Quantum Machine Learning using the ZXW-Calculus

Mark Koch

Lady Margaret Hall

University of Oxford

A thesis submitted for the degree of

Master of Science in Advanced Computer Science

Trinity 2022

Word count: $15,793$

Diagram count: 806

The word count was calculated using `texcount` via `perl texcount.pl -1 thesis.tex`. Note that in diagram equations, each step is counted as a separate diagram.

Abstract

The field of *quantum machine learning* (QML) explores how quantum computers can be used to more efficiently solve machine learning problems. As an application of hybrid quantum-classical algorithms, it promises a potential quantum advantages in the near term. In this thesis, we use the ZXW-calculus to *diagrammatically* analyse two key problems that QML applications face.

First, we discuss algorithms to compute gradients on quantum hardware that are needed to perform gradient-based optimisation for QML. Concretely, we give new diagrammatic proofs of the common 2- and 4-term parameter shift rules used in the literature. Additionally, we derive a novel, generalised parameter shift rule with $2n$ terms that is applicable to gates that can be represented with n parametrised spiders in the ZXW-calculus. Furthermore, to the best of our knowledge, we give the first proof of a conjecture by Anselmetti et al. by proving a no-go theorem ruling out more efficient alternatives to the 4-term shift rule.

Secondly, we analyse the gradient landscape of quantum ansätze for barren plateaus using both empirical and analytical techniques. Concretely, we develop a tool that automatically calculates the variance of gradients and use it to detect likely barren plateaus in commonly used quantum ansätze. Furthermore, we formally prove the existence or absence of barren plateaus for a selection of ansätze using diagrammatic techniques from the ZXW-calculus.

Acknowledgements

Contents

Chapter 1

Introduction

It is widely believed that quantum computers are capable of solving certain computational problems that are intractable for classical computers. While this potential quantum advantage was already recognised in the 1980s, the quantum devices available today still lack the scale and reliability to tackle many practical problems, with anticipated algorithms like Grover's search [1] or Shor's factorisation algorithm [2] remaining out of reach. Because of those limitations, there is increasing interest in *hybrid quantum-classical algorithms*. The rationale behind hybrid approaches is that the required quantum resources can be significantly reduced by implementing some subroutines on classical hardware. As a result, those algorithms are runnable on the *noisy intermediate-scale quantum* (NISQ) devices available today.

One area where hybrid algorithms promise a quantum advantage is the field of *machine learning* (ML). Roughly, ML is concerned with recognising and generalising patterns in statistical data. It has been shown that even relatively small quantum circuits can represent functions that are highly complex and difficult to express via classical means [3]. Hence, the hope is that quantum computers can capture certain data patterns more efficiently than classical computers, yielding a quantum advantage in ML. This field of study is commonly referred to as *quantum machine learning* (QML) [4].

Typically, hybrid QML algorithms rely on *parametrised quantum circuits*, i.e. circuits that depend on some tunable parameters. An optimisation algorithm running on a classical computer is used to find a parameter assignment such that the output of the quantum circuit minimises some cost function. For example, circuits can be trained to solve ML tasks like classification, regression, or generative modelling. There are many classical optimisation techniques that can be used to train quantum circuits. In the field of QML, one commonly uses gradient-based techniques like gradient descent, which have already been very successfully used in classical ML, especially for the training of neural networks. Notably, gradient-based methods have also been proven to improve convergence in the quantum domain [5]. However, compared to classical neural networks, training quantum circuits using gradient descent comes with a set of unique challenges.

First, one has to determine the gradient of parametrised circuits, i.e. compute how the output of a circuit changes when the parameters are altered. As it turns out, it is not feasible to perform this computation classically. Instead, gradients need to be evaluated on the quantum device itself. The quantum algorithms used for those gradient computations are called *gradient recipes* and are subject to a lot of research interest [6, 7, 8, 9]. Secondly, it has been shown that the gradient landscape of many quantum circuits is not amenable to learning. Concretely, the landscape is often exponentially flat [10], making gradient descent difficult or even impossible. Naturally, there is a lot of interest in determining which circuits exhibit those so-called *barren plateaus* [11, 12, 13].

This thesis is concerned with analysing both of these problems using *diagrammatic* means. The ZX-calculus [14] is a graphical language for reasoning about quantum computation that has been successfully applied to a wide range of tasks in the quantum domain, including circuit optimisation [15], compilation [16], and simulation [17]. The ZXW-calculus [18] is a variant of ZX that has recently been used to diagrammatically represent gradients and integrals [19]. Thus, it is particularly well-suited for our diagrammatic analysis of gradient based optimisation for QML.

1.1 Main Contributions

Below are the main contributions of this thesis with regard to gradient recipes:

- We derive a simplified version of Wang and Yeung's diagrammatic differentiation [19] for the special case of parametrised circuits (Theorem 3.6).

- We give a diagrammatic proof of the most general version of Schuld et al.'s [7] two-term parameter shift rule (Theorem 4.8) and Anselmetti et al.'s [8] four-term shift rule (Theorem 4.11).

- We derive a novel generalised $2n$-term shift rule for gates that can be represented with n parametrised spiders (Theorem 4.13)

- To the best of our knowledge, we give the first proof of a conjecture by Anselmetti et al. [8] showing that their shift rule is optimal. Concretely, we prove a no-go theorem ruling out shift rules with less than four terms for all gates whose Hermitian generators have eigenvalues of shape $-\lambda, 0, \lambda$ (Theorem 4.16).

On the topic of barren plateaus we make the following contributions:

- We develop a tool that automatically computes $\text{Var}\left(\frac{\partial \langle H \rangle}{\partial \theta_i}\right)$ and use it to empirically show that barren plateaus likely appear in 7 ansätze studied by Sim et al. [20] when measuring in the computational basis (Figures 5.3 and 5.4).

- We formally prove the existence of barren plateaus in three of the Sim ansätze and give necessary conditions on the measurement Hamiltonian for when they occur (Theorems 5.11, 5.12, and 5.14).

- We give a general framework for the barren plateau analysis of IQP circuits (Theorem 5.16) and use it to prove that the main circuit used by the quantum natural language processing library `lambeq` [21] has barren plateaus when measuring in the computational basis (Theorem 5.19).

1.2   Structure of this Thesis

We begin by discussing some of the background necessary to follow this thesis in Chapter 2 and introduce diagrammatic differentiation in Chapter 3. Chapter 4 is concerned with deriving gradient recipes using this diagrammatic technique. Subsequently, we study the gradient landscape of parametrised circuit with regard to barren plateaus in Chapter 5. Finally, we discuss our results and comment on future work in Chapter 6.

For presentation purposes, we move some of the proofs throughout the thesis to the appendix. This is remarked on underneath each such lemma. In the PDF version of this thesis one can easily jump to the corresponding proof by clicking on the arrow symbol (↓) on the right-hand side of the page.

The code to reproduce all numerical results and graphs in this thesis is available at

$$\texttt{https://github.com/mark-koch/msc-code}$$

Chapter 2

Background

In this chapter we give the necessary background to follow the thesis. Concretely, we give a brief introduction to quantum theory in Section 2.1 and discuss quantum machine learning in Section 2.2. Finally, we introduce the ZXW-calculus in Section 2.3.

2.1 An Introduction to Quantum Theory

2.1.1 States

The states of quantum systems are given by normalised vectors in a complex Hilbert space $\mathcal{H}$. We exclusively work within $\mathcal{H} = \mathbb{C}^{2^n}$ for this thesis, where states are given by column vectors of complex numbers. The *adjoint* $\psi^\dagger$ of a state ψ in this case is given by the conjugate-transpose of ψ. States and their adjoints are usually written in the *Dirac bra-ket notation*:

$$\psi \quad \rightsquigarrow \quad |\psi\rangle \qquad \psi^\dagger \quad \rightsquigarrow \quad \langle\psi|$$

The symbol $|\psi\rangle$ is called *ket* and $\langle\psi|$ is called *bra*. Plugging a bra into a ket yields the inner product of the two vectors which we denote by $\langle\psi|\phi\rangle := \langle\psi||\phi\rangle$ and call
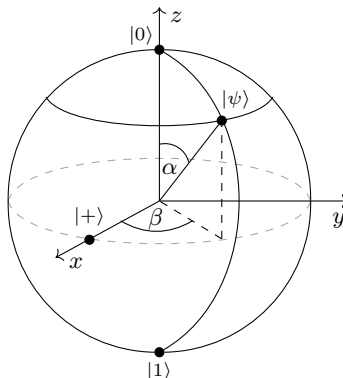
Figure 2.1: Visualisation of a qubit state $|\psi\rangle = x|0\rangle + y|1\rangle$ as a point on the Bloch sphere. We have $x = \cos(\frac{\alpha}{2})$ and $y = e^{i\beta}\sin(\frac{\alpha}{2})$.

bra-ket. The most elementary state is given by a single quantum bit, or *qubit*, which belongs to the two-dimensional Hilbert space $\mathbb{C}^2$ spanned by the standard basis

$$|0\rangle := \begin{pmatrix} 1 \\ 0 \end{pmatrix} \qquad |1\rangle := \begin{pmatrix} 0 \\ 1 \end{pmatrix}.$$

The states $|0\rangle$ and $|1\rangle$ are the quantum analogues of classical bits. Therefore, the basis $\{|0\rangle, |1\rangle\}$ is usually called *computational basis*. However, unlike classical bits, qubits can represent any linear combination of $|0\rangle$ and $|1\rangle$:

$$|\psi\rangle = x|0\rangle + y|1\rangle$$

for some $x, y \in \mathbb{C}$ with $|x|^2 + |y|^2 = 1$. We can picture the state $|\psi\rangle$ as a point on the so-called *Bloch sphere* as illustrated in Figure 2.1. We refer to those states "in-between" 0 and 1 as *superpositions*.

In order to unleash the full power of quantum computation, we describe interactions between multiple systems using the *tensor product* operation $\otimes$ corresponding to the Kronecker product. For example, the two-qubit system $\mathbb{C}^2 \otimes \mathbb{C}^2 = \mathbb{C}^4$ is spanned by the basis

$$|00\rangle := |0\rangle \otimes |0\rangle = (1, 0, 0, 0)^T \qquad |10\rangle := |0\rangle \otimes |1\rangle = (0, 1, 0, 0)^T$$
$$|01\rangle := |1\rangle \otimes |0\rangle = (0, 0, 1, 0)^T \qquad |11\rangle := |1\rangle \otimes |1\rangle = (0, 0, 0, 1)^T$$

where $|\psi\rangle \otimes |\phi\rangle$ is the *product state* of $|\psi\rangle$ and $|\phi\rangle$. We sometimes also write the computational basis vectors for $\mathbb{C}^{2^n}$ as $|j\rangle$ for $j = 0, 1, ..., 2^n - 1$.

2.1.2 Unitary Evolution

Definition 2.1. *A square matrix U is unitary if $UU^\dagger = U^\dagger U = I$.*

Computation on a quantum state $|\psi\rangle \in \mathbb{C}^{2^n}$ is done using *unitary evolutions*, i.e. acting on $|\psi\rangle$ according to a unitary matrix $U \in \mathbb{C}^{2^n \times 2^n}$. The resulting state is given by $|\psi'\rangle = U|\psi\rangle$. An example of a single-qubit action is the Hadamard operation

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \tag{2.1}$$

that maps the computational basis to the so-called *X-basis* $\{|+\rangle, |-\rangle\}$:

$$H|0\rangle = |+\rangle := \frac{|0\rangle + |1\rangle}{\sqrt{2}} \qquad\qquad H|1\rangle = |-\rangle := \frac{|0\rangle - |1\rangle}{\sqrt{2}}$$

Another example is the single-qubit $R_Z(\alpha)$ operation that corresponds to a Z-rotation on the Bloch sphere by an angle of α:

$$R_Z(\alpha) := \begin{pmatrix} e^{-i\frac{\alpha}{2}} & 0 \\ 0 & e^{i\frac{\alpha}{2}} \end{pmatrix} \tag{2.2}$$

$R_Z(\alpha)$ is an example of a parametrised unitary:

Definition 2.2. *A (strongly continuous) one-parameter unitary group is a family $\{U(\alpha)\}_{\alpha \in \mathbb{R}}$ of single-parameter unitary matrices that are strongly continuous ($\lim_{\alpha \to \alpha_0} U(\alpha) = U(\alpha_0)$ for all $\alpha_0 \in \mathbb{R}$) and homomorphisms ($U(\alpha + \beta) = U(\alpha) U(\beta)$).*

When speaking of (single-)parametrised unitaries, we generally refer to one-parameter unitary groups.

Definition 2.3. *A matrix H is self-adjoint, or Hermitian, if $H^\dagger = H$.*

Remarkably, there is a one-to-one correspondence between single-parameter unitaries and Hermitian operators:

Theorem 2.4 (Stone [22]). *Every strongly continuous one-parameter unitary group $\{U(\alpha)\}_{\alpha \in \mathbb{R}}$ is generated by a Hermitian operator H via $U(\alpha) = e^{i\alpha H}$.*

The matrix exponentials e^A for square matrices A used in this theorem are defined by $e^{iA} := \sum_{k=0}^{\infty} \frac{A^k}{k!}$ and satisfy

$$e^{\mathrm{diag}(a_1, ..., a_n)} = \mathrm{diag}(e^{a_1}, ..., e^{a_n}) \qquad\qquad e^{U^\dagger A U} = U^\dagger e^A U^\dagger \qquad (2.3)$$

for all unitaries U.

2.1.3 Measurements

In order to extract information from quantum systems, we need to perform *measurements*. Importantly, measuring a system usually also alters its state, making measurement a somewhat destructive process. Note that there are many different kinds of measurements one can perform. Mathematically, a measurement is specified by a set $\mathcal{M} = \{P_1, ..., P_k\}$ of projectors that sum up to the identity $\sum_i P_i = I$.

Definition 2.5. *A square matrix P is a projector if $P = P^\dagger = P^2$.*

Each projector represents a measurement outcome. Since measurement is a non-deterministic process, we get a probability distribution over the outcomes. When measuring $|\psi\rangle$, the probability of outcome P_i can be computed using the *Born rule*:

$$\mathrm{Prob}(i|\psi) = \langle\psi|P_i|\psi\rangle \qquad (2.4)$$

Example 2.6 (ONB Measurements). *The orthonormal basis measurement corresponding to a basis $\mathcal{B} = \{|\phi_i\rangle\}_i$ is given by $\mathcal{M}_\mathcal{B} = \{|\phi_i\rangle\langle\phi_i|\}_i$. For example, the two-dimensional computational basis yields $\mathcal{M} = \{|0\rangle\langle0|, |1\rangle\langle1|\}$. In that case, we have $Prob(i|\psi) = \langle\psi|i\rangle\langle i|\psi\rangle$. We can think of this as a measure of how "close" $|\psi\rangle$ is to $|0\rangle$ or $|1\rangle$: If $|\psi\rangle = x|0\rangle + y|1\rangle$ then $Prob(0|\psi) = (\overline{x}\langle0|0\rangle + \overline{y}\langle1|0\rangle)(x\langle0|0\rangle + y\langle0|1\rangle) = x\overline{x} = |x|^2$.*

An important observation is that states that are equal up to a *global phase* of $e^{i\alpha}$ behave exactly the same with regard to measurement: Let $|\phi\rangle := e^{i\alpha}|\psi\rangle$, then

$$\text{Prob}(j|\phi) = \langle e^{i\alpha}\psi|P_j|e^{i\alpha}\psi\rangle = e^{i\alpha}\langle\psi|P_j e^{-i\alpha}|\psi\rangle = \langle\psi|P_j|\psi\rangle = \text{Prob}(j|\psi).$$

Thus, there is no measurable difference between $|\phi\rangle$ and $|\psi\rangle$. Hence, states are not just vectors, but equivalence classes of vectors that are equal up to a global phase. One way to remove this redundancy is the *doubling* construction where we represent the states as $|\phi\rangle\langle\phi|$ and $|\psi\rangle\langle\psi|$ instead, which are actually equal. We will make heavy use of this when describing gradients of parametrised quantum circuits later.

Performing a single measurement corresponds to sampling from the distribution (2.4). However, often we are not necessarily interested in a single sample, but want to understand the broader distribution of outcomes. A useful tool for this is the *expectation value*. To motivate its definition, suppose we associate a real number x_j with each projector P_j. Then, we define random variable X that takes the value x_j whenever we get the measurement outcome j. The expectation value of our state $|\psi\rangle$ w.r.t. this operator then corresponds to the mean value of X:

$$\mathbf{E}(X) = \sum_{j=1}^{k} x_j \cdot \text{Prob}(j|\psi) \overset{(2.4)}{=} \sum_{j=1}^{k} x_j \cdot \langle\psi|P_j|\psi\rangle = \langle\psi|\left(\sum_{j=1}^{k} x_j P_j\right)|\psi\rangle$$

In order to estimate the expectation value on a quantum computer, one can compute the statistical mean of X by preparing and measuring the state $|\psi\rangle$ for a large number of executions. One commonly refers to the different executions as *shots*.

Interestingly, $\sum_{j=1}^{k} x_j P_j$ is self-adjoint. Conversely, every self-adjoint matrix H with eigenvectors $\lambda_1, ..., \lambda_k$ gives rise to a unique set of projectors $\mathcal{M}_H = \left\{\sum_{\phi\in\Phi_i}|\phi\rangle\langle\phi|\right\}_{i=1}^{k}$ where Φ_i is the set of eigenvectors of H corresponding to the eigenvalue λ_i. Because of this duality, it is often more convenient to describe measurements via Hermitian operators instead of projectors. In this context, H is commonly referred to as an *observable*, or *Hamiltonian* and the expectation value is denoted by

$$\langle H \rangle := \langle \psi | H | \psi \rangle.$$

Interestingly, every Hermitian matrix $H \in \mathbb{C}^{2^n}$ can be written as a real combination of Pauli operators $P \in \{X, Y, Z, I\}^{\otimes n}$. We will use this in Chapter 5 to simplify our barren plateau analysis.

2.1.4   The Quantum Circuit Model

The *quantum circuit model* is a model to describe quantum computation that is inspired by classical circuits. After preparing n qubits in a fixed state (usually $|0\rangle^{\otimes n}$) we apply *gates* that correspond to unitary operations on the qubits. Finally, we measure one or more qubits. Circuits are read from left to right and qubits are drawn as wires with gates on them:
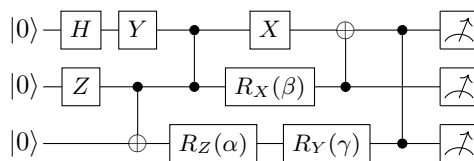
Figure 2.2: Example of a 3-qubit quantum circuit.

We have already seen the Hadamard gate H and the Z-rotation R_Z in (2.1) and (2.2), respectively. Similarly, the single-qubit gates R_X and R_Y correspond to X- and Y-rotations on the Bloch sphere:

$$R_X(\alpha) := \begin{pmatrix} \cos(\frac{\alpha}{2}) & -i\sin(\frac{\alpha}{2}) \\ -i\sin(\frac{\alpha}{2}) & \cos(\frac{\alpha}{2}) \end{pmatrix} \qquad R_Y(\alpha) := \begin{pmatrix} \cos(\frac{\alpha}{2}) & -\sin(\frac{\alpha}{2}) \\ \sin(\frac{\alpha}{2}) & \cos(\frac{\alpha}{2}) \end{pmatrix}$$

The special cases for $\alpha = 180°$ rotations around the Bloch sphere give rise to the so-called *Pauli matrices* (up to a global phase):

$$X := \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \qquad Y := \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \qquad Z := \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

Finally, controlled gates are gates where the first qubit controls whether a unitary U is applied to the remaining gates. They can be constructed via

$$C_U = \quad \begin{array}{c} \text{(circuit)} \end{array} \quad := (|0\rangle\langle 0| \otimes I) + (|1\rangle\langle 1| \otimes U) = \begin{pmatrix} I & 0 \\ 0 & U \end{pmatrix}.$$

In Figure 2.2, we have controlled X and Z gates that are usually called $CNOT$[1] and CZ, respectively. They have a special notation:

$$CNOT = \quad \begin{array}{c} \text{(circuit)} \end{array} = \begin{array}{c} \boxed{X} \end{array} := \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

$$CZ = \quad \begin{array}{c} \text{(circuit)} \end{array} = \begin{array}{c} \boxed{Z} \end{array} := \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix}$$

The $CNOT$ gate is drawn with a $\oplus$ symbol since it acts like $|x, y\rangle \mapsto |x, x \oplus y\rangle$ on the computational basis where $\oplus$ denotes XOR. The CZ gate is drawn with two black dots since it is symmetric in which qubit is the control. Both $CNOT$ and CZ are used to *entangle* the two qubits to which they are applied.

2.2 Quantum Machine Learning

The goal of *quantum machine learning* (QML) is to achieve a quantum advantage using the current *noisy intermediate-scale quantum* (NISQ) hardware. Typically, QML algorithms employ a hybrid approach where a quantum processor works in tandem with a classical computer. In this thesis, we focus on *variational algorithms* for QML. This approach relies on *parametrised quantum circuits* (PQCs), i.e. circuits that depend on tunable parameters. For example, the circuit in Figure 2.2 is a PQC if the parameters α, β, γ are not fixed. Given a PQC that depends on some parameters $\vec{\theta}$, machine learning techniques are used to find an optimal parameter

---

[1]This is because the Pauli X acts like negation on the computational basis.
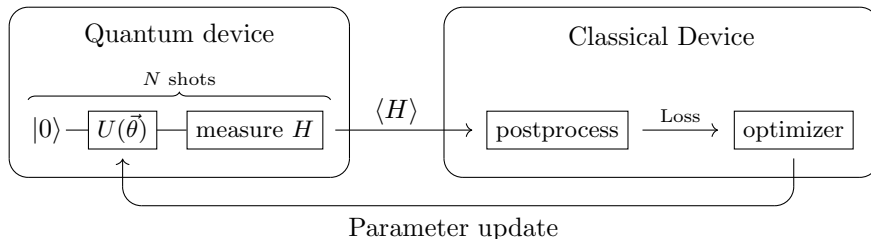
Figure 2.3: Pipeline for variational algorithms (adapted from Figure 1 in [13]).

assignment $\vec{\theta}^\star$ for which the circuit exhibits some desired behaviour. This could for example be fitting a dataset in a supervised classification or regression task [23, 6], or modelling a probability distribution for a generative task [24, 25, 26]. Other applications of variational algorithms include simulating quantum chemistry [27, 28], solving combinatorial optimisation problems [29], and performing natural language processing tasks [30, 21].

Figure 2.3 shows the schematic pipeline used by variational algorithms. Essentially, the PQC is trained using a classical optimiser in order to minimise some loss calculated based on the expectation value $\langle H \rangle$ produced by the quantum device. Because of the current NISQ hardware, this process is generally noisy. However, many optimisers developed for machine learning are resilient to a certain amount of noise which makes variational algorithms applicable in the NISQ era.

2.2.1   Types of Ansätze

The PQCs used for variational algorithms are typically referred to as *ansätze*. The term ansatz comes from mathematics and physics where it describes an initial strategy or approach to express a solution. Broadly, one can distinguish two different kinds of ansatz designs commonly used for QML which are depicted in Figure 2.4.

Tensor network ansätze arrange gates in a fixed layout inspired by tensor networks [31, 32]. For example, the blocks in Figure 2.4 are laid out in a tree architecture. *Layered ansätze* on the other hand consist of layers that are repeated one after the other for a fixed number of times. Commonly, each layer is made up of

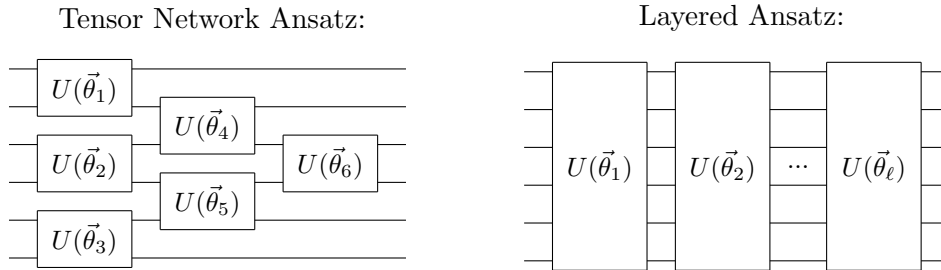Tensor Network Ansatz: Layered Ansatz:

Figure 2.4: Different ansatz layouts.

single qubit unitaries, preceded or followed by a block of entangling gates. Another commonly used type of layered ansatz is the *alternating operator ansatz* used in the *quantum approximate optimization algorithm* (QAOA) [29]. There, the layers are defined in terms of two Hamiltonians that encode a combinatorial optimisation problem which can be solved by training the circuit.

In this thesis, we focus on layered ansatz designs that have been shown to be more expressive than tensor network ansätze [33]. In particular, see Figure 5.1 in Chapter 5 for layered ansätze that are used in practice.

2.2.2 Gradient-Based Optimisation

There is a wide range of optimisation algorithms that can be used to train PQCs [34, 35, 36, 37, 38, 39]. In this thesis, we focus on gradient-based optimisation approaches which are commonly used in QML and provably improve convergence in variational algorithms [5].

Gradient-based optimisation techniques such as gradient descent have been proven to be widely successful in the domain of classical machine learning, in particular neural networks. Given the output $\vec{y}$ of a neural network, the gradient $\frac{\partial \mathcal{L}}{\partial w_i}(\vec{y})$ of some loss function $\mathcal{L}$ with respect to the weight w_i is computed via backpropagation and the weight is updated in the opposite direction of this gradient. We can transfer this approach to the quantum realm: Instead of the weights of a neural network, we train the parameters of an ansatz. The "output" of the quantum circuit is an expectation value $\langle H \rangle$. Hence, we want to compute $\frac{\partial \mathcal{L}}{\partial \theta_i}(\langle H \rangle)$ which by the chain rule depends on

$\frac{\partial \langle H \rangle}{\partial \theta_i}$. Unlike individual measurements, expectation values are continuous variables such that this gradient is well-defined. Finally, we update the circuit parameters according to the loss gradient.

However, gradient descent on quantum computers comes with a set of unique challenges. First, it is not feasible to compute $\frac{\partial \langle H \rangle}{\partial \theta_i}$ classically. In particular, the back-propagation algorithm is not available since quantum circuits have a fundamentally different structure than neural networks. Instead, the gradient must be computed on the quantum device itself. Quantum algorithms that solve this task are commonly referred to as *gradient recipes* and are subject of a lot research interest at the moment [6, 7, 8, 9]. We contribute to this in Chapter 5 by giving diagrammatic interpretations and proofs of existing recipes, and by proving a conjecture by Anselmetti et al. [8] establishing the optimality of a certain recipe.

