Karlsruhe Institute of Technology

Extending Gradient-Free Optimization of Parameterized Quantum Circuits to Controlled Pauli Gates

Bachelor's Thesis of

Maximilian Tim Schweikart

At the KIT Department of Informatics
Steinbuch Centre for Computing

First examiner:	Prof. Dr. Achim Streit
Second examiner:	Prof. Dr. Bernhard Neumair
First advisor:	Dr. Eileen Kühn
Second advisor:	Dr. Max Fischer

Aug. 22, 2023 – Dec. 22, 2023

Karlsruher Institut für Technologie
Fakultät für Informatik
Postfach 6980
76128 Karlsruhe

I declare that I have developed and written the enclosed thesis completely by myself. I have not used any other than the aids that I have mentioned. I have marked all parts of the thesis that I have included from referenced literature, either in their original wording or paraphrasing their contents. I have followed the by-laws to implement scientific integrity at KIT.

Karlsruhe, Dec. 22, 2023

. . . . . .

(Maximilian Tim Schweikart)

Contents

1. Introduction

While the first theoretical foundations of quantum computers have already been developed in the 1980s, quantum computers have only recently gathered widespread attention with the development and availability of real quantum computing hardware [13, 8]. The field is evolving rapidly, with new tools, algorithms, and hardware being released every month. Still, today's quantum devices are subject to high amounts of noise, have a very limited number of qubits, and are not fully connected. These limitations often further reduce the number of available qubits and gates needed to perform a given task [5]. Researchers often refer to these limited quantum devices as *noisy intermediate-scale quantum* (NISQ) devices [15].

One promising idea for harnessing the computational power of quantum devices within the NISQ era is to apply Machine Learning methodology to quantum computers [5]. Machine Learning can often be used to approximate complex functions without requiring intricate knowledge about their nature. Moreover, Machine Learning can compensate for or even benefit from a limited amount of noise [6].

A typical Machine Learning workflow uses a parameterizable function as a so-called model [3]. Machine Learning aims to choose parameters for the model so that the model maps any given input data closely to the expected output data. In many iterations, an *optimizer* evaluates the model function to compute the *loss value*, which indicates how much the calculated value deviates from the expected output value. The optimizer uses these evaluations to produce new parameter values and then repeats its assessment. This process is known as *training*. Many state-of-the-art optimizers use an approach called *gradient descent* to improve the model's parameters [1]. Machine Learning has recently made a tremendous leap due to the availability of computational resources and improvements in model function design [9]. These improvements allow Machine-Learning applications to train billions of parameters with terabytes of training data in many iterations.

A *Parameterized Quantum Circuit* (PQC) can replace the classical model function to transfer the machine-learning methodology to quantum computers. PQCs are the quantum equivalent of parameterized algorithms like neural networks [1]. Gradient-based optimizers can be used for Quantum Machine Learning (QML) too, but the current price of NISQ devices does not allow for large amounts of data and large numbers of iterations. Additionally, each evaluation of a quantum circuit comprises multiple shots to decrease statistical noise introduced by qubit measurement.

In "*Structure optimization for parameterized quantum circuits*," Ostaszewski et al. explore a different optimization approach [14]. They observed that the univariate expected value

of a PQC w.r.t. a single rotational Pauli gate parameter is always sinusoidal. Using three circuit evaluations with select parameter values, they could reconstruct this univariate loss function for every parameter value. Since the curve of sine functions is well-understood, they could then analytically calculate the parameter value that minimizes this univariate loss function. The `Rotosolve` optimizer uses this approach to optimize each parameter individually. While univariate optimization does not guarantee convergence to the global minimum, experiments show good results with this approach compared to state-of-the-art optimizers such as Adam and SPSA [10, 18, 14]. However, the original `Rotosolve` approach was limited to optimizing parameters of single-qubit gates.

These promising results naturally pose the question of whether this approach can be extended to other types of gates. Because of their similarity with the studied rotational gates, this bachelor's thesis presents `Crotosolve`, an extension of the `Rotosolve` optimization technique for controlled rotational Pauli gates.

To develop and investigate the `Crotosolve` idea, this thesis is structured as follows. First, in chapter 2, I summarize the theoretical background in quantum computing, Machine Learning, and Quantum Machine Learning required to understand the idea and proof behind `Crotosolve`. Chapter 4 analyzes the mathematical structure of a controlled rotational Pauli gate parameter's univariate effect on the expected value of a quantum circuit measurement. It also contains a constructive proof for an algorithm that can be used to determine the prefactors and offsets characterizing the effect function's specific curve.

The `Crotosolve` algorithm is evaluated in chapter 5. I present a proof-of-concept implementation in PennyLane, a major quantum computing SDK [2, 19]. Using this implementation and PennyLane's `Rotosolve` [14], Adam [10], Adagrad [7], and Stochastic Gradient Descent implementation, I create a benchmark of their loss curves. The data gathered in this benchmark shows that `Crotosolve` outperforms the other optimizers in most cases. Its loss value progresses towards the minimum value much quicker and more consistently in comparison to its competitors. However, in cases with high numbers of parameters, the gradient-based Adam optimizer shows a slightly better final result. Even in this case, `Crotosolve` progression towards low loss values is steeper. A comprehensive discussion of this evaluation can be found in section 5.3.

While I believe `Crotosolve` to be the first implementation that exploits the sinusoidal structure of CRP gate loss functions explicitly, other gradient-free approaches have been developed in the recent years. In chapter 3, I outline how contributions by Wierichs et al. allow the reconstruction of univariate loss functions of almost arbitrary PQCs [20]. These results have already been implemented in PennyLane's `Rotosolve` implementation and were evaluated as part of chapter 5.

Finally, in chapter 6, I present a summary of the contributions from this thesis. Furthermore, I outline how some of the optimizations of my `Crotosolve` implementation and the Wierichs-improved implementation of `Rotosolve` could be fused in an optimizer that combines their advantages.

The proof-of-concept implementation as well as the code and benchmark data of my evaluation are available on GitHub and Zenodo [16].

2. Background

This chapter introduces the theoretical background for understanding the idea behind `Crotosolve`. Rather than giving a complete overview of the field, this text is strictly focused on aspects critical for proving the optimizer's correctness and understanding its applications. I will first present an introduction to the theory of quantum information processing using qubits, gates, and circuits. Following this introduction, I will also give a quick introduction to the field of Machine Learning. I will show how Quantum Machine Learning applies the ideas from Machine Learning to quantum processing devices. The final section of this chapter will focus on state-of-the-art classical and quantum optimization techniques in particular. One of these optimization methods is `Rotosolve`, which I will present with further detail. These basic ideas are also fundamental for the `Crotosolve` algorithm, which will be developed in the next chapter.

2.1. A brief introduction to quantum computing

At its core, the operation of a quantum computer revolves around gates and qubits. Qubits (quantum bits) make up the data register of a quantum computer. Just like a classical bit can be in the 0 or 1 state, a qubit can be in a corresponding $|0\rangle$ or $|1\rangle$ state. Besides these two basis states, however, quantum bits can be in a superposition of these two states,

$$|\psi\rangle = \alpha_0 \cdot |0\rangle + \alpha_1 \cdot |1\rangle \ ,$$

where the probability amplitudes $\alpha_0, \alpha_1 \in \mathbb{C}$ can be any complex numbers with $|\alpha_0|^2 + |\alpha_1|^2 = 1$. The superposable character of a qubit essentially increases its expressibility in comparison to a classical, digital bit from the discrete set $\{0, 1\}$ to the complex, two-dimensional sphere surface $\left\{\vec{\alpha} \in \mathbb{C}^2 \mid |\vec{\alpha}|^2\right\}$. Thus, a quantum computer with a single qubit can work with continuous, multidimensional data, while a classical computer with a single bit can only work with a single boolean value.

Due to the laws of quantum mechanics, this multidimensional superposition state collapses into one of its discrete basis states upon observation. More specifically, this means that *measuring* the value of a qubit cannot reveal its probability amplitudes and instead turns the state into either $|0\rangle$ (i.e., $\alpha_0 = 1, \alpha_1 = 0$) or $|1\rangle$ (i.e., $\alpha_0 = 0, \alpha_1 = 1$)[1]. The probability of

[1]Technically, it is possible to measure the state in a different basis, like $|+\rangle = \frac{|0\rangle+|1\rangle}{\sqrt{2}}$ and $|-\rangle = \frac{|0\rangle-|1\rangle}{\sqrt{2}}$. However, this will not be necessary for the development of `Crotosolve`.

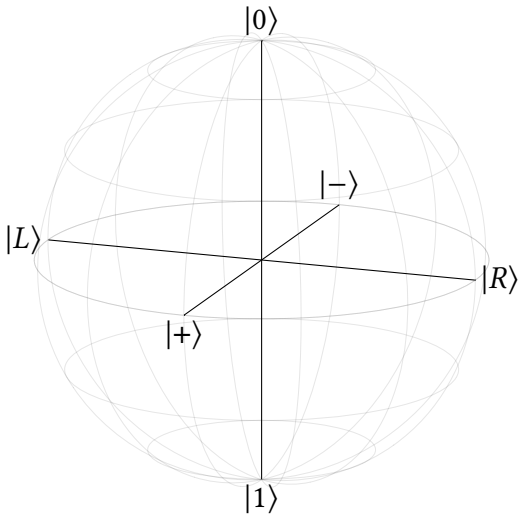


Figure 2.1.: The Bloch sphere can represent the state $|\psi\rangle = e^{i\gamma}\left(\cos\frac{\theta}{2}|0\rangle + e^{i\varphi}\sin\frac{\theta}{2}|1\rangle\right)$ of a single qubit, ignoring its global phase. φ specifies the rotation of the state around the Z axis ($|0\rangle \rightarrow |1\rangle$). θ specifies the rotation of the state relative to the XY plane (X axis: $|-\rangle \rightarrow |+\rangle$, Y axis: $|L\rangle \rightarrow |R\rangle$).

the qubit collapsing into either of these states is given by the squared absolute value of its probability amplitude,

$$\mathbb{P}(M = |n\rangle) = |\alpha_n|^2,\ n \in \{0, 1\}\,.$$

Thus, the normalization of the probability amplitudes, $\sum_{n\in\{0,1\}} |\alpha_n|^2 = 1$, is in fact a normalization of the probability distribution over the given basis states. While it is physically impossible to observe α_0 and α_1 directly, their squared absolute values still influence the outcome of observation M. If the same[2] quantum state can be prepared many times, $|\alpha_n|^2$ can be estimated as the relative frequency from an empirical probability distribution of M.

Since only the absolute value of α_0 and α_1 influences the measurement outcome, it is common to rewrite the state as follows.

$$\begin{aligned} |\psi\rangle &= \alpha_0|0\rangle + \alpha_1|1\rangle \\ &= e^{i\gamma}\left(\cos\frac{\theta}{2}|0\rangle + e^{i\varphi}\sin\frac{\theta}{2}|1\rangle\right) \end{aligned} \tag{2.1}$$

In this representation, $e^{i\gamma}$ has no effect on the outcome of measurement M because $\left|e^{i\gamma}\right| = 1$. That is why the *global phase* γ is often ignored. The remaining two variables θ and φ can be interpreted as coordinates on the so-called Bloch sphere, which is shown in Figure 2.1.

[2]Two quantum states $\sum \alpha_n |n\rangle$ and $\sum \beta_n |n\rangle$ are the same if all of their probability amplitudes are the same, i.e., $\alpha_n = \beta_n \forall n$.

