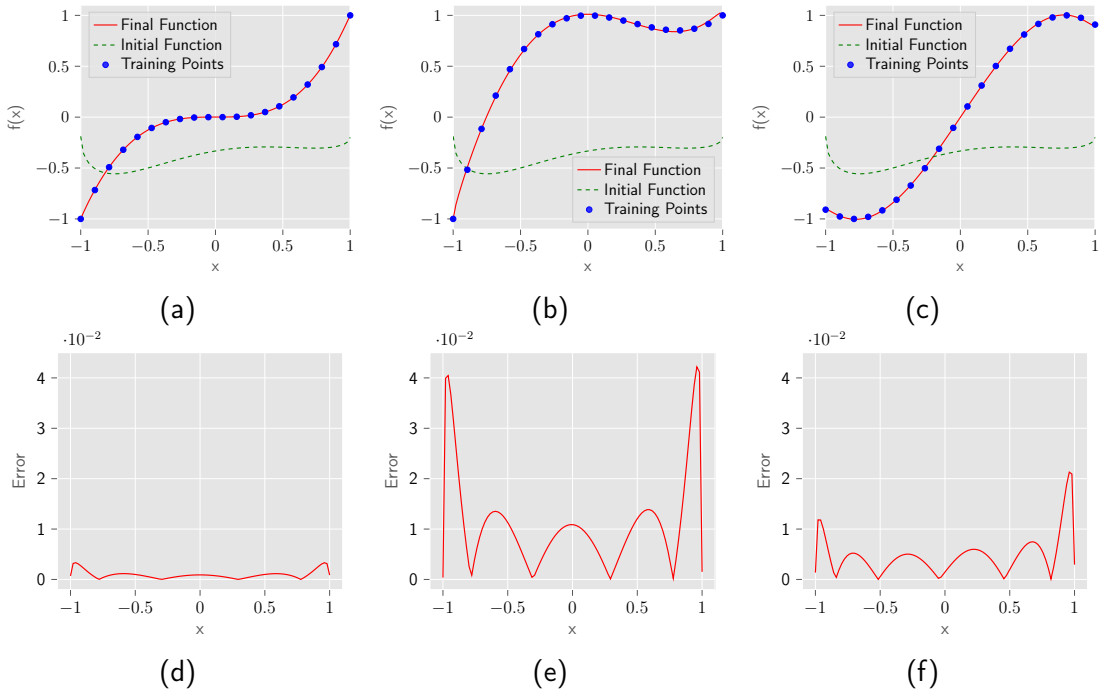


Figure 12: Function approximation on an exact simulator with a classical parameter σ_0 using $R_Y(\arcsin(x))$ data encoding with a qubit number $N = 3$ and a depth $D = 3$ with $f(x) = x^3$ (a), $f(x) = x^3 - x^2 + 1$ (b) and $f(x) = \sin(2x)$ (c). The cost function is evaluated on 20 equidistant training points and is minimized using SLSQP. The initial function shows the circuit output with the randomly chosen starting parameters. Additionally, in (d)-(f) the absolute values of the respective errors are plotted.

This significantly improves the ability to approximate more complicated functions. The errors are considerably reduced. The reasons for this improvement are discussed in more detail in Chapter 6. The magnitude of the errors is so small, that in the case of an execution on a real quantum computer, shot noise would far overshadow these

errors. Thus, the goal of this algorithm is to approximate the qualitative behavior of the functions and not to achieve an accuracy that comes close to classical methods.

5.1 Different Data Encoding Schemes

Up until this point, we only looked at the data encoding scheme in the form of $R_Y(\arcsin(x))$ rotations to generate a polynomial ansatz function with additional $\sqrt{1-x^2}$-terms [20]. This data encoding scheme is therefore also best suited to approximate polynomial functions.
Different circuits that generate different ansatz functions can be implemented, making them better suited for certain problems. Very complex approaches are possible, such as Chebyshev feature maps [21], which form a complete set of independent functions and make it possible to approximate a large number of functions with a relatively small number of qubits. Due to this high complexity, however, such functions can lead to overfitting, especially if noise is introduced. Overfitting is an undesirable behavior in variational algorithms that occurs when the algorithm provides accurate predictions for training data but not for new data. In the case of QCL, it means that the function approximation matches very well at the position of the training points, but deviates significantly from the desired function between those points. In addition, the optimization process for very complicated ansatz functions like Chebyshev feature maps takes notably longer. We therefore focus on simpler data encoding schemes like the $R_Y(\arcsin(x))$ rotations explained before.
Another very simple example, which we have not considered so far, can be seen in Figure 13.

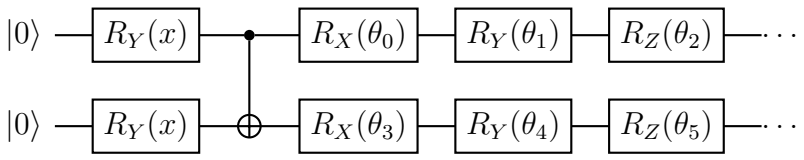


Figure 13: Two-qubit circuit with $R_Y(x)$-encoding followed by a controlled NOT gate and θ-parameterized x-, y- and z-rotations.

Here, the data encoding consists of just y-rotations according to the x-value and no function $\varphi(x)$ is included. Analogous to the explanation in Chapter 4.2, by introducing multiple qubits, the tensor product results in a higher-order ansatz function. For the example of two qubits the ansatz function can be expressed as

$$\xi_\beta(x) = \beta_0 + \beta_1 \cdot \sin(x) + \beta_2 \cdot \cos(x) + \beta_3 \cdot \sin(2x) + \beta_4 \cdot \cos(2x)\,, \tag{5.2}$$

with $\xi_\beta(x) \in [-1, 1]$. Again, the first qubit's Z expectation value $y(x)$ is a subset of $\xi_\beta(x)$.

The variational part allows us to get arbitrary functions $y(x) \in [-1, 1]$ with $x \in \mathbb{R}$ of the form of (5.2). Like before, a classical parameter σ_0 as introduced in (5.1) is multiplied by $y(x)$, which changes the value range to $y(x) \cdot \sigma_0 \in \mathbb{R}$. Due to the trigonometric ansatz function, this circuit is expected to perform better in approximating trigonometric functions. Three simple examples with a qubit number $N = 3$ and a depth $D = 3$ are shown in Figure 14.

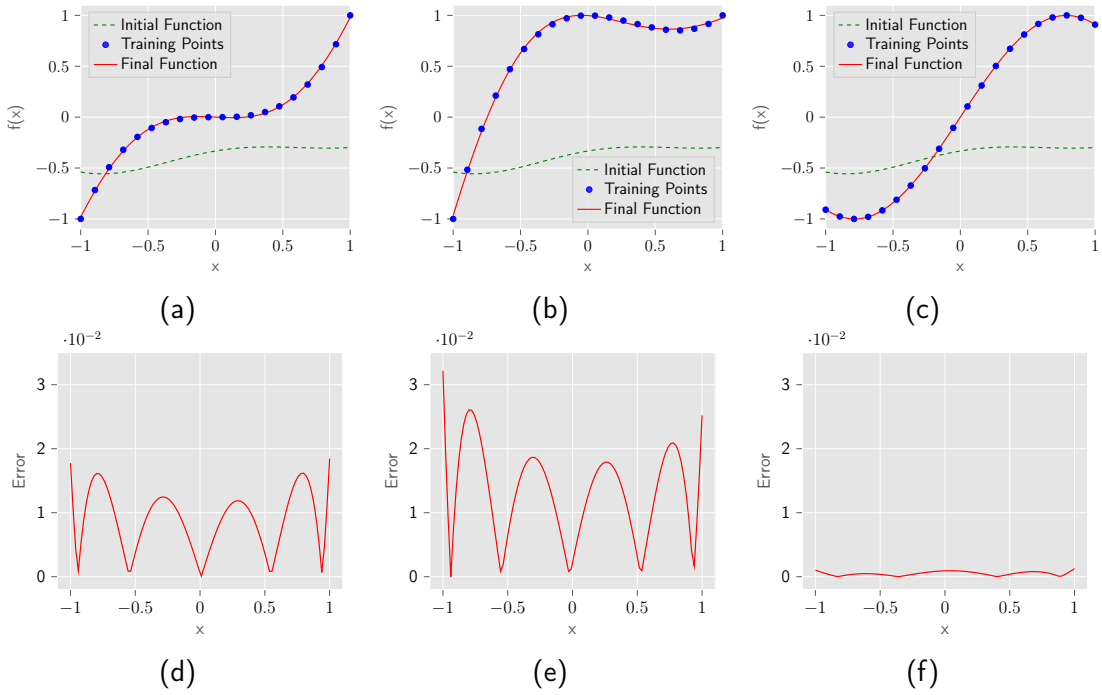


Figure 14: Function approximation on an exact simulator using $R_Y(x)$ data encoding with a qubit number $N = 3$ and a depth $D = 3$ with $f(x) = x^3$ (a), $f(x) = x^3 - x^2 + 1$ (b) and $f(x) = \sin(2x)$ (c) with a classical parameter σ_0. The cost function is evaluated on 20 equidistant training points and is minimized using SLSQP. The initial function shows the circuit output with the randomly chosen starting parameters. Additionally, in (d)-(f) the absolute values of the respective errors are plotted.

It can be seen that this ansatz is indeed more suited for trigonometric functions which results in lower errors.

In general, arbitrary encoding schemes can be used. Those encoding schemes can be tailored to specific problems. For example, a polynomial ansatz and a trigonometric ansatz can be combined to obtain a combination of polynomial and trigonometric ansatz functions. Such a circuit can be seen in Figure 15.

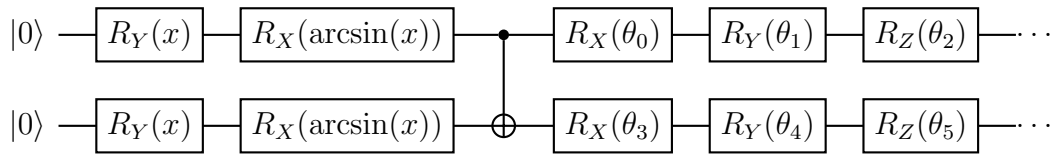


Figure 15: Two-qubit circuit with $R_Y(x)$- and $R_X(\arcsin(x))$-encoding followed by a controlled NOT gate and θ-parameterized x-, y- and z-rotations.

This example can be advantageous for more complex functions without having an overcomplicated ansatz and being prone to overfitting. An exemplary function, where this ansatz is advantageous, is the solution of a damped harmonic oscillator. This differential equation will be solved later in Chapter 9.3. For now, we only focus on the solution. With the conditions defined in Chapter 9.3, the solution of the differential equation is

$$f(x) = \cos(6x)e^{-x}\,. \tag{5.3}$$

The approximated function with $N = 4$ and $D = 4$ can be seen for different data encoding schemes in Figure 16. The number of training points is chosen to be 20.

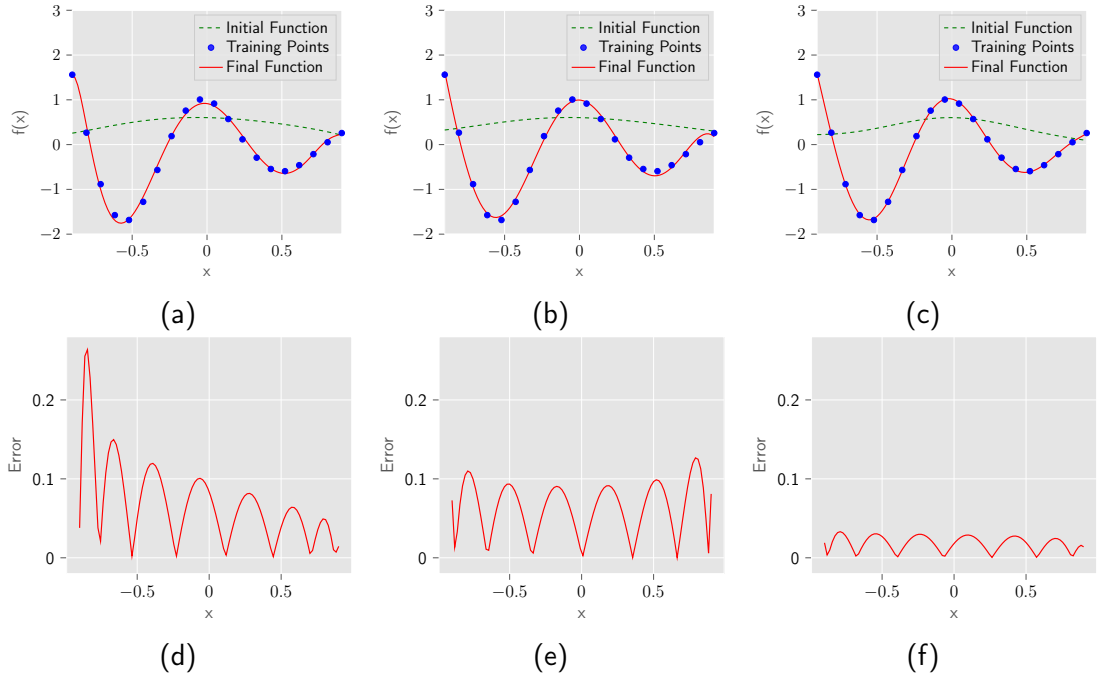


Figure 16: Function approximation of $f(x)$ given in (5.3) on an exact simulator using $R_Y(\arcsin(x))$ rotations (a), $R_Y(x)$ rotations (b) and $R_Y(x)$ and $R_X(\arcsin(x))$ rotations (c) with a qubit number $N = 4$ and a depth $D = 4$. The cost function is evaluated on 20 equidistant training points and is classically minimized using SLSQP. Additionally, in (d)-(f) the absolute values of the respective errors are plotted.

For the previously described combination of $R_Y(x)$ and $R_X(\arcsin(x))$ rotations, the

lowest error is observed. This is because both rotations play an important role here. The trigonometric part of the ansatz function can approximate the cosine well and the polynomial part of the ansatz function is better suited to model the exponential decay.

5.2 Multi Qubit Measurements

Until now, we have only used the Z expectation value of the first qubit to approximate one function. However, we found that it is possible to use the expectation values of different qubits to train different functions on the same circuit. Until now, this area was unexplored, lacking any existing literature addressing this possibility. Simulations are shown in Figure 17 for different functions on a circuit with a qubit number of $N = 3$, a depth of $D = 3$ and 30 equidistant training points.