The second issue lies with the geometry of the gradient landscape. It is hypothesised that gradient descent performs well on classical neural networks because their loss surface has few bad local minima [40]. The same can unfortunately not be said for PQCs [41]. Even worse, it has been shown that the gradient landscape of many ansätze is exponentially flat with respect to circuit size, making gradient descent difficult or even impossible [10]. Thus, there is a lot of interest in analysing which ansätze exhibit those *barren plateaus*. In Chapter 5 we apply a diagrammatic method to analyse ansätze for this problem.

2.3   The ZXW-Calculus

The *ZX-calculus* is graphical language for reasoning about quantum computation originally developed by Coecke and Duncan [14]. It is universal and complete [42] meaning that all quantum reasoning can be carried out in the realm of ZX diagrams. The ZX-calculus has been applied in a variety of areas, including circuit optimisation [15, 43], compilation [16, 44], simulation [17], measurement-based quantum computing [45, 46] and surface codes [47].

The *ZXW-calculus* [18] is a variant of ZX that has its roots in the *algebraic ZX-calculus* [48]. It has recently been used to express derivates and integrals [19] which makes it well-suited for our diagrammatic treatment of gradient-based QML.

2.3.1 Generators and String Diagrams

ZXW diagrams consist of generators that are wired together and connected to inputs and outputs. Following the circuit notation, we put the inputs on the left side and the outputs on the right. While diagrams can be studied as mathematical objects in their own right, for this thesis we are mainly interested in their interpretation as linear maps. Concretely, a diagram with n inputs and m outputs represents a $2^m \times 2^n$ complex matrix. There are also diagrams with zero inputs and outputs which thus represent single complex numbers.

We now give the three main generators of the ZXW-calculus:

$$n\left\{ \begin{array}{c} \vdots \end{array} \boxed{a} \begin{array}{c} \vdots \end{array} \right\}m := |0^m\rangle\langle 0^n| + a|1^m\rangle\langle 1^n| \qquad -\!\square\!- := \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \qquad \blacktriangleleft := \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 0 & 0 \end{pmatrix}$$

where $a \in \mathbb{C}$. We call the generators the *green box*, *Hadamard*, and *black triangle*, respectively. ZXW diagrams are formed by wiring these generators together. For this, we also introduce generators that allow us to bend and cross wires:

$$— := \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \qquad \subset := \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \end{pmatrix} \qquad \supset := \begin{pmatrix} 1 & 0 & 0 & 1 \end{pmatrix} \qquad \times := \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

We can wire the generators together using the sequential and parallel composition operators $\circ$ and $\otimes$, corresponding to matrix multiplication and tensor product on the underlying matrices. For example, we write $\overset{\boxed{2}}{\underset{\square}{\frown}} \supset := \supset \circ (-\boxed{2}\!- \otimes -\square\!-)$. Furthermore, the wires satisfy the *yanking equations*

$$\underset{}{\subset\!\supset} = — \qquad\qquad \subset\!\!\times = \subset$$

This means we can arbitrarily deform diagrams by moving the generators around the plane, bending and unbending wires as we go, without changing the underlying matrix. We only have to make sure that the inputs and outputs stay in the same order. This principle is summarised in the slogan *only connectivity matters*.

2.3.2   Additional Notation

Based on the generators, we define some additional notation. For example, the green spider from the original ZX-calculus can be defined via the green box:

$$\vdots \, \alpha \, \vdots \;\; := \;\; \vdots \, e^{i\alpha} \, \vdots \qquad\qquad \vdots \, \bigcirc \, \vdots \;\; := \;\; \vdots \, 0 \, \vdots \tag{2.5}$$

The red spiders from the original ZX-calculus can be defined by Hadamard conjugation:

$$\vdots \, \alpha \, \vdots \;\; := \;\; \vdots \, \alpha \, \vdots \qquad\qquad \vdots \, \bullet \, \vdots \;\; := \;\; \vdots \, 0 \, \vdots \tag{2.6}$$

If a diagram only contains spiders and no boxes or black triangles, we sometimes drop the "W" and speak of traditional *ZX-diagrams*. Often we only have spiders with phase $\alpha = 0$ or $\alpha = \pi$. For those cases, we define a special pink spider as a rescaled version of the red spider that only has integer components in its matrix:

$$n\left\{ \vdots \, \pi \, \vdots \right\}m \;\; := \;\; 2^{\frac{n+m-2}{2}} \, \vdots \, \pi \, \vdots \qquad\qquad n\left\{ \vdots \, \bullet \, \vdots \right\}m \;\; := \;\; 2^{\frac{n+m-2}{2}} \, \vdots \, \bullet \, \vdots \tag{2.7}$$

We give the scalars that are represented by commonly occurring diagrams below:

$$\bigcirc = 2 \qquad \bullet = 1 \qquad \pi = \pi = 0 \qquad \bullet\!\!-\!\!\alpha = 1 \qquad \pi\!\!-\!\!\alpha = e^{i\alpha}$$

Finally, we define the triangle and inverse triangle as well as their transposes:

$$\blacktriangleleft \;\; := \;\; \blacktriangleleft\!\!\bigcirc \;\; = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \qquad\qquad \blacktriangleright \;\; := \;\; \underset{}{\smallfrown\!\!\blacktriangleleft} \;\; = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}$$

$$\blacktriangleleft^{-1} \;\; := \;\; \blacktriangleleft\!\!\pi \;\; = \begin{pmatrix} 1 & -1 \\ 0 & 1 \end{pmatrix} \qquad\qquad \blacktriangleright^{-1} \;\; := \;\; \underset{}{\smallfrown\!\!\blacktriangleleft}^{-1} \;\; = \begin{pmatrix} 1 & 0 \\ -1 & 1 \end{pmatrix} \tag{2.8}$$
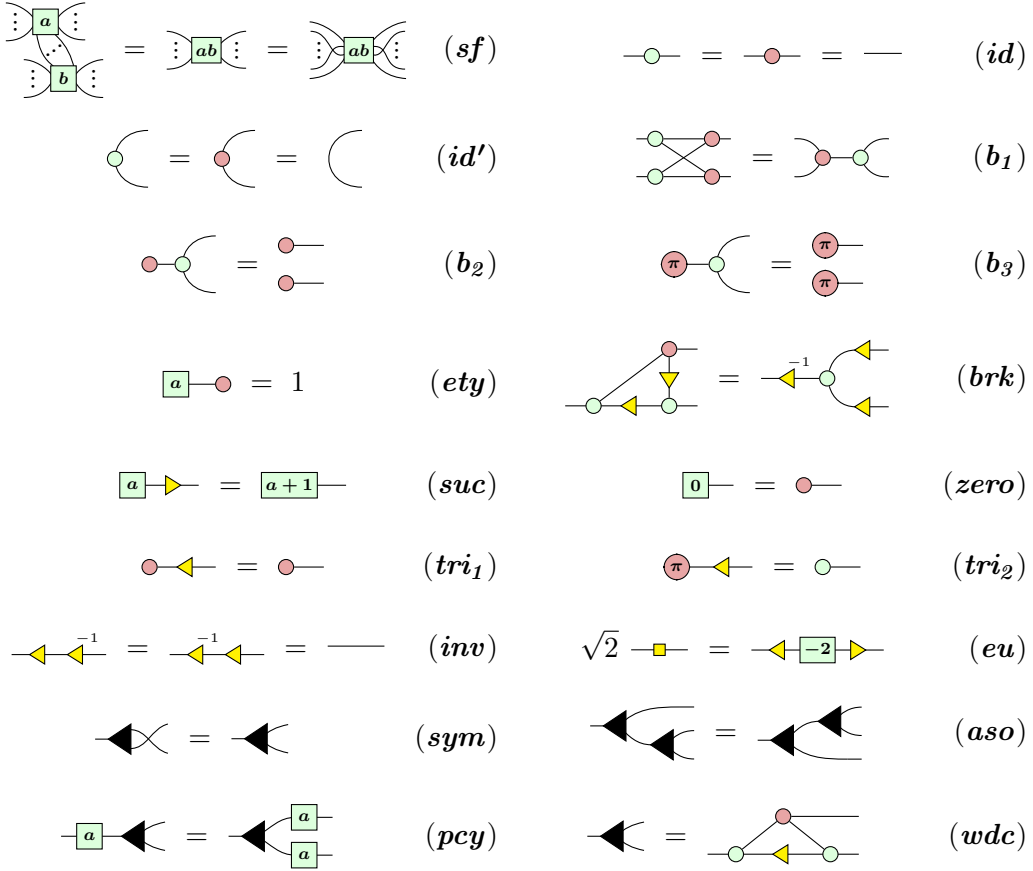
Figure 2.5: Rules of the ZXW-calculus for $a, b \in \mathbb{C}$.

2.3.3   Rules

So far, we have only seen ZX(W) diagrams as graphical representations of matrices. Their real power comes from the rewrite rules that allow us to do matrix calculations diagrammatically. The rules of the ZXW-calculus are listed in Figure 2.5.

Note that the equality signs in the rules mean that both sides represent exactly the same matrix. In the original ZX-calculus, many rules like (b_1), (b_2), or (b_3) only hold up to a (non-zero) scalar that is often ignored. However, for the purposes of this thesis we need to be precise about scalars. The fact that we can give many rules without them is thanks to the rescaled pink spider. As a trade-off, the colour-change rule now introduces scalars for pink spiders[2]:

---

[2] We prove this rule as well as other rules of the original ZX-calculus in Section 2.3.6.

$$n\left\{\;\vdots\;\tau\;\vdots\;\right\}m \overset{(\boldsymbol{cc})}{=} 2^{-\frac{n+m-2}{2}}\;\vdots\;\tau\;\vdots\qquad\qquad n\left\{\;\vdots\;\tau\;\vdots\;\right\}m \overset{(\boldsymbol{cc})}{=} 2^{\frac{n+m-2}{2}}\;\vdots\;\tau\;\vdots$$

Furthermore, fusing pink spiders that are connected by multiple wires also introduces a scalar:

$$\vdots\;\tau\;\vdots \overset{(\text{Lem. 2.13})}{=} 2^{n-1}\;\vdots\;\boldsymbol{\tau+\sigma}\;\vdots$$

We define a multi-legged version of the black triangle, which we call *W spider*:

$$\blacktriangleleft\;\vdots\; := \;\blacktriangleleft\blacktriangleleft\;\vdots\qquad\qquad \blacktriangleleft := \text{———}\qquad\qquad(2.9)$$

Because of the (**aso**) rule it actually does not matter in which order we plug the triangles together and it is easy to see that W spiders satisfy the following fusion rule:

$$\blacktriangleleft\;\vdots\;\blacktriangleleft\;\vdots\; = \;\blacktriangleleft\;\vdots\qquad\qquad(\boldsymbol{wf})$$

On top of this, as we will prove in Lemma 2.23, they interact with pink spiders in the following way:

$$\bullet\blacktriangleleft\;\vdots\; \overset{(\boldsymbol{w})}{=} \;\vdots\qquad\qquad \pi\blacktriangleleft\;\vdots\; \overset{(\boldsymbol{w})}{=} \;\boxed{\pi}\; + \;\boxed{\pi}\; + \;...\; + \;\boxed{\pi}$$

This property will prove to be crucial when discussing diagrammatic differentiation in Chapter 3.

2.3.4 Quantum Gates and Computation in ZXW

Next, we explain how quantum computation is expressed in ZX(W). First, note that pink and green spiders can describe the computational and the X-basis:

$$\bullet\!\!-\!\! = |0\rangle \qquad \textcolor{red}{\pi}\!\!-\!\! = |1\rangle \qquad \circ\!\!-\!\! = \sqrt{2}\,|+\rangle \qquad \textcolor{green}{\pi}\!\!-\!\! = \sqrt{2}\,|-\rangle \qquad (2.10)$$

Many matrix operations commonly used in quantum computing have elegant representations in ZXW. For example, transposing a matrix corresponds to mirroring the diagram horizontally and the conjugate matrix is obtained by conjugating the numbers in boxes and negating the phases in spiders. Thus the adjoint of a ZXW diagram is constructed by combining those two operations.

The Hadamard gate is given as a generator. We introduce the following notation, denoting edges with a Hadamard on them as dashed blue lines:

$$-\blacksquare- \quad \rightsquigarrow \quad \text{-----}$$

The Pauli matrices are represented by

$$X \;=\; -\textcolor{red}{\pi}- \qquad\qquad Y \;=\; i\;-\textcolor{green}{\pi}\!\!-\!\!\textcolor{red}{\pi}- \qquad\qquad Z \;=\; -\textcolor{green}{\pi}-$$

the rotation gates can be written as

$$R_Z(\alpha) \;=\; e^{-i\frac{\alpha}{2}}\;-\textcolor{green}{\alpha}- \qquad\qquad R_X(\alpha) \;=\; e^{-i\frac{\alpha}{2}}\;-\textcolor{red}{\alpha}-$$

$$R_Y(\alpha) \;=\; e^{-i\frac{\alpha}{2}}\;-\textcolor{green}{-\frac{\pi}{2}}\!\!-\!\!\textcolor{red}{\alpha}\!\!-\!\!\textcolor{green}{\frac{\pi}{2}}- \;=\; e^{-i\frac{\alpha}{2}}\;-\textcolor{red}{\frac{\pi}{2}}\!\!-\!\!\textcolor{green}{\alpha}\!\!-\!\!\textcolor{red}{-\frac{\pi}{2}}-$$

and common two-qubit gates are given by

$$CNOT \;=\; \quad = \quad \qquad\qquad CZ \;=\; \quad = \sqrt{2}\;$$

$$CR_Z(\alpha) \;=\; \boxed{R_Z(\alpha)} \;=\; \qquad\qquad (2.11)$$

Using those building blocks, we can easily turn quantum circuits into ZXW diagrams. However, recall from our discussion in Section 2.1.3 that the matrix representation of

quantum states has a certain redundancy in that states that only differ by a global phase behave exactly the same. To deal with this problem, we use the *doubling construction* to represent quantum circuits in ZXW. Concretely, whenever we want to express quantum circuits in ZXW, we first construct a diagram capturing the circuit structure, and then we double it. Doubling means tensoring the diagram with its complex conjugate, i.e.

$$\text{doubled} \left(\boxed{D} \right) := \begin{array}{c} \boxed{D} \\ \boxed{\overline{D}} \end{array}$$

This way, all global phases cancel out. See [49] for a more detailed description of doubling.

2.3.5 Pauli Boxes

A useful ZX construction related to Paulis are so-called *Pauli boxes* [50, 51]:

Definition 2.7. *[51] The Pauli boxes are defined as*

Note that we can treat the wire sticking out on top as either input or output. Plugging in a green π yields the corresponding Pauli:

Lemma 2.8. *[51] For all $P \in \{I, X, Y, Z\}$ we have*

Pauli boxes can be used to define a type of gate we have not mentioned so far. Given a *Pauli string* $\vec{P} \in \{I, X, Y, Z\}^{\otimes n}$, i.e. a tensor product of Paulis, we define the *Pauli exponential gate* $\vec{P}(\alpha)$ by

$$\vec{P}(\alpha) := e^{-i\frac{\alpha}{2}\vec{P}} \quad = $$

In the special case where $\vec{P} \in \{I, Z\}^{\otimes n}$, we call $\vec{P}(\alpha)$ a *phase gadget*. Paul exponentials based on the same Pauli string fuse together:

Lemma 2.9. *[50] For all Pauli strings $\vec{P}$ we have*

$$ = \tag{2.12}$$

In particular, this implies $\vec{P}(\alpha)\vec{P}(\beta) = \vec{P}(\alpha + \beta)$.

Pauli gadgets also have interesting commutation properties:

Lemma 2.10. *[50] Let $\vec{P}, \vec{Q}$ be n-qubit Pauli strings. If the number of positions i for which $P_i \neq Q_i$ and $P_i, Q_i \neq I$ is even, then*

$$ = $$

Otherwise,

$$ = \frac{1}{\sqrt{2}} $$

2.3.6   Useful Lemmas

We close the chapter by stating and proving some basic results that we will use throughout the thesis.

Lemma 2.11. *[48] Hadamard is involutive:*

$$\text{—}\blacksquare\text{—}\blacksquare\text{—} \;\; = \;\; \text{———} \qquad\qquad (\boldsymbol{hh})$$

Lemma 2.12. *Hadamards switch colours up to a scalar. For $\tau \in \{0, \pi\}$:*

$$n\left\{ \vcenter{\hbox{:}}\,\tau\,\vcenter{\hbox{:}} \right\}m \;=\; 2^{-\frac{n+m-2}{2}}\;\;\tau\qquad\qquad n\left\{ \vcenter{\hbox{:}}\,\tau\,\vcenter{\hbox{:}} \right\}m \;=\; 2^{\frac{n+m-2}{2}}\;\;\tau \qquad (\boldsymbol{cc})$$

The only scalar-free colour change happens for two legs:

$$\text{—}\blacksquare\!-\!\tau\!-\!\blacksquare\text{—} \;=\; \text{—}\,\tau\,\text{—} \qquad\qquad \text{—}\blacksquare\!-\!\tau\!-\!\blacksquare\text{—} \;=\; \text{—}\,\tau\,\text{—}$$

Proof.

$$n\left\{ \vcenter{\hbox{:}}\,\tau\,\vcenter{\hbox{:}} \right\}m \;\overset{(2.6)}{=}\; \tau \;\overset{(2.7)}{=}\; 2^{-\frac{n+m-2}{2}}\;\tau$$

$$n\left\{ \vcenter{\hbox{:}}\,\tau\,\vcenter{\hbox{:}} \right\}m \;\overset{(2.7)}{=}\; 2^{\frac{n+m-2}{2}}\;\tau \;\overset{(2.6)}{=}\; 2^{\frac{n+m-2}{2}}\;\tau$$

$$\overset{(\boldsymbol{hh})}{=}\; 2^{\frac{n+m-2}{2}}\;\tau \qquad\qquad\qquad \square$$

Lemma 2.13. *Pink spiders fuse together. We also call this rule $(\boldsymbol{sf})$.*

$$\tau \;\vcenter{\hbox{$\overset{\cdots n}{}$}}\; \sigma \;=\; 2^{n-1}\;\; \tau+\sigma \qquad\qquad (\boldsymbol{sf})$$

Proof.

Lemma 2.14. *The zero box disconnects:*

$$(2.13)$$

Proof.

$\square$

Lemma 2.15. *Pink spiders can be decomposed as follows:*

$$(2.14)$$

Proof.

$\square$

Lemma 2.16. *[48] Hopf rule:*

$$-\!\bigcirc\!\bullet\!- \;=\; -\!\bigcirc \;\; \bullet\!- \qquad\qquad (\boldsymbol{ho})$$

Lemma 2.17. *[48] Strong complementarity:*

$$\mathrel{\vcenter{\hbox{$\times$}}} \;=\; \mathrel{\vcenter{\hbox{$\bowtie$}}} \qquad\qquad (\boldsymbol{sc})$$

Lemma 2.18. *[48] Pink π copies through and negates phases:*

$$-\!\pi\!-\!\alpha\big)\vdots \;=\; e^{i\alpha}\; -\!\overline{-\alpha}\big)\vdots \qquad\qquad (\boldsymbol{\pi})$$

Lemma 2.19. *[48] For $x,y \in \{0,1\}$ we have*

$$x\pi\!-\!\boxed{a}\big)\vdots \;=\; a^{x}\; \vdots \qquad\qquad x\pi\!-\!y\pi\big)\vdots \;=\; (-1)^{xy}\; \vdots \qquad\qquad (\boldsymbol{cp})$$

Lemma 2.20. *[48] Pink π transposes the triangle:*

$$-\!\pi\!-\!\triangleleft\!- \;=\; -\!\triangleright\!-\!\pi\!- \qquad\qquad (2.15)$$

Lemma 2.21. *The triangle acts as a change of bases:*

$$\begin{aligned} \bullet\!-\!\triangleleft\!- &= \bullet\!- & \pi\!-\!\triangleleft\!- &= \circ\!- \\ \bullet\!-\!\triangleright\!- &= \circ\!- & \pi\!-\!\triangleright\!- &= \pi\!- \end{aligned} \qquad\qquad (\boldsymbol{tri})$$

Proof. The first two equations are just $(\boldsymbol{tri_1})$ and $(\boldsymbol{tri_2})$. The third equation has been proven in [48]:

$$\bullet\!-\!\triangleright\!- \;\overset{(\boldsymbol{zero})}{=}\; \boxed{0}\!\triangleright\!- \;\overset{(\boldsymbol{suc})}{=}\; \boxed{1}\!- \;=\; \circ\!-$$

Then, the second equations follow from

$$\pi \blacktriangleright \stackrel{(\textbf{\textit{sf}})}{=} \pi \blacktriangleright \stackrel{(2.15)}{=} \blacktriangleleft \pi \stackrel{(\textbf{\textit{tri}}_1)}{=} \pi \qquad \qquad \square$$

Lemma 2.22. *The two-legged W spider satisfies*

$$\blacktriangleleft = \qquad \qquad \pi \blacktriangleleft = \pi \qquad \qquad (2.16)$$

Proof.

$$\blacktriangleleft \stackrel{(\textbf{\textit{wdc}})}{=} \blacktriangleleft \stackrel{(\textbf{\textit{cp,sf}})}{=} \blacktriangleleft \stackrel{(\textbf{\textit{tri}})}{=} \stackrel{(\textbf{\textit{cp,sf}})}{=}$$

$$\pi \blacktriangleleft \stackrel{(\textbf{\textit{wdc}})}{=} \pi \blacktriangleleft \stackrel{(\textbf{\textit{cp,sf}})}{=} \pi \blacktriangleleft \stackrel{(\textbf{\textit{tri}})}{=} \pi \stackrel{(\textbf{\textit{id}})}{=} \pi \qquad \square$$

Lemma 2.23. *In general, the W spider acts on the computational basis as follows:*

$$\blacktriangleleft \vdots = \vdots \qquad \pi \blacktriangleleft \vdots = \pi \vdots + \pi \vdots + \dots + \pi \vdots \qquad (\textbf{\textit{w}})$$

Proof. We prove both equation simultaneously by induction on the number of outputs. If the W spider has a single output, the equations hold trivially:

$$\blacktriangleleft \stackrel{(2.9)}{=} \qquad \qquad \pi \blacktriangleleft \stackrel{(2.9)}{=} \pi$$

For the inductive step, we have

$$\blacktriangleleft \vdots \stackrel{(\textbf{\textit{wf}})}{=} \blacktriangleleft \blacktriangleleft \vdots \stackrel{(2.16)}{=} \blacktriangleleft \vdots \stackrel{(\text{IH})}{=} \vdots$$

$$\pi \blacktriangleleft \vdots \stackrel{(\textbf{\textit{wf}})}{=} \pi \blacktriangleleft \blacktriangleleft \vdots \stackrel{(2.16)}{=} \pi \blacktriangleleft \vdots \stackrel{(*)}{=} \pi \blacktriangleleft \vdots + \pi \blacktriangleleft \vdots$$

$$\stackrel{(IH)}{=} \quad + \quad + \ \dots \ + \quad$$

where the step $(\ast)$ follows from

$$
\begin{aligned}
&= |+\rangle\langle+| - |-\rangle\langle-| \\
&= \frac{1}{2}(|0\rangle + |1\rangle)(\langle0| + \langle1|) - \frac{1}{2}(|0\rangle - |1\rangle)(\langle0| - \langle1|) \\
&= |0\rangle\langle1| + |1\rangle\langle0| \\
&= \quad + \quad \qquad\qquad\qquad \square
\end{aligned}
$$

Lemma 2.24. *Plugging a pink dot into a two-legged spider produces identity:*

$$= \quad \underline{} \tag{2.17}$$

Proof.

$$\stackrel{(wdc)}{=} \quad \stackrel{(cp,sf)}{=} \quad \stackrel{(tri,sf)}{=} \quad \stackrel{(id)}{=} \quad \underline{} \qquad \square$$

Lemma 2.25. *Plugging a pink dot into a W spider makes the leg disappear:*

$$= \tag{2.18}$$

Proof. By induction on the number of outputs. The base case holds by (2.17). For the inductive step we have

$$\stackrel{(wf)}{=} \quad \stackrel{(IH)}{=} \quad \stackrel{(wf)}{=} \qquad \square$$

Lemma 2.26. *The two-legged W spider adds boxes:*

$$\boxed{\begin{matrix} a \\ b \end{matrix}} = \boxed{a+b} \tag{2.19}$$

Proof. If $a = 0$, we have

If $a \neq 0$, we have

Corollary 2.27. *The W spider adds boxes:*

$$\tag{2.20}$$

Proof. Follows by induction on n using (2.19). □

Chapter 3

Diagrammatic Differentiation

For our diagrammatic analysis of gradient-based QML, we crucially need a graphical representation of derivatives. This so-called *diagrammatic differentiation* for ZX-calculus was first discovered in [52] and [13] and subsequently generalised to tensor calculi based on monoidal categories [53]. Recently, Wang and Yeung [19] developed a more compact graphical representation of derivatives avoiding sums of diagrams using the ZXW-calculus.

We give an overview on diagrammatic differentiation in Section 3.1, following the treatment by Wang and Yeung [19]. In Section 3.2, we present a novel, simplified gradient representation for the special case of parametrised quantum circuits (Theorem 3.6) that we will use for the remainder of the thesis. Finally, we discuss some properties of this representation in Section 3.3.

3.1 Background

Recall that we can interpret every ZX diagram D with n inputs and m outputs as a matrix $\mathbb{C}^{2^m \times 2^n}$. The derivative of a parametrised diagram $D(\theta)$, written $\frac{\partial}{\partial \theta} D(\theta)$, is defined as the gradient of the matrix associated with $D(\theta)$. Consider for example a single-legged green spider:

$$\theta\!-\!\!- \;\; = \;\; \begin{pmatrix} 1 \\ e^{i\theta} \end{pmatrix} \qquad \Rightarrow \qquad \frac{\partial}{\partial\theta}\,[\,\theta\!-\!\!-\,] = \begin{pmatrix} 0 \\ ie^{i\theta} \end{pmatrix}$$

The goal of diagrammatic differentiation is to represent those gradients as ZX(W) diagrams. For example, by inspecting the gradient matrix of our single-legged spider, we observe that

$$\frac{\partial}{\partial\theta}\,[\,\theta\!-\!\!-\,] \;\; = \;\; ie^{i\theta}\; \pi\!-\!\!- \;\; \overset{(cp)}{=} \;\; i\; \pi\!-\!\theta\!-$$

In fact, this is one of the key equations of diagrammatic differentiation. However, to cover the most general case, we should also to consider the possibility that the phase of the spider is a different function in θ. The resulting equation is very similar to the rule above:

Lemma 3.1. *Let f be a differentiable real function. Then*

$$\frac{\partial}{\partial\theta}\,[\,f(\theta)\!-\!\!-\,] \;\; = \;\; if'(\theta)\; \pi\!-\!f(\theta)\!- \tag{3.1}$$

Proof. We have

$$\frac{\partial}{\partial\theta}\,[\,f(\theta)\!-\!\!-\,] \;\; \overset{(2.5)}{=} \;\; \frac{\partial}{\partial\theta}\left(|0\rangle + e^{if(\theta)}|1\rangle\right) = if'(\theta)\cdot e^{if(\theta)}|1\rangle \;\; \overset{(cp)}{=} \;\; if'(\theta)\; \pi\!-\!f(\theta)\!- \qquad \square$$

Furthermore, we note that when differentiating a larger diagram, we can ignore the parts that do not depend on θ. This property is called *linearity* [52]:

Lemma 3.2 (Linearity). *Let $D(\theta)$ be a parametrised ZX diagram depending on θ and let E be a ZX diagram in which θ does not occur. Then*

$$\frac{\partial}{\partial\theta}\left[\,\vdots\,\boxed{E}\,\vdots\,\boxed{D(\theta)}\,\vdots\,\right] = \frac{\partial}{\partial\theta}\left[\,\vdots\,\boxed{D(\theta)}\,\vdots\,\right] \circ \,\vdots\,\boxed{E}\,\vdots$$

$$\frac{\partial}{\partial\theta}\left[\begin{array}{c}\boxed{D(\theta)}\\ \boxed{E}\end{array}\right] = \frac{\partial}{\partial\theta}\left[\boxed{D(\theta)}\right]\begin{array}{c}\\ \boxed{E}\end{array}$$

The equations also hold when switching the order of $D(\theta)$ and E.