2.1.1. Multi-qubit-systems

The separated state of several qubits can be combined into a multi-qubit state through the Kronecker product. With two qubits, this results in

$$\begin{aligned}
&(\alpha_0 |0\rangle + \alpha_1 |1\rangle) \otimes (\beta_0 |0\rangle + \beta_1 |1\rangle) \\
=&\alpha_0\beta_0 |0\rangle \otimes |0\rangle + \alpha_0\beta_1 |0\rangle \otimes |1\rangle + \alpha_1\beta_0 |1\rangle \otimes |0\rangle + \alpha_1\beta_1 |1\rangle \otimes |1\rangle \\
=& \underbrace{\alpha_0\beta_0}_{=:\gamma_{00}} |00\rangle + \underbrace{\alpha_0\beta_1}_{=:\gamma_{01}} |01\rangle + \underbrace{\alpha_1\beta_0}_{=:\gamma_{10}} |10\rangle + \underbrace{\alpha_1\beta_1}_{=:\gamma_{11}} |11\rangle .
\end{aligned} \tag{2.2}$$

with $\sum_{n\in\{00,01,10,11\}} |\gamma_n|^2 = 1$. I use natural numbers instead of bitstrings to simplify the notation in the following. For example, Equation 2.2 can be compactly rewritten to the following.

$$\begin{aligned}
&\gamma_{00} |00\rangle + \gamma_{01} |01\rangle + \gamma_{10} |10\rangle + \gamma_{11} |11\rangle \\
&= \gamma_0 |0\rangle + \gamma_1 |1\rangle + \gamma_2 |2\rangle + \gamma_3 |3\rangle \\
&= \sum_{j=0}^{3} \gamma_j |j\rangle
\end{aligned} \tag{2.3}$$

Similarly to this two-qubit example, a system with n separate qubit states $|\psi_i\rangle$, $1 \le i \le n$ can be described with a product state. In this system, the probability amplitudes $\gamma_0, \ldots, \gamma_{2^n-1} \in \mathbb{C}$ of the standard basis states $|0\rangle, \ldots, |2^n - 1\rangle$ are again given by the multiplication and form a probability distribution through their squared absolute values.

$$|\psi\rangle = \bigotimes_{i=1}^{n} |\psi_i\rangle = \sum_{j=0}^{2^n-1} \gamma_j |j\rangle \quad \text{with} \quad \sum_{j=0}^{2^n-1} \left|\gamma_j\right|^2 = 1 \tag{2.4}$$

2.1.2. Gates and circuits

Unlike with classical bits and gates, it is physically impossible for a quantum computer to create copies of a qubit. While classical computers typically read (i.e., copy), transform (e.g., add two numbers), and write data from and to registers, quantum computers can only perform in-place operations.

Quantum states can be changed by applying operators to them. Since the state of an n-qubit-system can be described through its probability amplitudes γ_i, we can represent this state with a 2^n-dimensional complex vector. If we identify $|0\rangle \equiv \begin{pmatrix}1 & 0\end{pmatrix}^\top$ and $|1\rangle \equiv \begin{pmatrix}0 & 1\end{pmatrix}^\top$, equation 2.4 implicitly leaves us with

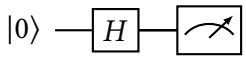


Figure 2.2.: In this single-qubit quantum circuit, a Hadamard gate is applied to the qubit before it is measured. This circuit will probabilistically output $|0\rangle$ just as often as $|1\rangle$.

$$|\psi\rangle \equiv \begin{pmatrix} \gamma_{0^n} \\ \vdots \\ \gamma_{1^n} \end{pmatrix}. \tag{2.5}$$

Universal quantum computers allow us to apply operations to these states that are described by unitary matrices[3]. Such an operation is referred to as a *gate* in quantum computing. Gates can act on one or multiple qubits, and bigger gates can be composed by computing the Kronecker product of gates.

The application of gates to a system of qubits can be visualized through quantum circuits. In these circuits, each qubit is displayed as a horizontal line, and gates are displayed as boxes that overlay the lines of the qubits they are applied to. Measurements are indicated by a box with a meter inside.

Take for example the Hadamard gate H,

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}. \tag{2.6}$$

Applying the Hadamard gate to a single qubit will transform the $|0\rangle$ state into $\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$ and $|1\rangle$ into $\frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle$, which are both equal superpositions of the original states. Figure 2.2 visualizes the application of a Hadamard gate in a quantum circuit.

All single-qubit gates $U \in \mathbb{C}^{2\times 2}$ can be represented by products of *rotational Pauli gates* $RX(\theta), RY(\theta)$ and $RZ(\theta)$,

$$RP(\theta) = \cos\left(\frac{\theta}{2}\right) \cdot I_2 - i \cdot \sin\left(\frac{\theta}{2}\right) \cdot P, \quad P \in \{X, Y, Z\} \tag{2.7}$$

These gates are called *parameterized gates* since they can be adjusted by a continuous parameter $\varphi \in [0, 2\pi]$. The Hadamard gate, for example, can be written as $H = RX(\pi) \cdot RY(0.5\pi)$.

Single-qubit operations can be simulated trivially by computing the product of the gate matrices and the state vector.

[3] A matrix $A \in \mathbb{C}^{N\times N}$ is called unitary if and only if its conjugate transpose is its inverse, i.e. $\overline{A^\top} = A^{-1}$.

The problem gets more interesting with two-qubit gates. For example, the controlled X gate (also known as $CNOT$) works on a system of two qubits and maps the basis states as follows:

$$CX = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \tag{2.8}$$

$$CX\,|00\rangle = |00\rangle \tag{2.9}$$

$$CX\,|01\rangle = |01\rangle \tag{2.10}$$

$$CX\,|10\rangle = |11\rangle \tag{2.11}$$

$$CX\,|11\rangle = |10\rangle \tag{2.12}$$

When applying the CX gate to states from the computational basis, the second bit is flipped (X is applied) if and only if the first bit is $|1\rangle$. States other than those from the computational basis might, however, not follow this simple principle. The $|{+}{+}\rangle$ state, for example, is transformed into the $|{-}{+}\rangle = CX\,|{+}{+}\rangle$ state, changing the control qubit but not the target qubit.

2.2. Quantum Machine Learning

Today's quantum devices are far from ideal implementations of the qubits, gates, and circuits presented in section 2.1. Noise is a major limitation of today's quantum devices. Noise is hard to describe completely since it can be the result of a multitude of factors. For example, noise can be accumulated over time by unwanted interactions of the physical qubit implementation with the outside world. Additionally, the number of qubits on today's quantum devices is limited to a couple dozen or a couple hundred. On quantum computers with high numbers of qubits, the connection topology is incomplete, as this simplifies the architecture of the physical system. This means that multi-qubit gates cannot be applied to arbitrary groups of qubits directly. This limitation can be mitigated through the application of swap gates prior to the multi-qubit gate application, however, this further increases the quantum computer's exposure to noise. The limitations of today's quantum devices are collectively referred to as the *noisy intermediate-scale quantum* (NISQ) era [15, 13].

Research suggests that Quantum Machine Learning and Variational Quantum Algorithms may produce useful results even in the presence of NISQ limitations [5]. In the following section, I will first introduce a few ideas from classical Machine Learning. Using these results, I will present several approaches to Quantum Machine Learning. I will focus on optimization techniques in particular, as from Chapter 4 onwards, I will introduce a new optimizer that can produce results in this regime with high efficiency.

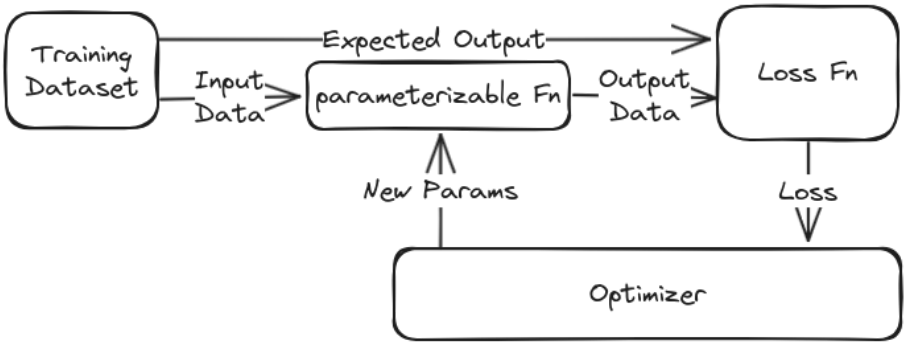


Figure 2.3.: In a typical supervised Machine Learning optimization loop, the optimizer runs the model with training data to compute a loss value. It updates the model's parameters to reduce the loss value in the next evaluation. Through many iterations of this process, the model learns the relation between input and output data [3].

2.2.1. Classical Machine Learning

According to Bishop [3], Machine Learning typically aims to find and reproduce a pattern in a dataset. In a *supervised* Machine Learning task, a set of input data is given together with its corresponding output data. This data could, for example, be the results of an experiment where the input represents the parameters of the experiment and the output represents the measurement results corresponding to these parameters. The goal of a supervised Machine Learning task is to recognize the relation between input and output data to predict output values from input values. We measure the quality of the Machine Learning results with a *loss value*, which is a real number representing how much the predicted values deviate from the correct values from the output data set.

Many approaches in Machine Learning use a parameterizable *model* function to express the mapping from input to output. An *optimizer* feeds data into the model and compares the model's prediction with the correct output to determine the loss value. Based on this loss value, it updates the parameters of the model aiming to produce a better loss value in the next model evaluation. The repeated process of evaluating test data with the model and updating the model's parameters based on the generated loss value is often referred to as the *optimization loop*. The optimization loop of a supervised Machine Learning process is visualized in Figure 2.3.

It should be noted that supervised learning is not the only type of Machine Learning task. *Unsupervised* learning tasks, for example, can be implemented without given output data. In this case, the loss value must be determined from the model output without relying on any given ideal output data. When applying Machine Learning to an optimization problem, the loss value is typically the negative objective value of the output proposed by the model function.

The success of a Machine Learning task depends on the careful selection of training data, loss function, model, and the optimizer. For the optimizer, a common choice is

Gradient Descent. In each optimization iteration, the optimizer computes the gradient of the loss value from the model output w.r.t. the model parameters. As the gradient points towards the steepest ascent, its negative points towards the steepest descent. Altering the parameters in this direction should then decrease the loss value. This parameter update can be expressed with the following equation, where θ_i is the parameter vector after the i-th iteration, E is the loss function, f is the model, (x, y) is the training data, and η is the learning rate.

$$\theta_{i+1} = \theta_i - \eta \nabla E(f_{\theta_i}(x), y) \tag{2.13}$$

2.2.2. Quantum Machine Learning

As Machine Learning can often learn patterns from given datasets even in the presence of noise, it is expected that Machine Learning applications can benefit from the computational power of quantum computers even in the NISQ era [6, 5].

One approach for using quantum computers for Machine Learning is to use PQCs as the model function of an otherwise classical optimization loop. This is referred to as a *hybrid* approach [1, 11].

2.2.3. Optimization techniques

Although gradient-based optimizers are widely used in the field of Machine Learning, other optimizers are available too. For the optimization of PQCs, Ostaszewski et al. present an alternative approach that optimizes model parameters independently [14]. In "*Structure optimization for parameterized quantum circuits*", they observe that the expected loss value $\mathbb{P}(M = |0\rangle)$ of a PQC w.r.t. a single rotational Pauli gate parameter θ always has the following mathematical structure.

$$\mathbb{P}(M = |0\rangle \,|\theta) = d_1 + d_3 \cdot \cos(\theta + d_2) \tag{2.14}$$

Here, $d_1, d_2, d_3 \in \mathbb{R}$ are unknown constants independent of θ. The authors could reconstruct these constants by evaluating the PQC with three select values for θ. As sinusoidal functions are well understood, the minimum of this reconstructed function can be calculated analytically. It is mathematically easy to calculate their minimum once the function is reconstructed.

3. Related work

The successful training of Quantum Machine Learning models relies on choosing an optimization technique that can minimize the loss value effectively (see section 2.2). Most optimizers rely on the computation or approximation of the model's loss gradient w.r.t. the model's parameters. On the other hand, `Rotosolve` optimizes parameters of rotational Pauli gates independently instead. This allows it to minimize these parameters analytically instead of altering parameters in the direction of steepest descent [14]. In this thesis, I present an extension of this method for a second type of parameterized gates (i.e., controlled rotational Pauli gates). Another approach has recently been presented by Wierichs et al..