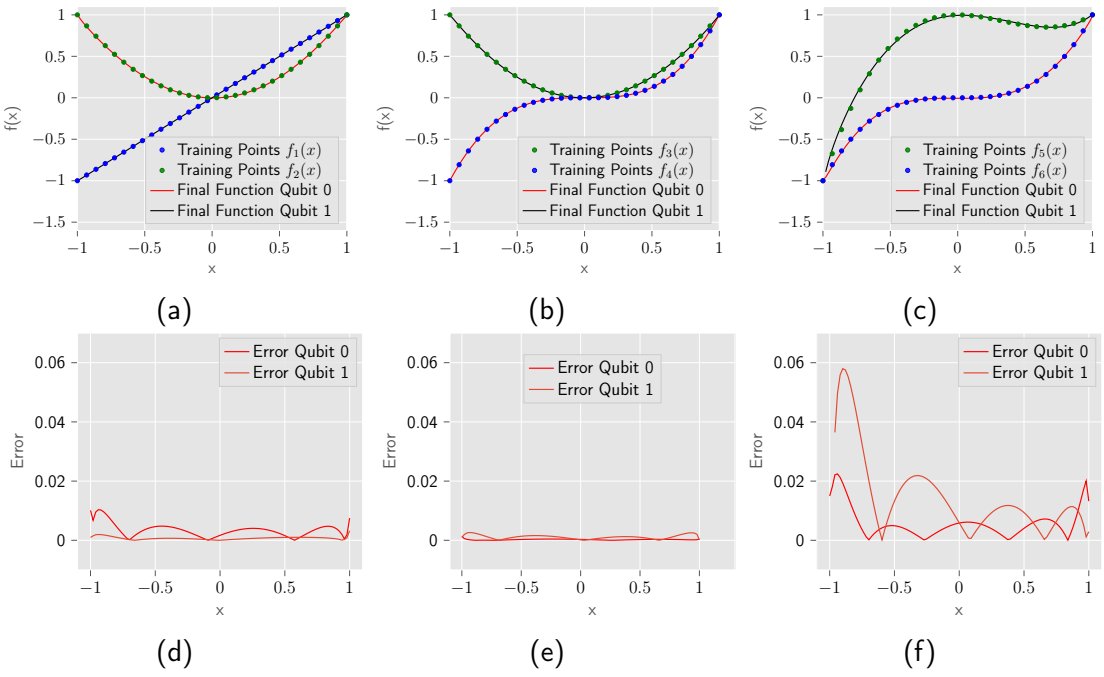


Figure 17: Function approximation of two different functions on an exact simulator using $R_Y(\arcsin x)$ data encoding with a qubit number $N = 3$ and a depth $D = 3$ with $f_1(x) = x$ and $f_2(x) = x^2$ (a), $f_3(x) = x^2$ and $f_4(x) = x^3$ (b) and $f_5(x) = x^3 - x^2 + 1$ and $f_6(x) = x^3$ (c) with a classical parameter σ_0. The cost function is evaluated on 30 equidistant training points and is minimized using SLSQP. Additionally, in (d)-(f) the absolute values of the respective errors are plotted.

Important conclusions can be drawn from these results. They indicate that we have sufficient variational parameters to approximate multiple functions simultaneously. However, it should be noted that in variational algorithms, an unnecessarily large

number of variational parameters is usually undesirable, as the optimization with a larger number of variational parameters becomes more computationally expensive. In this particular quantum algorithm, however, reducing the number of variational parameters turns out to be difficult, since these parameters are necessary to approximate complicated functions. Nevertheless, the ability to approximate multiple polynomials opens up potentially important applications. This effect will be revisited in Chapter 9.4 to successfully solve coupled differential equations with a single circuit.

6 Expressibility

To understand what expressibility means in the context of function approximations, we look again at the ansatz function

$$\xi_\alpha(x) = \alpha_0 \cdot x + \alpha_1 \cdot \sqrt{1-x^2} + \alpha_2 \cdot x^2 + \alpha_3 \cdot x \cdot \sqrt{1-x^2} + \alpha_4 \tag{6.1}$$

introduced in Chapter 4.2 for $R_Y(\arcsin(x))$ data encoding. This function is constrained by

$$\xi_\alpha(x) \in [-1, 1]. \tag{6.2}$$

The expressibility, as we define it, describes which part of the functional space of $\xi_\alpha(x)$ can be reached with the first qubit's Z expectation value $y(x)$ of a certain parameterized circuit. As mentioned before, this depends strongly on the depth D, but additional factors also play an important role. We will now implement a method that can be used to estimate the expressibility of small circuits.
Randomly chosen parameters $\vec{\theta}$ result in a random function of the form of the ansatz function (6.1). Those randomly generated functions can be subsequently fitted with this ansatz function using the least square method which is explained in Appendix B. With this approach, it is possible to compare different circuits in their ability to approximate as many different functions as possible.
To visualize the expressibility, the correlations of the fit parameters α_0-α_4 are plotted. The parameter correlations for the example of $R_Y(\arcsin(x))$ data encoding can be seen for $N = 2$ and different depths D in Figure 18. To make the graph more readable, the density of 1000 random parameter correlations is plotted. It is expected that the parameter correlations are limited because of the condition in (6.2) alone. However, the plots clearly show that many parameter correlations are limited beyond that. Thus, only a small part of the possible functions in Equation (6.1) can be obtained with a measurement. This improves, however, when the depth is increased. A depth of about $D = 4$ seems to be ideal since a further increase only slightly enhances the expressibility. This also shows why it is so important to add another classical parameter, as shown in Equation (5.1). This parameter gives another degree of freedom, allowing all combinations for a depth greater than $D = 2$. However, certain combinations are still difficult to obtain and require many optimization steps. To understand why it can be very difficult to achieve certain functions with these circuits,

one must look at the influence of the variational parameters. Consider the example in Figure 10, where we have already introduced the ansatz function in (4.3). Here, the observed behavior stems from the fact that each of these function parameters α_0 to α_4, depends on a large number of variational parameters $\vec{\theta}$ in the circuit. To change a single one of the function parameters α_0 to α_4, many variational parameters $\vec{\theta}$ must be changed, which in turn changes all other function parameters. It is therefore this complicated interplay of parameters, that makes the optimization so computationally expensive.

This behavior also explains the nature of the error plots. Due to the previously described behavior of the parameters, it is very unlikely to end up with only the desired function even for a simple function like $f(x) = x^3$, as seen in Figure 12 (d). There will always remain small proportions of different functions that lead to the unique structures of the error plots. It is important to take into consideration that the execution of this algorithm on a real quantum computer inevitably introduces shot noise, effectively overshadowing these minor errors. Hence, it is crucial to not overemphasize the significance of these errors.

Next, we look at another data encoding scheme. For the circuit in Figure 13, $R_Y(x)$ is used for data encoding. For this example, as introduced in Chapter 5.1, the ansatz function can be expressed as

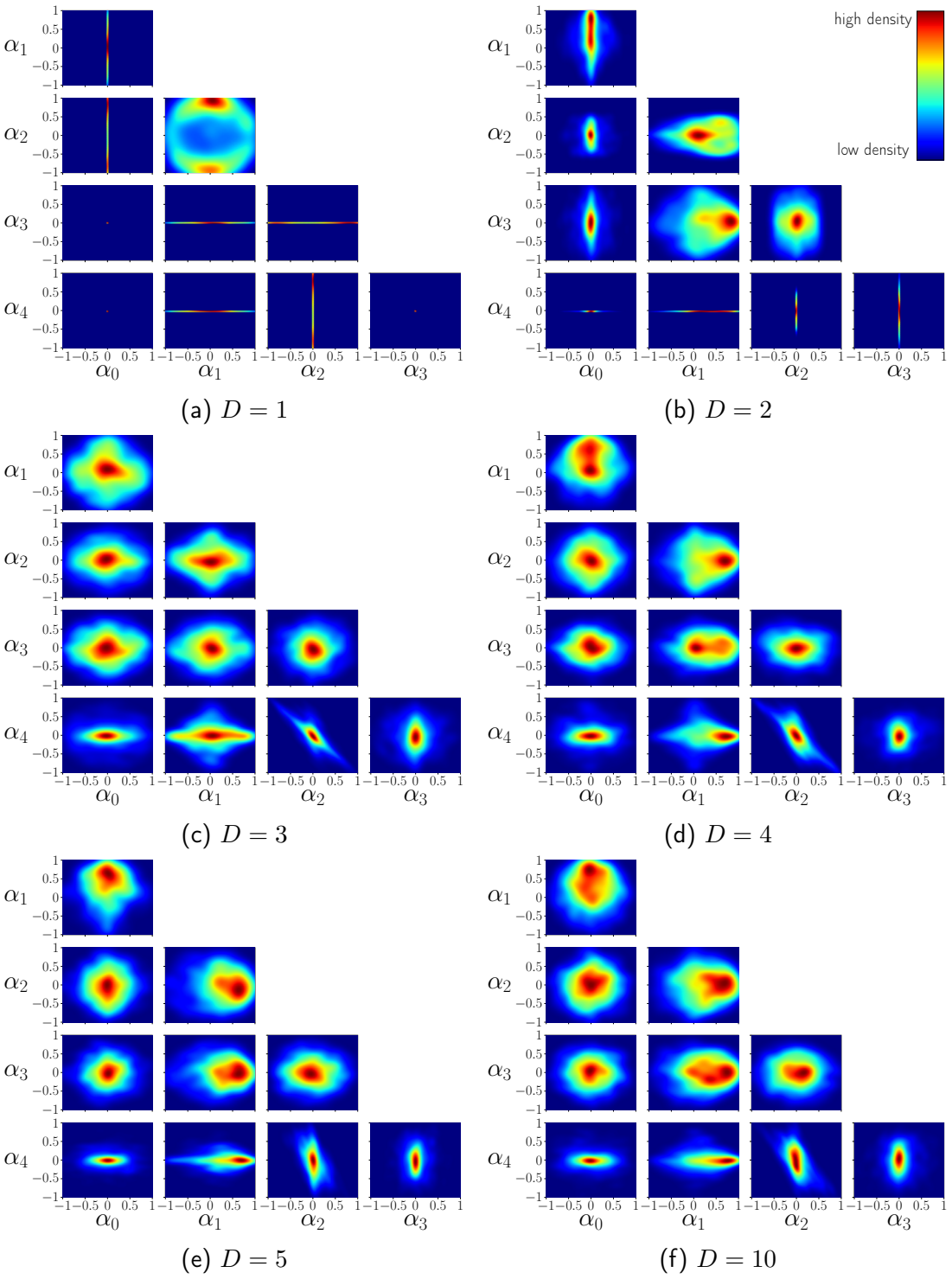
$$\xi_\beta(x) = \beta_0 \cdot \sin(x) + \beta_1 \cdot \cos(x) + \beta_2 \cdot \sin(2x) + \beta_3 \cdot \cos(2x) + \beta_4\,, \tag{6.3}$$

with

$$\xi_\alpha(x) \in [-1, 1].$$

Here, the same procedure as described in the previous example is applied and the parameter correlations for a qubit number $N = 2$ and a depth of $D = 4$ are shown in Figure 19. Analogous to before, the plots show that many parameter correlations are limited. This means that in the case of $R_Y(x)$ data encoding, it is also important to add a classical parameter σ_0. Looking at this figure and Figure 18, a pattern emerges. It seems that higher frequencies or higher-order polynomials are harder to achieve. This behavior ensures that this algorithm does not tend to overfit since overfitting is mostly due to unwanted higher orders/frequencies, which are suppressed here. Whether this effect leads to an advantage in more complex problems cannot be answered in general.

In this work, we only considered repeating parameters in the form of three different parameterized rotations. In Appendix A alternative methods are discussed.



(a) $D = 1$ (b) $D = 2$

(c) $D = 3$ (d) $D = 4$

(e) $D = 5$ (f) $D = 10$

Figure 18: Parameter correlations of the circuit in Figure 10 with a qubit number $N = 2$ and different depths D after fitting the Z expectation value $y(x)$ with the function given in (6.1) for 1000 random parameters $\vec{\theta}$ using the least square method. For readability, one-dimensional lines are slightly broadened to be visible in the plots.

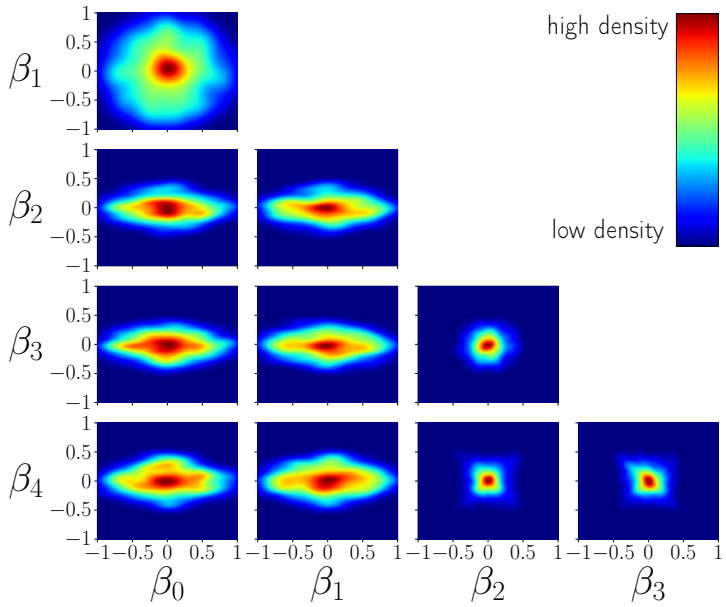


Figure 19: Parameter correlations of the circuit in Figure 13 with a qubit number $N = 2$ and a depth of $D = 4$ after fitting the Z expectation value $y(x)$ with the function given in (6.3) for 1000 random parameters $\vec{\theta}$ using the least square method.

7 Function Approximation: Real Quantum Computer

The function approximation is now also performed on the quantum computer "IBM Quantum System One Ehningen" which was introduced in Chapter 3.2. In the literature so far, the algorithm was executed with an exact simulator and the circuit with the final parameters was run on a real quantum computer [31]. In this work, for the first time, not one circuit with the final parameters but the full algorithm is executed on a real quantum computer. This means that every circuit evaluation is run on a real quantum computer. The previously utilized optimizer SLSQP is deliberately not used here, since the calculation of the gradient, inherent to the SLSQP optimizer, becomes too inaccurate with shot noise. For this application, Constrained Optimization BY Linear Approximation (COBYLA) [32] proves to be the most suitable. The results for the same functions used in Figure 12 are shown in Figure 20 on the IBM Quantum System One Ehningen. It is evident that the approximation can be successfully performed on a real quantum computer. In this example, the method succeeds in approximating the two polynomials very precisely. Hence, it is not necessary to let the optimization process continue since the error prevailing at the end is primarily due to the shot noise. In the case of the sine function, the approximation obtained is subject to greater inaccuracies since this function is not easily represented by polynomials. Nevertheless, it is possible to determine the qualitative behavior of the sine function on the quantum computer. This result can theoretically be improved by a higher shot number, which, however, prolongs the optimization. The algorithm already takes a long time with the chosen shot number of 2000. Although the number of function evaluations is very small, the execution of the examples shown here takes several hours (e.g. $\approx 7\,\mathrm{h}$ for $f_1(x)$). A large part of this is due to the classical optimization, but the execution on the quantum computer also takes a long time. The results shown here are still very promising since it has been demonstrated for the first time that this algorithm can be executed on today's quantum computers and is thus highly relevant for the NISQ era.