Proof. Directly follows from the linearity of matrix differentiation for multiplication and tensor product, since for matrices $D(\theta), E$ we have $\frac{\partial}{\partial\theta}\left[D(\theta)E\right] = \left(\frac{\partial}{\partial\theta}\left[D(\theta)\right]\right)E$ and $\frac{\partial}{\partial\theta}\left[D(\theta)\otimes E\right] = \left(\frac{\partial}{\partial\theta}\left[D(\theta)\right]\right)\otimes E$. □

In the following, we will use brackets to denote the parts of the diagram we are differentiating. Using the previous two lemmas, we can give the derivative of any green and red spider:

$$\frac{\partial}{\partial\theta}\left[\begin{array}{c}f(\theta)\end{array}\right] \stackrel{(sf,\text{Lem } 3.2)}{=} \frac{\partial}{\partial\theta}\left[\begin{array}{c}f(\theta)\end{array}\right] \stackrel{(3.1)}{=} if'(\theta)\begin{array}{c}\pi\\f(\theta)\end{array} \stackrel{(sf)}{=} if'(\theta)\begin{array}{c}\pi\\f(\theta)\end{array} \qquad (3.2)$$

$$\frac{\partial}{\partial\theta}\left[\begin{array}{c}f(\theta)\end{array}\right] \stackrel{(sf,cc,\text{Lem } 3.2)}{=} \frac{\partial}{\partial\theta}\left[\begin{array}{c}f(\theta)\end{array}\right] \stackrel{(3.1,cc,sf)}{=} \frac{if'(\theta)}{\sqrt{2}}\begin{array}{c}\pi\\f(\theta)\end{array} \qquad (3.3)$$

This allows us to differentiate all ZX diagrams with only a single parametrised spider. In general, given some ZX diagram $D(\theta)$ in which n spiders depend on θ, we can always fuse out the parametrised spiders similar to first step in equations (3.2) and (3.3) and obtain a diagram of shape

$$\begin{array}{c}f_1(\theta)\cdots f_n(\theta)\\ \boxed{D'}\end{array}$$

where θ does not occur in D'. However, if we want to differentiate such diagrams using Lemma 3.1, we have to make use of the product rule:

Lemma 3.3 (Product Rule). *Let $D(\theta)$ and $E(\theta)$ be parametrised ZX diagram depending on θ. Then*

$$\frac{\partial}{\partial\theta}\left[\vdots\boxed{D(\theta)}\vdots\boxed{E(\theta)}\vdots\right] = \vdots\boxed{E(\theta)}\vdots \circ \frac{\partial}{\partial\theta}\left[\vdots\boxed{D(\theta)}\vdots\right] + \frac{\partial}{\partial\theta}\left[\vdots\boxed{E(\theta)}\vdots\right] \circ \vdots\boxed{D(\theta)}\vdots$$

$$\frac{\partial}{\partial\theta}\left[\begin{array}{c}\vdots\boxed{D(\theta)}\vdots\\\vdots\boxed{E(\theta)}\vdots\end{array}\right] = \begin{array}{c}\frac{\partial}{\partial\theta}\left[\vdots\boxed{D(\theta)}\vdots\right]\\\vdots\boxed{E(\theta)}\vdots\end{array} + \begin{array}{c}\vdots\boxed{D(\theta)}\vdots\\\frac{\partial}{\partial\theta}\left[\vdots\boxed{E(\theta)}\vdots\right]\end{array}$$

Proof. Directly follows from the product rule for matrix differentiation. $\qquad\square$

This allows us to differentiate diagrams with multiple occurrences of θ, for example

Unfortunately, the ZX-calculus is not well-equipped to deal with such linear combinations of diagrams. In particular, there are no rewrite rules that involve sums, which means that we would need to rewrite and simplify each term separately. Clearly, it would be more convenient if we could express the derivative as a single diagram. Luckily, Wang and Yeung [19] developed a technique to achieve this in ZXW using the W spider:

Theorem 3.4 (Wang and Yeung [19]). *Let $f_1, ..., f_n$ be real differentiable functions and D a ZX diagram. Then*

Proof. Follows from $(\boldsymbol{w})$ and Lemma 3.1. The detailed proof can be found as Theorem 15 in [19]. □

3.2 Differentiating Quantum Circuits

While Theorem 3.4 can be used to obtain the derivative of any ZX diagram, for the purposes of this thesis, we are only interested in the special case of ZX diagrams representing parametrised quantum circuits. We prove a novel, simplified version of Theorem 3.4 for this special case that we will use for the remainder of this thesis.

To motivate the idea, recall that a two-legged green spider corresponds to the R_Z gate up to a global phase:

$$R_Z(\theta) = \begin{pmatrix} e^{-i\frac{\theta}{2}} & 0 \\ 0 & e^{i\frac{\theta}{2}} \end{pmatrix} = e^{-i\frac{\theta}{2}} \; \text{—}\boxed{\theta}\text{—}$$

Using Stone's theorem (see Theorem 2.4), we can see that the derivative of the R_Z gate is given by

$$\frac{\partial}{\partial\theta}R_Z(\theta) = \frac{\partial}{\partial\theta}e^{-i\frac{\theta}{2}Z} = -\frac{i}{2}Ze^{-i\frac{\theta}{2}Z} = -\frac{i}{2}ZR_Z(\theta)$$

$$= -\frac{i}{2}e^{-i\frac{\theta}{2}} \; \text{—}\boxed{\theta}\text{—}\boxed{\pi}\text{—} = -\frac{i}{2}e^{-i\frac{\theta}{2}} \; \text{—}\boxed{\theta+\pi}\text{—}$$

Compared to Lemma 3.1, we obtain the derivative by simply adding π to the phase. Similarly, we obtain an alternative version of Lemma 3.1 by adding a global phase:

Lemma 3.5. *Let f be a differentiable real function. Then*

$$\frac{\partial}{\partial\theta}\left[e^{-i\frac{f(\theta)}{2}}\;\boxed{f(\theta)}\!\!-\right] = -\frac{if'(\theta)}{2}e^{-i\frac{f(\theta)}{2}}\;\boxed{f(\theta)+\pi}\!\!- \tag{3.4}$$

Proof. We have

$$\frac{\partial}{\partial\theta}\left[e^{-i\frac{f(\theta)}{2}}\;\boxed{f(\theta)}\!\!-\right] \stackrel{(2.5)}{=} \frac{\partial}{\partial\theta}\left(e^{-i\frac{f(\theta)}{2}}|0\rangle + e^{i\frac{f(\theta)}{2}}|1\rangle\right)$$

$$= -\frac{if'(\theta)}{2}e^{-i\frac{f(\theta)}{2}}|0\rangle + \frac{if'(\theta)}{2}e^{i\frac{f(\theta)}{2}}|1\rangle$$

$$= -\frac{if'(\theta)}{2}e^{-i\frac{f(\theta)}{2}}(|0\rangle + e^{i(f(\theta)+\pi)}|1\rangle)$$

$$\stackrel{(2.5)}{=} -\frac{if'(\theta)}{2}e^{-i\frac{f(\theta)}{2}}\;\boxed{f(\theta)+\pi}\!\!- \qquad\qquad\square$$

Since we work with quantum circuits, we can ignore global phases as they cancel out because of the doubling construction (see Section 2.3.4). Therefore, we can use Lemma 3.5 instead of Lemma 3.1 to differentiate all spiders in a circuit, yielding a simplified version of Theorem 3.4:

Theorem 3.6. *The derivative of a parametrised quantum circuit can be expressed as the following diagram:*

In other words, we can replace the triangles in Theorem 3.4 with Hadamards.

Proof.

$$\overset{\text{(Lem 3.3,3.4)}}{=} \quad -\frac{if'_1(\theta)}{2} \quad \begin{array}{c} e^{-i\frac{f_1(\theta)}{2}} \;\boxed{f_1(\theta)+\pi} \\ \vdots \\ e^{-i\frac{f_n(\theta)}{2}} \;\boxed{f_n(\theta)} \end{array} \boxed{C} \begin{array}{c} \boxed{-f_1(\theta)}\; e^{i\frac{f_1(\theta)}{2}} \\ \vdots \\ \boxed{-f_n(\theta)}\; e^{i\frac{f_n(\theta)}{2}} \end{array}$$

$$+\frac{if'_1(\theta)}{2} \quad \begin{array}{c} e^{-i\frac{f_1(\theta)}{2}} \;\boxed{f_1(\theta)} \\ \vdots \\ e^{-i\frac{f_n(\theta)}{2}} \;\boxed{f_n(\theta)} \end{array} \boxed{C} \begin{array}{c} \boxed{-f_1(\theta)+\pi}\; e^{i\frac{f_1(\theta)}{2}} \\ \vdots \\ \boxed{-f_n(\theta)}\; e^{i\frac{f_n(\theta)}{2}} \end{array}$$

$$- \quad \dots$$
$$+ \quad \dots$$

$$-\frac{if'_n(\theta)}{2} \quad \begin{array}{c} e^{-i\frac{f_1(\theta)}{2}} \;\boxed{f_1(\theta)} \\ \vdots \\ e^{-i\frac{f_n(\theta)}{2}} \;\boxed{f_n(\theta)+\pi} \end{array} \boxed{C} \begin{array}{c} \boxed{-f_1(\theta)}\; e^{i\frac{f_1(\theta)}{2}} \\ \vdots \\ \boxed{-f_n(\theta)}\; e^{i\frac{f_n(\theta)}{2}} \end{array}$$

$$+\frac{if'_n(\theta)}{2} \quad \begin{array}{c} e^{-i\frac{f_1(\theta)}{2}} \;\boxed{f_1(\theta)} \\ \vdots \\ e^{-i\frac{f_n(\theta)}{2}} \;\boxed{f_n(\theta)} \end{array} \boxed{C} \begin{array}{c} \boxed{-f_1(\theta)}\; e^{i\frac{f_1(\theta)}{2}} \\ \vdots \\ \boxed{-f_n(\theta)+\pi}\; e^{i\frac{f_n(\theta)}{2}} \end{array}$$

$$= \quad -\frac{if'_1(\theta)}{2} \; \begin{array}{c}\boxed{f_1(\theta)+\pi}\\ \vdots \\ \boxed{f_n(\theta)}\end{array}\boxed{C}\begin{array}{c}\boxed{-f_1(\theta)}\\ \vdots \\ \boxed{-f_n(\theta)}\end{array} \;+\; \frac{if'_1(\theta)}{2} \; \begin{array}{c}\boxed{f_1(\theta)}\\ \vdots \\ \boxed{f_n(\theta)}\end{array}\boxed{C}\begin{array}{c}\boxed{-f_1(\theta)+\pi}\\ \vdots \\ \boxed{-f_n(\theta)}\end{array}$$

$$- \quad \dots \quad + \quad \dots$$

$$-\frac{if'_n(\theta)}{2} \; \begin{array}{c}\boxed{f_1(\theta)}\\ \vdots \\ \boxed{f_n(\theta)+\pi}\end{array}\boxed{C}\begin{array}{c}\boxed{-f_1(\theta)}\\ \vdots \\ \boxed{-f_n(\theta)}\end{array} \;+\; \frac{if'_n(\theta)}{2} \; \begin{array}{c}\boxed{f_1(\theta)}\\ \vdots \\ \boxed{f_n(\theta)}\end{array}\boxed{C}\begin{array}{c}\boxed{-f_1(\theta)}\\ \vdots \\ \boxed{-f_n(\theta)+\pi}\end{array}$$

$$= \quad -\frac{if'_1(\theta)}{2}\left(\begin{array}{c}\boxed{f_1(\theta)+\pi}\\ \vdots \\ \boxed{f_n(\theta)}\end{array}\boxed{C}\begin{array}{c}\boxed{-f_1(\theta)}\\ \vdots \\ \boxed{-f_n(\theta)}\end{array} \;-\; \begin{array}{c}\boxed{f_1(\theta)}\\ \vdots \\ \boxed{f_n(\theta)}\end{array}\boxed{C}\begin{array}{c}\boxed{-f_1(\theta)+\pi}\\ \vdots \\ \boxed{-f_n(\theta)}\end{array}\right)$$

$$- \quad \dots$$

$$-\frac{if'_n(\theta)}{2}\left(\begin{array}{c}\boxed{f_1(\theta)}\\ \vdots \\ \boxed{f_n(\theta)+\pi}\end{array}\boxed{C}\begin{array}{c}\boxed{-f_1(\theta)}\\ \vdots \\ \boxed{-f_n(\theta)}\end{array} \;-\; \begin{array}{c}\boxed{f_1(\theta)}\\ \vdots \\ \boxed{f_n(\theta)}\end{array}\boxed{C}\begin{array}{c}\boxed{-f_1(\theta)}\\ \vdots \\ \boxed{-f_n(\theta)+\pi}\end{array}\right)$$

$$\overset{(sf,hh)}{=} \quad -\frac{if'_1(\theta)}{2}2^n\left(\begin{array}{c}\pi\!-\!\boxed{f_1(\theta)}\\ \vdots \\ \circ\!-\!\boxed{f_n(\theta)}\end{array}\boxed{C}\begin{array}{c}\boxed{-f_1(\theta)}\!-\!\circ\\ \vdots \\ \boxed{-f_n(\theta)}\!-\!\circ\end{array} \;-\; \begin{array}{c}\circ\!-\!\boxed{f_1(\theta)}\\ \vdots \\ \circ\!-\!\boxed{f_n(\theta)}\end{array}\boxed{C}\begin{array}{c}\boxed{-f_1(\theta)}\!-\!\pi\\ \vdots \\ \boxed{-f_n(\theta)}\!-\!\circ\end{array}\right)$$

$$- \quad \dots$$

$$-\frac{if'_n(\theta)}{2}2^n\left(\begin{array}{c}\circ\!-\!\boxed{f_1(\theta)}\\ \vdots \\ \pi\!-\!\boxed{f_n(\theta)}\end{array}\boxed{C}\begin{array}{c}\boxed{-f_1(\theta)}\!-\!\circ\\ \vdots \\ \boxed{-f_n(\theta)}\!-\!\circ\end{array} \;-\; \begin{array}{c}\circ\!-\!\boxed{f_1(\theta)}\\ \vdots \\ \circ\!-\!\boxed{f_n(\theta)}\end{array}\boxed{C}\begin{array}{c}\boxed{-f_1(\theta)}\!-\!\circ\\ \vdots \\ \boxed{-f_n(\theta)}\!-\!\pi\end{array}\right)$$

$$\stackrel{(cp)}{=} \quad -2^{n-1}i \cdot \left(\begin{array}{c} \end{array} \right.$$

$$+ \quad \dots$$

$$+ \quad \left. \begin{array}{c} \end{array} \right)$$

$$\stackrel{(w)}{=} \quad -2^{n-1}i$$

$\square$

Thus, in order to differentiate a circuit, we just have to connect the following *differentiation gadget* to the parametrised spiders.

Definition 3.7. *The differentiation gadget is given by the following diagram:*
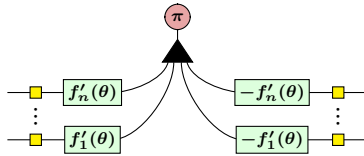
3.3   Properties of the Differentiation Gadget

We close this chapter by deriving some interesting properties of our differentiation gadget. First, we consider the case where all spider have the same phase $f(\theta)$:

Fact 3.8. *Let f be a differentiable real function. Then*

Proof. By Theorem 3.6, the derivative is given by

$$\stackrel{(\boldsymbol{cp})}{=} \quad -f'(\theta) \cdot 2^{n-1} i$$
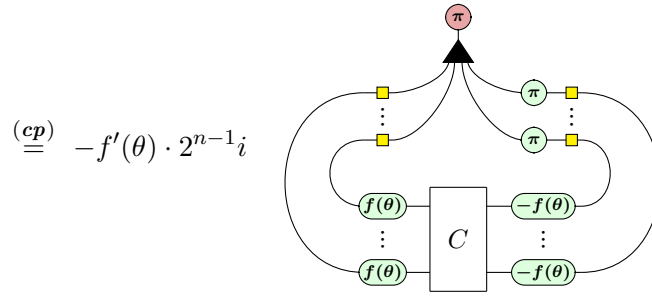
In the last step we also used the fact that $-\boxed{-1}- \;=\; -\bigcirc\pi\bigcirc-$. □

Note that this fact is essentially a version of the chain rule since we have shown

$$\frac{\partial}{\partial \theta} C(f(\theta)) = f'(\theta) \cdot \frac{\partial C}{\partial \theta}(f(\theta)).$$

Another interesting question is how the derivative in Theorem 3.6 behaves if one function f_i is constant, i.e. one of the differentiated spiders does not actually depend on θ. We can graphically show that such a spider does not contribute to the derivative:

Fact 3.9. *Let $f_1, ..., f_n$ be differentiable real functions where f_i is a constant function. Then*
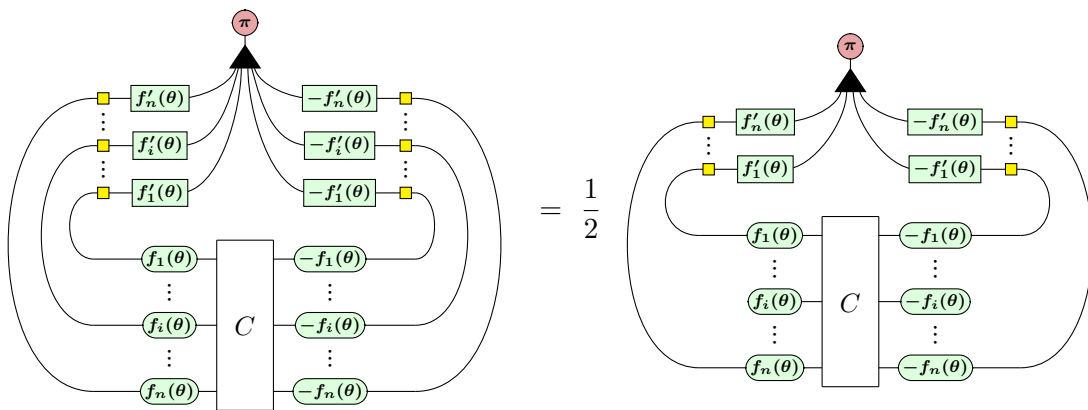
Proof. Note that $f'_i(\theta) = 0$ since f_i is constant. Thus
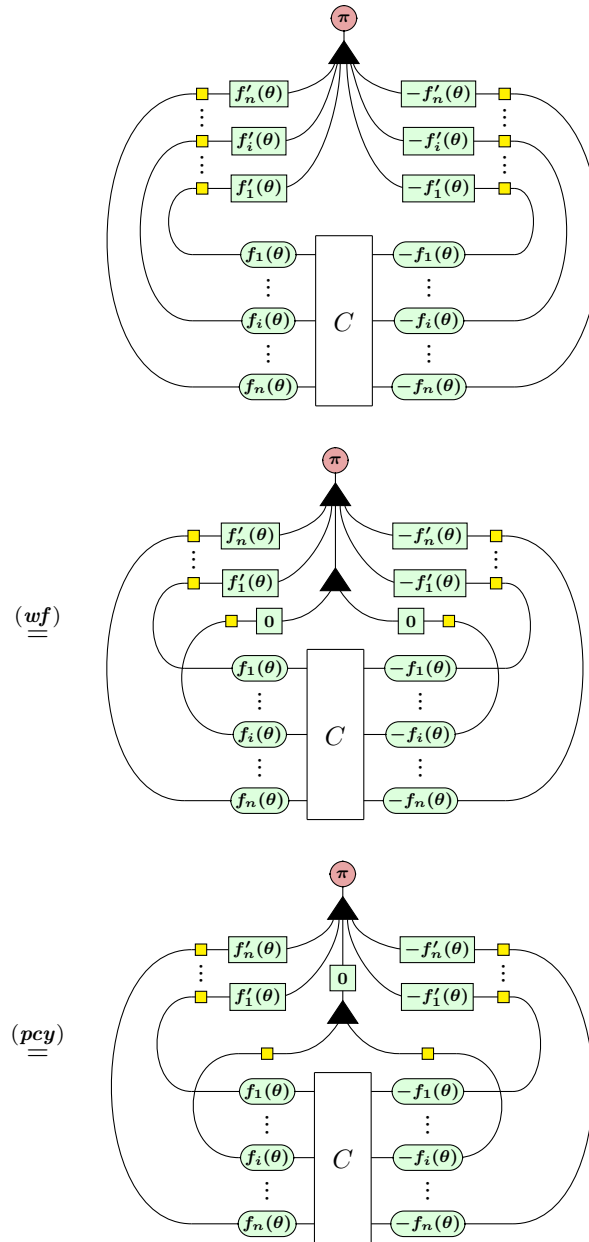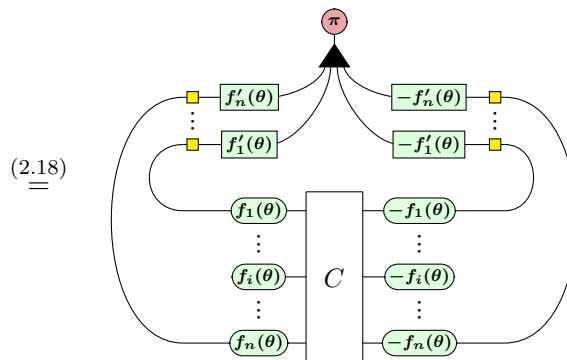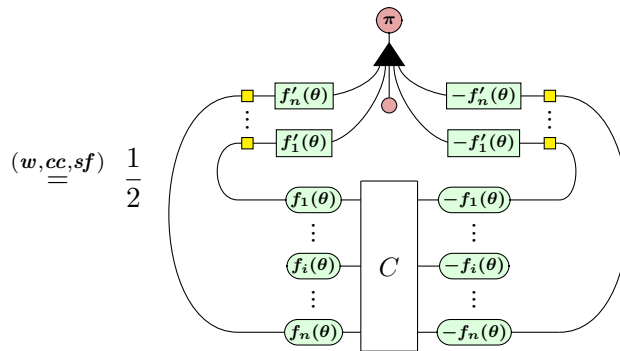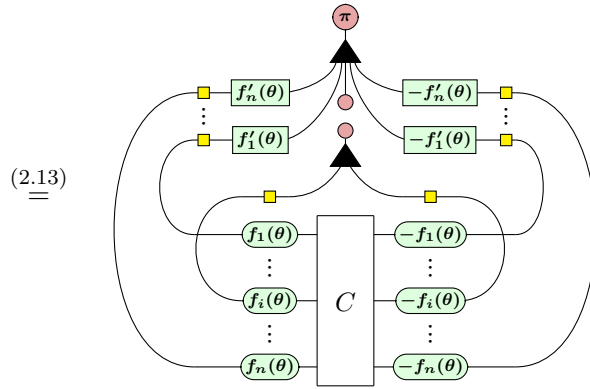
Finally, we emphasize that the ZX diagram representing a given linear map is not unique. In particular, diagrams representing the same parametrised circuit can differ in the number of parametrised spiders. A trivial example of this is evidenced by the spider fusion rule:

On the left-hand side we have 2 parametrised spiders, whereas we have 4 on the right-hand side. This also means that the differentiation gadgets that we plug into either side need to have a different number of legs. Of course, both representations still represent the same linear map. We can verify this graphically by showing that the differentiation gadget respects spider fusion. This requires the following auxiliary lemma:

Lemma 3.10. *For all $a, b \in \mathbb{C}$ we have*

Proof. If $a = 0$, then

If $a \neq 0$, then

Now, it easily follows that the differentiation gadget respects spider fusion:

Fact 3.11. *Let $f_1, ..., f_n, g, h$ be differentiable real functions. Then*

Proof.

$$\overset{(\boldsymbol{wf})}{=}$$

$$\overset{(\boldsymbol{sf},\boldsymbol{cc})}{=} \frac{1}{2}$$

$$\overset{\text{(Lem. 3.10,\textbf{\textit{wf}})}}{=} \quad \frac{1}{2}$$

Chapter 4

Gradient Recipes

This chapter deals with the problem of computing gradients of parametrised quantum circuits. Given a PQC $C(\vec{\theta})$, we are usually interested in the gradient of the expectation value w.r.t. a parameter θ_i, i.e. $\frac{\partial}{\partial \theta_i} \langle H \rangle$ for some Hamiltonian H. While we can represent this gradient as a ZXW diagram (c.f. Theorem 3.6), computing it classically is very hard, akin to simulating the quantum system. Therefore, the gradient computation should be ideally performed on quantum hardware. We can use linearity (and the product rule if multiple gates depend on θ_i) to break the gradient of the expectation value down to gradients of a single gate $U(\theta_i)$ that depends on θ_i: Suppose $C(\theta_i) = EU(\theta_i)D$, then

$$
\frac{\partial}{\partial \theta_i} \langle H \rangle = \frac{\partial}{\partial \theta_i} \langle 0 | D^\dagger U(\theta_i)^\dagger E^\dagger H E U(\theta_i) D | 0 \rangle
$$

$$
= \frac{\partial}{\partial \theta_i} \left[\begin{array}{c} \vdots \boxed{D} \vdots \boxed{U(\theta_i)} \vdots \boxed{E} \vdots \boxed{H} \vdots \boxed{E^\dagger} \vdots \boxed{U(\theta_i)^\dagger} \vdots \boxed{D^\dagger} \vdots \end{array} \right]
$$

$$
\overset{(\text{Lem. 3.2})}{=} \frac{\partial}{\partial \theta_i} \quad \begin{array}{c} \vdots \boxed{D} \vdots \boxed{U(\theta_i)} \vdots \boxed{E} \vdots \boxed{H} \\ \vdots \boxed{\overline{D}} \vdots \boxed{\overline{U(\theta_i)}} \vdots \boxed{\overline{E}} \vdots \end{array} \tag{4.1}
$$

Ideally, we would like to replace this gate with a new sub-circuit that represents the matrix $\frac{\partial}{\partial \theta_i} U(\theta_i)$. Running this modified circuit would then yield the desired gradient

$\frac{\partial}{\partial \theta_i} \langle H \rangle$. Unfortunately, the derivate of a parametrised unitary $U(\theta)$ is usually no longer unitary. For a trivial example of this, consider the $R_Z(\theta)$ gates whose derivate

$$\frac{\partial}{\partial \theta} R_Z(\theta) = \frac{\partial}{\partial \theta} \begin{pmatrix} e^{-i\frac{\theta}{2}} & 0 \\ 0 & e^{i\frac{\theta}{2}} \end{pmatrix} = \begin{pmatrix} -i\frac{\theta}{2} e^{-i\frac{\theta}{2}} & 0 \\ 0 & i\frac{\theta}{2} e^{i\frac{\theta}{2}} \end{pmatrix}$$

is clearly not unitary. One common approach to deal with this issue involves decomposing the gate into a linear combination of k unitaries that can be run on quantum hardware. Thus, computing the gradient in such a way involves k circuit executions. Such decompositions are called *gradient recipes*.