In "*General parameter-shift rules for quantum gradients*" [20], Wierichs et al. observe that the univariate expected value $E(\theta)$ of a Variational Quantum Algorithm measurement can be expressed as a finite Fourier series.

$$E(\theta) = a_0 + \sum_{l=1}^{R} a_l \cos(\Omega_l \theta) + b_l \sin(\Omega_l \theta)$$

Using a discrete Fourier transform, they computed the coefficients $a_l, b_l, 1 \leq l \leq R$ of this expression to reconstruct $E(\theta)$ from a finite amount of circuit evaluations. This approach is restricted to parameterized quantum circuits that can be expressed as a gate $U(\theta) = e^{i\theta G}$ defined by a Hermitian generator G. The R frequencies $\Omega_l, 1 \leq l \leq R$ can be derived from the eigenvalues of the Hermitian generator G.

To my knowledge, PennyLane is currently the only major quantum computing SDK implementing a variant of the `Rotosolve` algorithm [2, 14]. PennyLane's implementation extends the original `Rotosolve` approach by using Wierichs' general method to reconstruct univariate loss functions. This requires the calculation of the frequency spectrum Ω_l mentioned above prior to the first iteration of the optimizer. Only for parameters corresponding to rotational Pauli gates, PennyLane's `Rotosolve` implementation applies the eponymous analytic methodology from Ostaszewski's `Rotosolve` proposal. The `Crotosolve` optimizer presented in chapter 4 can be considered a special case of PennyLane's `Rotosolve` implementation, which has been optimized for controlled rotational Pauli gates.

4. `Crotosolve`: Gradient-free controlled rotational Pauli gate optimization

In chapter 2, I have presented how Quantum Machine Learning relies on optimizers to gradually minimize the expectation value of a parameterized quantum circuit. While many of these optimizers are gradient-based, the gradient-free `Rotosolve` optimizer by Ostaszewski et al. treats the different parameters of PQCs independently [14]. By reconstructing the univariate loss function of a rotational Pauli gate parameter, the minimizing parameter value can be calculated analytically.

In this chapter, I present `Crotosolve`, a similar method for parameters of controlled rotational Pauli gates. I first show in section 4.1 that the univariate loss curve of a controlled rotational Pauli gate can be described as the sum of two sinusoidal functions with different frequencies. To reproduce the concrete loss curve, the amplitudes of these functions as well as their offsets in x- and y-direction need to be determined. In section 4.3, I show how to determine these constants algorithmically using a minimum number of circuit evaluations. Finally, I outline how to minimize this reconstruction in section 4.4.

4.1. Effect

I analyze the mathematical structure of the loss curve with respect to the parameter θ of a single $CRP(\theta)$ gate. Without loss of generality, I will assume a quantum circuit with only two qubits and only a single $CRP(\theta)$ gate, as shown in Figure 4.1. Before and after the $CRP(\theta)$ gate, the circuit can contain arbitrary gates that do not depend on θ. These gates have been summarized as the U and V gates. Following the gates, a single qubit is

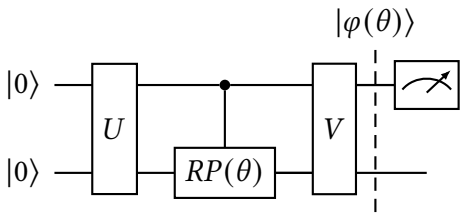


Figure 4.1.: The analyzed quantum circuit is composed of gates U, a controlled rotational Pauli gate $CRP(\theta)$, gates V and a single measurement. $|\varphi(\theta)\rangle$ denotes the quantum state immediately before the measurement.

measured. It should be noted that both the choice of the measured qubit and the choice of the controlling qubit in the $CRP(\theta)$ gate are irrelevant for this analysis, as different choices can be covered with this circuit structure by appending or prepending swap gates to U and V.

The possible outcomes of the measurement are distributed randomly, and this distribution depends on θ since the circuit depends on θ. The probability of measuring $|0\rangle$ in the measured qubit can be computed as the combination of all outcomes' probabilities where the measured qubit is $|0\rangle$.

$$\mathbb{P}(M_0 = |0\rangle \mid \theta) = \mathbb{P}(M = |00\rangle \mid \theta) + \mathbb{P}(M = |01\rangle \mid \theta) \tag{4.1}$$

To resolve this result, I will show in section 4.2 that there are $d_1, \ldots, d_5 \in \mathbb{R}$ so that $\mathbb{P}(M = |\alpha\rangle \mid \theta) = d_1 + d_3 \cos(\theta/2 + d_2) + d_5 \cos(\theta + d_4)$. Using trigonometric identities, Equation 4.1 can be simplified to a term of the same structure [4].

$$\begin{aligned}
\mathbb{P}(M_0 = |0\rangle \mid \theta) &\overset{(4.1)}{=} \mathbb{P}(M = |00\rangle \mid \theta) + \mathbb{P}(M = |01\rangle \mid \theta) \\
&\overset{(4.11)}{=} d_1^{00} + d_3^{00} \cos\left(\theta/2 + d_2^{00}\right) + d_5^{00} \cos\left(\theta + d_4^{00}\right) \\
&\quad + d_1^{01} + d_3^{01} \cos\left(\theta/2 + d_2^{01}\right) + d_5^{01} \cos\left(\theta + d_4^{01}\right) \\
&= d_1 + d_3 \cos(\theta/2 + d_2) + d_5 \cos(\theta + d_4)
\end{aligned} \tag{4.2}$$

4.2. Single outcome probabilities

I will show in this section that there are constants $d_1, \ldots, d_5 \in \mathbb{R}$ so that the probability of a single outcome can be expressed with the following equation.

$$\mathbb{P}(M = |\alpha\rangle \mid \theta) = d_1 + d_3 \cos(\theta/2 + d_2) + d_5 \cos(\theta + d_4) \tag{4.3}$$

To find this equation, I will start with the general equation for the probability of a single measurement and the definition of the $CRP(\theta)$ gate. Successively, I will absorb terms that are independent from θ into constants and combine sin and cos terms using trigonometric identities from [4]. In this context, I will call terms constant if they are independent from θ. Note that these terms are not constant w.r.t. $|\alpha\rangle$.

Let $|\alpha\rangle$ be any state from the basis of the measurement. The probability of the measurement to result in $|\alpha\rangle$ can be computed from its overlap with $|\varphi(\theta)\rangle$, as introduced in section 2.1. Here, $|\varphi(\theta)\rangle = V \cdot CRP(\theta) \cdot U\,|00\rangle$ denotes the quantum state just before the measurement.

$$
\begin{aligned}
\mathbb{P}\left(M = |\alpha\rangle \mid \theta\right) &= |\langle\alpha|\varphi(\theta)\rangle|^2 \\
&= |\langle\alpha| V \cdot CRP(\theta) \cdot U |0\rangle|^2 \\
&= \langle\alpha| V \cdot CRP(\theta) \cdot U |0\rangle \cdot \overline{\langle\alpha| \cdot V \cdot CRP(\theta) \cdot U |0\rangle} \\
&= \underbrace{\langle\alpha| V}_{=:\langle\tilde{\alpha}|} \cdot CRP(\theta) \cdot \underbrace{U |0\rangle \cdot \langle 0| \cdot U^\dagger}_{=:A} \cdot CRP(\theta)^\dagger \cdot \underbrace{V^\dagger |\alpha\rangle}_{=|\tilde{\alpha}\rangle} \\
&= \langle\tilde{\alpha}| CRP(\theta) \cdot A \cdot CRP(-\theta) |\tilde{\alpha}\rangle
\end{aligned}
\tag{4.4}
$$

Inserting the matrix definition of the controlled rotational Pauli gate, this term can be multiplied out further.

$$
\begin{aligned}
\mathbb{P}\left(M = |\alpha\rangle \mid \theta\right) &\overset{(4.4)}{=} \langle\tilde{\alpha}| CRP(\theta) \cdot A \cdot CRP(-\theta) |\tilde{\alpha}\rangle \\
&= \langle\tilde{\alpha}| \left(|0\rangle\langle 0| \otimes I + |1\rangle\langle 1| \otimes RP(\theta)\right) \\
&\quad \cdot A \cdot \left(|0\rangle\langle 0| \otimes I + |1\rangle\langle 1| \otimes RP(-\theta)\right) |\tilde{\alpha}\rangle \\
&= \underbrace{\langle\tilde{\alpha}| \left(|0\rangle\langle 0| \otimes I\right)}_{=:\langle\gamma|} \cdot A \cdot \underbrace{\left(|0\rangle\langle 0| \otimes I\right) |\tilde{\alpha}\rangle}_{=:|\delta\rangle} \\
&\quad + \underbrace{\langle\tilde{\alpha}| \left(|0\rangle\langle 0| \otimes I\right)}_{=\langle\gamma|} \cdot A \cdot \left(|1\rangle\langle 1| \otimes RP(-\theta)\right) |\tilde{\alpha}\rangle \\
&\quad + \langle\tilde{\alpha}| \left(|1\rangle\langle 1| \otimes RP(\theta)\right) \cdot A \underbrace{\left(|0\rangle\langle 0| \otimes I\right) |\tilde{\alpha}\rangle}_{=:|\delta\rangle} \\
&\quad + \langle\tilde{\alpha}| \left(|1\rangle\langle 1| \otimes RP(\theta)\right) \cdot A \cdot \left(|1\rangle\langle 1| \otimes RP(-\theta)\right) |\tilde{\alpha}\rangle \\
&= \langle\gamma| A |\delta\rangle \\
&\quad + \langle\gamma| A \cdot \left(|1\rangle\langle 1| \otimes RP(-\theta)\right) |\tilde{\alpha}\rangle \\
&\quad + \langle\tilde{\alpha}| \left(|1\rangle\langle 1| \otimes RP(\theta)\right) \cdot A |\delta\rangle \\
&\quad + \langle\tilde{\alpha}| \left(|1\rangle\langle 1| \otimes RP(\theta)\right) \cdot A \cdot \left(|1\rangle\langle 1| \otimes RP(-\theta)\right) |\tilde{\alpha}\rangle
\end{aligned}
\tag{4.5}
$$

As $|1\rangle\langle 1| \otimes RP(\theta)$ is a block matrix, any product $\langle x| \left(|1\rangle\langle 1| \otimes RP(\theta)\right) |y\rangle$ can be reduced to $\left\langle x^{\downarrow}\right| RP(\theta) \left|y^{\downarrow}\right\rangle$ where $\left|x^{\downarrow}\right\rangle$ contains the components from $|x\rangle$ corresponding to the non-zero matrix block. The summands in Equation 4.5 can be further simplified by using this reduction and inserting the matrix definition for $RP(\theta)$ gates, which was presented in Equation 2.7. With these transformations, the second summand from Equation 4.5 resolves to a sum of sin and cos functions of θ with frequency $1/2$.

$$
\begin{aligned}
&\langle\gamma|\, A \cdot (|1\rangle\langle 1| \otimes RP(-\theta))\, |\tilde{\alpha}\rangle \\
&= \left\langle \gamma^{\downarrow} \right| A^{\downarrow} \cdot RP(-\theta) \left| \tilde{\alpha}^{\downarrow} \right\rangle \\
&= \left\langle \gamma^{\downarrow} \right| A^{\downarrow} \cdot (\cos(-\theta/2)\, I - i \sin(-\theta/2)\, P) \left| \tilde{\alpha}^{\downarrow} \right\rangle \\
&= \cos(-\theta/2) \cdot \left\langle \gamma^{\downarrow} \right| A^{\downarrow} \cdot I \left| \tilde{\alpha}^{\downarrow} \right\rangle \\
&\quad - i \sin(-\theta/2) \cdot \left\langle \gamma^{\downarrow} \right| A^{\downarrow} \cdot P \left| \tilde{\alpha}^{\downarrow} \right\rangle \\
&= \cos(\theta/2) \cdot \underbrace{\left\langle \gamma^{\downarrow} \right| A^{\downarrow} \cdot I \left| \tilde{\alpha}^{\downarrow} \right\rangle}_{=:c_1} \\
&\quad + \sin(\theta/2) \cdot \underbrace{i \cdot \left\langle \gamma^{\downarrow} \right| A^{\downarrow} \cdot P \left| \tilde{\alpha}^{\downarrow} \right\rangle}_{=:c_2} \\
&= \cos(\theta/2) \cdot c_1 + \sin(\theta/2) \cdot c_2
\end{aligned}
\tag{4.6}
$$