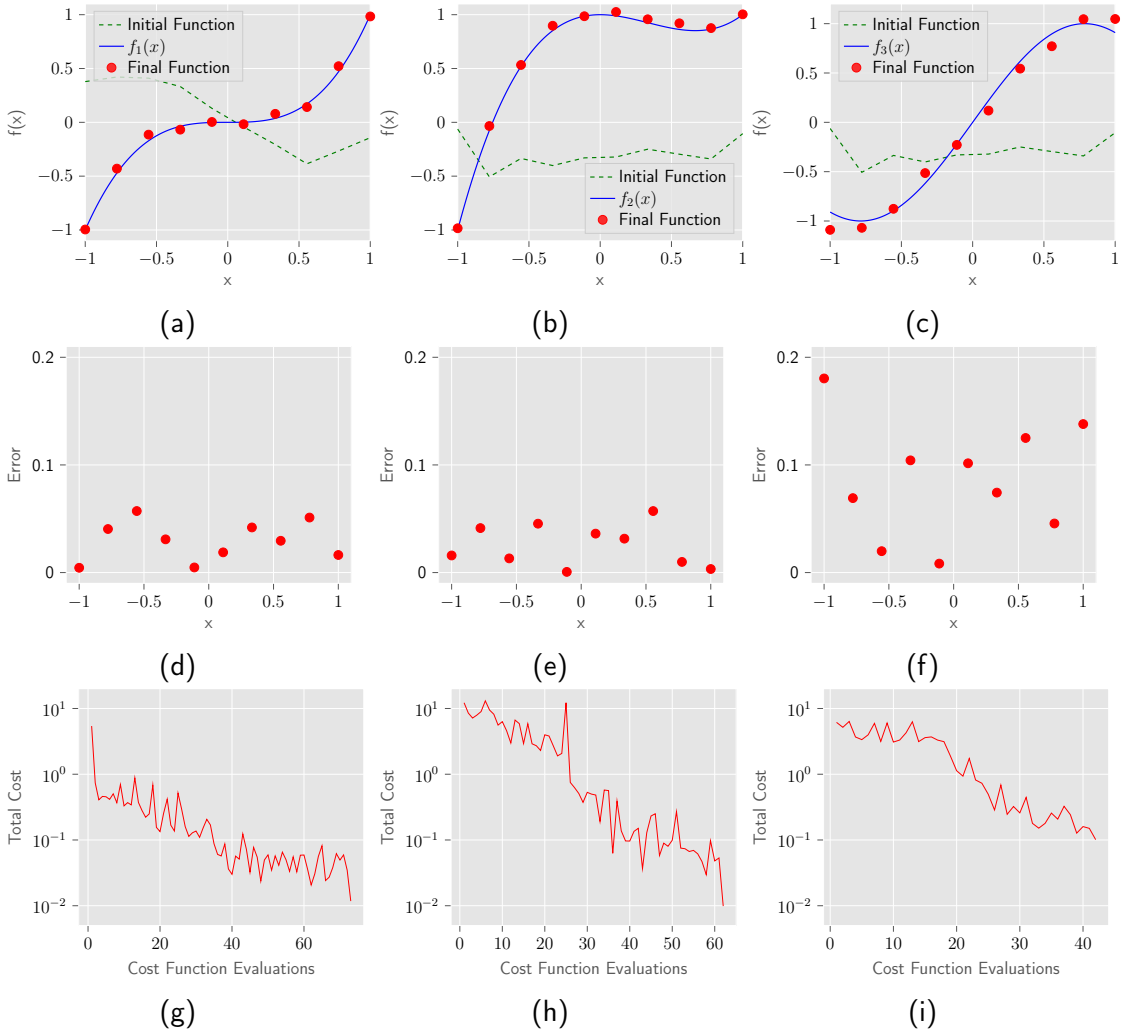


Figure 20: Function approximation on the IBM Quantum System One Ehningen of $f_1(x) = x^3$ (a), $f_2(x) = x^3 - x^2 + 1$ (b) and $f_3(x) = \sin(2x)$ (c) with $R_Y(\arcsin(x))$ rotations for data encoding, a qubit number of $N = 3$, 10 equidistant training points and 2000 shots. The parameters are classically optimized with COBYLA. The depth is chosen to be $D = 2$ for $f_1(x)$ and $D = 3$ for $f_2(x)$ and $f_3(x)$. Additionally, in (d)-(f) the absolute values of the respective errors are plotted and in (g)-(i) the respective values of the cost function versus the number of cost function evaluations are shown.

8 Parameter Shift Rule

As derived in Chapter 2.6 the parameter shift rule is described by

$$y'(x) = \frac{1}{2} \sum_j \left(y_j \left(\varphi(x) + \frac{\pi}{2} \right) - y_j \left(\varphi(x) - \frac{\pi}{2} \right) \right) \cdot \varphi'(x)$$

and can be used to calculate the derivative of the expectation value $y(x)$ of any parameterized circuit, where $\varphi(x)$ is the inner function that encodes the x-values in the circuit and j indicates the different gates with the parameter x. A simple three-qubit circuit is depicted in Figure 21.

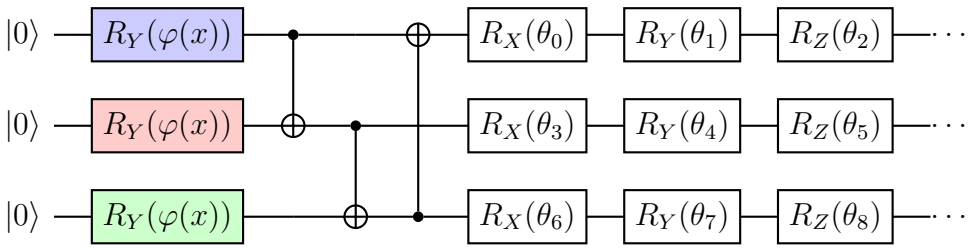


Figure 21: Three-qubit circuit with $R_Y(\varphi(x))$-encoding followed by three controlled NOT gates to achieve circular entanglement and $\vec{\theta}$-parameterized x-, y- and z-rotations.

The parameter shift rule can also be used to determine the derivative of the Z expectation value $y(x)$ of the first qubit, which is the basis for the algorithm in this work. To determine the derivative of this circuit, six additional circuit evaluations are needed. The full derivative can be expressed as

$$\begin{aligned} \frac{\partial y(x)}{\partial x} = \frac{1}{2} (& y_{q_0}(\varphi(x) + \tfrac{\pi}{2}) - y_{q_0}(\varphi(x) - \tfrac{\pi}{2}) \\ & + y_{q_1}(\varphi(x) + \tfrac{\pi}{2}) - y_{q_1}(\varphi(x) - \tfrac{\pi}{2}) \\ & + y_{q_2}(\varphi(x) + \tfrac{\pi}{2}) - y_{q_2}(\varphi(x) - \tfrac{\pi}{2})) \cdot \varphi'(x) \,, \end{aligned}$$

where $y_{qi}(\varphi(x) \pm \frac{\pi}{2})$ represents the Z expectation value $y(x)$ with the input of the data encoding gate of the ith qubit being shifted by $\pm\frac{\pi}{2}$. The data encoding layers of those six additional circuit evaluations are shown in Figure 22.

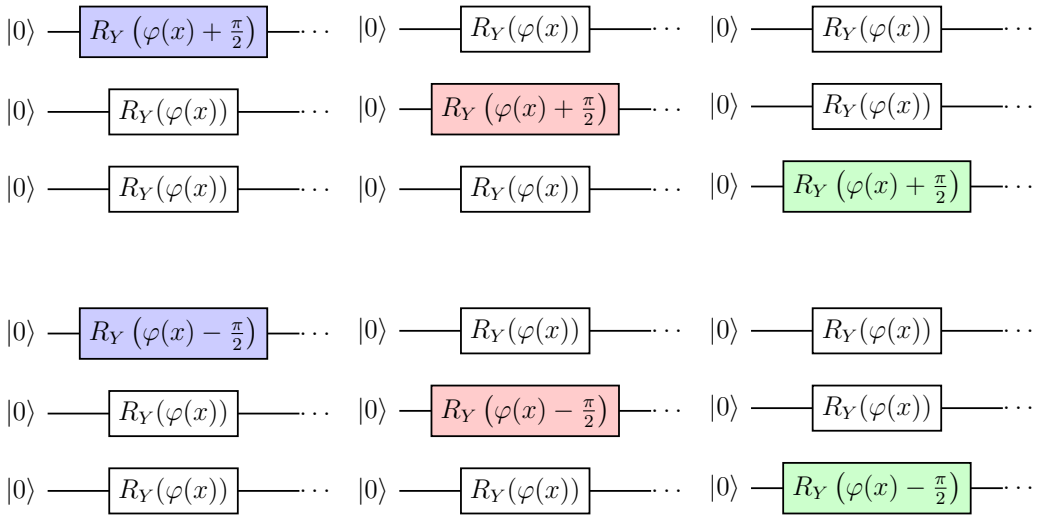


Figure 22: The data encoding layers of the six additional circuit evaluations required for the parameter shift rule of the example in Figure 21.

In general, the calculation of the first derivative requires $2M$ circuit evaluations, where M is the number of gates with the variable x. The second derivative can be determined by a further parameter shift on all gates. Therefore, the second derivative requires $4M^2 - 2M$ circuit evaluations. This means that for the example in Figure 9, the second derivative would require $4M^2 - 2M = 4N^2 - 2N = 30$ additional circuit evaluations. One powerful aspect of employing the parameter shift rule for these derivatives is that it yields the exact derivative and not an approximation like many classical methods do. An example with $\varphi(x) = \arcsin(x)$ on an exact simulator can be seen in Figure 23. Here the optimized parameters from the example shown in Figure 12 (a) are employed. Since the derivative of the inner function $\varphi(x) = \arcsin(x)$ is needed, which diverges for $x = -1$ and $x = 1$, a value range of $x \in$ [-0.9,0.9] is chosen in this example.

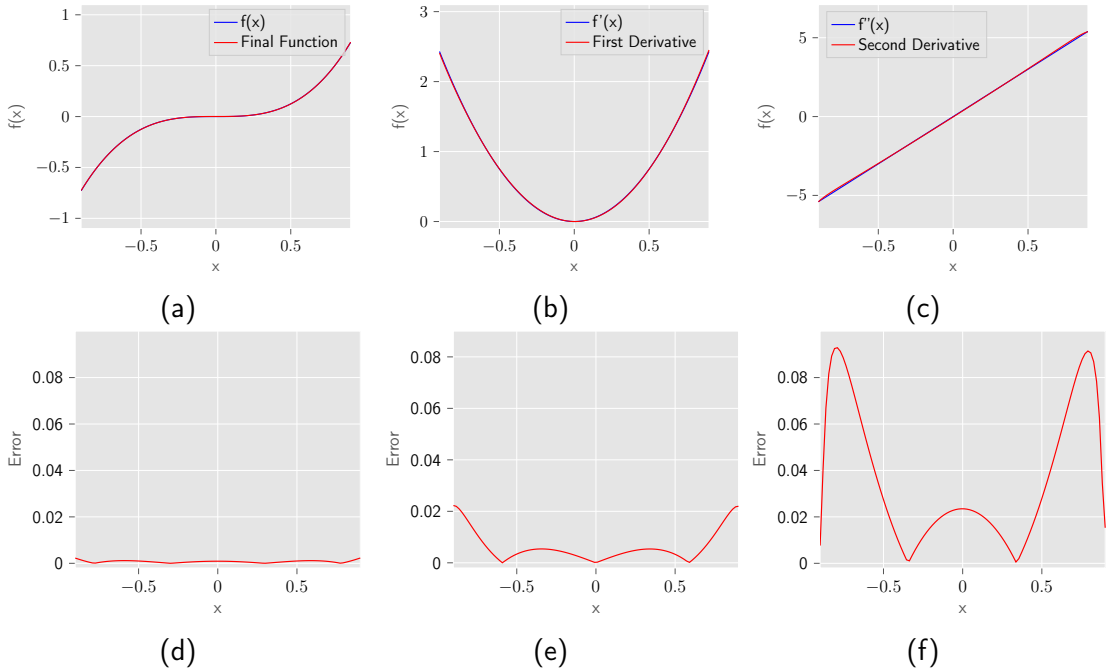


Figure 23: Derivatives obtained with the parameter shift rule on an exact simulator using the first example from Figure 12 (a) with the function $f(x) = x^3$ (a), the first derivative $f'(x) = 3x^2$ (b) and the second derivative $f''(x) = 6x$ (c). Additionally, in (d)-(f) the absolute values of the respective errors are plotted.

With this procedure, arbitrary derivatives can be calculated. The figure shows increased errors for the derivatives. However, this is not due to an error caused by the differentiation, since the parameter shift rule yields the exact derivative, but due to inaccuracies of the original function approximation in the example from Figure 12 (a). However, the increased circuit evaluations on a real quantum computer lead to increased errors. Figure 24 shows the results of the IBM quantum computer in Ehningen. Again, the parameters obtained with an exact simulation in the example from Figure 12 (a) are used and the circuit is evaluated on 10 points. It can be seen that although the qualitative behavior of the derivatives can be determined, there are very significant differences, especially in the areas close to $x = -1$ and $x = 1$. When we go into the realm of differential equations, these derivatives become very important. Solving these differential equations on a real quantum computer therefore involves large errors. Hence, in the following chapters, we will focus more on exact simulations.