An important detail to note here is that the gate $U(\theta_i)$ occurs doubled in (4.1), matching our previous discussion of quantum circuits in ZX. Thus, we actually have to study decompositions of doubled$(U(\theta_i))$ into a linear combination $\sum_{i=1}^{k} x_i \cdot$ doubled(V_i) of doubled unitaries. Thus, we can use the circuit differentiation machinery from Section 3.2 to analyse this problem.

After discussing a custom ZX representation of parametrised unitaries in Section 4.1, we focus on a popular class of gradient recipes in Section 4.2, the so-called *parameter-shift rules*. We give new proofs of various shift rules based on our diagrammatic gradient representation. Furthermore, we prove a conjecture by Anselmetti et al. [8] establishing that their 4-term recipe is optimal. For this, we prove a no-go theorem lower bounding the number of terms needed to compute gradients of a certain class of circuits in Section 4.2.3. Finally, we remark on a gradient recipe using ancillae in Section 4.3.

4.1 Parametrised Unitaries as ZX Diagrams

In the literature, gradient recipes are usually derived based on properties of the matrices representing the gates. For example, the validity of different parameter-shift rules for a unitary $e^{i\theta H}$ depends on the make-up of the eigenvalues of the Hermitian generator H. The goal of this section is to bridge the gap between this eigenvalue-

description and the higher-level ZX representation of parametrised unitaries. This is necessary for our derived rules to be comparable with the results in the literature.

In order to do this, we have to determine the number of parametrised spiders needed to implement a parametrised unitary $U(\theta)$. Parametrised spiders in this context refer to spiders whose phase is a (non-constant) function in θ. This number is important, because the cost of our recipes will depend on the number of legs that the differentiation gadget for the unitary has. We have already seen that this can vary because of the spider-fusion rule (c.f. Fact 3.11). However, there are also less trivial examples. For instance, consider the CU_1 gate that has the following two representations:

$$CU_1(\theta) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & e^{i\theta} \end{pmatrix} = \quad \raisebox{-1em}{} \quad = \quad \raisebox{-1em}{} \quad (4.2)$$

where $\raisebox{-0.3em}{}$ is the *and-gate* acting like conjunction on the computational basis.

4.1.1 Diagonalising Parametrised Unitaries

First, we reduce the problem to constructing diagrams for diagonal matrices. For this, let us consider an n-dimensional parametrised unitary $U(\theta)$. By Stone's theorem (2.4), we know that $U(\theta) = e^{i\theta H}$ for some self-adjoint n-dimensional matrix H. By the spectral theorem for finite dimensional self-adjoint matrices, H is diagonalisable. This means, there is an orthonormal basis $\mathcal{B} = \{|v_1\rangle, ..., |v_n\rangle\}$ called the *eigenbasis*, satisfying $H|v_j\rangle = \lambda_j|v_j\rangle$ for all j. Here, λ_j are the eigenvalues of H which must all be real since H is self-adjoint. We can now define a unitary $V := \sum_{j=1}^{n} |v_j\rangle\langle j|$ mapping each computational basis element $|j\rangle$ to the corresponding eigenvector $|v_j\rangle$ in the eigenbasis. Furthermore, we define $D := \mathrm{diag}(\lambda_1, ..., \lambda_n)$ as the diagonal matrix consisting of the eigenvalues of H. Noting that we can also write $D = \sum_{j=1}^{n} \lambda_j|j\rangle\langle j|$, we have

$$VDV^\dagger = \sum_{j_1,j_2,j_3=1}^{n} \lambda_{j_2}|v_{j_1}\rangle\langle j_1|j_2\rangle\langle j_2|j_3\rangle\langle v_{j_3}| = \sum_{j=1}^{n} \lambda_j|v_j\rangle\langle v_j| = H.$$

Therefore,

$$U(\theta) = e^{i\theta VDV^\dagger} = Ve^{i\theta D}V^\dagger = V\mathrm{diag}(e^{i\theta\lambda_1},...,^{i\theta\lambda_n})V^\dagger.$$

Note that only the diagonal matrix in the middle depends on the parameter θ. Hence, when trying to determine the number of parametrised spiders needed to implement $U(\theta)$ in the ZX-calculus, it suffices to look at $e^{i\theta D}$.

4.1.2   General Construction

Consider a 2^n-dimensional parametrised unitary $U(\theta) = e^{i\theta H}$ whose Hermitian generator H has m non-zero eigenvalues $\lambda_1,...,\lambda_m \neq 0$ (it does not matter if H has 0 as an additional eigenvalue). In this section, we describe a construction to realise $U(\theta)$ in the ZX-calculus using m parametrised spiders. Following the previous section, we perform a diagonalization $U(\theta) = V^\dagger e^{i\theta D}V$ where D is a diagonal matrix with non-zero entries $\lambda_1,...,\lambda_m$. We define Boolean functions $f_{\lambda_1},...,f_{\lambda_m}$ that match on the eigenvectors corresponding to $\lambda_1,...\lambda_m$:

$$f_{\lambda_j}(\vec{x}) = \begin{cases} 1 & \text{if } D|\vec{x}\rangle = \lambda_j|\vec{x}\rangle \\ 0 & \text{otherwise.} \end{cases}$$

This allows us to characterise the action of $e^{i\theta D}$ on computational basis states as follows:

Lemma 4.1. *For all $x \in \{0,1\}^n$, we have*

$$e^{i\theta D}|\vec{x}\rangle = \left(\prod_{j=1}^{m} e^{i\theta\lambda_j \cdot f_{\lambda_j}(\vec{x})}\right)|\vec{x}\rangle. \tag{4.3}$$

Proof. D is a diagonal matrix whose entries are either zero or one of the non-zero eigenvalues λ_j. Thus, we either have $D|\vec{x}\rangle = \vec{0}$, or $D|\vec{x}\rangle = \lambda_j|\vec{x}\rangle$ for some j.

- Suppose $D|\vec{x}\rangle = \vec{0}$, then $e^{i\theta D}|\vec{x}\rangle = |\vec{x}\rangle$. Furthermore, by definition $f_{\lambda_j}(\vec{x}) = 0$ for all j. Thus, $\left(\prod_{j=1}^m e^{i\theta\lambda_j \cdot f_{\lambda_j}(\vec{x})}\right)|\vec{x}\rangle = \left(\prod_{j=1}^m 1\right)|\vec{x}\rangle = |\vec{x}\rangle = e^{i\theta D}|\vec{x}\rangle$.

- Suppose $D|\vec{x}\rangle = \lambda_j|\vec{x}\rangle$ for some j, then $e^{i\theta D}|\vec{x}\rangle = e^{i\lambda_j\theta}|\vec{x}\rangle$. Furthermore, $f_{\lambda_j}(\vec{x}) = 1$ and $f_{\lambda_k}(\vec{x}) = 0$ for all $k \neq j$. Therefore, $\left(\prod_{j=1}^m e^{i\theta\lambda_j \cdot f_{\lambda_j}(\vec{x})}\right)|\vec{x}\rangle = e^{i\lambda_j\theta}|\vec{x}\rangle = e^{i\theta D}|\vec{x}\rangle$. $\square$

In order to realise this construction diagrammatically, we first need a result regarding the representably of our Boolean functions $f_{\lambda_1}, ..., f_{\lambda_m}$ in the ZX-calculus:

Lemma 4.2. *For every Boolean function $f : \{0,1\}^n \to \{0,1\}$ there is a ZX diagram such that for all $\vec{x} \in \{0,1\}^n$ we have*

Proof. We can express f as a propositional formula in variables $x_1, ..., x_n$ using only conjunction ($\wedge$) and negation ($\neg$) connectives.[1] We can implement this formula as a diagram using gates that act like conjunction, negation, and copying on the computational basis.[2] In ZX calculus, those are given by

By appropriately wiring those gate together, we get the desired ZX representation of f. $\square$

This yields the following construction in the ZX-calculus:

---

[1] This follows from the fact that conjunction and negation form a functionally complete set and can thus encode all possible truth tables [54].

[2] The copying is necessary since inputs might be used multiple times.

Theorem 4.3. *Let $U(\theta) = e^{i\theta H}$ be a parametrised unitary whose Hermitian generator H has m non-zero eigenvalues $\lambda_1, ..., \lambda_m$ and admits the diagonalization $H = V^\dagger D V$. Then*

$$U(\theta) = \quad$$

Proof. It suffices to show that the middle part is equal to $e^{i\theta D}$. We verify this by plugging in a computational basis state $|\vec{x}\rangle$:

$$\overset{(\boldsymbol{cp})}{=} \left(\prod_{j=1}^m e^{i\theta\lambda_j \cdot f_{\lambda_j}(\vec{x})}\right) \quad \overset{x_1\pi}{\underset{x_n\pi}{\vdots}} \quad = \quad \left(\prod_{j=1}^m e^{i\theta\lambda_j \cdot f_{\lambda_j}(\vec{x})}\right)|\vec{x}\rangle \overset{(4.3)}{=} e^{i\theta D}|\vec{x}\rangle. \qquad \square$$

As an example, consider the $CR_Z(\theta)$ and $CU_1(\theta)$ gate, whose Hermitian generators

$$H_{CR_Z} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & -\frac{1}{2} & 0 \\ 0 & 0 & 0 & \frac{1}{2} \end{pmatrix} \qquad H_{CU_1} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

have non-zero eigenvalues $-\frac{1}{2}, \frac{1}{2}$, and 1, respectively. The functions matching on the eigenvectors are given by $f_{-1/2}(x_1, x_2) = x_1 \wedge \neg x_2$ and $f_{1/2}(x_1, x_2) = f_1(x_1, x_2) = x_1 \wedge x_2$. Invoking Theorem 4.3, we get

$$CR_Z(\theta) = \quad$$

To summarise, we constructed an alternate two-spider representation for $CR_Z(\theta)$ different from (2.11) and recovered the one-spider representation of $CU_1(\theta)$ from (4.2).

Note that phase gadgets are a special case of Theorem 4.3 where the function computes the XOR of its inputs. Also note that in general, the construction from Theorem 4.3 is not optimal, in the sense that there might be representations that require less parametrised spiders. For example, consider the parametrised unitary

$$U(\theta) \;=\; \text{[diagram]} \;=\; \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & e^{i\alpha} & 0 & 0 \\ 0 & 0 & e^{2i\alpha} & 0 \\ 0 & 0 & 0 & e^{3i\alpha} \end{pmatrix}$$

whose Hermitian generator has three non-zero eigenvalues. However, we can show that the construction in Theorem 4.3 is in fact optimal if the eigenvectors are $-\lambda, 0, \lambda$:

Proposition 4.4. *It is not possible to represent a parametrised unitary $e^{i\theta H}$ whose Hermitian generator has eigenvalue $-\lambda, 0, \lambda$ with less than two parametrised spiders.*

Proof. See Appendix C. $\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\downarrow$

Furthermore, we discuss an improved optimal construction for unitaries with only two eigenvalues λ_1, λ_2 in the next section.

4.1.3 Special Case for Two Eigenvalues

In the special case where H has only two eigenvalues λ_1, λ_2, it is possible to implement $e^{i\theta H}$ using only a single parametrised spider. In particular, this is the case for all single-qubit unitaries.

Theorem 4.5. *Let $U(\theta) = e^{i\theta H}$ be a parametrised unitary whose Hermitian generator H has only eigenvalues λ_1, λ_2 and admits the diagonalization $H = V^\dagger D V$. Then*

$$U(\theta) = e^{i\lambda_2} \quad \text{[diagram]}$$

Proof. Follows from Theorem 4.3 and the observation that $f_{\lambda_2}(\vec{x}) = \neg f_{\lambda_1}(\vec{x})$ since either $D|\vec{x}\rangle = \lambda_1|\vec{x}\rangle$ or $D|\vec{x}\rangle = \lambda_2|\vec{x}\rangle$:

$$U(\theta) = \quad\text{[diagram]}\quad = \quad\text{[diagram]}$$

$$= e^{i\lambda_2}\quad\text{[diagram]}\quad \overset{(*)}{=}\ e^{i\lambda_2}\quad\text{[diagram]}$$

The step $(*)$ holds since function boxes acting on the computational basis form a bialgebra with the green spider and thus

$$\text{[diagram]} \qquad\qquad \square$$

This construction is optimal since it is clearly not possible to implement parametrised unitaries using zero parametrised spiders.

4.2 Parameter-Shift Rules

The first parameter-shift rule was discovered by Mitarai et al. [6] and extended by Schuld et al. [7]. It says that the derivate of a gate $U(\theta) = e^{i\theta H}$ whose Hermitian generator H has only two eigenvalues λ_1, λ_2 satisfies

$$\frac{\partial}{\partial \theta} U(\theta) = \frac{\lambda_1 - \lambda_2}{2\sin((\lambda_1 - \lambda_2)\alpha)}(U(\theta + \alpha) - U(\theta - \alpha)) \tag{4.4}$$

for an arbitrary shift angle α with $\sin((\lambda_1 - \lambda_2)\alpha) \neq 0$. Thus, computing the gradient requires two evaluations of the circuit on the quantum device with parameter values shifted by $\pm\alpha$. Remarkably, equation (4.4) is an exact representation of the gradient and should not be mistaken for a numerical gradient approximation, which might look similar:

$$\frac{\partial}{\partial \theta} U(\theta) \approx \frac{1}{2h}(U(\theta + h) - U(\theta - h))$$

Unlike this noisy approximation, parameter-shift rules provide an unbiased estimator for the gradient of the expectation value. Hence, they are widely used in practice.[3] There has also been a focus in recent years on finding shift rules for a wider class of gates going beyond two eigenvalues. For example, there is the four-term rule by Anselmetti et al. [8] for Hermitians with eigenvalues $-\lambda, 0, \lambda$ in addition to further generalisations depending on the differences between eigenvalues by Wierichs et al. [9].

In this section, we graphically derive the original rule by Schuld et al. and then move on to gates with more than two eigenvalues.

4.2.1 Two-Term Shift Rule

A diagrammatic proof for a simplified version of Schuld. et al.'s [7] parameter shift rule (4.4) has already been given in [53] and [19]. However, all previous ZX-

---

[3]For example by the QML library `pennylane` [55].

based proofs only derived the special case $\alpha = \frac{\pi}{2}$. Furthermore, they only consider simple rotation gates without generalising to arbitrary parametrised unitaries with two eigenvalues. We extend the proof to derive the two-term shift rule in its most general form:

Lemma 4.6. *For all $\alpha \in \mathbb{R}$ with $\alpha \neq \pi n$ for all $n \in \mathbb{Z}$, we have*

$$-\!\!\boxed{\pi}\!\!-\!\!\boxed{\pi}\!\!- \;=\; \frac{1}{2i\sin(\alpha)}\left(-\!\!\boxed{\alpha}\;\boxed{-\alpha}\!\!- \;-\; -\!\!\boxed{-\alpha}\;\boxed{\alpha}\!\!-\right). \tag{4.5}$$

Proof. See Appendix C. $\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\downarrow$

This allows us to decompose a version of the two-legged differentiation gadget:

Lemma 4.7. *For all $\alpha \in \mathbb{R}$ with $\alpha \neq \pi n$ for all $n \in \mathbb{Z}$, we have*

$$-i \quad \blacktriangle \quad = \quad \frac{1}{2\sin(\alpha)}\left(-\!\!\boxed{\alpha}\;\boxed{-\alpha}\!\!- \;-\; -\!\!\boxed{-\alpha}\;\boxed{\alpha}\!\!-\right). \tag{4.6}$$

Proof. We have

$$-i \quad \blacktriangle \quad \overset{(2.16)}{=} \quad -i -\!\!\boxed{\pi}\!\!-\!\!\boxed{\pi}\!\!- \quad \overset{(cc,hh)}{=} \quad -i -\!\!\boxed{\pi}\!\!-\!\!\boxed{\pi}\!\!- \quad \overset{(\pi)}{=} \quad i -\!\!\boxed{\pi}\!\!-\!\!\boxed{\pi}\!\!-$$

$$\overset{(4.5)}{=} \quad \frac{1}{2\sin(\alpha)}\left(-\!\!\boxed{\alpha}\;\boxed{-\alpha}\!\!- \;-\; -\!\!\boxed{-\alpha}\;\boxed{\alpha}\!\!-\right) \qquad\qquad\qquad \square$$

Combining this with the results from Section 4.1, we obtain the two-term shift rule:

Theorem 4.8 (Schuld et al. [7]). *Every parametrised circuit $C(\theta)$ described by the unitary $e^{i\theta H}$ whose Hermitian generator H has only two eigenvalues λ_1, λ_2 satisfies*

$$\frac{\partial}{\partial\theta}C(\theta) = \frac{\lambda_1 - \lambda_2}{2\sin((\lambda_1 - \lambda_2)\alpha)}\left(C(\theta + \alpha) - C(\theta - \alpha)\right)$$

for all $\alpha \in \mathbb{R}$ with $(\lambda_1 - \lambda_2)\alpha \neq \pi n$ for all $n \in \mathbb{Z}$.

Proof. Using the construction from Theorem 4.5, we can write $C(\theta)$ as
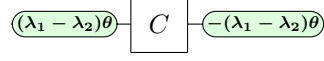
$$\boxed{(\lambda_1 - \lambda_2)\theta} - \boxed{C} - \boxed{-(\lambda_1 - \lambda_2)\theta}$$

Thus, we have

$$\frac{\partial}{\partial\theta}C(\theta) \;=\; \frac{\partial}{\partial\theta}\left[\,\boxed{(\lambda_1 - \lambda_2)\theta} - \boxed{C} - \boxed{-(\lambda_1 - \lambda_2)\theta}\,\right]$$

$$\stackrel{\text{(Fact 3.8)}}{=}\; -i(\lambda_1 - \lambda_2)\;\left[\,\boxed{(\lambda_1 - \lambda_2)\theta} - \boxed{C} - \boxed{-(\lambda_1 - \lambda_2)\theta}\,\right]$$

$$\stackrel{(4.6)}{=}\; \frac{\lambda_1 - \lambda_2}{2\sin((\lambda_1 - \lambda_2)\alpha)}\left(\;\left[\,\boxed{(\lambda_1 - \lambda_2)\theta} - \boxed{C} - \boxed{-(\lambda_1 - \lambda_2)\theta}\,\right]\right.$$
$$\left. -\;\left[\,\boxed{(\lambda_1 - \lambda_2)\theta} - \boxed{C} - \boxed{-(\lambda_1 - \lambda_2)\theta}\,\right]\,\right)$$

$$\stackrel{(sf)}{=}\; \frac{\lambda_1 - \lambda_2}{2\sin((\lambda_1 - \lambda_2)\alpha)}\left(C(\theta + \alpha) - C(\theta - \alpha)\right) \qquad\qquad \square$$

4.2.2 Shift Rules Beyond Two Terms

One way to extend the result from Theorem 4.8 to a wider class of circuits is to invoke the product rule. For example, this gives us

$$\frac{\partial}{\partial\theta}\left[\,\boxed{\begin{matrix}\theta\\\theta\end{matrix}} - \boxed{C} - \boxed{\begin{matrix}-\theta\\-\theta\end{matrix}}\,\right]$$

$$\stackrel{\text{(Lem. 3.3)}}{=}\; \frac{\partial}{\partial\theta}\left[\,\boxed{\begin{matrix}\theta\\\theta\end{matrix}} - \boxed{C} - \boxed{\begin{matrix}-\theta\\-\theta\end{matrix}}\,\right] \;+\; \frac{\partial}{\partial\theta}\left[\,\boxed{\begin{matrix}\theta\\\theta\end{matrix}} - \boxed{C} - \boxed{\begin{matrix}-\theta\\-\theta\end{matrix}}\,\right]$$

$$\stackrel{\text{(Thm. 3.6,4.6)}}{=}\; \frac{1}{2\sin(\alpha)}\left(\boxed{\begin{matrix}\theta+\alpha\\\theta\end{matrix}} - \boxed{C} - \boxed{\begin{matrix}-(\theta+\alpha)\\-\theta\end{matrix}} \;-\; \boxed{\begin{matrix}\theta-\alpha\\\theta\end{matrix}} - \boxed{C} - \boxed{\begin{matrix}-(\theta-\alpha)\\-\theta\end{matrix}}\right) \qquad (4.7)$$
$$+\;\frac{1}{2\sin(\beta)}\left(\boxed{\begin{matrix}\theta\\\theta+\beta\end{matrix}} - \boxed{C} - \boxed{\begin{matrix}-\theta\\-(\theta+\beta)\end{matrix}} \;-\; \boxed{\begin{matrix}\theta\\\theta-\beta\end{matrix}} - \boxed{C} - \boxed{\begin{matrix}-\theta\\-(\theta-\beta)\end{matrix}}\right)$$

The downside of this approach is that it requires the gate to be decomposed such that individual rotation angles can be shifted [56]. This introduces additional overhead if the considered gate is hardware-native. One example of this studied in the literature is the *fsim* gate native to Google's `gmon` architecture [57]. As argued in [58], it is more efficient to use a rule that shifts all parameter occurrences simultaneously, avoiding the depth increase invoked by decomposing *fsim* into elementary gates. Furthermore, it was proven in [9] that rules shifting all gates simultaneously sometimes require less shots to get accurate gradient estimates.

The natural way to extend our proof of Theorem 4.8 is to find decompositions of the differentiation gadget for more than two legs. Unfortunately, the proof of Lemma 4.7 does not scale since the simple representation of the W-state as a red π-spider no longer holds if we add more legs. Instead, we characterise the validity of all possible m-term shift rules for n-legged differentiation gadgets via a system of (complex) polynomial equations:

Lemma 4.9. *For $\vec{\xi}, \vec{\alpha} \in \mathbb{R}^m$, the diagram equation*

holds iff for all $k \in \{0, 1, ..., n\}$ we have

$$\sum_{j=1}^{m} \xi_j \cdot e^{ik\alpha_j} = ki.$$

Proof. The diagram equation holds iff both sides are equal when plugging in computational basis states. First, consider the left-hand side:

$$\overset{(2.20)}{=} -\frac{1}{2}i \;\; \boxed{\substack{\bullet \pi \\ \sum_{j=1}^{n}(-1)^{x_j} - (-1)^{y_j}}} \;\; = \;\; -\frac{1}{2}i \sum_{j=1}^{n}(-1)^{x_j} - (-1)^{y_j}$$

For the right-hand side, we get

$$\sum_{i=1}^{m} \xi_i \;\; \substack{\boxed{x_1\pi}-\boxed{\alpha_i} \;\; \boxed{-\alpha_i}-\boxed{y_n\pi} \\ \vdots \quad\quad \vdots \\ \boxed{x_n\pi}-\boxed{\alpha_i} \;\; \boxed{-\alpha_i}-\boxed{y_1\pi}} \;\; = \;\; \sum_{i=1}^{m} \xi_i \cdot e^{i\alpha_i \sum_{j=1}^{n} x_j - y_j}.$$

Equating both sides yields

$$-\frac{1}{2}i \sum_{j=1}^{n}(-1)^{x_j} - (-1)^{y_j} = \sum_{i=1}^{m} \xi_i \cdot e^{i\alpha_i \sum_{j=1}^{n} x_j - y_j}$$

$$\Leftrightarrow \; i \sum_{j=1}^{n} x_j - y_j = \sum_{i=1}^{m} \xi_i \cdot e^{i\alpha_i \sum_{j=1}^{n} x_j - y_j}$$

since $(-1)^{x_j} - (-1)^{y_j} = -2(x_j - y_j)$ for all $x_j, y_j \in \{0, 1\}$. The equation above must hold for all choices of $\vec{x}, \vec{y} \in \{0, 1\}^n$. Noting that $\sum_{j=1}^{n} x_j - y_j \in \{-n, ..., n\}$, we can represent this more compactly as

$$\sum_{j=1}^{m} \xi_j \cdot e^{\pm i k \alpha_j} = \pm k i$$

for $k \in \{0, ..., n\}$. Finally, we can drop the $\pm$ sign since negating just corresponds to taking the complex conjugate on both sides. $\qquad\square$

This general characterisation of shift rules will be useful for proving a no-go result in Section 4.2.3. However, for the purposes of this section it suffices to look at the special case of symmetric shifts as in Lemma 4.7. This simplifies the system of equations:

Corollary 4.10. *For $\vec{\xi}, \vec{\alpha} \in \mathbb{R}^m$, the diagram equation*

holds iff for all $k \in \{1, ..., n\}$ we have

$$\sum_{j=1}^{m} \xi_j \cdot \sin(k\alpha_j) = \frac{1}{2}k.$$

Proof. Invoking Lemma 4.9, we get the system

$$\sum_{j=1}^{m} \xi_j \left(e^{ik\alpha_j} - e^{\mp ik\alpha_j} \right) = ki \quad \Leftrightarrow \quad 2i \sum_{j=1}^{m} \xi_j \cdot \sin(k\alpha_j) = ki$$

for $k \in \{0, 1, ..., n\}$. Notice that the case $k = 0$ is now trivially satisfied. $\square$

To make the notation more concise, we will write this system of equations in matrix form as

$$\mathbf{S}_{\vec{\alpha}} \cdot \vec{\xi} = \frac{1}{2}\vec{\tau} \tag{4.8}$$

where

$$\mathbf{S}_{\vec{\alpha}} = \begin{pmatrix} \sin(\alpha_1) & ... & \sin(\alpha_m) \\ \sin(2\alpha_1) & ... & \sin(2\alpha_m) \\ \vdots & & \vdots \\ \sin(n\alpha_1) & ... & \sin(n\alpha_m) \end{pmatrix} \qquad \vec{\tau} = \begin{pmatrix} 1 \\ 2 \\ \vdots \\ n \end{pmatrix}$$