Similarly, the third summand from Equation 4.5 resolves to a sum of sin and cos functions of θ. While both the second and third summands have a similar mathematical shape, their amplitude coefficients c_1, c_2 and c_3, c_4 may be different.

$$
\begin{aligned}
&\langle\tilde{\alpha}|\, (|1\rangle\langle 1| \otimes RP(\theta)) \cdot A\, |\delta\rangle \\
&= \left\langle \tilde{\alpha}^{\downarrow} \right| RP(\theta) \cdot A^{\downarrow} \left| \delta^{\downarrow} \right\rangle \\
&= \left\langle \tilde{\alpha}^{\downarrow} \right| (\cos(\theta/2)\, I - i \sin(\theta/2)\, P) \cdot A^{\downarrow} \left| \delta^{\downarrow} \right\rangle \\
&= \cos(\theta/2) \cdot \left\langle \tilde{\alpha}^{\downarrow} \right| I \cdot A^{\downarrow} \left| \delta^{\downarrow} \right\rangle \\
&\quad - i \sin(\theta/2) \cdot \left\langle \tilde{\alpha}^{\downarrow} \right| P \cdot A^{\downarrow} \left| \delta^{\downarrow} \right\rangle \\
&= \cos(\theta/2) \cdot \underbrace{\left\langle \tilde{\alpha}^{\downarrow} \right| I \cdot A^{\downarrow} \left| \delta^{\downarrow} \right\rangle}_{=:c_3} \\
&\quad + \sin(\theta/2) \cdot \underbrace{(-i) \cdot \left\langle \tilde{\alpha}^{\downarrow} \right| P \cdot A^{\downarrow} \left| \delta^{\downarrow} \right\rangle}_{=:c_4} \\
&= \cos(\theta/2) \cdot c_3 + \sin(\theta/2) \cdot c_4
\end{aligned}
\tag{4.7}
$$

To resolve the fourth term, the same ideas have to be applied multiple times. The summand is left in a form of sin and cos functions, their squares and a product of sin and cos.

$$
\begin{aligned}
&\langle\tilde{\alpha}|\left(|1\rangle\langle 1|\otimes RP\left(\theta\right)\right)\cdot A\cdot\left(|1\rangle\langle 1|\otimes RP\left(-\theta\right)\right)|\tilde{\alpha}\rangle\\
&=\left\langle\tilde{\alpha}^{\downarrow}\right|RP\left(\theta\right)\cdot A^{\downarrow}\cdot RP\left(-\theta\right)\left|\tilde{\alpha}^{\downarrow}\right\rangle\\
&=\left\langle\tilde{\alpha}^{\downarrow}\right|\left(\cos\left(\theta/2\right)I-i\sin\left(\theta/2\right)P\right)\\
&\quad\cdot A^{\downarrow}\cdot\left(\cos\left(-\theta/2\right)I-i\sin\left(-\theta/2\right)P\right)\left|\tilde{\alpha}^{\downarrow}\right\rangle\\
&=\cos\left(\theta/2\right)\cos\left(-\theta/2\right)\left\langle\tilde{\alpha}^{\downarrow}\right|I\cdot A\cdot I\left|\tilde{\alpha}^{\downarrow}\right\rangle\\
&\quad+\cos\left(\theta/2\right)\sin\left(-\theta/2\right)\cdot(-i)\cdot\left\langle\tilde{\alpha}^{\downarrow}\right|I\cdot A\cdot P\left|\tilde{\alpha}^{\downarrow}\right\rangle\\
&\quad+\sin\left(\theta/2\right)\cos\left(-\theta/2\right)\cdot(-i)\cdot\left\langle\tilde{\alpha}^{\downarrow}\right|P\cdot A\cdot I\left|\tilde{\alpha}^{\downarrow}\right\rangle\\
&\quad+\sin\left(\theta/2\right)\sin\left(-\theta/2\right)\cdot(-i)^{2}\cdot\left\langle\tilde{\alpha}^{\downarrow}\right|P\cdot A\cdot P\left|\tilde{\alpha}^{\downarrow}\right\rangle\\
&=\cos\left(\theta/2\right)^{2}\cdot\underbrace{\left\langle\tilde{\alpha}^{\downarrow}\right|A\left|\tilde{\alpha}^{\downarrow}\right\rangle}_{=:c_5}\\
&\quad+\cos\left(\theta/2\right)\sin\left(\theta/2\right)\cdot\underbrace{i\cdot\left\langle\tilde{\alpha}^{\downarrow}\right|A\cdot P\left|\tilde{\alpha}^{\downarrow}\right\rangle}_{=:c_6}\\
&\quad+\sin\left(\theta/2\right)\cos\left(\theta/2\right)\cdot\underbrace{(-i)\cdot\left\langle\tilde{\alpha}^{\downarrow}\right|P\cdot A\left|\tilde{\alpha}^{\downarrow}\right\rangle}_{=:c_7}\\
&\quad+\sin\left(\theta/2\right)^{2}\cdot\underbrace{\left\langle\tilde{\alpha}^{\downarrow}\right|P\cdot A\cdot P\left|\tilde{\alpha}^{\downarrow}\right\rangle}_{=:c_8}\\
&=\cos\left(\theta/2\right)^{2}\cdot c_5+\sin\left(\theta/2\right)^{2}\cdot c_8+\cos\left(\theta/2\right)\sin\left(\theta/2\right)\cdot\left(c_6+c_7\right)
\end{aligned}
\tag{4.8}
$$

With Equations 4.6, 4.7, and 4.8, Equation 4.5 can be expressed as the following.

$$
\begin{aligned}
\mathbb{P}\left(M = |\alpha\rangle \mid \theta\right) &\overset{(4.5)}{=} \underbrace{\langle\gamma| A |\delta\rangle}_{=:c_9} \\
&\quad + \langle\gamma| A \cdot \left(|1\rangle\langle 1| \otimes RP\left(-\theta\right)\right) |\tilde{\alpha}\rangle \\
&\quad + \langle\tilde{\alpha}| \left(|1\rangle\langle 1| \otimes RP\left(\theta\right)\right) \cdot A |\delta\rangle \\
&\quad + \langle\tilde{\alpha}| \left(|1\rangle\langle 1| \otimes RP\left(\theta\right)\right) \cdot A \cdot \left(|1\rangle\langle 1| \otimes RP\left(-\theta\right)\right) |\tilde{\alpha}\rangle \\
&\overset{(4.6)}{\overset{(4.7)}{\overset{(4.8)}{=}}} c_9 \\
&\quad + \cos\left({}^{\theta}/_{2}\right) \cdot c_1 + \sin\left({}^{\theta}/_{2}\right) \cdot c_2 \\
&\quad + \cos\left({}^{\theta}/_{2}\right) \cdot c_3 + \sin\left({}^{\theta}/_{2}\right) \cdot c_4 \\
&\quad + \cos\left({}^{\theta}/_{2}\right)^2 \cdot c_5 + \sin\left({}^{\theta}/_{2}\right)^2 \cdot c_8 \\
&\quad + \cos\left({}^{\theta}/_{2}\right) \sin\left({}^{\theta}/_{2}\right) \cdot \left(c_6 + c_7\right) \\
&= c_9 \\
&\quad + \cos\left({}^{\theta}/_{2}\right) \cdot \left(c_1 + c_3\right) + \sin\left({}^{\theta}/_{2}\right) \cdot \left(c_2 + c_4\right) \\
&\quad + \cos\left({}^{\theta}/_{2}\right)^2 \cdot c_5 + \sin\left({}^{\theta}/_{2}\right)^2 \cdot c_8 \\
&\quad + \cos\left({}^{\theta}/_{2}\right) \sin\left({}^{\theta}/_{2}\right) \cdot \left(c_6 + c_7\right)
\end{aligned}
\tag{4.9}
$$

This equation is composed of many $\sin\left({}^{\theta}/_{2}\right)$ and $\cos\left({}^{\theta}/_{2}\right)$ terms which occur linearly or in quadratic form. These products of $\sin\left({}^{\theta}/_{2}\right)$ and $\cos\left({}^{\theta}/_{2}\right)$ terms can be combined into linear sin and cos with different frequencies. To calculate these product terms, the following trigonometric identities are used [4].

$$\cos^2\left(\theta\right) = \frac{1}{2} + \frac{1}{2}\cos\left(2\theta\right) \tag{4.10a}$$

$$\sin^2\left(\theta\right) = \frac{1}{2} - \frac{1}{2}\cos\left(2\theta\right) \tag{4.10b}$$

$$\cos\left(\theta\right)\sin\left(\psi\right) = \frac{1}{2}\sin\left(\theta + \psi\right) + \frac{1}{2}\sin\left(\theta - \psi\right) \tag{4.10c}$$

$$a\cos x + b\sin x = sgn(a)\sqrt{a^2 + b^2}\cos\left(x + \arctan\left(-\frac{b}{a}\right)\right) \tag{4.10d}$$

Applying these identities to Equation 4.9 leaves the equation for the probability of a single measurement outcome in the desired form.

$$
\begin{aligned}
\mathbb{P}\left(M = |\alpha\rangle \mid \theta\right) &\overset{(4.9)}{=} c_9 \\
&\quad + \cos\left({}^{\theta}/_{2}\right) \cdot (c_1 + c_3) + \sin\left({}^{\theta}/_{2}\right) \cdot (c_2 + c_4) \\
&\quad + \cos\left({}^{\theta}/_{2}\right)^2 \cdot c_5 + \sin\left({}^{\theta}/_{2}\right)^2 \cdot c_8 \\
&\quad + \cos\left({}^{\theta}/_{2}\right) \sin\left({}^{\theta}/_{2}\right) \cdot (c_6 + c_7) \\
&= c_9 \\
&\quad + \cos\left({}^{\theta}/_{2}\right) \cdot (c_1 + c_3) + \sin\left({}^{\theta}/_{2}\right) \cdot (c_2 + c_4) \\
&\quad + \left(\frac{1}{2} + \frac{1}{2}\cos(\theta)\right) \cdot c_5 + \left(\frac{1}{2} - \frac{1}{2}\cos(\theta)\right) \cdot c_8 \\
&\quad + \frac{1}{2}\sin\underbrace{\left(\frac{\theta}{2} + \frac{\theta}{2}\right)}_{=\theta} + \underbrace{\frac{1}{2}\sin\left(\frac{\theta}{2} - \frac{\theta}{2}\right)}_{=0} \\
&= c_9 + \frac{c_5}{2} + \frac{c_8}{2} \\
&\quad + \cos\left({}^{\theta}/_{2}\right) \cdot (c_1 + c_3) + \sin\left({}^{\theta}/_{2}\right) \cdot (c_2 + c_4) \\
&\quad + \cos(\theta) \cdot \left(\frac{c_5}{2} - \frac{c_8}{2}\right) + \sin(\theta) \cdot \frac{1}{2} \\
&= \underbrace{c_9 + \frac{c_5}{2} + \frac{c_8}{2}}_{=:d_1} \\
&\quad + \cos\left({}^{\theta}/_{2} + \underbrace{\arctan\left(-\frac{c_2 + c_4}{c_1 + c_3}\right)}_{=:d_2}\right) \cdot \underbrace{sgn\left(c_1 + c_3\right)\sqrt{(c_1 + c_3)^2 + (c_2 + c_4)^2}}_{=:d_3} \\
&\quad + \cos\left(\theta + \underbrace{\arctan\left(-\frac{\frac{1}{2}}{\frac{c_5}{2} - \frac{c_6}{2}}\right)}_{=:d_4}\right) \cdot \underbrace{sgn\left(\frac{c_5}{2} - \frac{c_6}{2}\right)\sqrt{\left(\frac{c_5}{2} - \frac{c_6}{2}\right)^2 + \left(\frac{1}{2}\right)^2}}_{=:d_5} \\
&= d_1 + \cos\left({}^{\theta}/_{2} + d_2\right) \cdot d_3 + \cos\left(\theta + d_4\right) \cdot d_5
\end{aligned}
\tag{4.11}
$$

4.3. Determining the constants

Equation 4.2 describes the effect of the rotation angle parameter on the expected value of a single qubit measurement. The simple structure of this formula comes at the cost of five unknown constants $d_1, \ldots, d_5$. While it is possible to compute those constants from their definitions in Equation 4.5 - 4.11, this computation is as computationally expensive as simulating the execution of the quantum circuit.