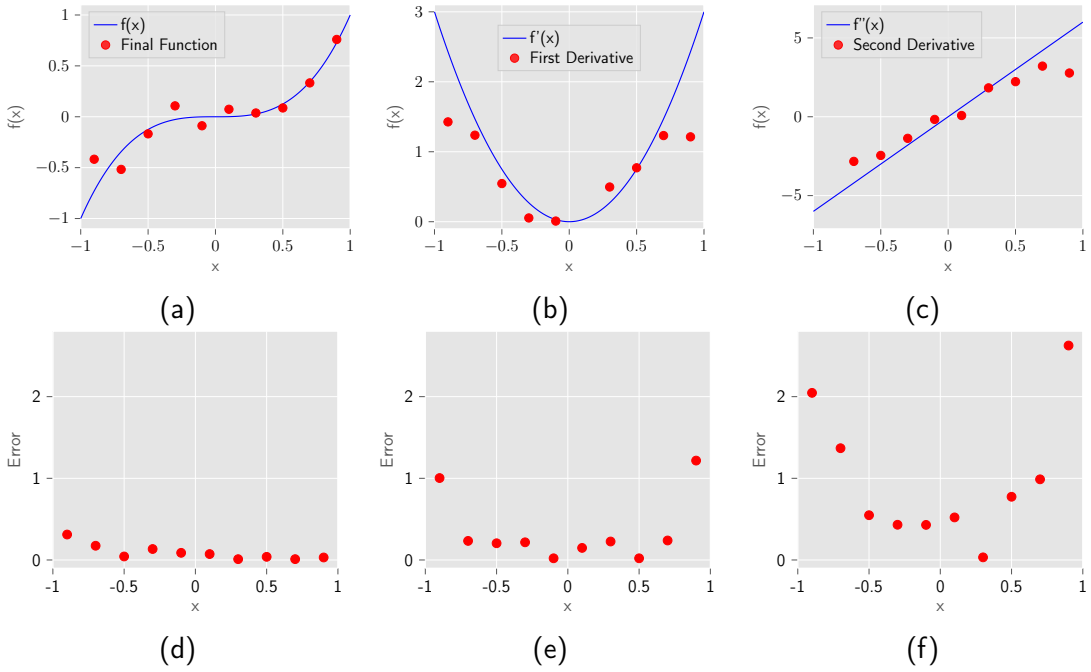


Figure 24: Derivatives obtained with the parameter shift rule on the real IBM quantum computer in Ehningen with 1024 shots using the first example from Figure 12 with the function $f(x) = x^3$ (a), the first derivative $f'(x) = 3x^2$ (b) and the second derivative $f''(x) = 6x$ (c). Additionally, in (d)-(f) the absolute values of the respective errors are plotted.

9 Differential Equations: Simulations

The previously explained parameter shift rule allows us to determine arbitrary derivatives of the expectation value $y(x)$. Instead of using it to obtain the derivative of an already trained function, the derivative terms can also be integrated directly into the cost function. This capacity to include derivatives into the cost function opens up the potential for solving differential equations. These circuits were coined as differentiable quantum circuits (DQCs) [21]. In this chapter, we delve deeper into this methodology, improving it through insights gathered in the earlier chapters. Furthermore, we conduct a comprehensive comparison of this approach against different quantum algorithms for solving differential equations, providing a thorough discussion of its strengths and weaknesses. First, however, we focus on a simple example to understand how DQCs work.

9.1 Logistic Differential Equation

A simple example of a differential equation is the so-called logistic differential equation

$$\begin{cases} u'(x) = k \cdot u(x)(G - u(x)) \,, \\ u(0) = u_0 \,, \end{cases} \tag{9.1}$$

where k is the growth constant and G is the limit for the function value. The solution is the logistic function

$$u(x) = G \cdot \frac{1}{1 + e^{-kGx}(\frac{G}{u_0} - 1)} \,.$$

This function describes a representation of saturation processes and has a multitude of applications in many fields of science. To be able to solve the logistic differential equation with DQCs the cost function

$$L = \sum_i \left(|u'(x_i) - ku(x_i)(G - u(x_i))|^2 + \mu \cdot |u(0) - u_0|^2 \right)$$

is defined, where i sums over the training points μ is a problem specific weight factor and $u(x_i) = y(x_i) \cdot \sigma_0$. As described in Chapter 4.2, $y(x_i)$ is the z expectation value

of the first qubit.

The weight factor μ penalizes any deviation from the boundary condition in the cost function L. The advantage here is that any number of arbitrary boundary conditions can be included. A disadvantage is that the optimal weight factor μ is problem-specific and must be found by trial and error for every new problem. Another possibility is the so-called floating boundary handling [21]. Here the boundary condition is not included as a penalty term in the cost function, but the whole function is shifted along the y-axis after the measurement so that the boundary condition is fulfilled. This has the great advantage that the boundary condition is always exactly matched. Additionally, no suitable weight factor must be determined. However, with this method, it is only possible to consider one boundary condition of the form $u(x_0) = u_0$. Multiple boundary conditions or boundary conditions that apply to a derivative are not implementable. Therefore, the method described first is applied in this work.

To solve the logistic differential equation a circuit with $N = 4$ and $D = 4$ is used. Here, a combination of $R_Y(x)$- and $R_X(\arcsin(x))$-encoding is used, as introduced in Figure 15. This circuit can be seen in Figure 25.

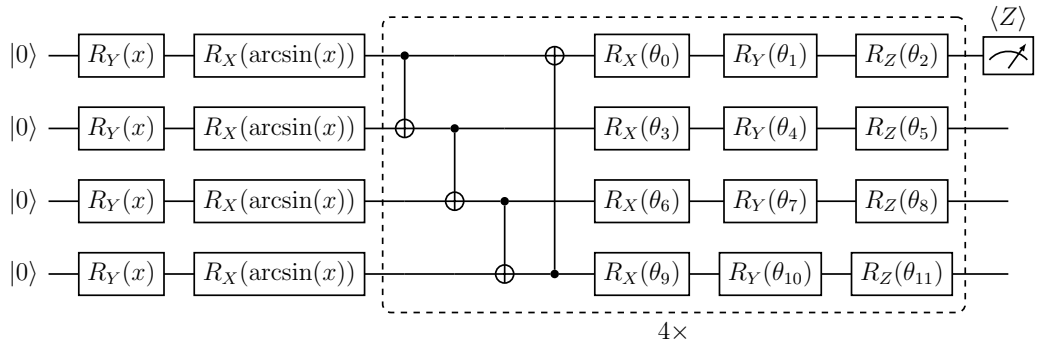
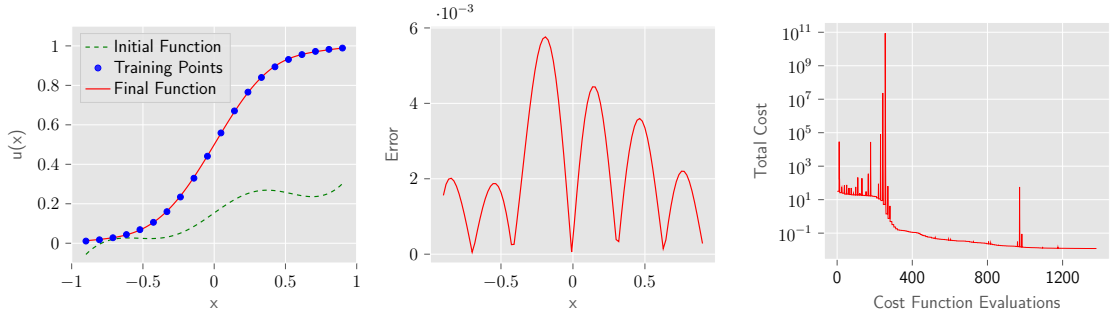


Figure 25: Four-qubit DQC with $R_Y(x)$- and $R_X(\arcsin(x))$-encoding and a measurement of the Z expectation value of the first qubit.

Since the derivative is obtained with the parameter shift rule, the circuit must be executed several times with shifts in the data encoding, as explained in Chapter 8. We use this circuit to solve the logistic differential equation for $G = 1$, $k = 5$ and the boundary condition $u_0 = \frac{1}{2}$. The result for a weight factor $\mu = 10$ and 20 equidistant training points is shown in Figure 26. The x-value range is chosen to be $x \in [-0.9, 0.9]$ because the derivative of $\varphi(x) = \arcsin(x)$ is not differentiable at $x = -1$ and $x = 1$.



(a) (b) (c)

Figure 26: (a) Solution of the differential Equation (9.1) with $\mu = 10$, $N = 4$, $D = 4$ and 20 equidistant training points using DQCs with $R_Y(x)$ and $R_X(\arcsin(x))$ rotations for data encoding. (b) The absolute values of the error. (c) Values of the cost function versus the number of cost function evaluations.

The final result matches the solution very well with a very small error. However, a lot of function evaluations are necessary to arrive at this solution. It also needs to be considered that because of the parameter shift rule, each function evaluation is translated into several circuit evaluations for each training point. This means in this example, where the first derivative has to be determined and where the variable x appears in 6 different gates, that for each function evaluation and for each training point 17 circuit evaluations must be performed. This does not include the shot number.

9.2 Harmonic Oscillator

The harmonic oscillator is a central concept in physics to describe many natural phenomena. It serves as a fundamental model for understanding oscillating motion and periodic behavior in a wide variety of physical systems, from a swing pendulum to the propagation of light. Therefore, the mathematical description of harmonic oscillation is ubiquitous in science and engineering, making it an interesting example problem. The differential equation is described by

$$\begin{cases} u''(t) + \omega_0^2 u(t) = 0\,, \\ u(t=0) = u_0\,, \\ u'(t=0) = u_0'\,, \end{cases} \tag{9.2}$$

where ω_0 is the circular frequency.

The parameter shift rule can also calculate the second derivative as explained in Chapter 8. This makes it possible to solve the differential equation in (9.2). Since

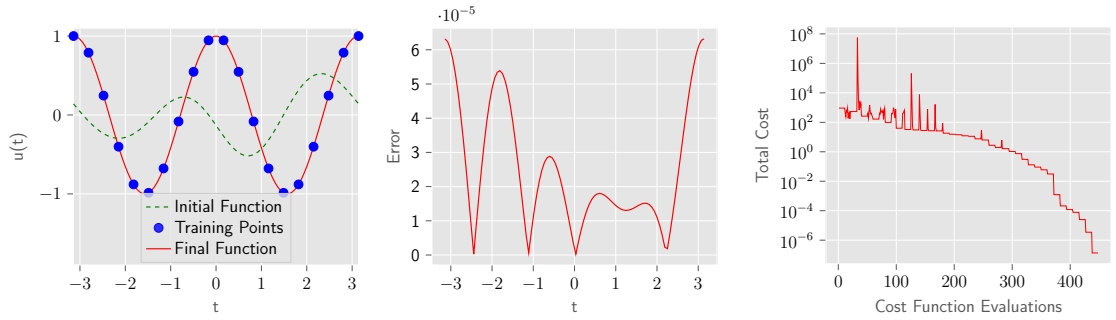
the harmonic oscillator usually describes an evolution in time, the variable x is named t in this section. From this differential equation, the cost function

$$L = \sum_i \left(\left| u''(t_i) + \omega_0^2 u(t_i) \right|^2 + \mu \left(|u(0) - u_0|^2 + |u'(0) - u_0'|^2 \right) \right)$$

can be formed. For this example, we choose the initial conditions $u_0 = 1$ and $u_0' = 0$. This results in the solution

$$u(t) = \cos(\omega_0 t) \, .$$

The differential equation for $\omega_0 = 2$ is now solved with DQCs. The result obtained with a circuit with $N = 3$, $D = 3$, a weight factor $\mu = 20$ and 20 equidistant training points is shown in Figure 27. A $R_Y(t)$-encoding is used, because it is better suited for trigonometric functions. In this example, we consider the time interval $t \in [-\pi, \pi]$.



(a) (b) (c)

Figure 27: (a) Solution of the differential Equation (9.2) with $\mu = 20$, $N = 3$ and $D = 3$ using DQCs with $R_Y(t)$ rotations for data encoding. (b) The absolute values of the error. (c) Values of the cost function versus the number of cost function evaluations.

As expected, the solution which consists of only one cosine term, can be matched very well. Since the solution of the differential equation, just like the rotational gate $R_Y(t)$, is 2π-periodic, the circuit gives accurate results for $t \in \mathbb{R}$. Analogous to before, a lot of cost function evaluations are needed. Since the cost function involves the second derivative, each cost function evaluation requires many circuit evaluations. In fact, 36 evaluations are needed for every training point.

Comparison to non-variational algorithm

The differential equation for a harmonic oscillator is a second-order differential equation. One way to solve a harmonic oscillator with a non-variational algorithm is to transform the higher-order differential equation into a first-order system, which can then be solved on a quantum computer. As we will discuss, this is only possible for

this very simple example and is not transferable to general differential equations. To achieve this, we formulate the system in the form

$$\begin{cases} \vec{v}' = A\vec{v}\,, \\ \vec{v}(0) = \vec{v}_0\,. \end{cases} \tag{9.3}$$

Once we have expressed the system in this form, the solution is given by

$$\vec{v}(t) = e^{tA}\vec{v}_0\,. \tag{9.4}$$

Now, we transform the differential equation in (9.2) into a first-order system by defining

$$\vec{v} = \begin{pmatrix} u \\ u' \end{pmatrix} \quad \text{and} \quad A = \begin{pmatrix} 0 & 1 \\ -\omega_0^2 & 0 \end{pmatrix}.$$

However, the matrix A is not skew-symmetric, which results in the problem that e^{tA} in (9.7) is not a unitary operator and is therefore not executable on a quantum computer. To circumvent this problem, we define the modified vector and the modified matrix

$$\vec{a} = \begin{pmatrix} \omega_0 u \\ u' \end{pmatrix} \quad \text{and} \quad S = \begin{pmatrix} 0 & \omega_0 \\ -\omega_0 & 0 \end{pmatrix} = i\omega Y.$$

For this newly defined system

$$\begin{cases} \vec{a}' = S\vec{a}\,, \\ \vec{a}(0) = \vec{a}_0 \end{cases}$$

is fulfilled and the matrix S is skew-symmetric. This expression is analogous to (9.3) and because S is skew-symmetric, e^{tS} is a unitary matrix that can be executed on a quantum computer. Finally, the time evolution can be expressed as

$$\vec{a}(t) = e^{tS}\vec{a}_0 = e^{i\omega_0 tY}\vec{a}_0 = R_Y(-2\omega_0 t)\vec{a}_0$$

with

$$\vec{a}_0 = \begin{pmatrix} \omega_0 u_0 \\ u_0' \end{pmatrix}.$$

To obtain the solution on a quantum computer, it is necessary to prepare the initial state which contains the initial conditions. The initial state which corresponds to $\vec{a}_0$ is

$$|a_0\rangle = \frac{1}{||\vec{a}_0||}(\omega_0 u_0 |0\rangle + u_0' |1\rangle)\,.$$

Following this, the $R_Y(-2\omega_0 t)$-rotation is applied to this state which results in the state

$$|a(t)\rangle = R_Y(-2\omega_0 t)\,|a_0\rangle = \frac{1}{||\vec{a}_0||}(\omega_0 u(t)\,|0\rangle + u'(t)\,|1\rangle)\,. \tag{9.5}$$

Now, we want to obtain $u(t)$ with a measurement. We see from (9.5) that

$$\frac{\omega_0 u(t)}{||\vec{a}_0||} = \langle 0|a(t)\rangle = \langle 0|U(t)|0\rangle\ ,$$

where $U(t)$ is the combination of the state preparation and the $R_Y(-2\omega_0 t)$-rotation

$$U(t) = R_Y(-2\omega_0 t)\cdot U_{\text{state-prep}}\,.$$

From this follows that we can obtain $u(t)$ with

$$u(t) = \frac{||\vec{a}_0||}{\omega_0}\,\langle 0|U(t)|0\rangle\ .$$

For this purpose, the Hadamard test is implemented, which is explained in Appendix C. At its core, the Hadamard test can be used to obtain the expectation value $\langle\psi|U|\psi\rangle$. Here, $|\psi\rangle = |0\rangle$ and $U = U(t)$. With the values $\omega_0 = 2$, $x(0) = 1$ and $\dot{x}(0) = 0$ from before, the state to prepare is $|0\rangle$, which makes the addition of a state preparation obsolete. The final circuit of $U(t)$ can be seen in Figure 28.