If $m = n$, the system is square and solvable under some mild conditions on the α_i. For example, in the case $n = m = 1$, we get the single equation $\xi \sin(\alpha) = \frac{1}{2}$ whose solution $\xi = \frac{1}{2\sin(\alpha)}$ is exactly the shift rule from Lemma 4.7. In the case $n = m = 2$, we get the system

$$\xi_1 \sin(\alpha_1) + \xi_2 \sin(\alpha_2) = \frac{1}{2} \qquad \xi_1 \sin(2\alpha_1) + \xi_2 \sin(2\alpha_2) = 1$$

which for $\alpha_1 \neq \alpha_2$ and $\sin(2\alpha_1), \sin(2\alpha_2) \neq 0$ is solved by

$$\xi_1 = \frac{2\sin(\alpha_2) - \sin(2\alpha_2)}{2(\sin(2\alpha_1)\sin(\alpha_2) - \sin(\alpha_1)\sin(2\alpha_2))}$$

$$\xi_2 = \frac{\sin(2\alpha_1) - 2\sin(\alpha_1)}{2(\sin(\alpha_1)\sin(2\alpha_2) - \sin(2\alpha_1)\sin(\alpha_2))}.$$

This allows us to immediately derive the four-term shift rule given by Anselmetti et al. [8]:

Theorem 4.11 (Anselmetti et al. [8]). *Every parametrised circuit $C(\theta)$ described by the unitary $e^{i\theta H}$ whose Hermitian generator H has eigenvalues $-\lambda, 0, \lambda$ satisfies*

$$\frac{\partial}{\partial\theta}C(\theta) = \xi_1\left(C(\theta + \alpha_1) - C(\theta - \alpha_1)\right) + \xi_2\left(C(\theta + \alpha_2) - C(\theta - \alpha_2)\right)$$

for $\xi_1 = \frac{2\sin(\lambda\alpha_2) - \sin(2\lambda\alpha_2)}{2(\sin(2\lambda\alpha_1)\sin(\lambda\alpha_2) - \sin(\lambda\alpha_1)\sin(2\lambda\alpha_2))}$ and $\xi_2 = \frac{\sin(2\lambda\alpha_1) - 2\sin(\lambda\alpha_1)}{2(\sin(\lambda\alpha_1)\sin(2\lambda\alpha_2) - \sin(2\lambda\alpha_1)\sin(\lambda\alpha_2))}.$

Proof. Using the construction from Theorem 4.3, we can write $C(\theta)$ as
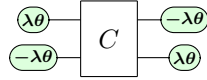
Thus, we have
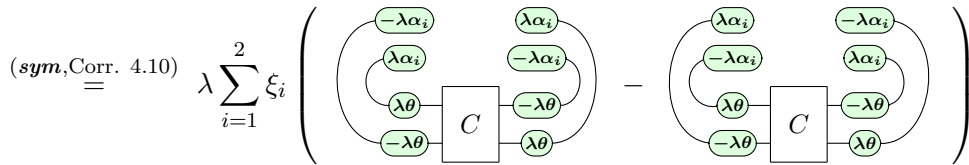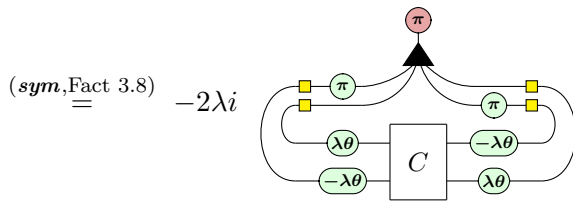
$$\stackrel{(\boldsymbol{sf})}{=} \lambda\xi_1\left(C(\theta + \alpha_1) - C(\theta - \alpha_1)\right) + \lambda\xi_2\left(C(\theta + \alpha_2) - C(\theta - \alpha_2)\right) \qquad \square$$

Similarly, solving the system (4.8) for $m = n = 3, 4, \ldots$ yields $2n$-term shift rules for circuits with more than 2 parametrised spiders. Unfortunately, we are not aware of a closed-form solution for the coefficients $\vec{\xi}$ for arbitrary n and $\vec{\alpha}$. However, if we fix equidistant shift angles $\alpha_j = \frac{j\pi}{n+1}$, we can in fact derive a closed-form solution for $\vec{\xi}$:

Lemma 4.12. *If* $\alpha_j = \frac{j\pi}{n+1}$, *then* $\mathbf{S}_{\vec{\alpha}} \cdot \vec{\xi} = \frac{1}{2}\vec{\tau}$ *has the solution* $\xi_j = \frac{1}{n+1} \sum_{k=1}^{n} k \cdot \sin\left(\frac{kj\pi}{n+1}\right)$.

Proof. For equidistant angles, the equations correspond to a type-I discrete sine transform (DST-I) [59, 60]

$$x_k = \sum_{j=1}^{n} \xi_j \cdot \sin\left(\frac{kj\pi}{n+1}\right)$$

where $x_k = \frac{1}{2}k$. Since the inverse of the DST-I is again given by the DST-I scaled by $\frac{2}{n+1}$, we get

$$\xi_j = \frac{2}{n+1} \sum_{k=1}^{n} x_k \cdot \sin\left(\frac{kj\pi}{n+1}\right) = \frac{1}{n+1} \sum_{k=1}^{n} k \cdot \sin\left(\frac{kj\pi}{n+1}\right). \qquad \square$$

This corresponds to the following generalised parameter-shift rule:

Theorem 4.13. *Let* $C(\theta)$ *be a parametrised circuit that is represented by*
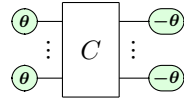
Then

$$\frac{\partial}{\partial \theta} C(\theta) = \lambda \sum_{j=1}^{n} \xi_j \left(C(\theta + \alpha_j) - C(\theta - \alpha_j)\right)$$

if $\vec{\alpha}$ *and* $\vec{\xi}$ *satisfy the equations* $\mathbf{S}_{\vec{\alpha}} \cdot \vec{\xi} = \frac{1}{2}\vec{\tau}$. *One possible solution is given by* $\alpha_j = \frac{j\pi}{n+1}$ *and* $\xi_j = \frac{1}{n+1} \sum_{k=1}^{n} k \cdot \sin\left(\frac{kj\pi}{n+1}\right)$.

Proof. Similar to Theorem 4.11, this follows from Fact 3.8 and Corollary 4.10. □

A similar generalised shift rule has been proven by Wierichs et al. [9]. Their rule requires $2k$ terms where k is the number of unique eigenvalue differences of the Hermitian generator, whereas the cost of our rule depends on the number of parametrised spiders needed to implement the gate. We already established a connection between eigenvalues and ZX diagrams by upper-bounding the number of parametrised spiders by the number of non-zero eigenvalues (c.f. Theorem 4.3). Furthermore, we lower-bounded the number of spiders for eigenvalues $0, \pm\lambda$ (c.f. Proposition 4.4). An interesting future research direction would be to explore whether there are tighter bounds for more general cases and any deeper relationships between eigenvalues and parametrised spiders in diagrams. This could possibly lead to a diagrammatic proof of Wierichs et al.'s version of the generalised shift rule. Note that this might also require decompositions of the differentiation gadget where the spider-phases do not all have the same absolute value as assumed in Theorem 4.13.

4.2.3 Proof of Anselmetti's No-Go Conjecture

A general pattern in parameter shift rules seems to be that the number of terms required is the same as when using the naive approach of the product rule combined with two-term shifts (see equation (4.7)). This holds true for our generalised rule (Theorem 4.13), as well as for Wierichs et al.'s general rule [9] and Anselmetti et al.'s four-term rule [8] (Theorem 4.11). An obvious question at this point is whether we can do any better than that.

As far as we are aware, no results regarding the optimality of shift rules in this sense have been proven in the literature. In particular, Anselmetti et al. [8] conjecture that their four-term rule is optimal, but do not give a proof. In this section, we give the first proof (to our knowledge) of this conjecture. We show that it is indeed impossible to compute gradients for gates whose generators have eigenvalue $-\lambda, 0, \lambda$ using less than four shifts.

For this, we first look at an example. One common gate whose generator has eigenvalues of this shape is $CR_Z(\theta)$. Recall that when calculating derivatives, we always have to work with the doubling construction. Thus, we define $U(\theta) :=$ doubled($CR_Z(2\theta)$) $= CR_Z(2\theta) \otimes \overline{CR_Z(2\theta)}$. We multiply θ by 2 to avoid fractions in the matrix:

$$U(\theta) = \text{diag}(1, 1, e^{i\theta}, e^{-i\theta}, 1, 1, e^{i\theta}, e^{-i\theta}, e^{-i\theta}, e^{-i\theta}, 1, e^{-i\theta}, e^{i\theta}, e^{i\theta}, e^{2i\theta}, 1)$$

The corresponding derivative is thus given by

$$\frac{\partial}{\partial\theta}U(\theta) = \text{diag}(0, 0, ie^{i\theta}, -ie^{-i\theta}, 0, 0, ie^{i\theta}, -ie^{-i\theta}, -ie^{-i\theta}, -ie^{-i\theta}, 0, -ie^{-i\theta}, ie^{i\theta}, ie^{i\theta}, 2ie^{2i\theta}, 0)$$

Suppose we had a three-term shift rule for $U(\theta)$, i.e. $\xi_1, \xi_2, \xi_3, \alpha, \beta, \gamma \in \mathbb{R}$ such that $\frac{\partial}{\partial\theta}U(\theta) = \xi_1 U(\theta + \alpha) + \xi_2 U(\theta + \beta) + \xi_3 U(\theta + \gamma)$. By comparing the matrix elements, this rule would need to satisfy the following equations:

$$\xi_1 + \xi_2 + \xi_3 = 0 \tag{4.9}$$
$$\xi_1 e^{i(\theta+\alpha)} + \xi_2 e^{i(\theta+\beta)} + \xi_3 e^{i(\theta+\gamma)} = ie^{i\theta} \tag{4.10}$$
$$\xi_1 e^{-i(\theta+\alpha)} + \xi_2 e^{-i(\theta+\beta)} + \xi_3 e^{-i(\theta+\gamma)} = -ie^{-i\theta} \tag{4.11}$$
$$\xi_1 e^{2i(\theta+\alpha)} + \xi_2 e^{2i(\theta+\beta)} + \xi_3 e^{2i(\theta+\gamma)} = 2ie^{2i\theta} \tag{4.12}$$

Note that equation (4.10) is redundant since it is the complex conjugate of equation (4.11). Furthermore, multiplying (4.11) with $e^{-i\theta}$ and (4.12) with $e^{-2i\theta}$ yields the following simplified system:

$$\xi_1 + \xi_2 + \xi_3 = 0$$
$$\xi_1 e^{i\alpha} + \xi_2 e^{i\beta} + \xi_3 e^{i\gamma} = i$$
$$\xi_1 e^{2i\alpha} + \xi_2 e^{2i\beta} + \xi_3 e^{2i\gamma} = 2i$$

Surprisingly, this is the exact same system we get in Lemma 4.9 for decomposing the differentiation gadget. We can show that this system is in fact not solvable:

Lemma 4.14. *This system of equations has no solution for $\xi_1, \xi_2, \xi_3, \alpha, \beta, \gamma \in \mathbb{R}$:*

$$\begin{pmatrix} 1 & 1 & 1 \\ e^{i\alpha} & e^{i\beta} & e^{i\gamma} \\ e^{2i\alpha} & e^{2i\beta} & e^{2i\gamma} \end{pmatrix} \begin{pmatrix} \xi_1 \\ \xi_2 \\ \xi_3 \end{pmatrix} = \begin{pmatrix} 0 \\ i \\ 2i \end{pmatrix}$$

Proof. First, note that the system is given by a Vandermonde matrix. Thus, it has full rank if α, β, γ are pairwise distinct angles. In that case, the solution to the system is unique. Using the shorthand $a = e^{i\alpha}, b = e^{i\beta}, c = e^{i\gamma}$, Gaussian elimination yields

$$\xi_1 = -i\frac{b+c-2}{(a-b)(a-c)} \qquad \xi_2 = -i\frac{a+c-2}{(b-a)(b-c)} \qquad \xi_3 = -i\frac{a+b-2}{(c-a)(c-b)}.$$

Suppose that ξ_1, ξ_2, ξ_3 are real. This means that

$$\bar{\xi}_1 \cdot \xi_1 = \xi_1^2$$

$$\Leftrightarrow -\frac{b^{-1}+c^{-1}-2}{(a^{-1}-b^{-1})(a^{-1}-c^{-1})} \cdot \frac{b+c-2}{(a-b)(a-c)} = \frac{(b+c-2)^2}{(a-b)^2(a-c)^2}$$

$$\Leftrightarrow \frac{a^2(b+c-2)(2bc-b-c)}{(a-b)^2(a-c)^2} = \frac{(b+c-2)^2}{(a-b)^2(a-c)^2}$$

$$\Leftrightarrow a^2 = \frac{b+c-2}{2bc-b-c}$$

Note that we have $2bc - b - c \neq 0$ since the equation $2e^{i(\beta+\gamma)} = e^{i\beta} + e^{i\gamma}$ has the only angle solution $\beta = \gamma = 0$ which is ruled out by the assumption that β and γ are distinct angles.

Similarly, we get $b^2 = \frac{a+c-2}{2ac-a-c}$ and $c^2 = \frac{a+b-2}{2ab-a-b}$. Thus, we have a system of quadratic equations which we can solve using a computer algebra system. Using a Mathematica program, we find that $a = b = c = -\sqrt[3]{-1}$ and $a = b = c = (-1)^{\frac{2}{3}}$ are the only solutions. This violates the assumption that α, β, and γ are distinct angles.

Next, we consider the case where the angles are not distinct. W.l.o.g. assume that $\gamma = \alpha$. This means that the last column of the matrix becomes redundant and we can simplify the system to

$$\xi_1 + \xi_2 = 0 \qquad\qquad \xi_1 e^{i\alpha} + \xi_2 e^{i\beta} = i \qquad\qquad \xi_1 e^{2i\alpha} + \xi_2 e^{2i\beta} = 2i$$

Since ξ_1 and ξ_2 are real, we know that the conjugate equations $\xi_1 e^{-i\alpha} + \xi_2 e^{-i\beta} = -i$ and $\xi_1 e^{-2i\alpha} + \xi_2 e^{-2i\beta} = -2i$ also hold. Adding those conjugate equations to the original versions and using $\xi_2 = -\xi_1$ yields $\cos(\alpha) - \cos(\beta) = 0$ and $\cos(2\alpha) - \cos(2\beta) = 0$. This is only satisfied for $\alpha = \beta = \pi$. But then $\xi_1 e^{i\alpha} + \xi_2 e^{i\beta} = 0 \neq i$, which violates the second equation. $\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\square$

With this lemma we have shown two things at once: First, we cannot decompose the four-legged differentiation gadget into less than four shifts. Secondly, $CR_Z(\theta)$ does not satisfy a shift rule with less than four terms.

Now the question is how to extend this result to arbitrary gates with generator eigenvalues $-\lambda, 0, \lambda$? The answer is surprisingly simple: It relies on the fact that each such gate can be used to "simulate" $CR_Z(\theta)$:

Lemma 4.15. *Let $U(\theta) = e^{i\theta H}$ be an n-qubit unitary whose Hermitian generator H has eigenvalue $-\lambda, 0, \lambda$ with corresponding eigenvectors $|\vec{x}_{-\lambda}\rangle$, $|\vec{x}_0\rangle$, and $|\vec{x}_\lambda\rangle$. Define a Boolean function $f : \{0,1\}^2 \to \{0,1\}^n$ by*

$$f(0,0) = \vec{x}_0 \qquad f(0,1) = \vec{x}_0 \qquad f(1,0) = \vec{x}_{-\lambda} \qquad f(1,1) = \vec{x}_\lambda.$$

Then, we have

Proof. We check how the diagram acts on computational basis states:

- For $|00\rangle$ we get $\langle \vec{x}_0 | U(\theta) | \vec{x}_0 \rangle |00\rangle = e^{i \cdot 0 \cdot \frac{\theta}{2\lambda}} |00\rangle = |00\rangle = CR_Z(\theta)|00\rangle$.

- For $|01\rangle$ we get $\langle \vec{x}_0 | U(\theta) | \vec{x}_0 \rangle |01\rangle = e^{i \cdot 0 \cdot \frac{\theta}{2\lambda}} = |01\rangle = CR_Z(\theta)|01\rangle$.

- For $|10\rangle$ we get $\langle \vec{x}_{-\lambda} | U(\theta) | \vec{x}_{-\lambda} \rangle |10\rangle = e^{-i\lambda \frac{\theta}{2\lambda}} |10\rangle = e^{-i\frac{\theta}{2}} |10\rangle = CR_Z(\theta)|10\rangle$.

- For $|11\rangle$ we get $\langle \vec{x}_\lambda | U(\theta) | \vec{x}_\lambda \rangle |11\rangle = e^{i\lambda \frac{\theta}{2\lambda}} |11\rangle = e^{i\frac{\theta}{2}} |11\rangle = CR_Z(\theta)|11\rangle$. $\qquad\square$

This allows us to immediately conclude Anselmetti et al.'s conjecture:

Theorem 4.16 (Anselmetti's No-Go Conjecture). *The shift rule in Theorem 4.11 is optimal, i.e. it is not possible to compute the gradient of gates with generator eigenvalues $-\lambda, 0, \lambda$ using less than four shifts.*

Proof. Suppose there is such a gate $U(\theta)$ that admits a 3-term shift rule. But by Lemma 4.15 this would also yield a 3-term rule for $CR_Z(\theta)$ which we have shown is not possible (Lemma 4.14). $\qquad\square$

Remark 4.17. *This "proof by example" technique also generalises to the optimality of other shift rules. As soon as we can prove that a shift rule is optimal for an example gate, this immediately implies that the rule is optimal for all gates with the same eigenvalues. This could be used to generalise this no-go theorem to capture the cost of shift rules for all gates. The main difficulty lies in characterising for which combinations of n and m the system in Lemma 4.9 is solvable.*

4.3 Ancilla Recipes

One of the initial motivations for using the ZX calculus to study gradient recipes was the hope that the graphical representation of derivatives might make it easier to discover new recipes that possibly go beyond parameter shift rules. Generally, this

requires decomposing the differentiation gadget into doubled maps. We have found the following promising decomposition:

$$
\overset{(wf)}{=} \qquad \overset{(2.16)}{=} \qquad \overset{(sf,pcy)}{=}
$$

$$
\overset{(4.5)}{=} \; \frac{1}{2\sin(\alpha)}\left(\qquad - \qquad \right)
$$

$$
= \; \frac{1}{2\sin(\alpha)}\left(\text{doubled}\left(\qquad \right) \; - \; \text{doubled}\left(\qquad \right) \right) \tag{4.13}
$$

Unfortunately, applying this decomposition to gates yields non-unitary terms in general. The same holds true for other compositions of the gadget we investigated. However, (4.13) suggests a general algorithm to compute gradients using *ancillae*. We illustrate this on the example of two R_Z gates. Note that

$$
\overset{(id,sf)}{=}
$$
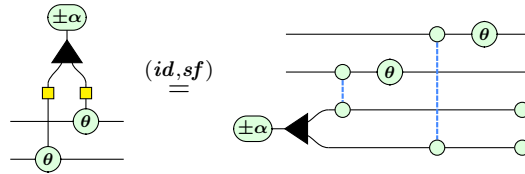
Thus, we can perform this computation on a quantum computer by preparing two ancillae, i.e. extra qubits, in the state $\pm\alpha$◀⊏,[4] connecting them to the original qubits via CZs, and then performing *post-selection*. Post-selection means that we measure and ignore all executions where the outcome is not the one specified in the circuit. This allows us to perform non-unitary operations like the gradient above.

This yields a 2-term recipe for the gate above which would have required four terms using the regular shit rules. Generally, any gate with n parametrised spider can be

---

[4] We discuss how to construct this state in Appendix A

differentiated this way using 2 terms and n ancilla qubits. However, note that each term is more expensive to execute, requiring more shots to get accurate estimates of the expectation value because of the post-selection. Furthermore, the linear requirement of ancillae is a significant limitation of this rule since qubits are a very scarce recourse on current quantum devices. Thus, while theoretically interesting, the practical applicability of this rule is limited and shift rules should probably be preferred.

Chapter 5

Barren Plateaus

After studying how gradients can be computed, we now turn to the question of how the gradient landscape of quantum circuits looks like. A common challenge when training PQCs using gradient-based methods is the so-called *barren plateau phenomenon*. Roughly, it describes the problem that the gradient landscape of many quantum circuits, unlike classical neural networks [40], flattens exponentially with increasing circuit sizes. In other words, the probability that the gradient $\frac{\partial \langle H \rangle}{\partial \theta_i}$ is non-zero to some fixed precision is exponentially small with regards to the number of qubits [10]. As a result of this, gradient-based optimisation becomes increasingly difficult or even numerically impossible. Thus, identifying and studying which circuits exhibit this undesirable behaviour has been a major focus of QML research [10, 11].

Recently, Wang and Yeung proposed a new method to detect barren plateaus using the ZXW-calculus [19]. However, they only demonstrate their method on a trivial example circuit. In this chapter we apply their diagrammatic approach to ansätze actually used in QML research (see Figure 5.1). After formally defining the barren plateau phenomenon in Section 5.1 and the circuits we study in Section 5.2, we introduce a technique to empirically detect barren plateaus in Section 5.3.

- We develop a tool using the QuiZX library [17] in Rust that automatically

computes the variance of the expectation value gradients.

- We use the tool to numerically analyse 7 circuits used by Sim et al. [20] and conclude that they likely all already have barren plateaus for a single layer (Figures 5.3 and 5.4).

- We also study 3 IQP circuits, concluding that two of them likely have barren plateaus while one does not (Figures 5.5, 5.6, and 5.7).

We verify our empirical hypotheses in Section 5.4 by diagrammatically proving the existence of barren plateaus:

- We prove that the first Sim ansatz has barren plateaus even when only using a single layer if we measure on $\Theta(n)$ qubits (Theorem 5.11) and derive similar conditions for the second Sim ansatz (Theorem 5.12).

- We derive a general result that can be used to analyse any single-layer IQP circuit for barren plateaus (Theorem 5.16) and apply it to prove the existence of barren plateaus for 3 single-layer IQP ansätze (Theorem 5.18), including the main ansatz used by the quantum natural language processing library `lambeq` [21] (Theorem 5.19).

- We also prove that one of the IQP ansätze does not have barren plateaus for any number of layers, with the variance converging to a constant in the limit (Corollary 5.25).

Finally, we give a brief overview of barren plateau mitigation techniques presented in the literature in Section 5.5.

5.1 Background

Consider a parametrised quantum circuit $U(\vec{\theta})$ on n qubits and a Hamiltonian H. We assume that the parameters of U are independently and uniformly distributed over the interval $[-\pi, \pi]$ since this is a common initialisation strategy. One can

show that the mean gradient of U's expectation value with regards to H is zero in that case, i.e. $\mathbf{E}\left(\frac{\partial \langle H \rangle}{\partial \theta_i}\right) = 0$ [13]. Now, if furthermore $\mathrm{Var}\left(\frac{\partial \langle H \rangle}{\partial \theta_i}\right) \approx 0$ then it is likely that the training starts in a barren plateau where the gradient $\frac{\partial \langle H \rangle}{\partial \theta_i} \approx 0$. Formally, we say that barren plateaus are present if $\mathrm{Var}\left(\frac{\partial \langle H \rangle}{\partial \theta_i}\right) \in O\left(\frac{1}{2^{\mathrm{poly}(n)}}\right)$, i.e. the variance vanishes exponentially as a function of the number of qubits n. Then, Chebyshev's inequality implies that $\mathrm{Pr}\left(\left|\frac{\partial \langle H \rangle}{\partial \theta_i}\right| \geq \varepsilon\right) \leq \mathrm{Var}\left(\frac{\partial \langle H \rangle}{\partial \theta_i}\right)/\varepsilon^2$. In other words, the probability that the gradient $\frac{\partial \langle H \rangle}{\partial \theta_i}$ is non-zero up to some precision ε is exponentially small in n.

The barren plateau phenomenon was first studied by McLean et al. [10] who proved that barren plateaus appear if an ansatz is sufficiently random such that its parametrisations match the uniform distribution of unitaries (the so-called *Haar-measure*) up to the second moment, i.e. they form a *unitary 2-design*. The distance between the distribution of unitaries generated by an ansatz and the Haar distribution can be seen as a measure for ansatz expressivity since it captures how uniformly an ansatz explores the unitary space [11, 20]. Sim et al. [20] studied the expressiveness of several commonly used ansätze. We will analyse a selection of these in Section 5.3 and Section 5.4. Holmes et al. relate the existence of barren plateaus to the expressiveness of an ansatz [11], showing that more expressive ansätze have flatter gradient landscapes. Concretely, they upper-bound the variance of the gradient in terms of how far an ansatz is from a 2-design, implying a trade-off between ansatz expressiveness and trainability. Interestingly, the existence of barren plateaus also depends on the Hamiltonian H: If we only measure a subset of qubits (i.e. we use a so-called *local cost function*), then some ansätze can avoid barren plateaus up to logarithmic circuit depth in n [12].