Instead, because of the sinusoidal nature of the function, the constants can be determined through a few evaluations of the quantum circuit. To simplify the syntax, I will use $y(\theta)$ to refer to the total expected value of the measurement and $y_1(\theta), y_2(\theta)$ to refer to the two sinusoidal functions that $y(\theta)$ is composed of.

$$\begin{aligned} y(\theta) &:= \mathbb{P}(M_0 = |0\rangle \mid \theta) \\ &= d_1 + \underbrace{d_3 \cos(\theta/2 + d_2)}_{=:y_1(\theta)} + \underbrace{d_5 \cos(\theta + d_4)}_{=:y_2(\theta)} \\ &= d_1 + y_1(\theta) + y_2(\theta) \end{aligned} \tag{4.12}$$

We can analyze y_1 and y_2 separately by constructing interferences of y with a phase-shifted version of itself. Note that $\cos(\psi + \pi) = -\cos(\psi), \cos(\psi + 2\pi) = \cos(\psi)$.

$$\begin{aligned} y(\theta) + y(\theta + 2\pi) &= d_1 + d_3 \cos(\theta/2 + d_2) + d_5 \cos(\theta + d_4) \\ &\quad + d_1 + d_3 \cos(\theta/2 + d_2 + \pi) + d_5 \cos(\theta + d_4 + 2\pi) \\ &= d_1 + d_3 \cos(\theta/2 + d_2) + d_5 \cos(\theta + d_4) \\ &\quad + d_1 - d_3 \cos(\theta/2 + d_2) + d_5 \cos(\theta + d_4) \\ &= 2d_1 + 2d_5 \cos(\theta + d_4) \\ &= 2d_1 + 2y_2(\theta) \\ \Rightarrow y_2(\theta) &= \frac{1}{2}(y(\theta) + y(\theta + 2\pi) - 2d_1) \end{aligned} \tag{4.13}$$

Again, the interference of this function with itself can be used to eliminate y_2.

$$\begin{aligned} & y(\theta) + y(\theta + \pi) + y(\theta + 2\pi) + y(\theta + 3\pi) \\ &= y(\theta) + y(\theta + 2\pi) + y(\theta + \pi) + y((\theta + \pi) + 2\pi) \\ &\overset{4.13}{=} 2d_1 + 2d_5 \cos(\theta + d_4) + 2d_1 + 2d_5 \cos(\theta + \pi + d_4) \\ &= 2d_1 + 2d_5 \cos(\theta + d_4) + 2d_1 - 2d_5 \cos(\theta + d_4) \\ &= 4d_1 \\ \Rightarrow d_1 &= \frac{1}{4}(y(\theta) + y(\theta + \pi) + y(\theta + 2\pi) + y(\theta + 3\pi)) \end{aligned} \tag{4.14}$$

With d_1, we can now work with y_2 to determine d_4 and $_5$. To do so, we first need to catch an edge case. If $d_5 = 0$, then $y_2(\theta)$ is 0 for all angles θ and d_4 can be chosen arbitrarily. Since sin and cos have distinct zeros, we can check for this condition with two evaluations $y_2(\theta)$ and $y_2(\theta + 3/2\pi)$.

$$d_5 = 0 \quad \Leftrightarrow \quad \bigwedge \begin{cases} 0 = d_5 \cos(\theta + d_4) = y_2(\theta) \\ 0 = d_5 \sin(\theta + d_4) = \cos(\theta + d_4 + 3/2\pi) = y_2(\theta + 3/2\pi) \end{cases} \tag{4.15}$$

If $y_2(\theta) = 0$ and $y_2(\theta + {}^3\!/_2\pi) \neq 0$, we can derive d_4 and d_5 as follows. Note that we restrict the zeros of cos to be ${}^1\!/_2\pi$ or ${}^3\!/_2\pi$ without loss of generality since cos is 2π-periodic. We can further eliminate ${}^3\!/_2\pi$ since choosing ${}^3\!/_2\pi$ over ${}^1\!/_2\pi$ only changes the sign of the function, which can also be chosen through its amplitude d_5.

$$\begin{aligned} 0 = y_2(\theta) &= d_5 \cos(\theta + d_4) \\ \overset{d_5 \neq 0}{\Rightarrow} \quad & \theta + d_4 = {}^1\!/_2\pi \\ \Rightarrow \quad & d_4 = {}^1\!/_2\pi - \theta \end{aligned} \tag{4.16}$$

The corresponding amplitude can then be computed as

$$\begin{aligned} y_2(\theta + {}^3\!/_2\pi) &= d_5 \cos(\theta + d_4 + {}^3\!/_2\pi) \\ &= d_5 \cos({}^1\!/_2\pi + {}^3\!/_2\pi) \\ &= d_5 \cos(2\pi) \\ &= d_5 \,. \end{aligned} \tag{4.17}$$

If $y_2(\theta) \neq 0$, we can use the inverse of the tan function to compute d_4.

$$\begin{aligned} \frac{y_2(\theta + {}^3\!/_2\pi)}{y_2(\theta)} &= \frac{d_5 \cos(\theta + {}^3\!/_2\pi + d_4)}{d_5 \cos(\theta + d_4)} \\ &= \frac{\sin(\theta + d_4)}{\cos(\theta + d_4)} \\ &= \tan(\theta + d_4) \\ \Rightarrow \theta + d_4 &= \arctan\left(\frac{y_2(\theta + {}^3\!/_2\pi)}{y_2(\theta)}\right) \\ \Rightarrow d_4 &= \arctan\left(\frac{y_2(\theta + {}^3\!/_2\pi)}{y_2(\theta)}\right) - \theta \end{aligned} \tag{4.18}$$

The computation of d_5 then needs no additional circuit evaluations.

$$\begin{aligned} y_2(\theta) &= d_5 \cos(\theta + d_4) \\ \Rightarrow d_5 &= \frac{y_2(\theta)}{\cos(\theta + d_4)} \end{aligned} \tag{4.19}$$

A similar approach can be used to determine the remaining constants, d_2 and d_3. Again, we use an interference of y with a phase-shifted version of itself to find an equation with only the missing constants.

$$\begin{aligned} y(\theta) - y(\theta + 2\pi) &= d_1 + d_3 \cos(\theta/2 + d_2) + d_5 \cos(\theta + d_4) \\ &\quad - d_1 - d_3 \cos(\theta/2 + \pi + d_2) - d_5 \cos(\theta + 2\pi + d_4) \\ &= d_1 + d_3 \cos(\theta/2 + d_2) + d_5 \cos(\theta + d_4) \\ &\quad - d_1 + d_3 \cos(\theta/2 + d_2) - d_5 \cos(\theta + d_4) \\ &= 2d_3 \cos(\theta/2 + d_2) \\ &= 2y_1(\theta) \\ \Rightarrow y_1(\theta) &= \frac{1}{2}\left(y(\theta) - y(\theta + 2\pi)\right) \end{aligned} \tag{4.20}$$

Similar to 4.13 to 4.18, the missing constants can be derived from this equation. First, we handle the case where y_1 zeroes out, and d_2 can be chosen arbitrarily.

$$d_3 = 0 \quad \Leftrightarrow \quad \bigwedge \begin{cases} 0 = d_3 \cos(\theta/2 + d_2) = y_1(\theta) \\ 0 = d_3 \sin(\theta/2 + d_2) = d_3 \cos(\theta/2 + d_2 + 3/2\pi) = y_1(\theta + 3\pi) \end{cases} \tag{4.21}$$

Second, if $y_1(\theta) = 0$ but $y_1(\theta + 3\pi) \neq 0$, d_4 and d_5 can be determined by choosing $\theta/2 + d_2$ as a zero point of cos.

$$\begin{aligned} 0 = y_1(\theta) &= d_3 \cos(\theta/2 + d_2) \\ \overset{d_3 \neq 0}{\Rightarrow} 1/2\pi &= \theta/2 + d_2 \\ \Rightarrow 1/2\pi - \theta/2 &= d_2 \end{aligned} \tag{4.22}$$

$$\begin{aligned} y_1(\theta + 3\pi) &= d_3 \cos(\theta/2 + d_2 + 3/2\pi) \\ &= d_3 \cos(1/2\pi + 3/2\pi) \\ &= d_3 \cos(2\pi) \\ &= d_3 \end{aligned} \tag{4.23}$$

And third, we can determine d_2 and d_3 with the use of tan's inverse if $y_1(\theta) \neq 0$.

$$\begin{aligned} \frac{y_1(\theta + 3\pi)}{y_1(\theta)} &= \frac{d_3 \cos(\theta/2 + 3/2\pi + d_2)}{d_3 \cos(\theta/2 + d_2)} \\ &= \frac{\sin(\theta/2 + d_2)}{\cos(\theta/2 + d_2)} \\ &= \tan(\theta/2 + d_2) \\ \Rightarrow \theta/2 + d_2 &= \arctan\left(\frac{y_1(\theta + 3\pi)}{y_1(\theta)}\right) \\ \Rightarrow d_2 &= \arctan\left(\frac{y_1(\theta + 3\pi)}{y_1(\theta)}\right) - \theta/2 \end{aligned} \tag{4.24}$$

$$\begin{aligned} y_1(\theta) &= d_3 \cos({}^{\theta}\!/_{2} + d_2) \\ \Rightarrow d_3 &= \frac{y_1(\theta)}{\cos({}^{\theta}\!/_{2} + d_2)} \end{aligned} \tag{4.25}$$

Therefore, the constants $d_1, \ldots, d_5$ can be determined with a total of six quantum circuit evaluations $y(\theta), y(\theta+\pi), y(\theta+{}^{3}\!/_{2}\pi), y(\theta+2\pi), y(\theta+3\pi)$ and $y(\theta+{}^{7}\!/_{2}\pi)$. For convenience, they are summarized in the following. In cases where the loss function is independent of a parameter d_i, the arbitrary choice has been denoted with an asterisk $*$.