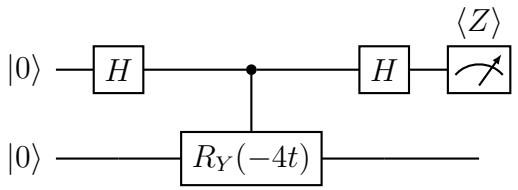


Figure 28: Circuit to simulate a harmonic oscillator with $\omega_0 = 2$, $x(0) = 1$ and $\dot{x}(0) = 0$.

As expected, using an exact simulator results in the exact solution of the harmonic oscillator without any errors, as shown in Figure 29.

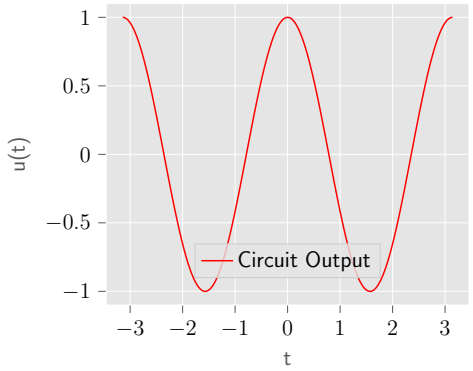


Figure 29: First qubit's Z expectation value of the circuit in Figure 28 for $t \in [-\pi, \pi]$.

It is clear, that in this simple case, the non-variational method is superior. The solution can be determined exactly with a much simpler circuit and without the need to optimize it classically. However, it should be noted that for general initial values a state preparation is necessary for the non-variational algorithm, which can be very costly and which is not necessary for DQCs. Besides that, controlled U gates have to be implemented, which cannot be executed directly on a quantum computer and have to be translated into controlled NOT gates and single-qubit rotations. For this simple example, this circuit can be seen in Figure 30.

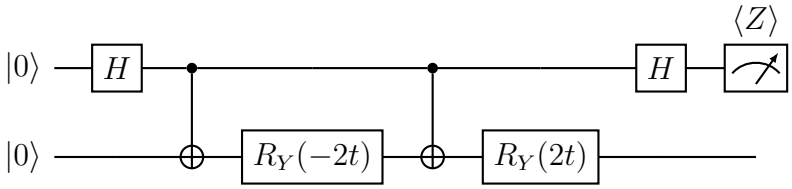


Figure 30: Circuit in Figure 28 without the controlled rotational gate.

Overall, the non-variational circuit is the method of choice for this simple example. In the further course of this chapter, we will investigate more complex differential equations where the advantages of DQCs come to light.

9.3 Damped Harmonic Oscillator

A damped harmonic oscillator is an important concept in physics and engineering, that describes the behavior of a system that performs an oscillation while losing energy to its surroundings. This phenomenon occurs in a variety of natural and engineered systems, such as a swinging pendulum that gradually loses energy due to friction.

The damped harmonic oscillator can be described by a differential equation of the form

$$\begin{cases} u''(t) + 2\gamma u'(t) + \omega_0^2 u(t) = 0\,, \\ u(0) = u_0\,, \\ u'(0) = u_0'\,, \end{cases} \tag{9.6}$$

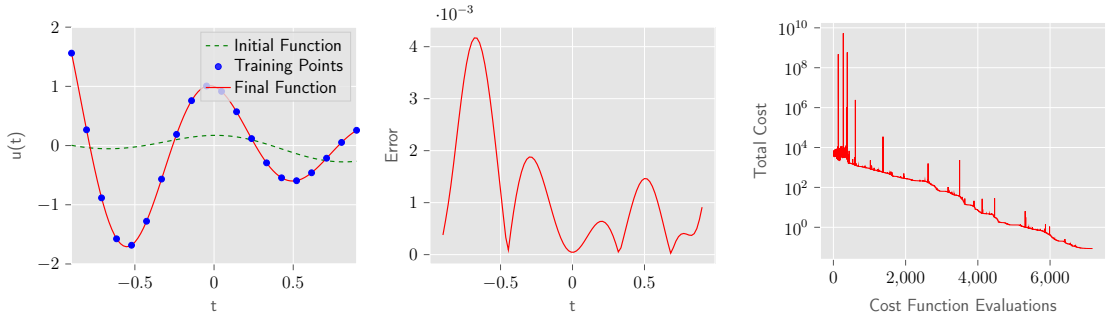
where ω_0 is the undamped angular frequency of the oscillator and γ is the damping coefficient. From this differential equation, we can define the cost function

$$L = \sum_i \Big(\left| u''(t_i) + 2\gamma u'(t_i) + \omega_0^2 u(t_i) \right|^2 + \mu \left| u(0) - u_0 \right|^2 + \left| u'(0) - u_0' \right|^2 \Big)\,.$$

For the case $\gamma = 1, \omega_0^2 = 37$ with $u_0 = 1$ and $u_0' = -1$ the solution of the differential equation is given by

$$u(t) = \cos(6t) e^{-t}\,.$$

We insert the selected initial conditions into the cost function and solve the system with DQCs. The result obtained with a circuit with $N = 4$, $D = 4$, a weight factor of $\mu = 100$ and 20 equidistant training points is shown in Figure 31. Here, a combination of $R_Y(t)$- and $R_X(\arcsin(t))$-encoding is used because it is well suited for this function, as shown in Figure 16. The time interval is chosen to be $t \in [-1, 1]$.



(a) (b) (c)

Figure 31: (a) Solution of the differential Equation (9.6) with $\mu = 100$, $N = 4$ and $D = 4$ using DQCs with $R_Y(t)$ and $R_X(\arcsin(t))$ data encoding. (b) The absolute values of the error. (c) Values of the cost function versus the number of cost function evaluations.

We get a result with very high accuracy. However, many function evaluations are necessary as seen in Figure 31 (c). In the following, we will discuss why other, non-variational methods are also associated with great difficulties.

Comparison to non-variational algorithm

We formulate the system (9.6) in the form of

$$\begin{cases} \vec{v}' = A\vec{v}\,, \\ \vec{v}(0) = \vec{v}_0\,, \end{cases}$$

for which the solution is given by

$$\vec{v}(t) = e^{tA}\vec{v}_0\,. \tag{9.7}$$

For this we define

$$\vec{v} = \begin{pmatrix} u \\ u' \end{pmatrix} \quad \text{and} \quad A = \begin{pmatrix} 0 & 1 \\ -\omega_0^2 & -2\gamma \end{pmatrix}.$$

However, this system cannot be solved on a quantum computer because e^{tA} is not a unitary operator. To make this possible, we define the vector

$$\vec{a} = \begin{pmatrix} \omega_0 u \\ u' \end{pmatrix}$$

and the matrix

$$S = \begin{pmatrix} 0 & \omega_0 \\ -\omega_0 & -2\gamma \end{pmatrix} = \begin{pmatrix} 0 & \omega_0 \\ -\omega_0 & 0 \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ 0 & -2\gamma \end{pmatrix}$$

where

$$\begin{cases} \vec{a}' = S\vec{a}\,, \\ \vec{a}(0) = \vec{a}_0 \end{cases}$$

is fulfilled. Now, S consists of a skew-symmetric part

$$S_1 = \begin{pmatrix} 0 & \omega_0 \\ -\omega_0 & 0 \end{pmatrix} = i\omega Y$$

and a symmetric part

$$S_2 = \begin{pmatrix} 0 & 0 \\ 0 & -2\gamma \end{pmatrix}.$$

To obtain the solution of the differential equation, we can use the approximation

$$\vec{a}(t) = e^{St}\vec{a}_0 \approx e^{S_1 t} \cdot e^{S_2 t}\vec{a}_0\,. \tag{9.8}$$

However, we have the problem that the matrix

$$e^{S_2 t} = \begin{pmatrix} 1 & 0 \\ 0 & e^{-2\gamma t} \end{pmatrix}$$

is still non-unitary. Non-unitary operations can not be directly applied on a quantum computer. One approach to implementing them on a quantum computer is, for example, utilizing probabilistic post-processing.
Before that, we have to initialize the state in

$$|a_0\rangle = \frac{1}{||\vec{a}_0||}(\omega_0 u_0 |0\rangle + u_0' |1\rangle) .$$

With the boundary conditions from above, we get the state

$$|a_0\rangle = \frac{1}{\sqrt{38}}(\sqrt{37}|0\rangle - |1\rangle).$$

This time we have to apply a state preparation. The state can be initialized with

$$|a_0\rangle = U_{\text{state-prep}} |0\rangle$$

with the unitary operation

$$U_{\text{state-prep}} = \begin{pmatrix} \sqrt{\frac{37}{38}} & \frac{1}{\sqrt{38}} \\ -\frac{1}{\sqrt{38}} & \sqrt{\frac{37}{38}} \end{pmatrix} = R_y \left(-2 \arccos \left(\sqrt{\frac{37}{38}} \right) \right) .$$

The final circuit is shown in Figure 32.

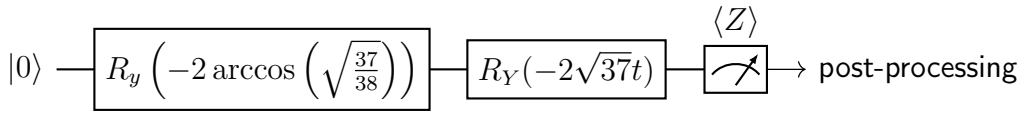


Figure 32: Circuit to simulate a damped harmonic oscillator with $\omega_0^2 = 37$, $\gamma = 1$ $x(0) = 1$ and $\dot{x}(0) = -1$.

We do not implement the Hadamard test in this example because it is not easily applicable due to the post-selection. The consequence of this is that only the squared function can be obtained by measuring the probability of state $|0\rangle$. This can be seen together with the exact solution in Figure 33.

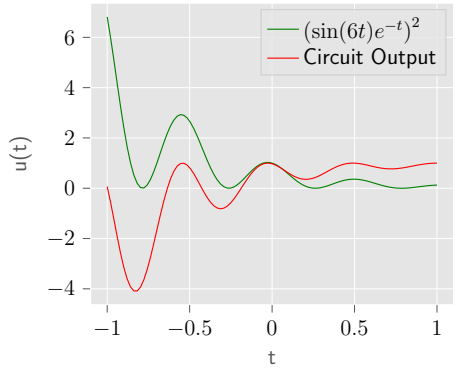


Figure 33: Damped harmonic oscillator solved with the circuit in Figure 32 (red line) together with the exact solution (green line).

Since the solution is only an approximation around the zero point, it diverges significantly in the other areas. So this approach alone is not sufficient for a useful solution. Because the approximation in (9.8) is only accurate for small t, time-stepping (Lie product formula) is usually applied to significantly increase the accuracy for different times. This process is called trotterization [33, 34]. Time stepping is implemented by repeating (9.8) multiple times

$$\vec{a}(t) \approx \left(e^{S_1 \frac{t}{r}} \cdot e^{S_2 \frac{t}{r}}\right)^r \vec{a}_0 \,,$$

where t/r defines the size of the time steps. The time steps should be as small as possible for a close approximation. However, the non-unitary operation now occurs r times in the circuit, which makes probabilistic post-processing unfeasible. Hence, for this problem, we have only marginally succeeded in solving a damped harmonic oscillator with a non-variational algorithm. However, there are no easily implementable algorithms to solve a damped harmonic oscillator.
In general, it is possible to embed non-unitary matrices in larger unitary matrices by adding ancillary qubits [35]. This is called block encoding [36]. This method has received a lot of attention in recent years, as it can solve a number of previously inaccessible problems. However, it requires very deep circuits with many ancilla qubits, which makes them unattractive in the context of the NISQ era and is therefore ruled out as a meaningful comparison and is not discussed in more detail in this thesis.

9.4 Coupled Harmonic Oscillator

The coupled harmonic oscillator is the mathematical formulation of a physical system that describes the motion of multiple oscillating and interacting systems. Here, a complex energy transfer between the systems takes place which we will solve using

DQCs.

We consider a coupled harmonic oscillator with two masses m, two springs of spring strength k and one spring of spring strength s. The arrangement can be seen in Figure 34.

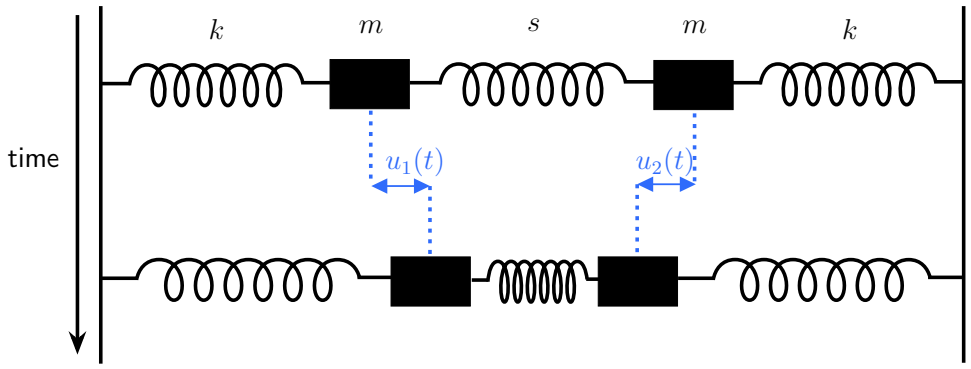


Figure 34: Structure of the coupled harmonic oscillator.