Zhao and Gao [13] were the first to employ the ZX-calculus to analyse barren plateaus. They express $\mathrm{Var}\left(\frac{\partial \langle H \rangle}{\partial \theta_i}\right)$ as a linear combination of diagrams with an exponential number of terms, which they handle using tensor networks. Wang and Yeung [19] improve on this by expressing the variance in a single diagram, allowing

the analysis of barren plateaus to be carried out entirely within the framework of ZX. They consider ansätze $U(\vec{\theta})$ where each parameter only occurs a single time and introduce the following notation for the expectation value:

$$\langle H \rangle = \langle 0|U^\dagger(\vec{\theta})HU(\vec{\theta})|0\rangle = \quad =:$$

Since $\mathbf{E}\left(\frac{\partial\langle H\rangle}{\partial\theta_i}\right) = 0$ (a diagrammatic proof of this is given as Lemma 25 in [19]), we have

$$\text{Var}\left(\frac{\partial\langle H\rangle}{\partial\theta_i}\right) = \mathbf{E}\left(\left(\frac{\partial\langle H\rangle}{\partial\theta_i}\right)^2\right) = \frac{1}{(2\pi)^m}\int_{-\pi}^{\pi}\cdots\int_{-\pi}^{\pi}\left(\frac{\partial\langle H\rangle}{\partial\theta_i}\right)^2 d\theta_1...d\theta_m.$$

Using their graphical integration approach, Wang and Yeung express those nested integrals as the following diagram:

Theorem 5.1 (Wang and Yeung [19]).
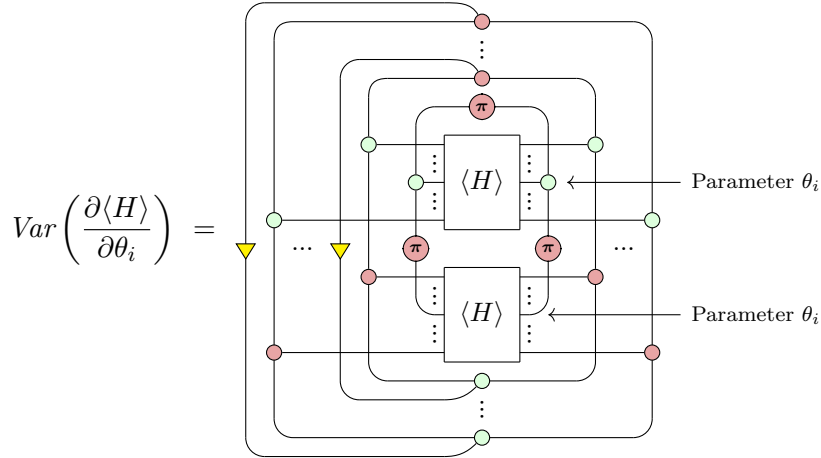
Proof. See Theorem 28 in [19]. $\qquad\square$

However, Wang and Yeung only apply Theorem 5.1 to a small toy circuit with two qubits and four parameters. In particular, they perform no actual barren plateau analysis which would require computing the variance for an arbitrary number of qubits n. The goal of this chapter is to apply Theorem 5.1 to ansätze that are used in practice and characterise when barren plateaus show up.
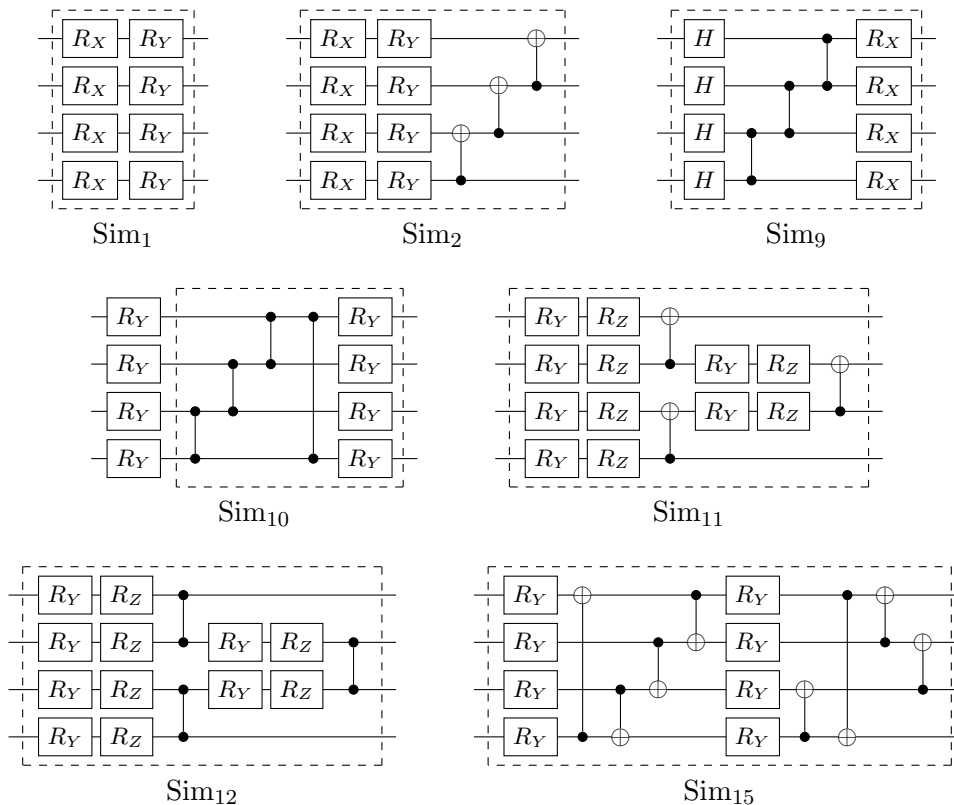
Figure 5.1: Circuits from Sim et al. [20] we study in this chapter. The numbering follows Figure 2 in [20]. The dashed box indicates a single layer that can be repeated multiple times where each layer has unique parameters. We omitted the parameters in the R_X, R_Y, and R_Z gates for brevity.

5.2   Studied Ansätze

In this chapter, we perform barren plateau analyses for two different classes of ansätze. First, we consider a selection of circuits studied by Sim et al. [20]. Concretely, we analyse all ansätze for which Wang and Yeung's [19] ZX-based variance computation from Theorem 5.1 is applicable. They are depicted in Figure 5.1. The remaining circuits in [20] use controlled rotation gates which (as we have proven in Proposition 4.4) need at least two parametrised spiders to be implemented. Thus, Theorem 5.1 does not apply there. Sim et al. [20] calculated the expressiveness of their ansätze, which makes them interesting cases to study as they are good candidates to empirically test the expressiveness vs. trainability trade-off described by Holmes et al [11].
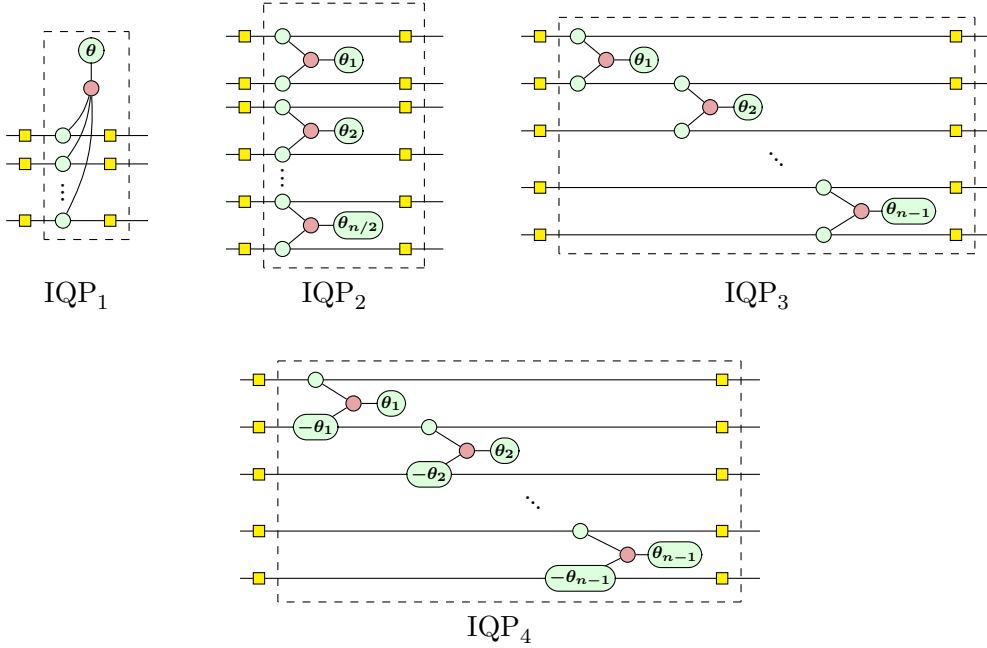
Figure 5.2: Various n-qubit IQP ansätze we study in this chapter given in ZX notation. Similar to Figure 5.1, the dashed boxes can be repeated multiple times, where each layer has unique parameters. Note that IQP_2 is only defined for even n.

Secondly, we study *instantaneous quantum polynomial* (IQP) circuits. First introduced in [61], IQPs consist of layers made up of diagonal gates, separated by columns of Hadamards, i.e.

$$U(\vec{\theta}_1, ..., \vec{\theta}_\ell) = \begin{array}{c}\boxed{H}\\ \vdots \\ \boxed{H}\end{array} \boxed{D(\vec{\theta}_1)} \begin{array}{c}\boxed{H}\\ \vdots \\ \boxed{H}\end{array} \boxed{D(\vec{\theta}_2)} \begin{array}{c}\boxed{H}\\ \vdots \\ \boxed{H}\end{array} \cdots \boxed{D(\vec{\theta}_\ell)} \begin{array}{c}\boxed{H}\\ \vdots \\ \boxed{H}\end{array}$$

where the blocks $D(\vec{\theta}_i)$ only contain gates with diagonal matrices. Thus, all gates that make up $D(\vec{\theta}_i)$ commute with each other. Therefore, it does not matter in which order they are executed which is the reason why this type of ansatz is called *instantaneous*. Remarkably, classical weak simulation of IQP circuits has been shown to be #P-hard [62, 63] and an efficient simulation algorithm would collapse the polynomial hierarchy to the third level [64]. Thus, the simple structure of IQP circuits already captures a quantum advantage, which makes them an interesting class of circuits to study.

For the purpose of this chapter, we use phase gadgets for the diagonal gates that make up the blocks. This is motivated by the fact that they have an elegant representation in ZX an can be nicely reasoned about. Concretely, Figure 5.2 shows the IQP ansätze we study in this chapter. IQP_1 and IQP_2 are of more theoretical interest and will serve as demonstrations for our analytical techniques. On the other hand, IQP_3 has been suggested in [30] and IQP_4 is the default ansatz for the quantum natural language processing (QNLP) library `lambeq` [21]. Thus, the barren plateau analysis for this ansatz is of great practical interest. Recalling (2.11), IQP_4 can be seen as a ladder of CR_Z gates. Also note that each parameter occurs twice in IQP_4 which means that Theorem 5.1 is not directly applicable. However, we can still compute the variance in some special cases which we discuss in Section 5.4.6.

5.3   Numerical Barren Plateau Detection

In this section, we develop a method to empirically test ansätze for barren plateaus by computing $\text{Var}(\frac{\partial \langle H \rangle}{\partial \theta_i})$.

5.3.1   Method

Our numerical barren plateau detection method relies on the following representation of the triangle in ZX calculus:

Lemma 5.2.
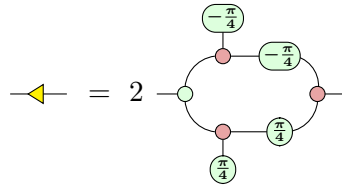
Proof. See Appendix C.   ↓

This allows us to represent the variance from Theorem 5.1 as a Clifford+T diagram[1]

---

[1]A ZX diagram is Clifford+T if all spider phases are multiples of $\frac{\pi}{4}$. The variance diagram is of course only Clifford+T if the ansatz is (not considering parametrised spiders), but this is the case for almost all ansätze used in practice.

which in turn allows us to use the ZX contraction techniques from Kissinger et al. [65] to compute the scalar represented by the diagram. Originally developed for classical simulation of quantum circuits, they employ decompositions of so-called *magic states* and *cat states* to successively simplify ZX diagrams leading to a runtime of $O(2^{\alpha t})$ where $\alpha \approx 0.396$ and t is the number spiders with phase $\pm\frac{\pi}{4}$ or $\pm\frac{3\pi}{4}$.

Thus, contracting the variance diagram from Theorem 5.1 using this method has complexity $O(2^{4\alpha(p-1)})$ where p is the number of parameters in the ansatz. While the runtime scales exponentially, in practice the method is fast enough to handle a wide range of ansätze. Concretely, all experiments in this section combined take roughly two hours to run using a single core on a standard desktop computer equipped with an Intel Core i7-8700k and 16Gb of RAM. Furthermore, the execution speeds up linearly by utilising multiple CPU cores.

We implement the ansätze from Figure 5.1 and 5.2, and the variance diagram from Theorem 5.1 in the Rust programming language using the QuiZX library [17]. Note that QuiZX uses cyclotomic rational numbers [66] and has a special treatment for powers of $\sqrt{2}$. Therefore, all scalars that occur during the ZX contraction can be represented exactly, thus avoiding the imprecisions of floating point arithmetic.

Finally, we want to point out that one could also compute $\text{Var}\left(\frac{\partial\langle H\rangle}{\partial\theta_i}\right)$ by computing the gradient $\frac{\partial\langle H\rangle}{\partial\theta_i}$ for many random parameter samples using the shift rules discussed in Chapter 4 and then compute the numerical variance. However, this would require actually running the circuit on a quantum device or simulator for a large number of shots. Furthermore, this method only yields noisy estimates of the variance (in particular when using a NISQ device) whereas our tool computes exact values for $\text{Var}\left(\frac{\partial\langle H\rangle}{\partial\theta_i}\right)$.

5.3.2 Note on Zero Variance

Before discussing our numerical results, we remark that we sometimes observe $\text{Var}\left(\frac{\partial\langle H\rangle}{\partial\theta_i}\right) = 0$, meaning that the gradient is constant. We show in Section 5.4.2
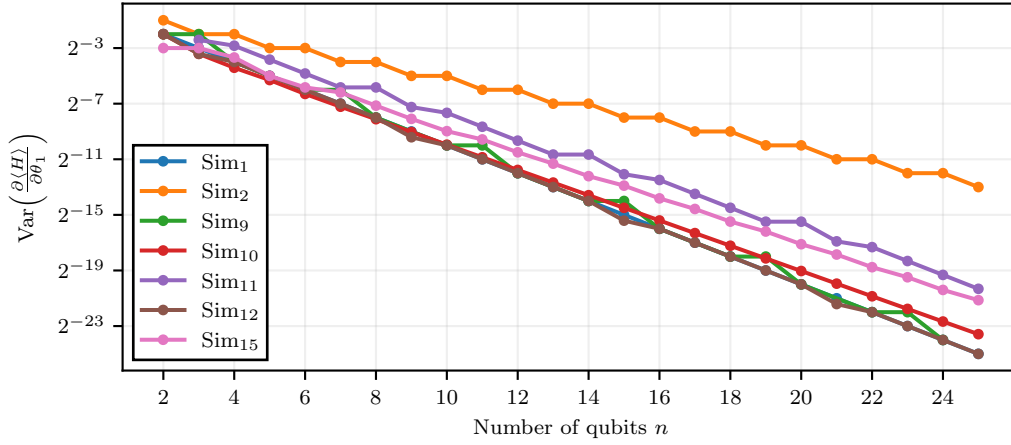
Figure 5.3: Gradient variance as function of qubits for a single layer of the different Sim ansätze for the Hamiltonian $H = Z^{\otimes n}$. Concretely, the variance $\mathrm{Var}(\frac{\partial \langle H \rangle}{\partial \theta_1})$ for the first parameter θ_1 (i.e. the top-left rotations in Figure 5.1) is plotted. Note that the y-axis has a logarithmic scale.

that in those cases the gradient is actually zero, meaning that varying the parameter θ_i does not change $\langle H \rangle$. A trivial example of this is using the Hamiltonian $H = I^{\otimes n}$, i.e. performing no measurement. However, there are also non-trivial cases where some parameters do not influence the expectation value. Training such parameters is of course pointless. Therefore, we can exclude them from the barren plateau analysis. See Remark 5.10 for more details on this.

5.3.3   Results

Sim Ansätze

We begin by analysing the Sim ansätze from Figure 5.1. Figure 5.3 shows the gradient variance for a single layer of the circuits when measuring with the Hamiltonian $H = Z^{\otimes n}$. As we can see, the gradient variance of all ansätze seems to vanish exponentially with increasing n. This suggests that the Sim ansätze have barren plateaus for $H = Z^{\otimes n}$, even when using only a single layer.

However, note that Figure 5.3 only plots the gradient variance w.r.t. the first parameter θ_1. It might be the case that other parameters do not vanish exponentially
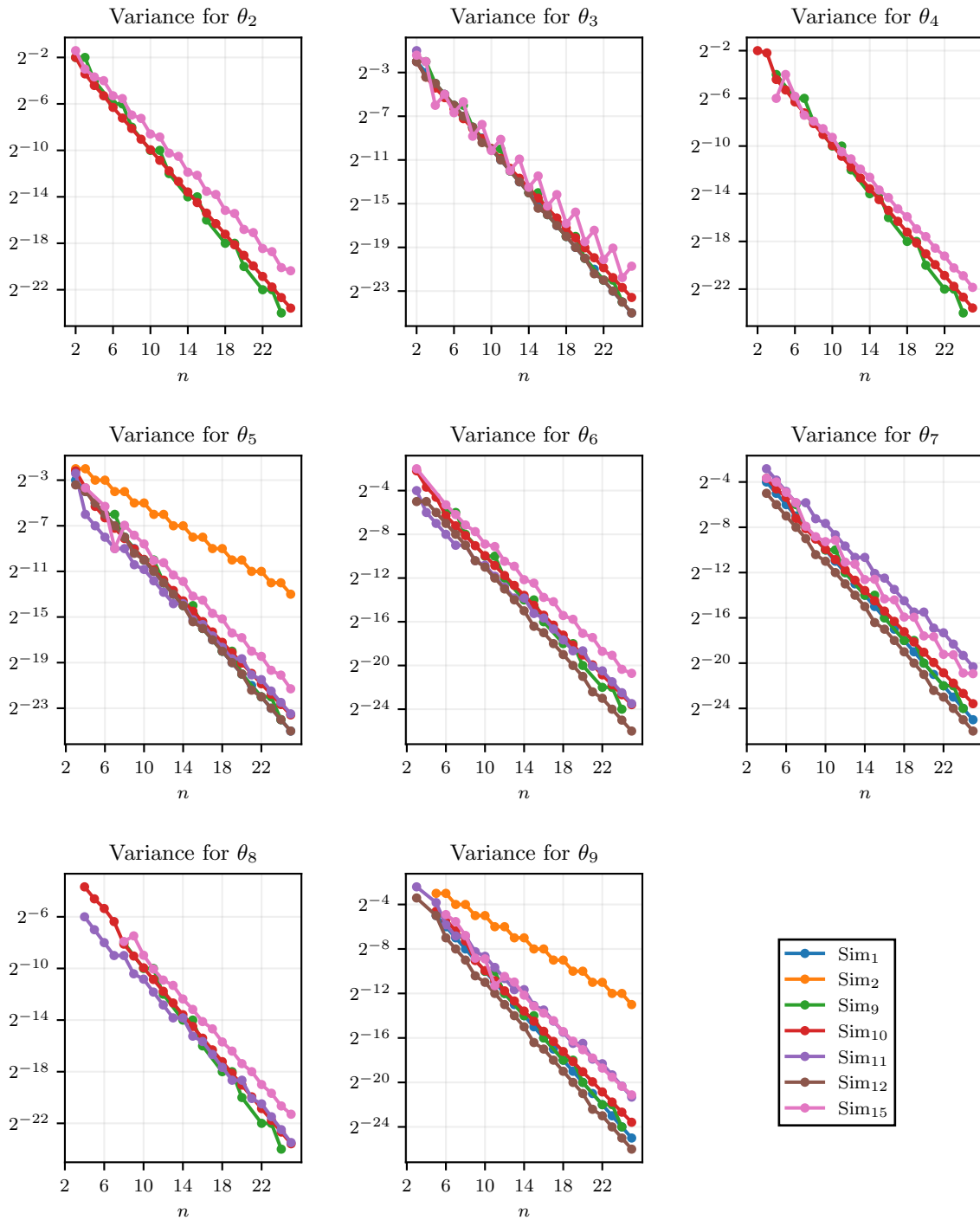
Figure 5.4: Gradient variance for different parameters θ_i as a function of qubits for a single layer of the different Sim ansätze for the Hamiltonian $H = Z^{\otimes n}$. We do not plot points if the parameter does not exist or $\mathrm{Var}(\frac{\partial \langle H \rangle}{\partial \theta_i}) = 0$.

which would make learning possible. To investigate this, we run the same experiment for different parameters θ_i. The results are shown in Figure 5.4. As we can see, as long as $\mathrm{Var}\left(\frac{\partial \langle H \rangle}{\partial \theta_i}\right) \neq 0$ we get exponentially vanishing variances for all cases. This leads us to stating the following hypothesis:

Hypothesis 5.3. *For $H = Z^{\otimes n}$, all single-layer Sim ansätze from Figure 5.1 have barren plateaus on all parameters.*

This matches with the expressiveness results computed by Sim et al. [20]: All ansätze in Figure 5.1 have a similar expressiveness for a single layer. Differences in expressiveness only show up when additional layers are added, with some circuits gaining more expressiveness by this than others. However, our experiments suggest that even a single layer already suffices to generate barren plateaus.

We want to stress that just looking at graphs is of course not a proof for the existence of a barren plateau. It could for example be the case that the curve in Figure 5.3 starts to flatten after some point n_0 outside of the range we investigated. However, the number of qubits used in QML experiments today is limited. Thus, in practical terms, the variance behaviour for small n is most relevant to make statements about the trainability of ansätze. Here, our experiments suggest that the Sim ansätze scale badly and might benefit from using barren plateau mitigation techniques (see Section 5.5).

While our empirical results are of practical use, there is also significant value in formal statements regarding the existence of barren plateaus. We turn to this question in Section 5.4 where we formally analyse Sim_1, Sim_2, and Sim_9 and prove Hypothesis 5.3 for those three ansätze. Furthermore, we generalise to arbitrary Hamiltonians, moving beyond the case $H = Z^{\otimes n}$ considered here.
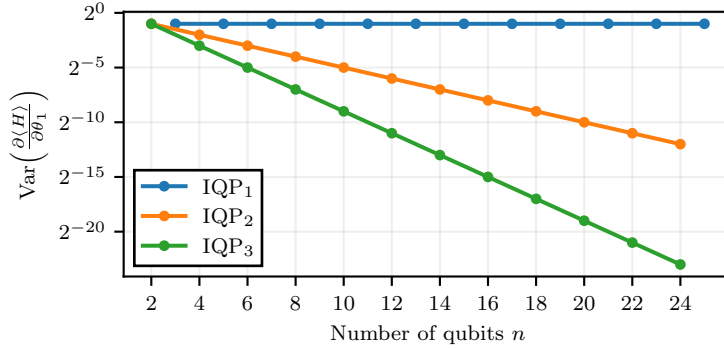
Figure 5.5: Gradient variance as a function of qubits for a single layer of the different IQP ansätze. We do not plot points if $\text{Var}(\frac{\partial \langle H \rangle}{\partial \theta_i}) = 0$.

IQP Ansätze

We run similar experiments for a single layer of IQP_1, IQP_2, and IQP_3.[2] However, while we still use the Hamiltonian $H = Z^{\otimes n}$ for IQP_1, we use an alternating Hamiltonian $H = Y \otimes X \otimes Y \otimes X...$ for IQP_2 and IQP_3. This is because we get $\text{Var}(\frac{\partial \langle H \rangle}{\partial \theta_i}) = 0$ otherwise.[3] The results are shown in Figure 5.5. Similar to the Sim ansätze, IQP_2 and IQP_3 appear to have exponentially vanishing gradient variances. Surprisingly, rerunning the experiment for different parameters θ_i yields the exact same numerical variance values.[4] Thus, we make the following hypothesis:

Hypothesis 5.4. *A single layer of IQP_2 and IQP_3 has barren plateaus on all parameter for $H = Y \otimes X \otimes Y \otimes X...$.*

However, the more interesting observation from Figure 5.5 is that the gradient variance of IQP_1 does not vanish. To investigate whether using more than one layer makes a barren plateau appear, we rerun the IQP_1 experiment for increasing numbers of layers. But the results in Figure 5.6 show that this is not the case. While adding more layers changes the variance, it stays constant with increasing n. Finally, we plot the variance for $n = 3$ as a function of ℓ in Figure 5.7. As we can see,

---

[2]We cannot apply our method to IQP_4 since multiple spiders share the same parameter, however we will derive some theoretical results in Section 5.4.6.

[3]For a theoretical explanation of this see the proof of Theorem 5.18.

[4]We will prove later that this is in fact true for *all* single layer IQP ansätze (see Theorem 5.16 and Remark 5.17).
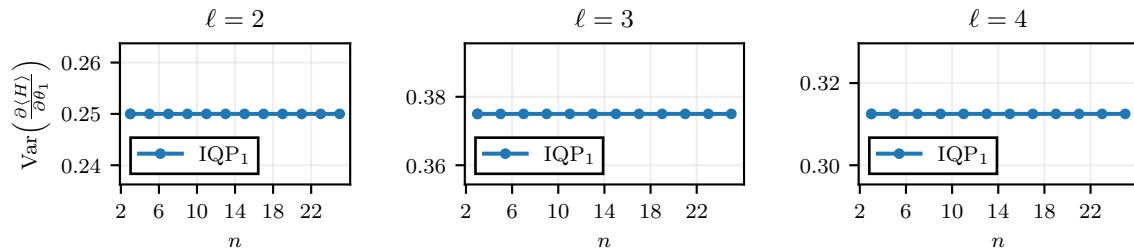
Figure 5.6: Gradient variance of IQP_1 as a function of qubits for different number of layers ℓ. We do not plot points if $\mathrm{Var}(\frac{\partial \langle H \rangle}{\partial \theta_i}) = 0$.
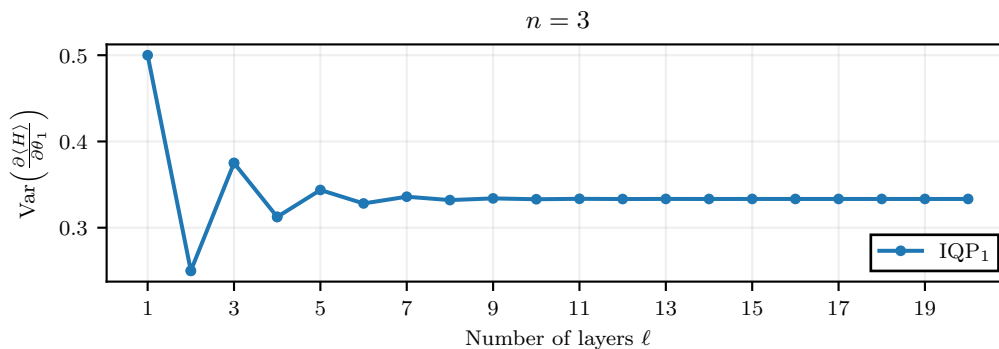

Figure 5.7: Gradient variance of IQP_1 as a function of layers for $n = 3$.

the variance seems to converge with increasing ℓ. To summarise, we can make the following hypothesis:

Hypothesis 5.5. *IQP_1 does not have barren plateaus for $H = Z^{\otimes n}$. More specifically, the variance for θ_1 is constant in n and converges for $\ell \to \infty$.*

Again following the trade-off described by Holmes et al. [11], this observation might be explained by the fact that IQP_1 is a very simple ansatz with limited expressiveness.

We will prove both Hypothesis 5.4 and Hypothesis 5.5 in in the next section (see Theorem 5.18 and Corollary 5.25).

5.4 Analytical Barren Plateau Detection

After investigating the gradient landscape of ansätze numerically, we now turn to the formal analysis of barren plateaus using Theorem 5.1. For this, we introduce a bit of terminology to refer to the structure of the variance diagram from Theorem 5.1: Note that it is made up of two main building blocks which we call *cycles*:
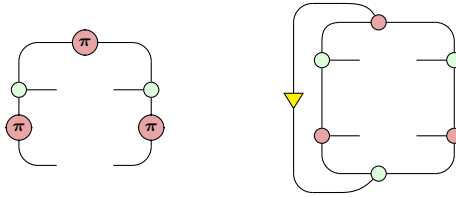
The left cycle is plugged into the positions corresponding to the variance parameter θ_i. The right cycle with the triangle is plugged into every other position, corresponding to parameters θ_j with $j \neq i$. The remainder of this section will largely be concerned with simplifying those kinds of cycles for different expectation value diagrams plugged in the middle. This will allow us to contract the diagram and obtain a numerical value for the variance as a function of n or ℓ.