$$d_1 = \frac{1}{4}(y(\theta) + y(\theta+2\pi) + y(\theta+\pi) + y(\theta+3\pi)) \tag{4.26a}$$

$$d_2 = \begin{cases} *, & \text{if } y_1(\theta) = 0 = y_1(\theta+3\pi) \\ {}^{1}\!/_{2}\pi - {}^{\theta}\!/_{2}, & \text{if } y_1(\theta) = 0 \neq y_1(\theta+3\pi) \\ \arctan\left(\frac{y_1(\theta+3\pi)}{y_1(\theta)}\right) - {}^{\theta}\!/_{2}, & \text{if } y_1(\theta) \neq 0 \end{cases} \tag{4.26b}$$

$$d_3 = \begin{cases} 0, & \text{if } y_1(\theta) = 0 = y_1(\theta+3\pi) \\ y_1(\theta+3\pi), & \text{if } y_1(\theta) = 0 \neq y_1(\theta+3\pi) \\ \frac{y_1(\theta)}{\cos({}^{\theta}\!/_{2}+d_2)}, & \text{if } y_1(\theta) \neq 0 \end{cases} \tag{4.26c}$$

$$d_4 = \begin{cases} *, & \text{if } y_2(\theta) = 0 = y_2(\theta+{}^{3}\!/_{2}\pi) \\ {}^{1}\!/_{2}\pi - \theta, & \text{if } y_2(\theta) = 0 \neq y_2(\theta+{}^{3}\!/_{2}\pi) \\ \arctan\left(\frac{y_2(\theta+{}^{3}\!/_{2}\pi)}{y_2(\theta)}\right) - \theta, & \text{if } y_2(\theta) \neq 0 \end{cases} \tag{4.26d}$$

$$d_5 = \begin{cases} 0, & \text{if } y_2(\theta) = 0 = y_2(\theta+{}^{3}\!/_{2}\pi) \\ y_2(\theta+{}^{3}\!/_{2}\pi), & \text{if } y_2(\theta) = 0 \neq y_2(\theta+{}^{3}\!/_{2}\pi) \\ \frac{y_2(\theta)}{\cos(\theta+d_4)}, & \text{if } y_2(\theta) \neq 0 \end{cases} \tag{4.26e}$$

with

$$y_1(\theta) = \frac{1}{2}(y(\theta) - y(\theta+2\pi)) \tag{4.27a}$$

$$y_1(\theta+3\pi) = \frac{1}{2}(y(\theta+3\pi) - y(\theta+\pi)) \tag{4.27b}$$

$$y_2(\theta) = \frac{1}{2}(y(\theta) + y(\theta+2\pi) - 2d_1) \tag{4.27c}$$

$$y_2(\theta+{}^{3}\!/_{2}\pi) = \frac{1}{2}(y(\theta+{}^{3}\!/_{2}\pi) + y(\theta+{}^{7}\!/_{2}\pi) - 2d_1)\,. \tag{4.27d}$$

4.4. Minimizing the reconstruction

The previous sections have shown that the effect of a controlled rotational Pauli gate parameter on the expected value of a circuit measurement can be expressed as a sum of two sinusoidal functions. `Crotosolve` can use this cost-efficient reconstruction to optimize a single CRP gate parameter. Therefore, the optimizer needs to find

$$\theta_{min} = \underset{\theta \in [0, 4\pi]}{\operatorname{argmin}} \, y(\theta) \, . \tag{4.28}$$

Since $y(\theta)$ is a univariate, real, and smooth function, a simple minimization strategy is to use a numerical optimizer like the `Nelder-Mead` method [12]. As a sum of two sinusoidal functions with frequencies 1 and $1/2$, $y(\theta)$ is 4π-periodic and has either one or two local minima[1] within the $[0, 4\pi]$ bounds. Thus, it is possible that the numerical optimizer converges to a non-global minimum. The chances of running into the global minimum can be increased by providing an estimate of θ_{min} as the initial value of the minimizer.

The calculation of such an estimate could be investigated in future work. Although it is without proof or evidence from substantial experiments, I want to present a characteristic of the $y(\theta)$ function that I have observed to be a helpful initial value for the minimization. Within the boundaries of $[0, 4\pi]$, $y_1(\theta)$ has a single minimum θ_1 and $y_2(\theta)$ has two equal minima θ_{2a}, θ_{2b}. Let θ_2 be the value of θ_{2a} or θ_{2b} that is the closest to θ_1 and let θ^*_{min} be in the middle of θ_1 and θ_2. Then, the global minimum of $y(\theta)$ is between θ_1 and θ_2 and choosing θ^*_{min} as an initial value allows the numerical minimizer to converge towards the global minimum θ_{min}. Here, the periodicity of $y(\theta)$ needs to be considered in the calculation of the distance and the middle of two values. Otherwise, a global minimum near $\theta = 0$ may not be recognized.

[1] If y is constant, the number of minima is infinite. In this case, however, θ is irrelevant and any minimization result is optimal.

5. Evaluation

To evaluate the quality of the algorithm proposed in chapter 4, I have implemented a proof-of-concept optimizer. I will compare `Crotosolve` to several other optimizers frequently used in QML by creating a loss curve benchmark for all optimizers. The benchmark features parameterizable quantum circuits with various degrees of expressibility and from various research applications.

5.1. Proof of Concept

To put the approach proposed in chapter 4 to test, I have implemented a proof-of-concept application with PennyLane [2]. PennyLane is a popular quantum computing SDK with extensive documentation and a library of readily implemented optimizers [19]. Particularly, PennyLane is currently the only quantum SDK to implement the `Rotosolve` optimizer, to which I want to compare my `Crotosolve` optimizer in section 5.2.

Like the `Rotosolve` algorithm, `Crotosolve` optimizes each parameter individually. For each parameter corresponding to a controlled rotational Pauli gate, `Crotosolve` first reconstructs the univariate cost function (see section 4.3) and then uses a numerical optimizer to find the minimizing parameter value (see section 4.4). Each parameter corresponding to an uncontrolled rotational Pauli gate can be optimized following the exact approach presented by Ostaszewski et al. [14]. The pseudocode for this algorithm is stated in algorithm 1.

This algorithm allows the optimization of parameterized quantum circuits where all parameterized gates are either RP or CRP gates. Furthermore, it is assumed that all gate parameters are used without preprocessing (e.g., having a $RP(x^2)$ gate for a parameter x) in only a single gate. The algorithm needs to know which parameters belong to CRP and which belong to RP gates. Technically, the optimization approach presented for CRP gates could also be used for RP gates since a CRP gate's loss function structure is a generalization of an RP gate's loss function structure. However, using the CRP optimization approach for RP gates would double the number of circuit evaluations required to reconstruct its loss function. While it is trivial to see if a parameter is used in a CRP or RP gate on paper, looking up this relationship in a given PennyLane circuit is non-trivial, although certainly possible. Thus, the proof-of-concept implementation always stores the RP and CRP gate parameters as separate arrays.

Algorithm 1: The `Crotosolve` algorithm updates parameters individually

Data: $prev\_params \in \mathbb{R}^n$, $circuit \in$ QNode
Result: $params \in \mathbb{R}^n$

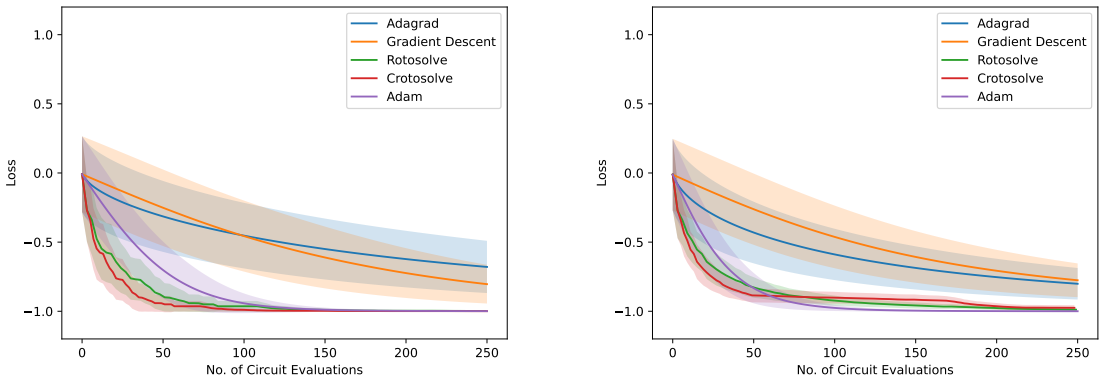
```
params ← copy(prev_params);
for p ← 1 to n do
    uni ← MakeUnivariate(circuit, params, p);
    if p belongs to controlled gate then
        recon ← ReconstructCrp(uni) ;                    /* see section 4.3 */
        params[p] ← argmin_{x∈[0,4π]} recon(x) ;   /* using numerical minimizer */
    else
        params[p] ← RotosolveUpdate(uni);
    end
end
```

5.2. Optimizer comparison

I will examine `Crotosolve`'s performance by comparing its loss curve with the loss curves of other optimizers frequently used in the field of QML. This comparison includes the Gradient Descent family of optimizers, namely standard gradient descent with a fixed learning rate, `Adam` [10] and `Adagrad` [7]. Additionally, I will test my `Crotosolve` implementation against PennyLane's implementation of the `Rotosolve` algorithm [14, 2], which is extended by Wierichs' paper on "*General parameter-shift rules for quantum gradients*" to support further types of gates [20].

To get meaningful results, I chose to run these optimizers on well-known and widely-used quantum circuits. Specifically, I use the set of circuit templates presented in a paper on "Expressibility and Entangling Capability [...]" metrics by Sukin Sim et al. [17]. As shown in this paper, these circuits have various degrees of expressibility and entangling capability.

The optimizers are compared through loss curves generated from exemplary circuits. These loss curves are recorded with respect to the number of circuit evaluations, even though it would be more common in classical Machine Learning to use the number of iterations or taken time instead of the evaluations. The time taken to run these quantum circuits on classical hardware using state-vector simulation is hardly representative of the actual runtime on quantum devices since this simulation task is proven to be inefficient. Additionally, the optimizers benefit to varying degrees from optimizations made in the simulation. For example, gradient-based optimizers benefit from PennyLane's gradient evaluation optimization, which is unavailable on real quantum hardware. Meanwhile, gradient-free optimizers like `Crotosolve` and `Rotosolve` cannot take advantage of these simulation tricks. On the other hand, iterations put `Crotosolve` and `Rotosolve` in favor since each of their iterations encompasses many univariate optimizations. Circuit evaluations, however, are proportional to the execution time on real quantum hardware as long as the classical work required between evaluations is negligible.



(a) Circuit 11

(b) Circuit 14

Figure 5.1.: Various optimizers minimize the expectation of two circuits from [17] with four qubits and three layers. Full lines show the average loss curve. Transparent areas show the error, which is the average ± its standard deviation.

I have generated 100 random initial parameter sets for each circuit to reduce statistical noise through averaging. Each pair of circuits and parameter values is run with each optimizer to record the optimizer's loss curve as a result. These results are then aggregated in average loss curves for each pair of circuits and optimizers.

Figure 5.1 shows the average loss curves for each optimizer on circuits 11 and 14. I have selected these charts as they show all the key characteristics described in the following, but the charts for all other evaluated circuits can be found in Appendix A. Furthermore, the circuit and initial values data for all evaluated circuits, as well as the recorded loss curve datasets are available on GitHub [16].

Results

`Crotosolve`'s loss curve is generally very similar to `Rotosolve`'s loss curve, although `Crotosolve` initially descends towards the minimum value slightly quicker. For RP gates, this is easy to explain since both algorithms are identical. For CRP gates, both algorithms use the same strategy, although different methods to achieve their common goal. PennyLane's implementation of the `Rotosolve` algorithm proposed by Ostaszewski et al. is augmented with the work of Wierichs et al. [20, 2]. This generalization allows the algorithm to reconstruct the univariate loss function from almost any type of gate (see chapter 3). While both `Crotosolve` and `Rotosolve` cache an evaluation value between RP gate optimization steps, only `Crotosolve` does so for CRP gates too. This enhanced caching ensures that `Crotosolve`'s loss curve is always slightly better than `Rotosolve`'s.