This system can be described with the two coupled differential equations

$$\begin{cases} u_1''(t) = -\frac{k}{m}u_1(t) + \frac{s}{m}(u_2(t) - u_1(t))\,, \\ u_1(0) = u_{1,0}\,, \\ u_1'(0) = u_{1,0}' \end{cases} \tag{9.9}$$

and

$$\begin{cases} u_2''(t) = -\frac{k}{m}u_2(t) - \frac{s}{m}(u_2(t) - u_1(t))\,, \\ u_2(0) = u_{2,0}\,, \\ u_2'(0) = u_{2,0}'\,. \end{cases} \tag{9.10}$$

For the following, we choose the initial conditions $u_{1,0} = 1$, $u_{1,0}' = 0$, $u_{2,0} = 0$ and $u_{2,0}' = 0$. This results in the solutions

$$\begin{aligned} u_1(t) &= \frac{1}{2}(\cos(\omega_0 t) - \cos(\omega_1 t)) \\ u_2(t) &= \frac{1}{2}(\cos(\omega_0 t) + \cos(\omega_1 t))\,, \end{aligned}$$

with the frequencies $\omega_0^2 = k/m$ and $\omega_1^2 = k/m + 2s/m$.

A coupled differential equation can be solved with DQCs using a designated circuit for each equation [21]. We want to build upon this idea and solve a coupled differential equation with only one circuit. We already saw in Figure 17 that it is possible to approximate different functions with the same circuit. We now employ this concept for coupled differential equations. For this, we utilize the circuit shown in Figure

35.

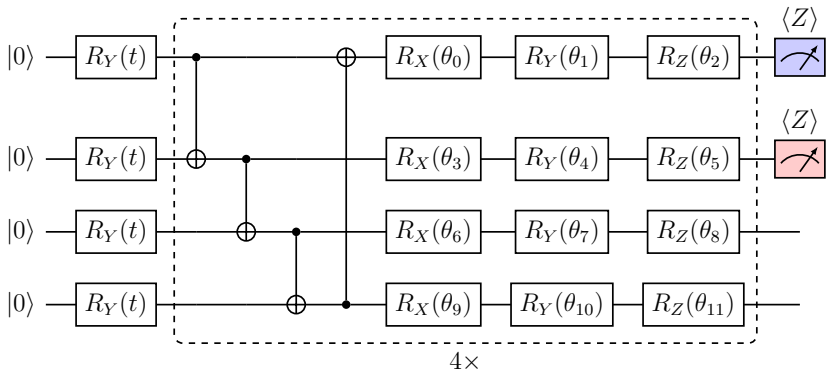
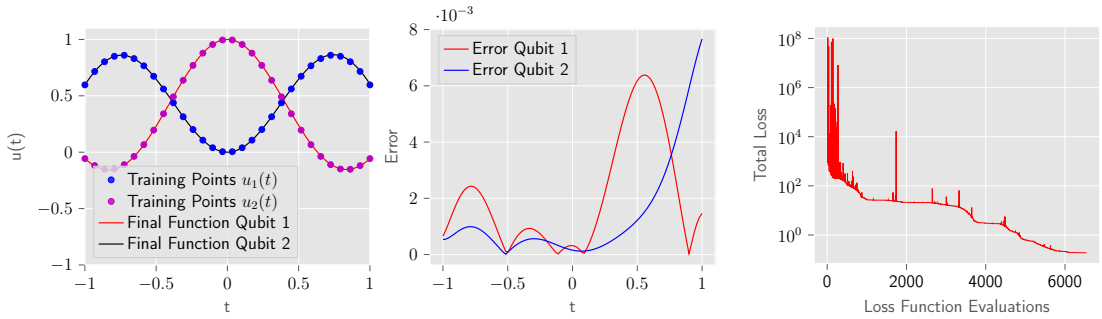


Figure 35: Four-qubit DQC with $R_Y(t)$-encoding where the first and the second qubit are measured.

In this circuit, the first and second qubit is measured. Apart from that, the process is the same as described in the previous examples. The coupled harmonic oscillator with the chosen initial conditions can be translated into the cost function

$$\begin{aligned} L = \sum_i \Bigg(& \left| u_1''(t_i) + \frac{k}{m} u_1(t_i) - \frac{s}{m}(u_2(t_i) - u_1(t_i)) \right|^2 \\ & + \left| u_2''(t_i) + \frac{k}{m} u_2(t_i) + \frac{s}{m}(u_2(t_i) - u_1(t_i)) \right|^2 \\ & + \mu \left(|u_{1,0} - 1|^2 + |u_{2,0}|^2 + \left|u_{1,0}'\right|^2 + \left|u_{2,0}'\right|^2 \right) \Bigg), \end{aligned}$$

where $u_1 = y_{\text{Qubit1}}(t) \cdot \sigma_1$ and $u_2 = y_{\text{Qubit2}}(t) \cdot \sigma_2$ are the Z expectations values of the first and second qubit, multiplied by the classical factors σ_1 and σ_2, respectively. The solved coupled harmonic oscillator for $\omega_0^2 = 1$ and $\omega_1^2 = 16$ and 30 equidistant training points is shown in Figure 36.



(a) (b) (c)

Figure 36: (a) Solution of the differential equations (9.9) and (9.10) with $\mu = 20$, $N = 4$ and $D = 4$ using DQCs with $R_Y(t)$ rotations for data encoding. The parameters are classically optimized with COBYLA. (b) The absolute values of the error. (c) Values of the cost function versus the number of cost function evaluations.

We can see that it is possible to solve a coupled differential equation with only one circuit. The counter-phase oscillatory behavior, which is to be expected with the selected initial conditions, can be nicely observed. The advantage of using only one circuit is that the total number of circuit evaluations is halved since it is not necessary for two different circuits to be evaluated. In addition, the variational parameters in both circuits would have to be optimized and thus the entire algorithm would be more computationally intensive in terms of the classical optimization. Thus, this is a useful simplification of the algorithm, which could significantly reduce the computational effort, especially for very complicated systems with even more coupled equations.

10 Differential Equations: Real Quantum Computer

So far, we have successfully simulated differential equations with DQCs on exact simulators. The goal of this chapter is to show that it is possible to solve differential equations with DQCs on a real quantum computer. For this purpose, a very simple example of a differential equation is considered in order to minimize the computation time, which is limited on the IBM Quantum System One Ehningen. We focus on the differentiation equation

$$\begin{cases} u'(x) = 3x^2\,, \\ u(0) = u_0\,. \end{cases} \tag{10.1}$$

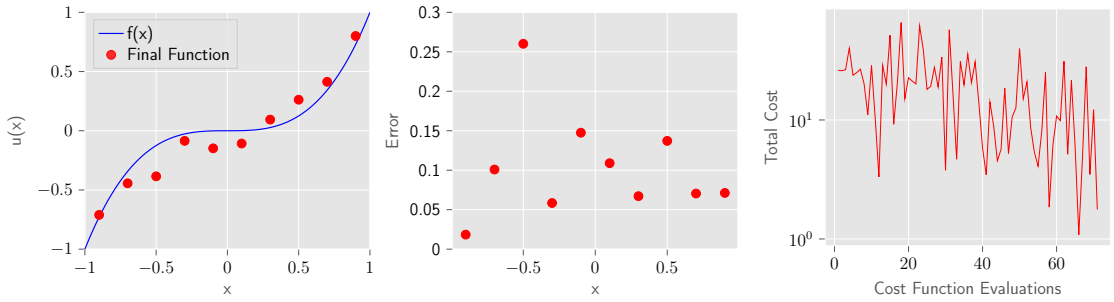
The solution of this differential equation is

$$u(x) = x^3 + u_0\,.$$

To be able to solve it with DQCs, we use Equation (10.1) to form the cost function

$$L = \sum_i \left(\left| u'(x_i) - 3x_i^2 \right|^2 + \mu \cdot |u(0) - u_0|^2 \right)\,,$$

where μ is a problem individual weight factor.
We solve the differential equation for $u_0 = 0$. The result for a circuit with $N = 3$, $D = 3$, a weight factor $\mu = 10$, 10 equidistant training points and 2000 shots is shown in Figure 37. The value range is chosen to be $x \in [-0.9, 0.9]$ because $R_X(\arcsin(x))$-encoding is used.



(a) (b) (c)

Figure 37: (a) Solution of the differential Equation (10.1) on the IBM Quantum System One Ehningen with $\mu = 10$, $N = 3$, $D = 3$, 10 equidistant training points and 2000 shots using DQCs with $R_Y(\arcsin(x))$ rotations for data encoding. (b) The absolute values of the error. (c) Values of the cost function versus the number of cost function evaluations.

It can be seen that the differential equation is solvable on the real quantum computer. The qualitative behavior can be observed well. The errors here are slightly higher than in the case of function approximations on the IBM Quantum System One Ehningen in Figure 20. This is because the derivatives are included in the cost function which results in higher errors as observed in Figure 24. In addition, with a direct function approximation, as seen in Figure 20, systematic errors can be compensated as explained in Chapter 4.1. That is not the case here, since only the derivative of the circuit is optimized.

11 Conclusions

11.1 Summary

In this work we conducted a comprehensive investigation of quantum circuit learning (QCL), wherein we thoroughly compared different circuit designs and evaluated their effectiveness in approximating specific functions. Here, efficient circuit designs could be developed for certain problems and multiple functions could be approximated with small errors. Especially the combination of $R_y(x)$ and $R_x(\arcsin(x))$ rotations has proven to be a very good encoding method to approximate complicated functions without choosing a too complex ansatz function, which can result in overfitting. The basic principles of this method were investigated in depth and conclusions could be drawn, which are relevant for a great number of variational quantum algorithms. For example, we analyzed how the depth of the circuit and the number of parameterized gates affect the expressibility and thus the resulting function approximation. A design of three parameterized rotations followed by circular entanglement, where the same parameters are repeated, proved to be the most effective. In addition, the importance of a classical parameter was shown, which greatly enhances the ability to approximate a large set of functions. Circuits designed with this newly gained knowledge were tested on the IBM Quantum System One in Ehningen. Here, promising results could be obtained. For a number of functions, the qualitative behavior could be approximated with good accuracy and relatively few function evaluations. The fact that the algorithm can already be executed on today's quantum computers shows that it is highly relevant in the current NISQ era.
Building upon this work, we investigated QCL in combination with the parameter shift rule to solve differential equations, coined as differentiable quantum circuits (DQCs). With the previously gained knowledge on function approximations, these circuits were also improved and the method was subjected to thorough testing across a multitude of differential equations. It has been shown that this method is able to solve a wide range of differential equations with high accuracy. The ability to solve nonlinear differential equations is particularly intriguing because there are few available quantum algorithms for this purpose. Here, a nonlinear differential equation could be solved. Additionally, a damped harmonic oscillator was solved, which causes problems with conventional solution methods. Subsequently, the differential equation of a coupled

harmonic oscillator could be solved by using only a single circuit, which reduced the number of circuit evaluations and the number of variational parameters.
However, it was also found that this algorithm requires many circuit evaluations, especially for high-order derivatives. This makes the classical optimization very computationally expensive. The comparison to different quantum algorithms for solving differential equations reinforces this impression. The strength of DQCs lies in the ability to solve differential equations that pose significant challenges for alternative algorithms. Moreover, they remain very NISQ-friendly even for complicated differential equations. This is, among other factors, because there is no input and output problem for this algorithm and the circuits remain very shallow even for large problems. On top of that, there is no need to create a quantum oracle, which would involve considerable computational effort. However, due to the expensive classical computational part and the large number of circuit evaluations, no quantum advantage is foreseeable for the simple problems presented in this work. This may change for more complex problems like a large number of coupled differential equations.

11.2 Outlook

The method described in this work has a lot of potential due to the combination of easy-to-generate ansatz functions and the exact differentiability with the parameter shift rule. However, the high classical computational effort due to the optimization of the parameters must be considered. This is a common problem that applies not only to this algorithm but to all variational quantum algorithms. Therefore, a lot of research focuses on this problem, for example in the expansion and development of more suitable classical optimizers [37]. Such methods could result in a significant reduction of computation time and make the algorithm even more relevant.
The algorithm can also be combined and extended with various common methods from quantum machine learning. For example, quantum kernels can be used, which are routinely applied to classify data in supervised quantum machine learning [38]. This could accelerate the algorithm and among other benefits, increase the noise resistance [39].
Moreover, the algorithm could offer significant advantages when applied to data that intrinsically embodies quantum mechanical properties. This is where the greatest advantages in quantum machine learning can be observed [40]. A similar advantage could also be observable for function approximations. However, proving such reasoning and finding suitable problems in the real world is very difficult and the existence of a potential quantum advantage is an open question.

Bibliography

[1] George Finlay Simmons. *Differential equations with applications and historical notes*. International series in pure and applied mathematics. New York: McGraw-Hill, 1972. ISBN: 0070573751.

[2] J. W. Thomas. *Numerical Partial Differential Equations: Finite Difference Methods*. Vol. 22. Springer eBook Collection Mathematics and Statistics. New York, NY: Springer, 1995. ISBN: 978-1-4419-3105-4. DOI: 10.1007/978-1-4899-7278-1.

[3] Ivan Dimov, István Faragó, and Lubin Vulkov, eds. *Finite Difference Methods. Theory and Applications: 7th International Conference, FDM 2018, Lozenetz, Bulgaria, June 11-16, 2018, Revised Selected Papers*. Vol. 11386. SpringerLink Bücher. Cham: Springer International Publishing, 2019. ISBN: 978-3-030-11538-8. DOI: 10.1007/978-3-030-11539-5.

[4] John Boyd, To Marilyn, and Paraphrasing Eliot. "Chebyshev and Fourier Spectral Methods". In: (2000).

[5] Peter W. Shor. "Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer". In: *SIAM Journal on Computing* 26.5 (1997), pp. 1484–1509. ISSN: 0097-5397. DOI: 10.1137/S0097539795293172.

[6] Lov K. Grover. "A fast quantum mechanical algorithm for database search". In: *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing - STOC '96*. Ed. by Gary L. Miller. New York, New York, USA: ACM Press, 1996, pp. 212–219. ISBN: 0897917855. DOI: 10.1145/237814.237866.

[7] Dominic W. Berry. "High-order quantum algorithm for solving linear differential equations". In: *Journal of Physics A: Mathematical and Theoretical* 47.10 (2014), p. 105301. ISSN: 1751-8113. DOI: 10.1088/1751-8113/47/10/105301.

[8] Ashley Montanaro and Sam Pallister. "Quantum algorithms and the finite element method". In: *Physical Review A* 93.3 (2016). ISSN: 2469-9926. DOI: 10.1103/PhysRevA.93.032324.