We make use of the following two lemmas throughout this section:

Lemma 5.6. *For all $x, y \in \{0, 1\}$ we have*

$$(5.1)$$

Proof. See Appendix C. ↓

Lemma 5.7.

$$(5.2)$$

Proof. See Appendix C. ↓

5.4.1   Introductory Example

Before discussing the Sim and IQP ansätze, we first show how to diagrammatically
compute the variance of a smaller example ansatz:

For the Hamiltonian $H = X^{\otimes n}$, we get the following expectation value:

$$\overset{(\mathbf{sf})}{=} \begin{cases} 0 & \text{if } n \text{ is even} \\[2em] \frac{1}{4} \; \text{[diagram]} & \text{if } n \text{ is odd} \end{cases}$$
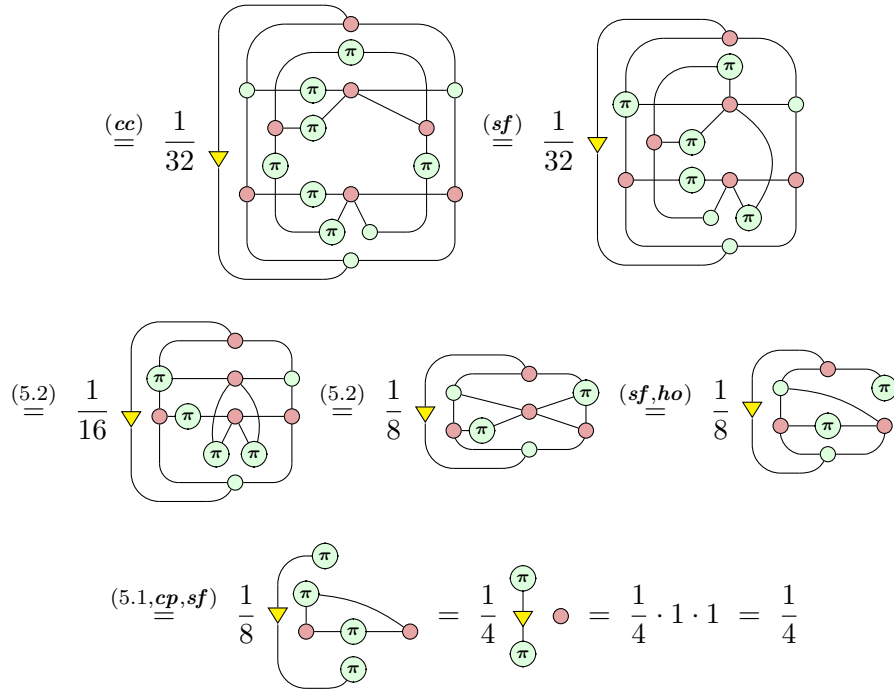
If n is even, we have $\langle H \rangle = 0$ and thus $\frac{\partial \langle H \rangle}{\partial \theta_1} = \frac{\partial \langle H \rangle}{\partial \theta_2} = 0$ such that $\mathrm{Var}\left(\frac{\partial \langle H \rangle}{\partial \theta_1}\right) = \mathrm{Var}\left(\frac{\partial \langle H \rangle}{\partial \theta_2}\right) = 0$. If n is odd, we can calculate the variance of $\frac{\partial \langle H \rangle}{\partial \theta_i}$ diagrammatically using Theorem 5.1:

$$\mathrm{Var}\left(\frac{\partial \langle H \rangle}{\partial \theta_1}\right) \;=\; \frac{1}{16}\;\text{[diagram]} \quad\overset{(\mathbf{sf},5.1)}{=}\quad \frac{1}{16}\;\text{[diagram]}$$

$$\overset{(\mathbf{id},\boldsymbol{\pi},\mathbf{sf})}{=} -\frac{1}{16}\;\text{[diagram]} \quad\overset{(\mathbf{sf},\mathbf{id})}{=}\quad -\frac{1}{8}\;\text{[diagram]}$$

$$\overset{(\mathbf{cc})}{=} -\frac{1}{8}\;\text{[diagram]} \quad\overset{(\boldsymbol{\pi},\mathbf{sf})}{=}\quad -\frac{1}{8}\;\text{[diagram]} \quad\overset{(\mathbf{sf},\mathbf{ho})}{=}\quad -\frac{1}{8}\;\text{[diagram]}$$

$$\overset{(\mathbf{tri})}{=} -\frac{1}{8}\;\text{[diagram]} \;=\; -\frac{1}{8}\cdot(-1)\cdot 2 \;=\; \frac{1}{4}$$

Similarly, for $\frac{\partial \langle H \rangle}{\partial \theta_2}$ we get

$$\mathrm{Var}\left(\frac{\partial \langle H \rangle}{\partial \theta_2}\right) \;=\; \frac{1}{16}\;\text{[diagram]} \quad\overset{(\mathbf{id},\mathbf{cc})}{=}\quad \frac{1}{32}\;\text{[diagram]}$$

$$\stackrel{(\textbf{\textit{cc}})}{=} \frac{1}{32} \quad \blacktriangledown \qquad\qquad \stackrel{(\textbf{\textit{sf}})}{=} \frac{1}{32} \quad \blacktriangledown$$

$$\stackrel{(5.2)}{=} \frac{1}{16} \quad \blacktriangledown \qquad\qquad \stackrel{(5.2)}{=} \frac{1}{8} \quad \blacktriangledown \qquad\qquad \stackrel{(\textbf{\textit{sf}},\textbf{\textit{ho}})}{=} \frac{1}{8} \quad \blacktriangledown$$

$$\stackrel{(5.1,\textbf{\textit{cp}},\textbf{\textit{sf}})}{=} \frac{1}{8} \quad \blacktriangledown \qquad = \frac{1}{4} \quad \blacktriangledown \qquad = \frac{1}{4} \cdot 1 \cdot 1 = \frac{1}{4}$$

In both cases, the variance of the gradient does not vanishes exponentially. Thus, we can conclude that the barren plateau phenomenon does not appear in this ansatz when measuring using the Hamiltonian $H = X^{\otimes n}$. The diagrammatic calculation in this example was relatively straightforward since the ansatz $U(\theta_1, \theta_2)$ has a fixed number of parameters, independent of the number of qubits n. Next, we will consider ansätze where the number of parameters increases when increasing n.
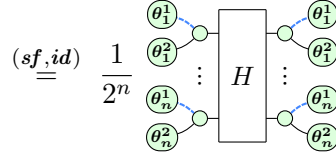
5.4.2   Sim 1

A single layer of Sim_1 can be represented in the ZX-calculus as

$$\mathrm{Sim}_1(\vec{\theta}) \;=\; \begin{matrix} \theta_1^1 & \theta_1^2 \\ \vdots & \\ \theta_n^1 & \theta_n^2 \end{matrix}$$

Given some Hamiltonian H, the corresponding expectation value is given by

$$\langle H \rangle \;=\; \boxed{H} \quad \stackrel{(\textbf{\textit{sf}},\textbf{\textit{cc}})}{=} \frac{1}{2^n} \quad \boxed{H}$$

$$(\underset{=}{\boldsymbol{sf},\boldsymbol{id}}) \quad \frac{1}{2^n}$$

Recall that each Hamiltonian can be written as a sum of Pauli strings $\{X, Y, Z, I\}^{\otimes n}$. Thus, it suffices to compute $\mathrm{Var}\left(\frac{\partial\langle H\rangle}{\partial\theta_i}\right)$ for $H = P_1 \otimes ... \otimes P_n$ where $P_j \in \{X, Y, Z, I\}$. To make the following derivations more concise, we represent all three cases in a single diagram

$$\boxed{P_j} \quad = \quad i^{a_j b_j} \; \boxed{a_j \pi}\;\boxed{b_j \pi}$$

where

$$a_j = \begin{cases} 1 & \text{if } H_j = Y, Z \\ 0 & \text{if } H_j = X, I \end{cases} \qquad b_j = \begin{cases} 1 & \text{if } H_j = X, Y \\ 0 & \text{if } H_j = Z, I. \end{cases}$$

Thus, we can write the expectation value as

$$\langle H\rangle \quad = \quad \frac{i^{\sum a_j b_j}}{2^n}$$

Next, we consider the different types of cycles that show up in the variance diagram:

Lemma 5.8. *We have*

$$(-1)^{a_i b_i} \quad = \quad \begin{cases} 0 & \text{if } H_i = I \\ 1 & \text{if } H_i = Y, X \\ 2 & \text{if } H_i = Z \end{cases}$$

$$(-1)^{a_i b_i} \quad = \quad \begin{cases} 0 & \text{if } H_i = Z, I \\ 1 & \text{if } H_i = X, Y \end{cases}$$

$$(-1)^{a_i b_i} \quad \text{[diagram]} \quad = \begin{cases} 1 & \text{if } H_i = X \\ 2 & \text{if } H_i = Z, Y \\ 4 & \text{if } H_i = I \end{cases}$$

Proof. See Appendix C.   ↓

This leads to the following result regarding the variance of the gradients:

Fact 5.9. *Let* $h_P = |\{H_j \mid H_j = P, j \in \{1, ..., n\} \setminus \{i\}\}|$ *be the number of times the Pauli* $P \in \{X, Y, Z, I\}$ *occurs in* H, *excluding the position* H_i. *Then*

$$Var\left(\frac{\partial \langle H \rangle}{\partial \theta_i^1}\right) = \begin{cases} 0 & \text{if } H_i = I \\ \frac{1}{4^n} \cdot 2^{h_Z + h_Y} \cdot 4^{h_I} & \text{if } H_i = X, Y \\ \frac{2}{4^n} \cdot 2^{h_Z + h_Y} \cdot 4^{h_I} & \text{if } H_i = Z \end{cases}$$

$$Var\left(\frac{\partial \langle H \rangle}{\partial \theta_i^2}\right) = \begin{cases} 0 & \text{if } H_i = Z, I \\ \frac{1}{4^n} \cdot 2^{h_Z + h_Y} \cdot 4^{h_I} & \text{if } H_i = X, Y \end{cases}$$

Proof. We start with the first equation where the gradient is w.r.t. θ_i^1. By Theorem 5.1, $\text{Var}\left(\frac{\partial \langle H \rangle}{\partial \theta_i^1}\right)$ is given by

$$(-1)^{\sum a_j b_j}$$
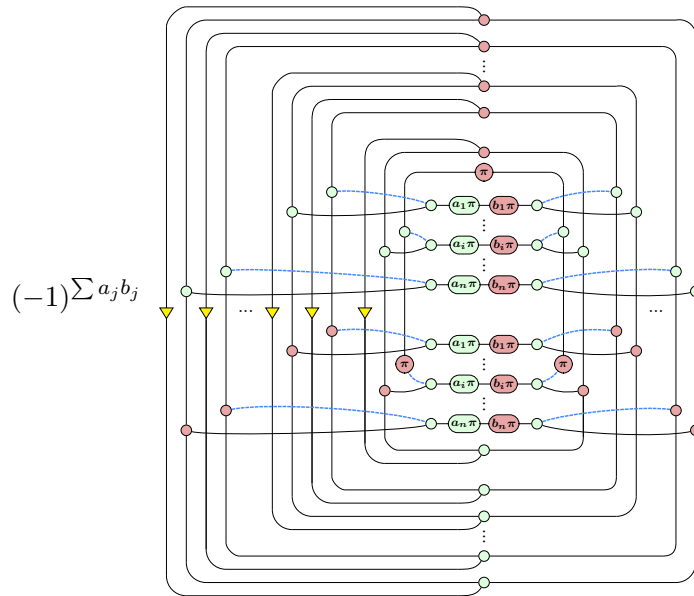
Note that the different cycles are not connected with each other, which means we can arrange them as follows:
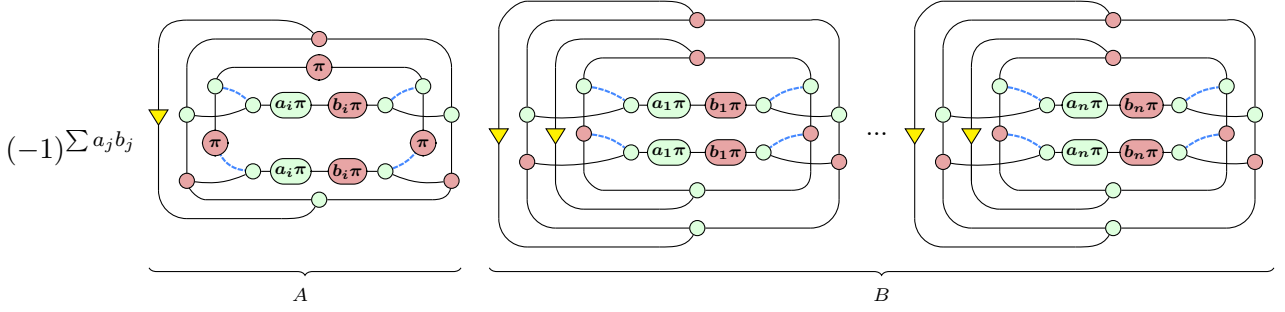
By Lemma 5.8, we have

$$A = \begin{cases} 0 & \text{if } H_i = I \\ 1 & \text{if } H_i = Y, X \\ 2 & \text{if } H_i = Z \end{cases} \qquad B = \prod_{\substack{j=1 \\ j \neq i}}^{n} \begin{cases} 1 & \text{if } H_j = X \\ 2 & \text{if } H_j = Z, Y \\ 4 & \text{if } H_j = I \end{cases}$$

$$= 1^{h_X} \cdot 2^{h_Z + h_X} \cdot 4^{h_I}$$

such that $A \cdot B$ corresponds to desired equation. The proof for $\mathrm{Var}\left(\frac{\partial \langle H \rangle}{\partial \theta_i^2}\right)$ is analogous. $\qquad \square$

Remark 5.10. *One might wonder why the variance is zero in some of the cases. If $H_i = I$, this corresponds to performing no measurement on qubit i. In this case, the value $\langle H \rangle$ actually does not depend on θ_i^1 and θ_i^2 since*

Therefore, we have $\frac{\partial \langle H \rangle}{\partial \theta_i^1} = \frac{\partial \langle H \rangle}{\partial \theta_i^2} = 0$ and thus $\mathrm{Var}\left(\frac{\partial \langle H \rangle}{\partial \theta_i^1}\right) = \mathrm{Var}\left(\frac{\partial \langle H \rangle}{\partial \theta_i^2}\right) = 0$. Similarly, if $H_i = Z$ we have

such that θ_i^2 does not contribute to the expectation value and thus $\mathrm{Var}\left(\frac{\partial \langle H \rangle}{\partial \theta_i^2}\right) = 0$.

Finally, Fact 5.9 immediately yields the condition for Sim_1 to have a barren plateau:

Theorem 5.11. *The barren plateau phenomenon appears in Sim_1 if we measure on $\Theta(n)$ qubits. In particular, this implies Hypothesis 5.3 for Sim_1.*
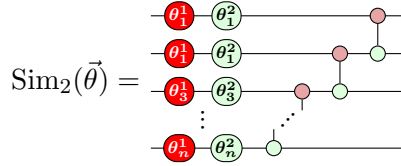
Proof. By Fact 5.9, the variance for all parameters (ignoring the scalar 2 in the case θ_i^1) is either 0 or

$$\frac{2^{h_Z+h_Y} \cdot 4^{h_I}}{4^n} = \frac{1}{2^{2n-h_Z-h_Y-2h_I}} = \frac{1}{2^{2h_X+h_Y+h_Z}}$$
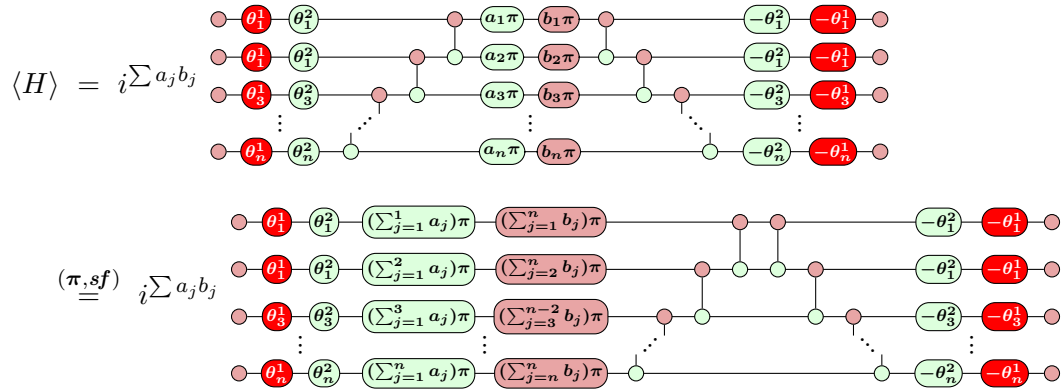
Since we measure on $\Theta(n)$ qubits, we must have $2h_X + h_Y + h_Z = \Theta(n)$ such that the variance vanishes exponentially. $\square$
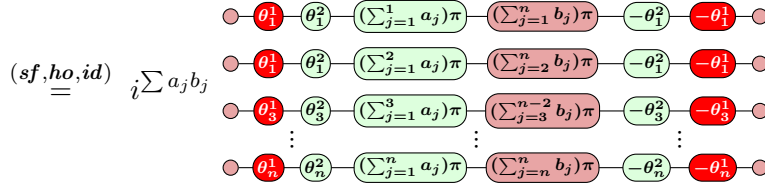
5.4.3 Sim 2

A single layer of Sim_2 can be represented in the ZX-calculus as

Using the same representation for a Hamiltonian H as in Section 5.4.2, we can write the expectation value as

$$\overset{(\mathbf{sf},\mathbf{ho},\mathbf{id})}{=} \quad i^{\sum a_j b_j}$$

Notice that this diagram has the same shape as the expectation value for Sim_1. In fact, the only difference is the Hamiltonian in the middle, which in this case is given by

$$H_i' = \left(\sum_{j=i}^{n} b_j\right) X \cdot \left(\sum_{j=1}^{i} a_j\right) Z.$$

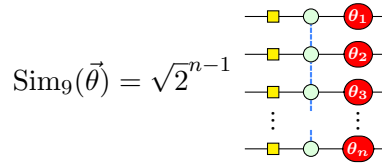In other words, $\langle H \rangle_{\text{Sim}_2} = \langle H' \rangle_{\text{Sim}_1}$. Thus, we can use Theorem 5.11 to characterise the barren plateaus in Sim_2:

Theorem 5.12. *The barren plateau phenomenon appears in Sim_2 if $H_i' \neq I$ at $\Theta(n)$ positions. In particular, this implies Hypothesis 5.3 for Sim_2.*
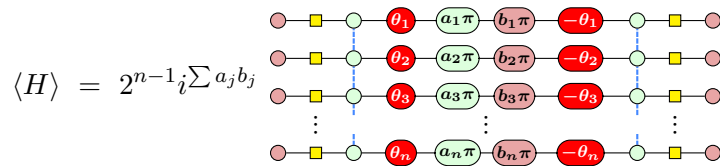
Proof. Follows from Theorem 5.11. Note that for $H = Z^{\otimes n}$ we have $H' = Z \otimes I \otimes Z \otimes I...$ such that the theorem applies and Hypothesis 5.3 is true. $\qquad \square$

5.4.4 Sim 9

Sim_9 can be represented in the ZX-calculus as

$$\text{Sim}_9(\vec{\theta}) = \sqrt{2}^{n-1}$$

yielding the expectation value

$$\langle H \rangle = 2^{n-1} i^{\sum a_j b_j}$$

As before, we consider how the cycles simplify:

Lemma 5.13. *We have*

Proof. See Appendix C.   ↓

Unfortunately, this makes it difficult to give a closed-form expression of the gradient variance in terms of a general Hamiltonian H as we have done in Fact 5.9. However, we can easily investigate concrete instances. For example we can verify Hypothesis Hypothesis 5.4 for Sim_9:

Theorem 5.14. *Hypothesis 5.3 holds for Sim_9, i.e. Sim_9 has barren plateaus for $H = Z^{\otimes n}$.*

Proof. By Theorem 5.1, we have

$$\mathrm{Var}\left(\frac{\partial \langle H \rangle}{\partial \theta_i}\right) = \frac{1}{2^{2n+2}}$$

Using Lemma 5.13, we can simplify this to

$$\frac{2^{n-1}}{2^{2n+2}} \quad \overset{(2.14)}{=} \quad \frac{2^{n-1}}{2^{3n+2}} \sum_{\vec{x} \in \{0,1\}^n}$$

Each of those "lines" can only represent the scalars 0, ± 1, and ± 2. Thus

$$\mathrm{Var}\left(\frac{\partial \langle H \rangle}{\partial \theta_i}\right) \leq \frac{2^{n-1}}{2^{3n+2}} \sum_{\vec{x} \in \{0,1\}^n} 16 = \frac{2^{n-1}}{2^{3n+2}} \cdot 2^n \cdot 16 = \frac{1}{2^{n-1}}$$

such that we have a barren plateau. $\qquad\square$

5.4.5 Single-Layer IQP Ansätze

After discussing some of the Sim ansätze, we now move to IQPs. In this section, we prove a general result that allows us to compute the gradient variance of any single-

layer IQP circuit with single parameter occurrences. To motivate our approach, we
first look at an example IQP circuit:

$$
U(\vec{\theta}) \;=\; \tag{5.3}
$$

The corresponding diagram for the expectation value is given by

$$
\langle H \rangle \;=\; i^{\sum a_j b_j}
$$

$$
\overset{(cc,hh)}{=}\; \frac{i^{\sum a_j b_j}}{2^4}
$$

$$
\overset{(\boldsymbol{\pi},sf)}{=}\; \frac{i^{\sum a_j b_j}}{2^4}
$$

$$
\overset{(sf)}{=}\; \frac{i^{\sum a_j b_j}}{2^4}
$$

The main insight is that the cycles for diagrams of this shape simplify nicely:

Lemma 5.15. *The cycles from single-layer IQP circuits simplify as follows:*
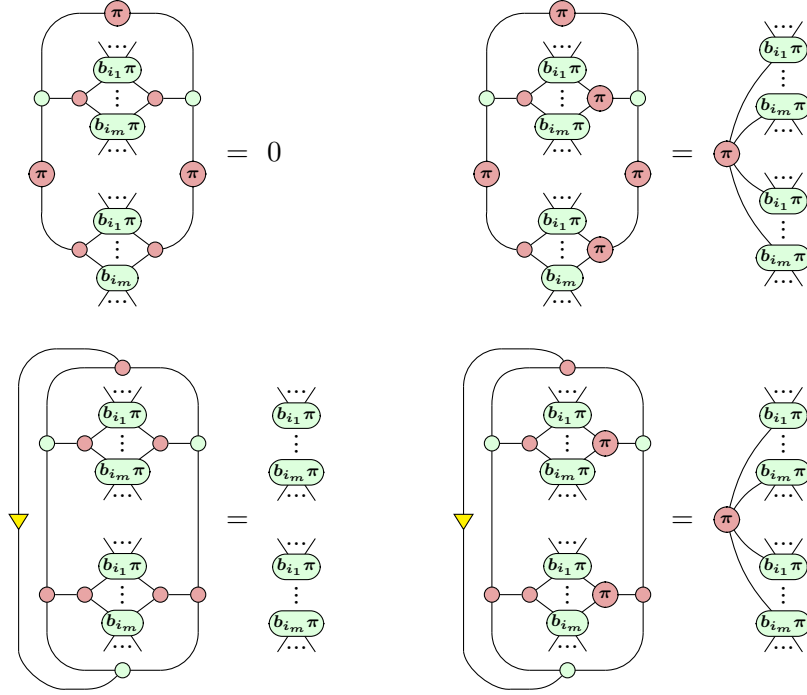
Proof. See Appendix C. ↓

To illustrate the application of Lemma 5.15, we show how to compute $\mathrm{Var}\left(\frac{\partial\langle H\rangle}{\partial\theta_2}\right)$ for the example circuit (5.3) for the Hamiltonian $H = X\otimes Z\otimes X\otimes Y$ which corresponds to $a_1 = b_2 = a_3 = 0$ and $b_1 = a_2 = b_3 = b_4 = a_4 = 1$ yielding the expectation value

$$\langle H\rangle = \frac{i}{2^4}\ (5.4)$$

Invoking Theorem 5.1 we get the following diagram for $\mathrm{Var}\left(\frac{\partial\langle H\rangle}{\partial\theta_2}\right)$:

$$-\frac{1}{2^8} \quad \overset{\text{(Lem. 5.15)}}{=} \quad -\frac{1}{2^8}$$

$$\overset{\text{(Lem. 5.15)}}{=} \quad -\frac{1}{2^8} \quad \overset{\text{(Lem. 5.15)}}{=} \quad -\frac{1}{2^8} \quad \overset{(\boldsymbol{cp,sf})}{=} \quad -\frac{1}{2^8}$$

$$\overset{(\boldsymbol{cp,sf})}{=} \quad \frac{1}{2^8} \; \circ\;\circ\;\circ\;\circ\;\circ\;\circ \; = \; \frac{1}{4}$$
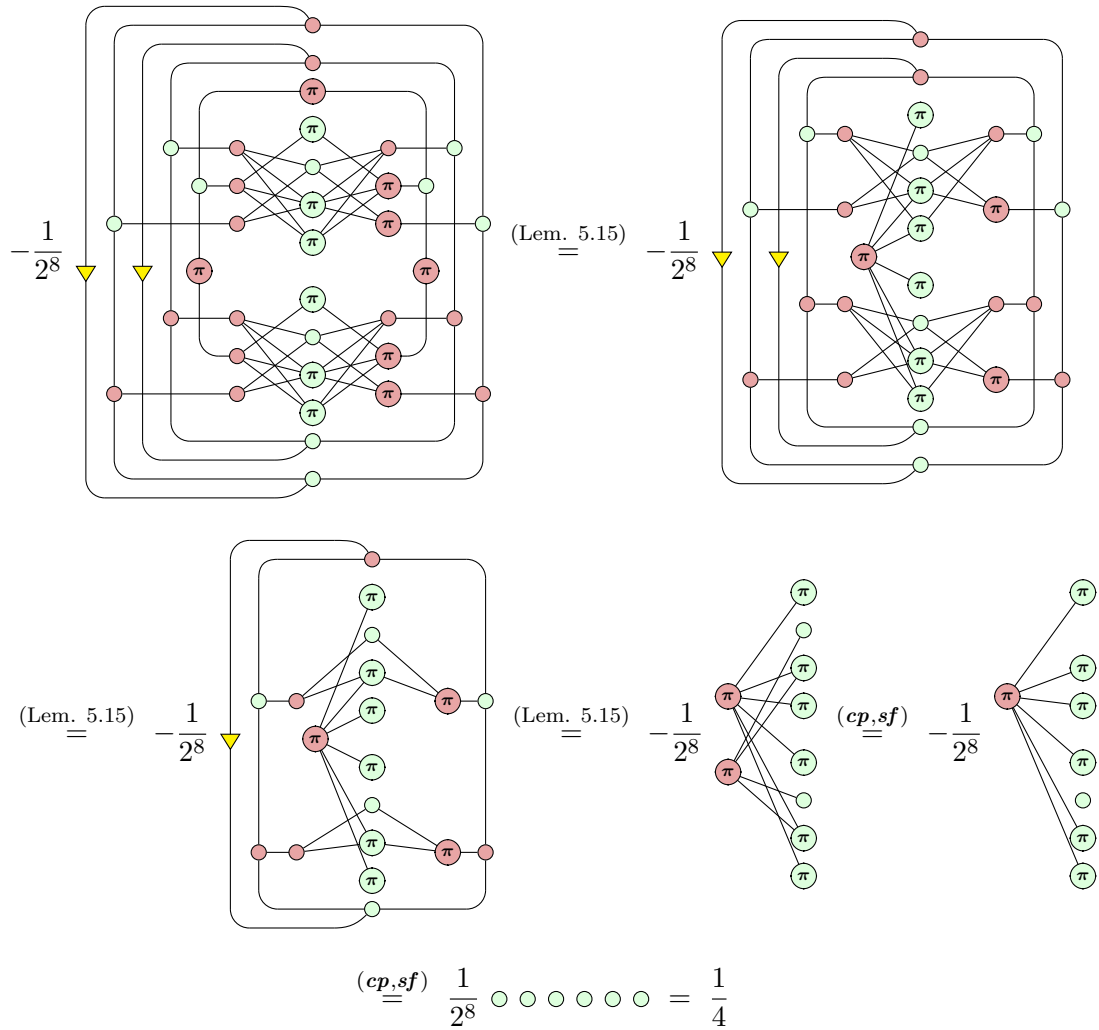
Generalising to arbitrary IQPs