`Crotosolve`'s loss curves are also well below loss curves from gradient-based optimizers in most instances. While `Crotosolve`, `Rotosolve` and `Adam` typically approach the minimum value, -1, within the first 250 circuit evaluations, `Stochastic Gradient Descent` and

Adagrad's loss curves progress much slower towards this value. In direct comparison with Adam, Crotosolve's loss curve typically descends much quicker in the beginning and reaches a point close to the minimum much more quickly. Still, Adam consistently reaches final loss values as low as Crotosolve's within the shown 250 iterations. Adam can even reach lower final values in some cases. The cases in which Adam outperforms Crotosolve match the cases in which circuit templates with many parameters have been used (i.e., circuit templates 05, 06, 13, and 14; see Appendix A). However, further investigation would be required to confirm this correlation between the number of parameters and optimizer performance.

It is worth noting that the loss curves of both Rotosolve and Crotosolve have a low variance throughout the optimization process in all evaluated instances. This means that the quick descent to the minimum value happens consistently across the evaluated instances. On the other hand, the gradient-based optimizers show a much higher variance in these experiments. More specifically, Stochastic Gradient Descent and Adagrad typically demonstrate a standard deviation of more than 0.2 throughout the entire optimization process. Meanwhile, Adam shows a similar standard deviation in the beginning and quickly progresses towards a standard deviation near zero as it approaches the 250 iterations.

5.3. Discussion

As demonstrated in this evaluation, Crotosolve can outperform gradient-based methods consistently on the circuits considered here. However, it should be noted that the gradient-based optimizers have been used with their default hyperparameter values for this evaluation. In many cases, achieving good results with Machine Learning optimizers requires elaborate hyper-parameter tweaking. While this suggests that these optimizers could potentially yield better results with hyper-parameter tweaking, it also demonstrates that Crotosolve produces competitive and consistent results without relying on hyper-parameter settings.

While Crotosolve shows slightly better results, PennyLane's Rotosolve implementation is more general due to Wierichs' work on the reconstruction of arbitrary univariate loss functions. This generality enables two use cases that are not covered by Crotosolve. First, it allows Rotosolve to optimizer parameters used in gate types not explicitly covered in the optimizer implementation, whereas Crotosolve requires specific implementations for each parameterized gate type. Second, it enables Rotosolve to optimize parameters used in multiple gates. While this is uncommon in parameterized quantum circuits designed to be used with one parameter per gate, other variational algorithms like QAOA use parameters to configure whole groups of gates. However, this generality comes at the cost of having to compute the frequency spectrum for each parameter. This is computationally expensive but still polynomial. It is sufficient to compute these frequencies once before the first Rotosolve iteration.

6. Conclusion

In this thesis, I have presented a generalization of Ostaszewski's `Rotosolve` algorithm to controlled rotational Pauli gates. Through mathematical analysis, I have characterized the structure of a measurement expectation with respect to a CRP gate parameter. I have demonstrated how targeted evaluations of the circuit with special parameter values can be used to reconstruct this function.

Using this technique, I have implemented a proof-of-concept optimizer for parameterized quantum circuits with parameterized rotational Pauli gates and parameterized controlled rotational Pauli gates. In a benchmark with circuit templates [17] commonly used in the field, I have shown that my `Crotosolve` implementation consistently converges quicker towards an optimal value than other optimizers.

Meanwhile, PennyLane's `Rotosolve` implementation uses results from Wierichs et al. [20] to reconstruct univariate loss functions of arbitrary PQCs. It can handle PQCs where multiple gates have a shared parameter and does not rely on specialized implementations for different types of parameterized gates. While PennyLane's `Rotosolve` implementation is more general than the `Crotosolve` extension, `Crotosolve`'s loss curves are consistently below theirs.

As `Rotosolve` and `Crotosolve` show competitive results compared to gradient-based optimizers, their application should continue to be investigated. Future work on the approach presented in this thesis could extend the optimizer to support more types of PQCs. Three extensions come to mind: 1) the extension to support parameterized gates with more than two qubits, for example the Deutsch gate $CCU(\theta)$, 2) the extension to PQCs where multiple gates may be parameterized by the same parameter, and 3) the extension to support gates that preprocess their parameter, for example $CRP(5\theta^2)$. An actual implementation of `Crotosolve` could also be improved by removing the need to separate parameters for *RP* and *CRP* gates. To remove this separation, the implementation needs to detect which parameter belongs to which type of gate. Furthermore, `Crotosolve` could benefit from further work on the minimization of CRP loss functions. As pointed out in section 4.4, the specific knowledge about CRP loss functions could be used to compute the minimum value analytically or to set a better initial value for the numerical minimizer.

In PennyLane's `Rotosolve` implementation, many of these improvements have already been incorporated through the use of general parameter-shift rules [20]. Thus, future work could combine Wierichs' general reconstruction approach with the optimizations presented in this thesis.

Bibliography

[1] Marcello Benedetti et al. "Parameterized quantum circuits as machine learning models". In: *Quantum Science and Technology* 4.4 (Nov. 2019). Publisher: IOP Publishing, p. 043001. ISSN: 2058-9565. DOI: 10.1088/2058-9565/ab4eb5.

[2] Ville Bergholm et al. *PennyLane: Automatic differentiation of hybrid quantum-classical computations*. 2018. DOI: 10.48550/arXiv.1811.04968.

[3] Christopher M. Bishop. *Pattern recognition and machine learning*. Information science and statistics. New York, NY: Springer, 2006. ISBN: 978-0-387-31073-2. URL: https://link.springer.com/book/9780387310732.

[4] Il'ja N. Bronštejn. *Taschenbuch der Mathematik*. Ed. by Konstantin A. Semendjaev, Gerhard Musiol, and Heiner Mühlig. 10., überarbeitete Auflage. Edition Harri Deutsch. Haan-Gruiten: Verlag Europa-Lehrmittel - Nourney, Vollmer GmbH & Co. KG, 2016. ISBN: 978-3-8085-5790-7. URL: https://d-nb.info/1081907711.

[5] M. Cerezo et al. "Variational quantum algorithms". In: *Nature Reviews Physics* 3.9 (Sept. 1, 2021), pp. 625–644. ISSN: 2522-5820. DOI: 10.1038/s42254-021-00348-9.

[6] Carlo Ciliberto et al. "Quantum machine learning: a classical perspective". In: *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* 474.2209 (2018). DOI: 10.1098/rspa.2017.0551.

[7] John Duchi, Elad Hazan, and Yoram Singer. "Adaptive Subgradient Methods for Online Learning and Stochastic Optimization". In: *J. Mach. Learn. Res.* 12 (July 2011), pp. 2121–2159. ISSN: 1532-4435.

[8] Jack D. Hidary. *Quantum Computing: An Applied Approach*. Second edition. Cham, Switzerland: Springer, 2021. ISBN: 978-3-030-83273-5. URL: https://doi.org/10.1007/978-3-030-83274-2.

[9] M. I. Jordan and T. M. Mitchell. "Machine learning: Trends, perspectives, and prospects". In: *Science* 349.6245 (2015), pp. 255–260. DOI: 10.1126/science.aaa8415.

[10] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2017. DOI: 10.48550/arXiv.1412.6980.

[11] K. Mitarai et al. "Quantum circuit learning". In: *Physical Review A* 98.3 (Sept. 2018). Publisher: American Physical Society (APS). DOI: 10.1103/physreva.98.032309.

[12] J. A. Nelder and R. Mead. "A Simplex Method for Function Minimization". In: *The Computer Journal* 7.4 (Jan. 1965), pp. 308–313. ISSN: 0010-4620. DOI: 10.1093/comjnl/7.4.308.

[13] Michael A. Nielsen and Isaac L. Chuang. *Quantum computation and quantum information*. 9th ed. Cambridge: Cambridge Univ. Press, 2007. ISBN: 978-0-521-63503-5. URL: https://www.cambridge.org/9780521635035.

[14] Mateusz Ostaszewski, Edward Grant, and Marcello Benedetti. "Structure optimization for parameterized quantum circuits". In: *Quantum* 5 (Jan. 2021). Publisher: Verein zur Förderung des Open Access Publizierens in den Quantenwissenschaften, p. 391. ISSN: 2521-327X. DOI: 10.22331/q-2021-01-28-391.

[15] John Preskill. "Quantum Computing in the NISQ era and beyond". In: *Quantum* 2 (Aug. 2018). Publisher: Verein zur Forderung des Open Access Publizierens in den Quantenwissenschaften, p. 79. ISSN: 2521-327X. DOI: 10.22331/q-2018-08-06-79.

[16] Max Schweikart. *schweikart/crotosolve*. Version v0.1.0. Dec. 2023. URL: https://doi.org/10.5281/zenodo.10413938.

[17] Sukin Sim, Peter D. Johnson, and Alán Aspuru-Guzik. "Expressibility and Entangling Capability of Parameterized Quantum Circuits for Hybrid Quantum-Classical Algorithms". In: *Advanced Quantum Technologies* 2.12 (2019). DOI: https://doi.org/10.1002/qute.201900070. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/qute.201900070.

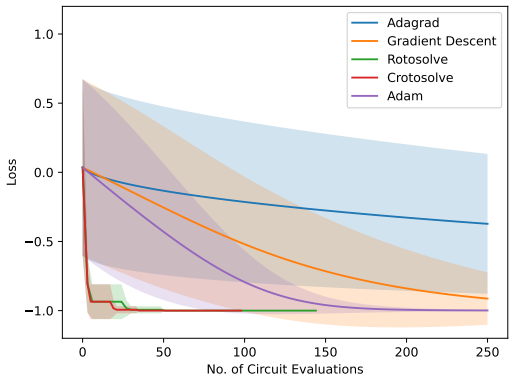
[18] J.C. Spall. "Multivariate stochastic approximation using a simultaneous perturbation gradient approximation". In: *IEEE Transactions on Automatic Control* 37.3 (1992), pp. 332–341. DOI: 10.1109/9.119632.

[19] Unitary Fund Team. *Results Are in for the 2022 Quantum Open Source Software Survey!* Unitary Fund Blog. Nov. 7, 2022. URL: https://unitary.fund/posts/2022_survey_results/ (visited on 11/26/2023).

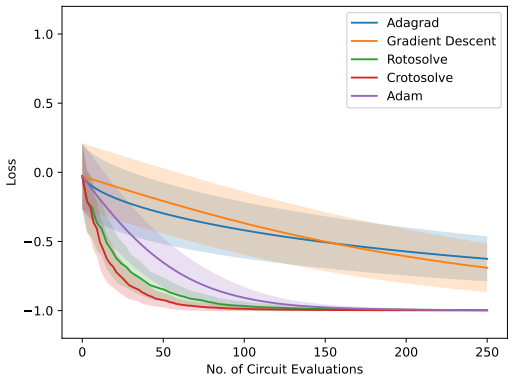
[20] David Wierichs et al. "General parameter-shift rules for quantum gradients". In: *Quantum* 6 (Mar. 2022). Publisher: Verein zur Forderung des Open Access Publizierens in den Quantenwissenschaften, p. 677. DOI: 10.22331/q-2022-03-30-677.

A. Appendix

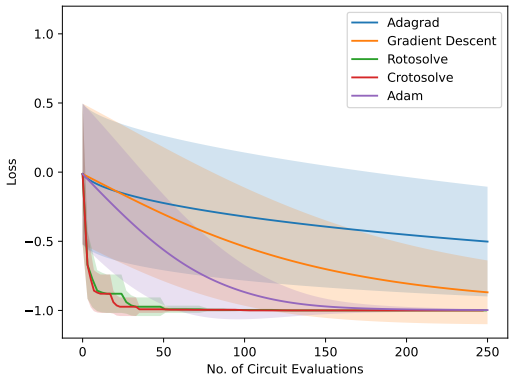
In section 5.2, I presented a characterization of the `Crotosolve` loss curve in comparison to other optimizers used in Quantum Machine Learning. While the analysis was illustrated with a single example, this section contains the loss curves of further PQCs. The circuit ids referenced in the captions refer to the circuit template numbers from [17]. All tests use circuits with four qubits and three template layers. The source code for the dataset generation and visualization is available with the `Crotosolve` implementation [16].