[9] Dominic W. Berry et al. "Quantum Algorithm for Linear Differential Equations with Exponentially Improved Dependence on Precision". In: *Communications in Mathematical Physics* 356.3 (2017), pp. 1057–1081. ISSN: 0010-3616. DOI: 10.1007/s00220-017-3002-y.

[10] Frank Gaitan. "Finding flows of a Navier–Stokes fluid through quantum computing". In: *npj Quantum Information* 6.1 (2020). DOI: 10.1038/s41534-020-00291-0.

[11] Bolesław Kacewicz. "Almost optimal solution of initial-value problems by randomized and quantum algorithms". In: *Journal of Complexity* 22.5 (2006), pp. 676–690. ISSN: 0885064X. DOI: 10.1016/j.jco.2006.03.001.

[12] Alexei Y. Kitaev. "Quantum measurements and the Abelian Stabilizer Problem". In: *Electron. Colloquium Comput. Complex.* TR96 (1995). URL: https://api.semanticscholar.org/CorpusID:17023060.

[13] Jacob Biamonte et al. "Quantum machine learning". In: *Nature* 549.7671 (2017), pp. 195–202. DOI: 10.1038/nature23474.

[14] Seth Lloyd et al. *Quantum algorithm for nonlinear differential equations*. 2020. DOI: 10.48550/arXiv.2011.06571.

[15] John Preskill. "Quantum Computing in the NISQ era and beyond". In: *Quantum* 2 (2018), p. 79. DOI: 10.22331/q-2018-08-06-79.

[16] Alberto Peruzzo et al. "A variational eigenvalue solver on a photonic quantum processor". In: *Nature communications* 5 (2014), p. 4213. DOI: 10.1038/ncomms5213.

[17] Nikolaj Moll et al. "Quantum optimization using variational algorithms on near-term quantum devices". In: *Quantum Science and Technology* 3.3 (2018), p. 030503. DOI: 10.1088/2058-9565/aab822.

[18] Maria Schuld, Ilya Sinayskiy, and Francesco Petruccione. "An introduction to quantum machine learning". In: *Contemporary Physics* 56.2 (2015), pp. 172–185. ISSN: 0010-7514. DOI: 10.1080/00107514.2014.964942.

[19] Maxwell Henderson et al. "Quanvolutional neural networks: powering image recognition with quantum circuits". In: *Quantum Machine Intelligence* 2.1 (2020). ISSN: 2524-4906. DOI: 10.1007/s42484-020-00012-y.

[20] K. Mitarai et al. "Quantum circuit learning". In: *Physical Review A* 98.3 (2018). ISSN: 2469-9926. DOI: 10.1103/PhysRevA.98.032309.

[21] Oleksandr Kyriienko, Annie E. Paine, and Vincent E. Elfving. "Solving nonlinear differential equations with differentiable quantum circuits". In: *Physical Review A* 103.5 (2021). ISSN: 2469-9926. DOI: 10.1103/PhysRevA.103.052416.

[22] Michael A. Nielsen and Isaac L. Chuang. *Quantum computation and quantum information*. 10th anniversary ed. Cambridge: Cambridge Univ. Press, 2010. ISBN: 978-1-107-00217-3. URL: https://ebookcentral.proquest.com/lib/subhh/detail.action?docID=647366.

[23] F. T. Arecchi et al. "Atomic Coherent States in Quantum Optics". In: *Physical Review A* 6.6 (1972), pp. 2211–2237. ISSN: 2469-9926. DOI: 10.1103/PhysRevA.6.2211.

[24] Maria Schuld et al. "Evaluating analytic gradients on quantum hardware". In: *Physical Review A* 99 (2019). ISSN: 2469-9926. DOI: 10.1103/PhysRevA.99.032331.

[25] Gavin N. Nop, Durga Paudyal, and Jonathan D. H. Smith. "Ytterbium ion trap quantum computing: The current state-of-the-art". In: *AVS Quantum Science* 3.4 (2021), p. 044101. DOI: 10.1116/5.0065951.

[26] Dorit Aharonov and Michael Ben-Or. "Fault-Tolerant Quantum Computation with Constant Error Rate". In: *SIAM Journal on Computing* 38.4 (2008), pp. 1207–1282. ISSN: 0097-5397. DOI: 10.1137/S0097539799359385.

[27] Andrew W. Cross et al. "Validating quantum computers using randomized model circuits". In: *Physical Review A* 100.3 (2019). ISSN: 2469-9926. DOI: 10.1103/PhysRevA.100.032328.

[28] Youngseok Kim et al. "Evidence for the utility of quantum computing before fault tolerance". In: *Nature* 618.7965 (2023), pp. 500–505. DOI: 10.1038/s41586-023-06096-3.

[29] J. H. Mathews and K. D. Fink. *Numerical Methods Using MATLAB*. Pearson Education. Prentice Hall, 1999. ISBN: 9780132700429. URL: https://books.google.de/books?id=F1sZAQAAIAAJ.

[30] D. Kraft. *A Software Package for Sequential Quadratic Programming*. Deutsche Forschungs- und Versuchsanstalt für Luft- und Raumfahrt Köln: Forschungsbericht. Wiss. Berichtswesen d. DFVLR, 1988. URL: https://books.google.de/books?id=4rKaGwAACAAJ.

[31] Kan Hatakeyama-Sato et al. *Quantum circuit learning as a potential algorithm to predict experimental chemical properties*. 2022. DOI: 10.26434/chemrxiv-2022-cz7wr-v2.

[32] M. J. D. Powell. "A Direct Search Optimization Method That Models the Objective and Constraint Functions by Linear Interpolation". In: *Advances in Optimization and Numerical Analysis*. Ed. by Susana Gomez and Jean-Pierre Hennart. Dordrecht: Springer Netherlands, 1994, pp. 51–67. ISBN: 978-90-481-4358-0. DOI: 10.1007/978-94-015-8330-5{\textunderscore}4.

[33] H. F. Trotter. "On the Product of Semi-Groups of Operators". In: *Proceedings of the American Mathematical Society* 10.4 (1959), p. 545. ISSN: 00029939. DOI: 10.2307/2033649.

[34] Masuo Suzuki. "Generalized Trotter's formula and systematic approximants of exponential operators and inner derivations with applications to many-body problems". In: *Communications in Mathematical Physics* 51.2 (1976), pp. 183–190. ISSN: 0010-3616. DOI: 10.1007/BF01609348.

[35] Guang Hao Low and Isaac L. Chuang. "Hamiltonian Simulation by Qubitization". In: *Quantum* 3 (2019), p. 163. DOI: 10.22331/q-2019-07-12-163.

[36] Shantanav Chakraborty, András Gilyén, and Stacey Jeffery, eds. *The Power of Block-Encoded Matrix Powers: Improved Regression Techniques via Faster*

Hamiltonian Simulation: Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik GmbH, Wadern/Saarbruecken, Germany. 2019. DOI: 10.4230/LIPIcs.ICALP.2019.33.

[37] Marco Wiedmann et al. *An Empirical Comparison of Optimizers for Quantum Machine Learning with SPSA-based Gradients*. 2023. DOI: 10.48550/arXiv.2305.00224.

[38] Vojtech Havlicek et al. "Supervised learning with quantum enhanced feature spaces". In: (2018). DOI: 10.48550/arXiv.1804.11326.

[39] Valentin Heyraud et al. "Noisy quantum kernel machines". In: *Physical Review A* 106.5 (2022). ISSN: 2469-9926. DOI: 10.1103/PhysRevA.106.052421.

[40] Frank Arute et al. "Quantum supremacy using a programmable superconducting processor". In: *Nature* 574.7779 (2019), pp. 505–510. DOI: 10.1038/s41586-019-1666-5.

[41] R. Cleve et al. "Quantum algorithms revisited". In: *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences* 454.1969 (1998), pp. 339–354. ISSN: 1364-5021. DOI: 10.1098/rspa.1998.0164.

A Appendix: Number of Parameters

In the main part of the work, only repeating parameters, in the form of three parameterized rotations were considered. Here, we briefly discuss different possibilities. The parameter correlations can also be used to assess whether it makes sense to repeat the parameters according to the depth, as explained in Chapter 4.2, instead of selecting new parameters for every layer. A circuit with new parameters is shown in Figure 38.

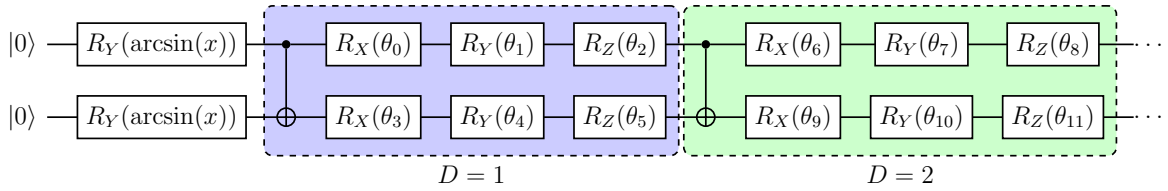


Figure 38: Two-qubit circuit with $R_Y(\arcsin(x))$-encoding with new parameters for each repeated block for depth D.

Figure 41 shows the parameter correlations for this case. There, we can see that non-repeating parameters lead to a slight improvement in expressibility compared to Figure 10. With the significantly higher number of parameters, the complexity and thus the duration of the classical optimization would increase significantly in the case of function approximations. Since the classical optimization is already very computationally intensive, this trade-off does not make sense. So it seems to be most reasonable to use the former method of repetitive parameters.
The number and type of rotations also leave room for investigation. In the previous part, we worked with three rotations that together can represent any unitary operation. However, instead of three rotations per qubit, only two rotations can be used, as shown in Figure 39.

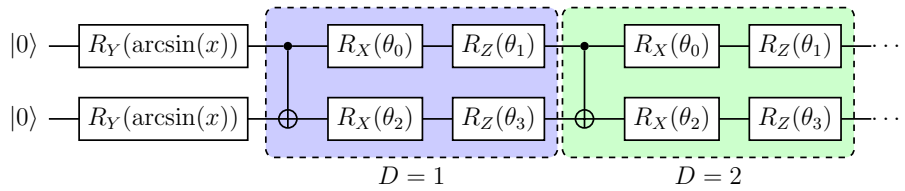


Figure 39: Two-qubit circuit with $R_Y(\arcsin(x))$-encoding followed by blocks of a controlled NOT gate and θ-parameterized y- and z-rotations.

The corresponding parameter correlations can be seen in Figure 42. The expressibility is significantly lower than before, and even with high depths, the expressibility does not approach that in Figure 18. Numerous simulations, which will not be included here, have shown that the reduced number of parameters and the slightly lower depth of the circuit are not worth this drop in expressibility. Analogous to before, new parameters can be used for each block, as shown in Figure 40.

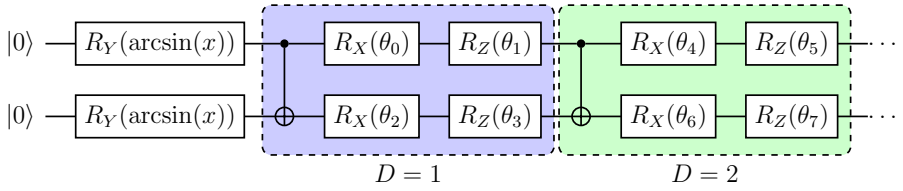
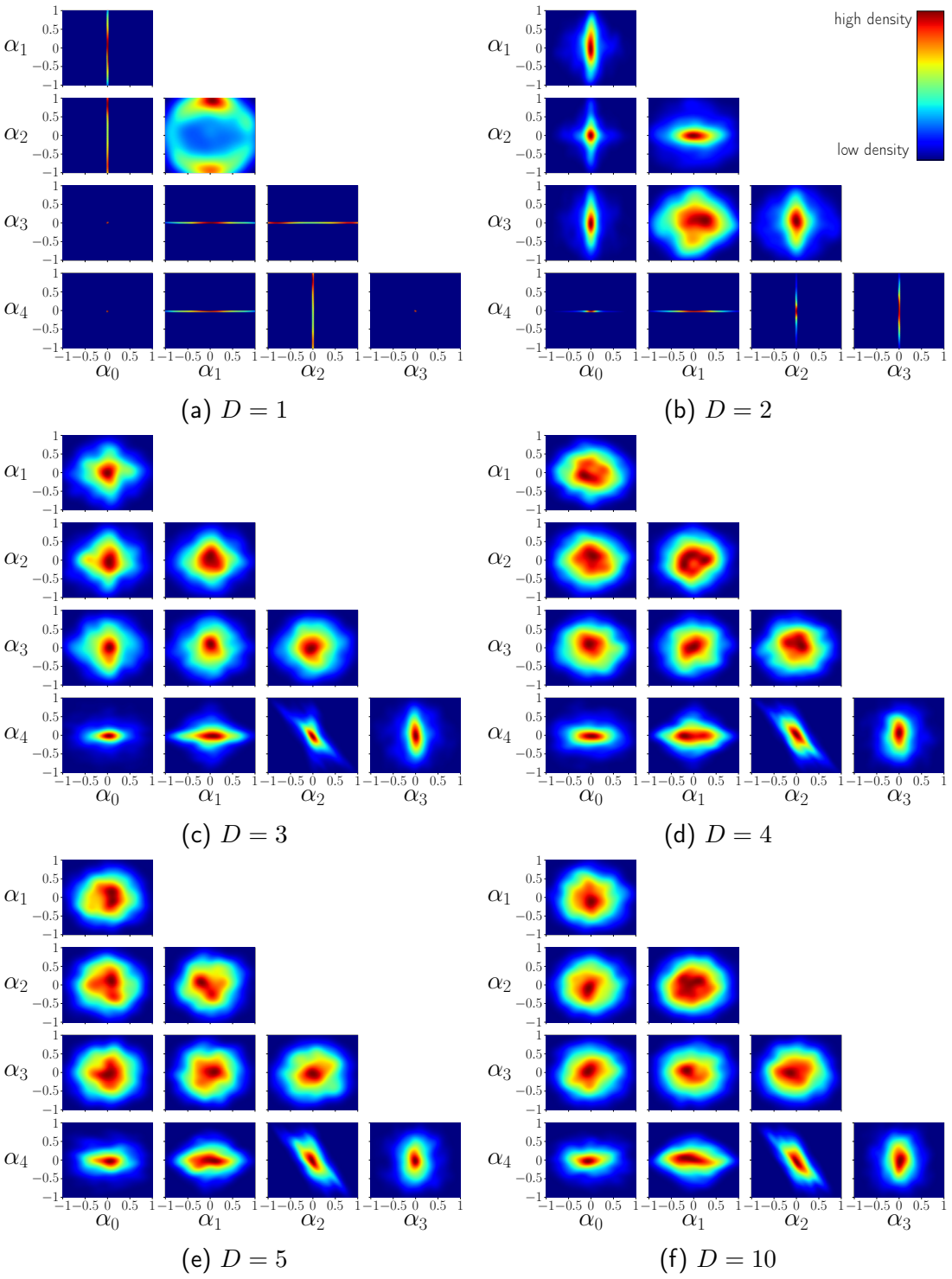


Figure 40: Two-qubit circuit with $R_Y(\arcsin(x))$-encoding followed by blocks of a controlled NOT gate and θ-parameterized y- and z-rotations with new parameters for each repeated block.

The parameter correlations can be seen in Figure 43. Contrary to before, a significant improvement can be observed here. Overall, however, no notable enhancement can be seen in comparison to the example of three parameters in Figure 18, which means that this setup is also not useful. In conclusion, it seems that two parameters are not sufficient for an efficient function approximation.

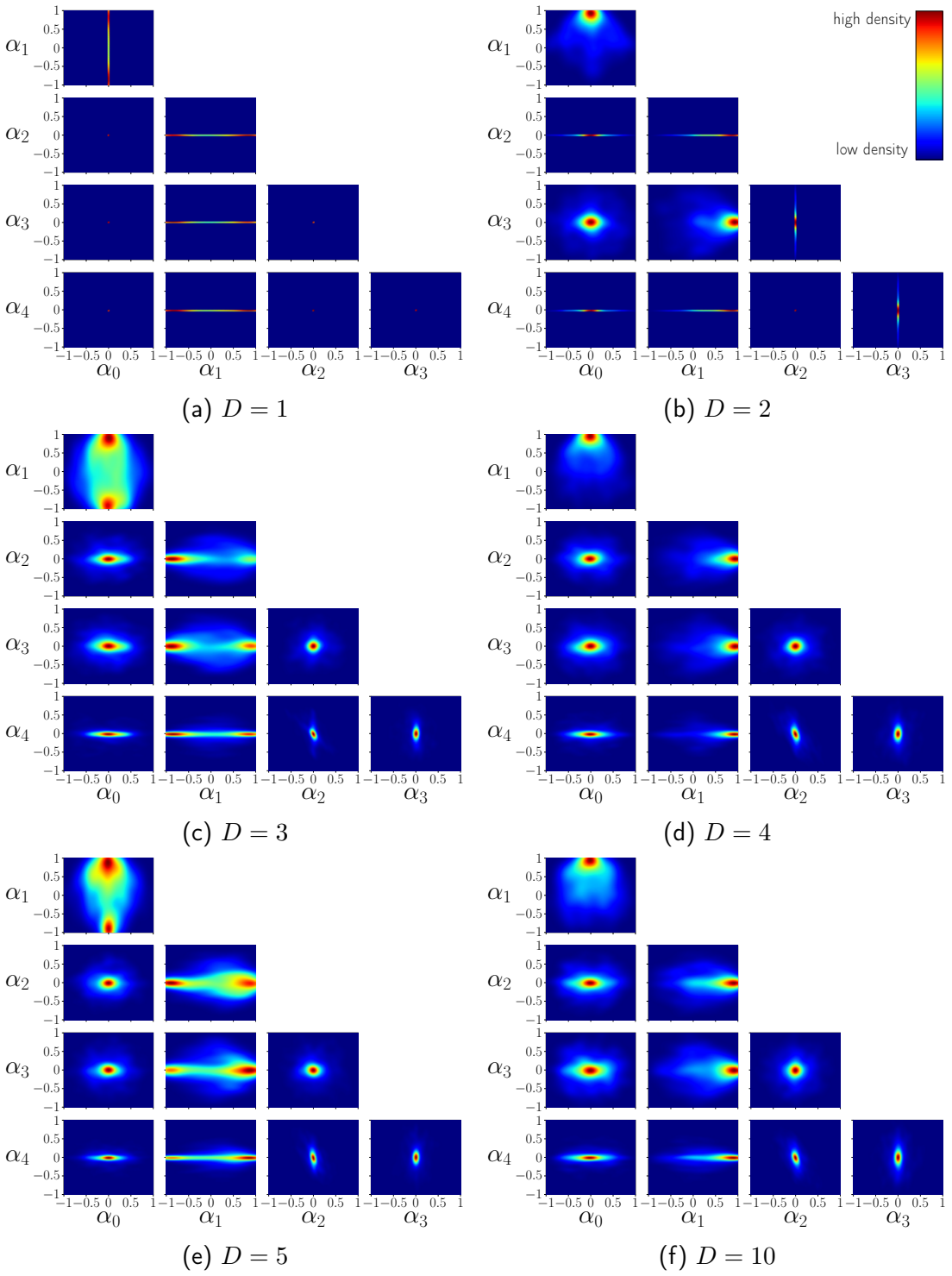


(a) $D = 1$ (b) $D = 2$

(c) $D = 3$ (d) $D = 4$

(e) $D = 5$ (f) $D = 10$

Figure 41: Parameter correlations of the circuit in Figure 38 with a qubit number $N = 2$ and different depths D after fitting the Z expectation value $y(x)$ with the function given in (4.3) for 1000 random sets of parameters $\vec{\theta}$ using the least square method. For readability, one-dimensional lines are slightly broadened to be visible in the plots.

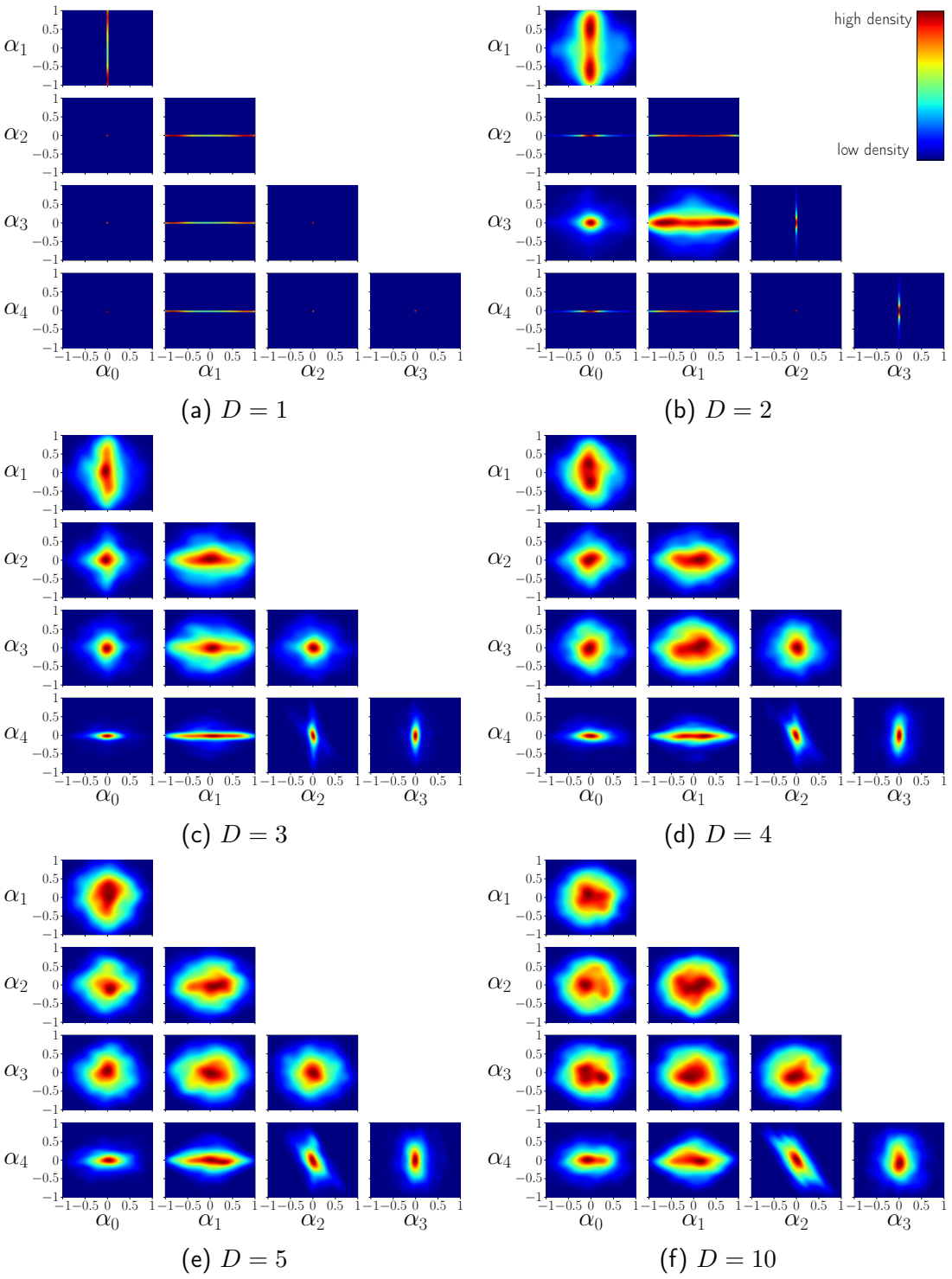


(a) $D = 1$ (b) $D = 2$

(c) $D = 3$ (d) $D = 4$

(e) $D = 5$ (f) $D = 10$

Figure 42: Parameter correlations of the circuit in Figure 39 with a qubit number $N = 2$ and different depths D after fitting the Z expectation value $y(x)$ with the function given in (4.3) for 1000 random sets of parameters $\vec{\theta}$ using the least square method. For readability, one-dimensional lines are slightly broadened to be visible in the plots.



(a) $D = 1$ (b) $D = 2$

(c) $D = 3$ (d) $D = 4$

(e) $D = 5$ (f) $D = 10$

Figure 43: Parameter correlations of the circuit in Figure 40 with a qubit number $N = 2$ and different depths D after fitting the Z expectation value $y(x)$ with the function given in (4.3) for 1000 random sets of parameters $\vec{\theta}$ using the least square method. For readability, one-dimensional lines are slightly broadened to be visible in the plots.

B Appendix: Least Square Method

The least squares method is a classical procedure for determining the best fit for a set of data points. The method works by minimizing the sum of the offsets (residuals) of the points of a curve. The goal is to fit the parameters of a model function to match the data. A simple data set of a two-dimensional function consists of n pairs (x_i, y_i) with $i = 1, ..., n$, where x_i is an independent variable and y_i is a dependent variable. The value of the dependent variable is determined by observation and we want to describe the behavior of the data points with a model function. The model function is of the form $f(x, \vec{a})$, where $\vec{a}$ consists of m free parameters. The goal is to find the parameter values so that the model function best describes the data. How good a particular parameter choice is, can be measured by the residual of the data points, which is defined as the difference between the observed value of the dependent variable and the predicted value of the model

$$r_i(\vec{a}) = y_i - f(x_i, \vec{a})\,.$$

Here, the model function can have any form. The optimal parameter values are determined by minimizing the sum of squared residuals. In the case of a linear function of the form

$$f(x, \vec{a}) = \sum_{j=1}^{m} a_j \phi_j(x)\,,$$

where the function ϕ_j is a function of x, we can solve the problem by writing it the form of matrices. We define

$$X_{ij} = \phi_j(x_i)$$

and form the vector $\vec{y}$ with the dependant variables y_i. Now, the sum of residuals can be calculated as

$$\begin{aligned} L(\vec{a}) &= ||\vec{y} - X\vec{a}||^2 = (\vec{y} - X\vec{a})^\intercal(\vec{y} - X\vec{a}) \\ &= \vec{y}^\intercal\vec{y} - \vec{y}^\intercal X\vec{a} - \vec{a}^\intercal X^\intercal\vec{y} + \vec{a}^\intercal X^\intercal X\vec{a}\,. \end{aligned}$$

To solve the least square problem, $L(\vec{a})$ is minimized so that the gradient becomes zero. The gradient can be expressed as

$$\frac{\partial L(\vec{x}, \vec{y}, \vec{a})}{\partial \vec{a}} = -2X^\intercal \vec{y} + 2X^\intercal X\vec{a}\,.$$

Setting the gradient to zero and solving for $\vec{a}$ we obtain

$$\vec{a} = (X^\intercal X)^{-1} X^\intercal \vec{y}\,.$$

This means that the parameters of the model function can now be determined so that the function best describes the data points.

C Appendix: Hadamard Test

We consider an arbitrary state $|\psi\rangle$ and a unitary operator U acting on this state. The Hadamard test [41] is a commonly used part of quantum algorithms with the purpose of generating a qubit state whose Z expectation value is equal to the real part of $\langle\psi|U|\psi\rangle$. The circuit of the Hadamard test is shown in Figure 44.

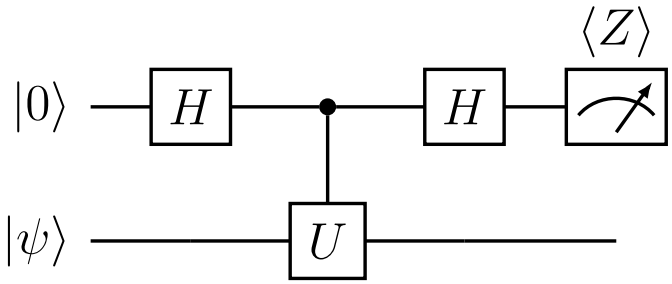


Figure 44: Circuit of the Hadamard test for an arbitrary state $|\psi\rangle$.

To understand the Hadamard test, we will look at the development of the state. The first Hadamard gate creates the state

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

Following this, the unitary operator U is applied on the state $|\psi\rangle$, conditioned on the first qubit. This results in the state

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

After applying the second Hadamard gate, the final state is

$$\frac{1}{2}(|0\rangle \otimes (I+U)\,|\psi\rangle + |1\rangle \otimes (I-U)\,|\psi\rangle)\,.$$

This means that the probability of measuring the state $|0\rangle$ and $|1\rangle$ on the first qubit is

$$P(|0\rangle) = \frac{1}{4}\,\langle\psi|(I+U^\dagger)(I+U)|\psi\rangle$$

and

$$P(|1\rangle) = \frac{1}{4} \langle\psi|(I - U^\dagger)(I - U)|\psi\rangle \ ,$$

respectively. If we now calculate the Z expectation value of the first qubit with this knowledge, we obtain

$$\langle Z\rangle = P(|0\rangle) - P(|1\rangle) = \frac{1}{2} \langle\psi|U^\dagger + U|\psi\rangle \ ,$$

which is equivalent to $\Re(\langle\psi|U|\psi\rangle)$.