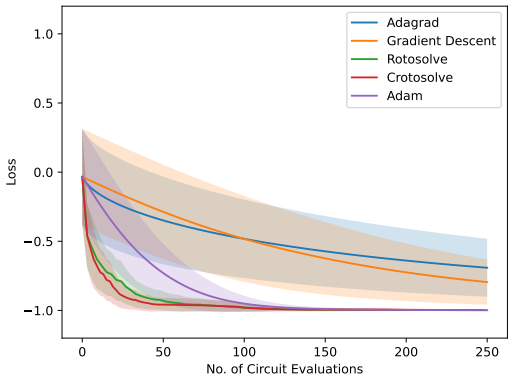
(a) Circuit 01 with 4 qubits and 3 layers



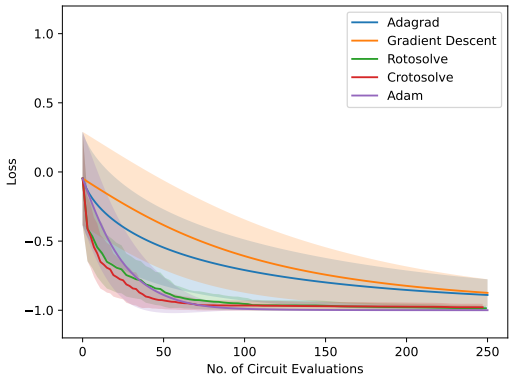
(b) Circuit 02 with 4 qubits and 3 layers



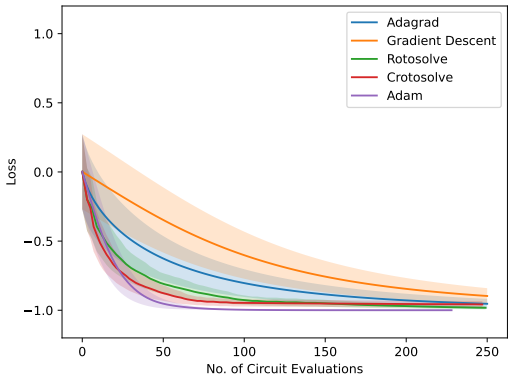
(c) Circuit 03 with 4 qubits and 3 layers



(d) Circuit 04 with 4 qubits and 3 layers

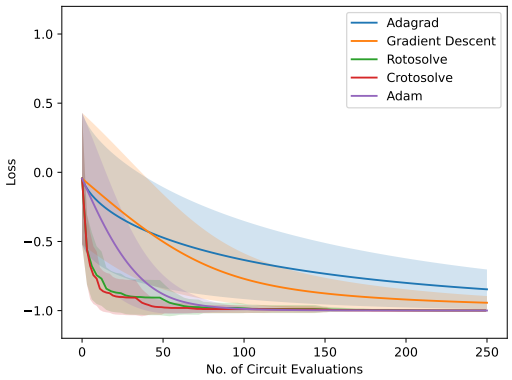


(e) Circuit 05 with 4 qubits and 3 layers

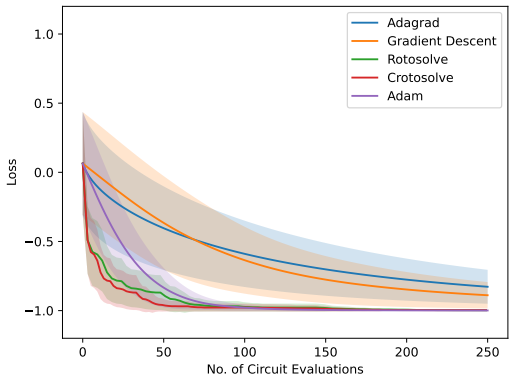


(f) Circuit 06 with 4 qubits and 3 layers

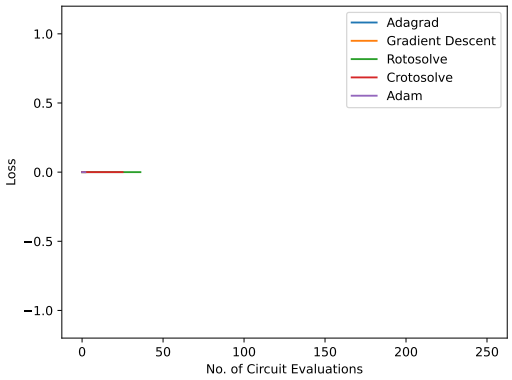
Figure A.1.: Complete evaluation results



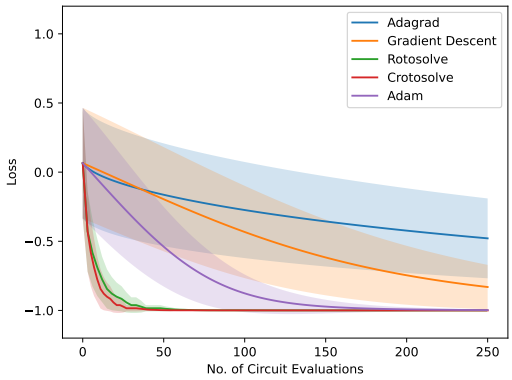
(g) Circuit 07 with 4 qubits and 3 layers



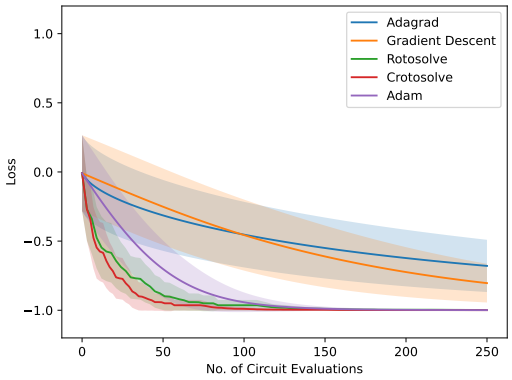
(h) Circuit 08 with 4 qubits and 3 layers



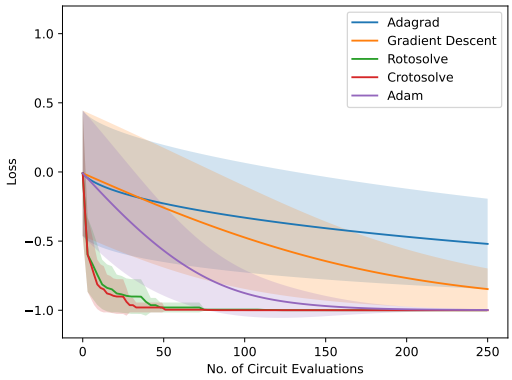
(i) Circuit 09 with 4 qubits and 3 layers



(j) Circuit 10 with 4 qubits and 3 layers

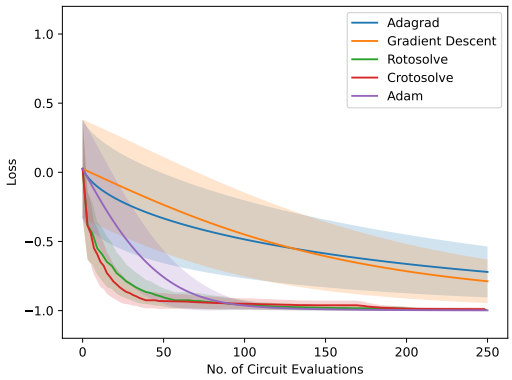


(k) Circuit 11 with 4 qubits and 3 layers

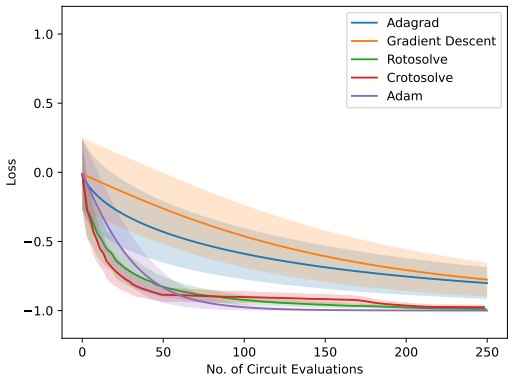


(l) Circuit 12 with 4 qubits and 3 layers

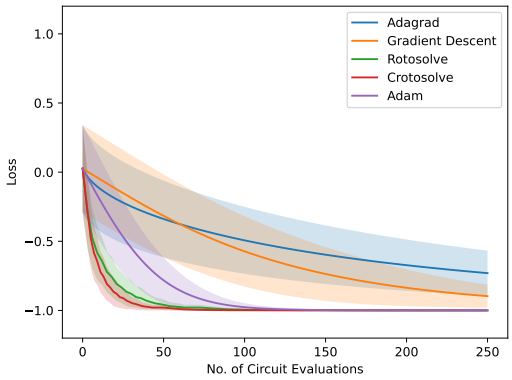
Figure A.1.: Complete evaluation results *(cont.)*



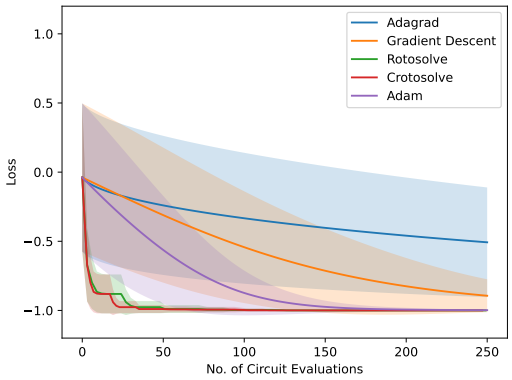
(m) Circuit 13 with 4 qubits and 3 layers



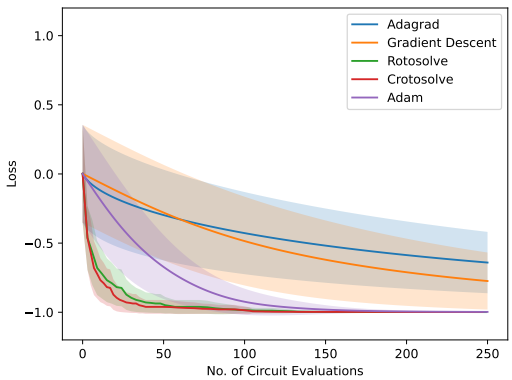
(n) Circuit 14 with 4 qubits and 3 layers



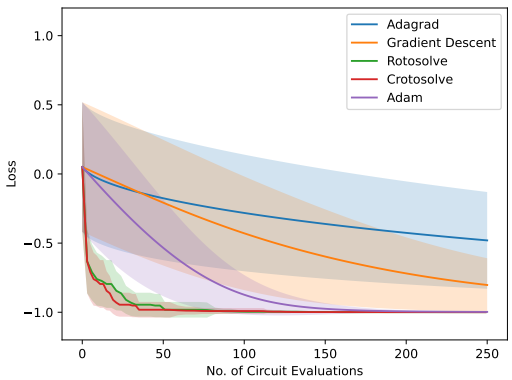
(o) Circuit 15 with 4 qubits and 3 layers



(p) Circuit 16 with 4 qubits and 3 layers

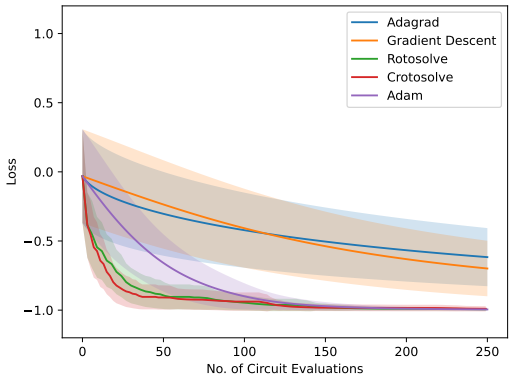


(q) Circuit 17 with 4 qubits and 3 layers



(r) Circuit 18 with 4 qubits and 3 layers

Figure A.1.: Complete evaluation results *(cont.)*



(s) Circuit 19 with 4 qubits and 3 layers

Figure A.1.: Complete evaluation results *(cont.)*