Following the technique from the example circuit, we can compute the variance for arbitrary single-layer IQP circuits. In the general case, our ansatz $U(\vec{\theta})$ consists of m phase gadgets, given by exponentials of Pauli strings $P_1, ..., P_m \in \{I, Z\}^{\otimes n}$:

$$U(\vec{\theta}) \quad = \quad$$
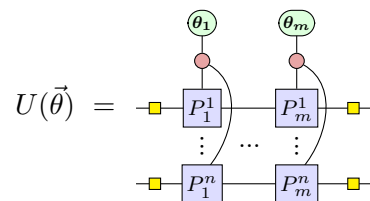
Furthermore fix a Hamiltonian H and define

$$k_i = \sum_{\substack{1 \leq j \leq n \\ P_i^j = Z}} a_j$$

such that
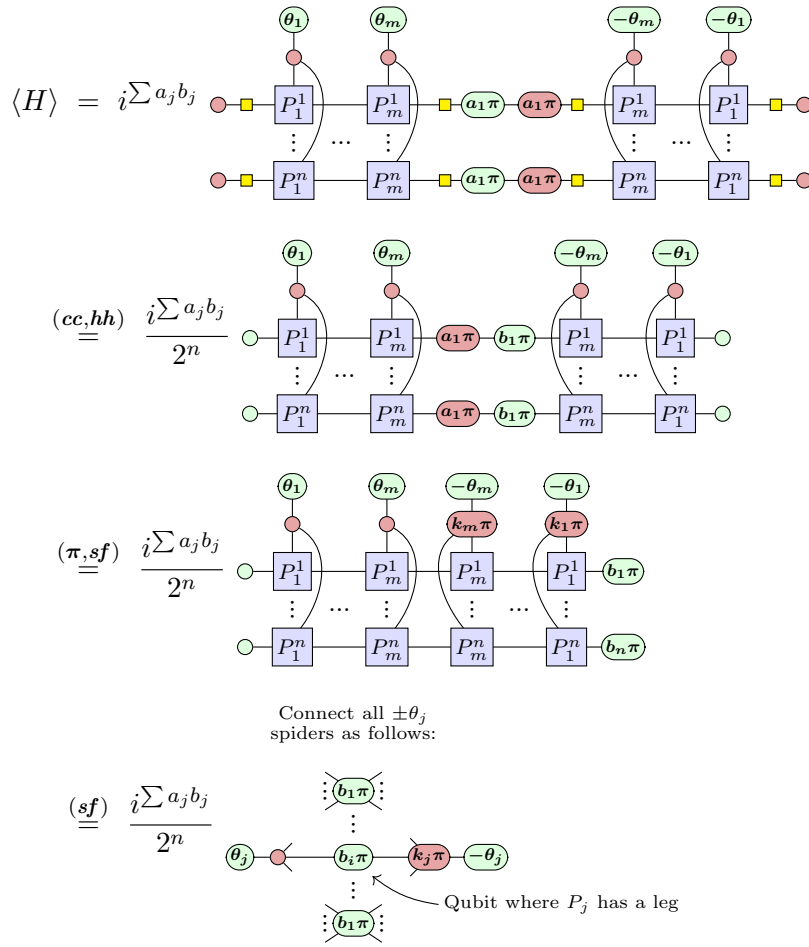
Compare this with (5.4): The green spiders in the middle represent one qubit each. Furthermore, we get pink spiders on the left and right side for each parameter θ_j. Those pink spiders are connected to all the qubits where the gadget associated with θ_j has legs.

This leads to the following characterisation of the variance:

Theorem 5.16. *If $k_i = 0$, then $Var\left(\frac{\partial\langle H\rangle}{\partial\theta_i}\right) = 0$. If $k_i = 1$, then*

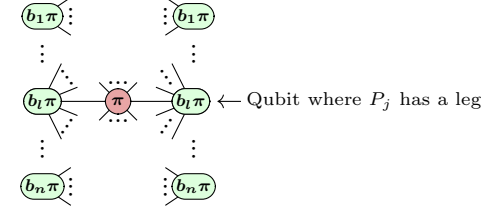$$\text{For all } j \text{ with } k_j = 1:$$

$$Var\left(\frac{\partial\langle H\rangle}{\partial\theta_i}\right) \; = \; \frac{(-1)^{\sum_{j=1}^n a_j b_j}}{4^n}$$

Proof. Follows by simplifying the cycles in the variance diagram according to Lemma 5.15. □

Remark 5.17. *A remarkable consequence of Theorem 5.16 is that the variance for every parameter is either zero, or the same as all other parameters with non-zero variance. Thus, when determining whether an ansatz exhibits the barren plateau phenomenon, it suffices to look at a single parameter whose gradient has non-zero variance.*

Using Theorem 5.16, we can analyse the IQP ansätze for barren plateaus:

Theorem 5.18. *Hypothesis 5.4 is true, i.e. we get the following results for single-layer IQPs:*

- *IQP_1 does not have barren plateaus.*

- *IQP_2 has barren plateaus for $H = (Y \otimes X)^{\otimes n/2}$.*

- *IQP_3 has barren plateaus for $H = (Y \otimes X)^{\otimes n/2}$.*

Proof.

- IQP$_1$: If $k = \sum a_j = 0$, we get $\text{Var}\left(\frac{\partial\langle H\rangle}{\partial\theta}\right) = 0$. If $k = 1$, Theorem 5.16 gives us

$$\mathrm{Var}\left(\frac{\partial \langle H\rangle}{\partial \theta}\right) \;=\; \frac{(-1)^{\sum a_j b_j}}{4^n}\; \text{[ZX diagram]}$$

$$\overset{(\boldsymbol{cp},\boldsymbol{sf})}{=}\; \frac{(-1)^{\sum a_j b_j}}{4^n}(-1)^{b_1}\; \text{[ZX diagram]}$$

This is either 0, or $\frac{1}{2}$. Thus, the variance does not vanish exponentially.

- IQP_2: For IQP_2, we have $k_1 = a_1 + a_2$, $k_2 = a_3 + a_4$, ..., $k_{n/2} = a_{n-1} + a_n$. By Theorem 5.16 we get $\mathrm{Var}\left(\frac{\partial \langle H\rangle}{\partial \theta_i}\right) = 0$ if k_i is even. Otherwise

$$\mathrm{Var}\left(\frac{\partial \langle H\rangle}{\partial \theta_i}\right) \;=\; \frac{(-1)^{\sum a_j b_j}}{4^n}\; \text{[ZX diagram]}$$

exists if k_1 is odd, i.e. $a_1 \neq a_2$

exists if k_2 is odd, i.e. $a_3 \neq a_4$

exists if $k_{n/2}$ is odd, i.e. $a_{n-1} \neq a_n$

where the pink spiders only exist if the annotated condition is met. Concretely, for $H = (Y \otimes X)^{\otimes n/2}$ we have $a_1 = a_3 = ... = a_{n-1} = 1$, $a_2 = a_4 = ... = a_n = 0$, $b_j = 1$ for all j and hence

$$\mathrm{Var}\left(\frac{\partial \langle H\rangle}{\partial \theta_i}\right) \;=\; \frac{(-1)^{n/2}}{4^n}\left(\text{[ZX diagram]}\right)^{n/2} \overset{(\boldsymbol{cp})}{=} \frac{(-1)^{n/2}}{4^n}\left(-\,\text{[ZX diagram]}\right)^{n/2}$$

$$=\; \frac{8^{n/2}}{4^n} \;=\; \frac{1}{2^{n/2}}.$$

Note that this exactly matches with the numerical data from Figure 5.5. We conclude that we have a barren plateau.

- IQP_3: Again, Theorem 5.16 yields

$$\text{Var}\left(\frac{\partial\langle H\rangle}{\partial\theta_i}\right) \ = \ \frac{(-1)^{\lfloor n/2\rfloor}}{4^n} \ \ \overset{(\boldsymbol{cp,sf})}{=} \ -\frac{(-1)^{\lfloor n/2\rfloor}}{4^n}$$

$$\overset{(\boldsymbol{cp,sf})}{=} \ -\frac{(-1)^{\lfloor n/2\rfloor}}{4^n} \qquad = \ \dots \ = \ \frac{1}{4^n} \quad \longleftarrow \text{only if } n \text{ is odd}$$

Thus, if n is odd we get variance 0. If n is even we get variance $\frac{2^n}{4^n} = \frac{1}{2^n}$. This exactly matches with the numerical data from Figure 5.5. Thus, we have a barren plateau. $\qquad\qquad\square$

5.4.6   Dealing with multiple parameter occurrences

The biggest limitation to the current ZX based analysis of barren plateaus is the fact that Theorem 5.1 only applies if each parameter occurs once in the diagram. However, many circuits of interest (for example IQP$_4$) require multiple spiders with the same parameter. Ideally, one would want alternate versions of Theorem 5.1 that support all possible combinations of parameter occurences. This would require extending and generalising the integration results by Wang an Yeung [19]. While this is principally possible using the summing technique from [18], the main challenge is finding a representation that is amenable to rewriting and offering a way to break up cycles.

In this section, we describe a trick that can sometimes be used instead to compute variances using Theorem 5.1, even if parameters occur multiple times. The idea is that in some special cases, one can choose a Hamiltonian for which the extra parameter occurrences cancel out. We demonstrate this using the IQP$_4$ ansatz with the Hamiltonian $H = Z^{\otimes n}$. In this case, we can rewrite the expectation value as follows:
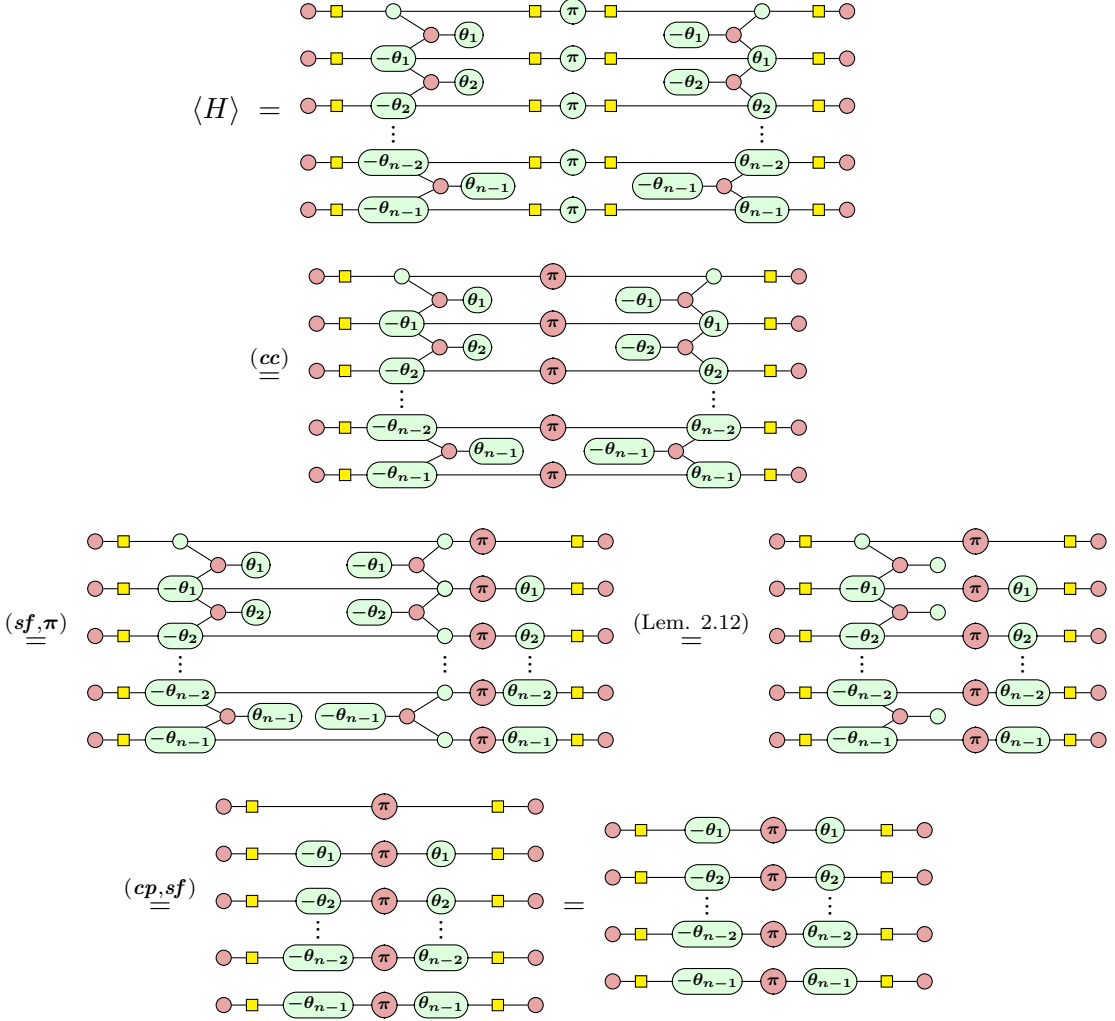
Thus, we got rid of all two-legged phase gadgets. This is now amenable for barren plateau analysis using Theorem 5.1 and Lemma 5.15:

Theorem 5.19. *The barren plateau phenomenon appears in IQP_4 for $H = Z^{\otimes n}$.*

Proof. By Theorem 5.1 and Lemma 5.15, we have

5.4.7   Commuting Multi-Layer IQP Ansätze

So far, we only studied single-layered circuits. In this section we analyse a special case where the multi-layers analysis of IQPs is straightforward. Note that since the layers are separated by Hadamards, we can view multi-layer IQPs as alternating layers of Z- and X-Pauli exponentials. For example, consider IQP_2 for an even number of layers:

Recalling the commutation properties of Pauli boxes (see Lemma 2.10), we see that X- and Z-layers commute with each other for this ansatz. In this special case, computing the variance is actually not difficult since we can fuse all odd and even layers together via Lemma 2.9:

As a result, an ℓ-layer IQP_2 circuit with even ℓ[5] is equivalent to a 2-layer IQP_2 circuit. The barren plateau analysis in this case is straightforward:

---

[5]We focus on the case where ℓ is even. For odd ℓ, the derivations are orthogonal, noting that the fused second layer will not have Hadamards at the end.

Theorem 5.20. *The barren plateau phenomenon appears in IQP$_2$ for $H = (Z \otimes Y)^{\otimes n/2}$ for any number of layers.*

Proof. As discussed before, it suffices to consider the case $\ell = 2$. The expectation value is given by

This allows us to use Lemma 5.15 to remove the cycles showing up during the variance calculation via Theorem 5.16. Concretely, we get

Thus, we have a barren plateau. □

This result is not surprising since we showed that the single-layer version of IQP$_3$ already has barren plateaus (see Theorem 5.18). The more interesting question is how IQP$_1$ behaves for multiple layers since it does not have barren plateaus for a single layer. Unfortunately, the layers of IQP$_1$ only commute for an even number of qubits. Hence, the technique discussed in this section is not applicable if n is odd.

In that case, the necessary calculations become significantly more involved, which we explore in the next section.

5.4.8   Non-Commuting Multi-Layer IQP Ansätze

Apart from specifically designed examples like IQP_2, it is uncommon that IQP layers fully commute. For example, the QNLP ansätze IQP_3 and IQP_4 do not form commuting layers in a multi-layer configuration. In that case, the variance computation becomes significantly more difficult. Since we have already shown that the QNLP ansätze have barren plateaus even for a single layer, we will not consider them in this section. Instead, we focus on IQP_1 to demonstrate our variance computation technique for non-commuting layers. IQP_1's layers do not commute for odd n and its single-layer version does not have barren plateaus which makes it an interesting case to study.

To make the diagrams more concise, we introduce the following notation to denote layers of IQP_1:
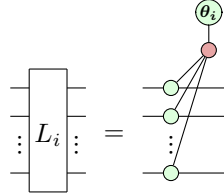
The diagram for the expectation value is then given by

where

$$k_i = \begin{cases} a_1 + ... + a_n & \text{if } i \text{ has same parity as } \ell \\ b_1 + ... + b_n & \text{otherwise} \end{cases}$$

$$c_i = \begin{cases} a_i & \text{if } \ell \text{ is even} \\ b_i & \text{if } \ell \text{ is odd} \end{cases} \qquad d = \begin{cases} 0 & \text{if } \ell \text{ is even} \\ \sum_{i=1}^{n} a_i b_i & \text{if } \ell \text{ is odd} \end{cases}$$

Similar to the calculations we did before, we have pushed the Hamiltonian through the layers on the right-hand side, occasionally adding a phase of π to the phase gadgets. The factor $(-1)^d$ is introduced because if ℓ is odd, we get the following situation after pushing the Hamiltonian through:
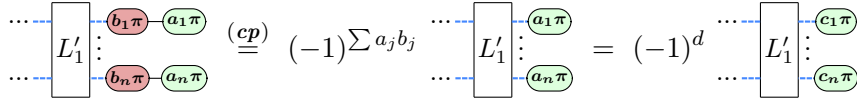
However, the factor $(-1)^d$ does not really matter since it cancels out when computing the variance. In order to draw the variance diagram, we add a wire coming out of each layer that replaces the parametrised spider:
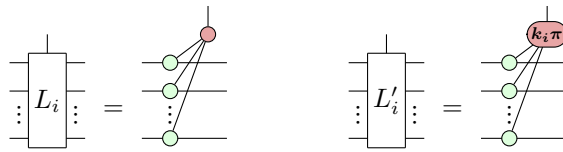
Note that in the following variance diagram, we only explicitly draw the cycle connecting L_ℓ and L'_ℓ. We only hint at remaining cycles using dots:

$$\overset{(\mathbf{sf})}{=} \quad (-1)^{\sum a_j b_j}$$
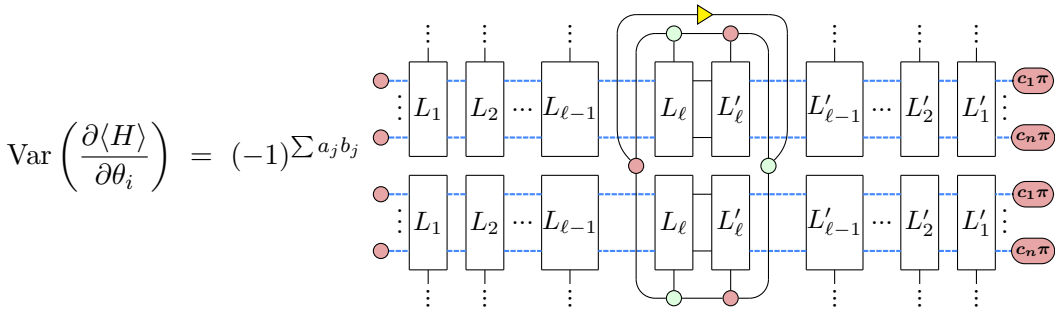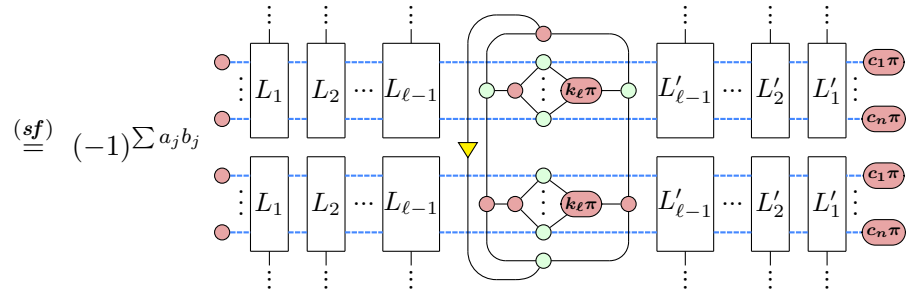
We can cut this cycle using our existing simplification strategy from Lemma 5.15. Concretely, if $k_\ell = 0$, we get

$$(-1)^{\sum a_j b_j}$$
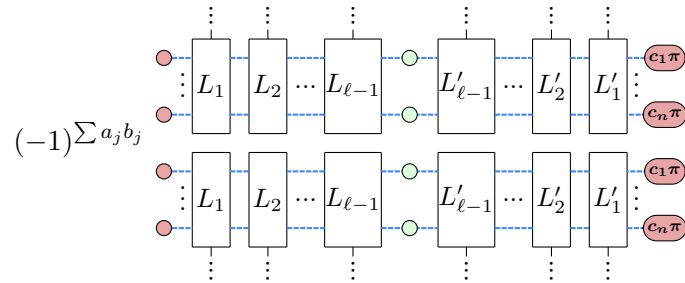
In this case, $L_{\ell-1}$ and $L'_{\ell-1}$ are now directly next to each other and we can continue the same argument recursively.

However, if $k_\ell = 1$ is odd, we get a red π-spider:

$$(-1)^{\sum a_j b_j}$$
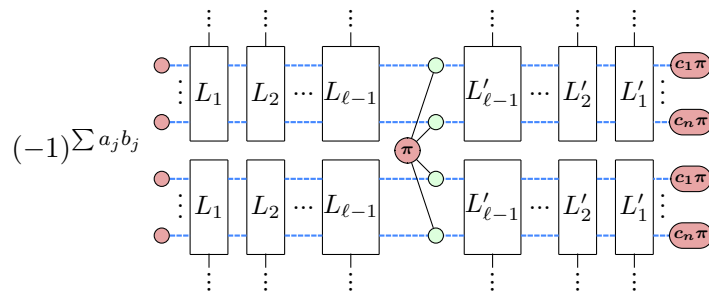
Assuming that n is odd, commuting $L_{\ell-1}$ past this will add some extra Hadamard wires according to Lemma 2.10:
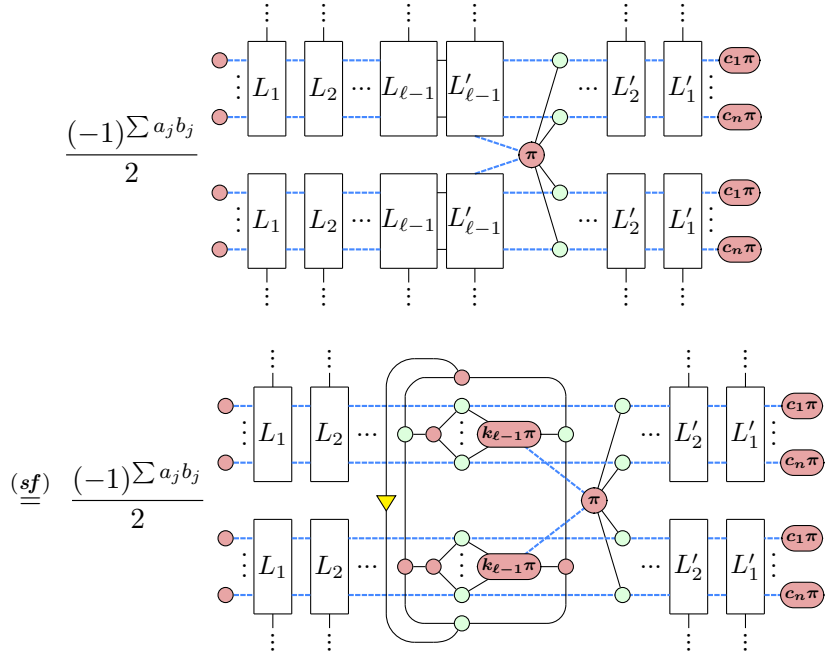
Note that the two new Hadamard wires connected to the red spiders in $L'_{\ell-1}$ come with a scalar of $\frac{1}{\sqrt{2}}$ each. In order to proceed from here, we need a new cycle cutting lemma that applies when the right side is connected to shared pink spider(s):

Lemma 5.21. *We have*
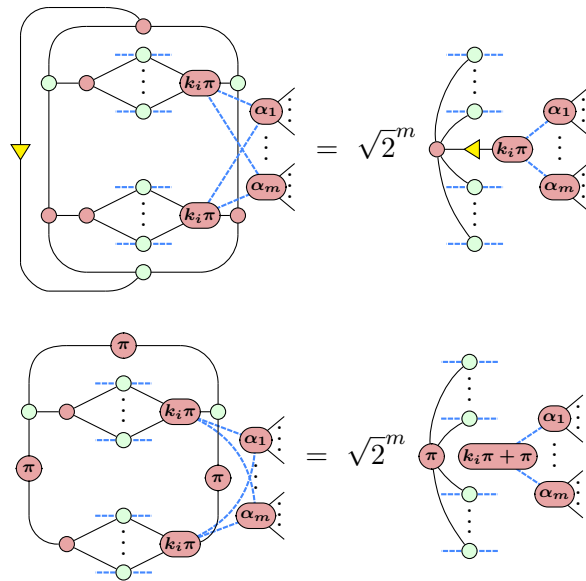
Proof. See Appendix C. ↓

Note that in the special case $m = 0$, Lemma 5.21 exactly corresponds to Lemma 5.15. We can now use Lemma 5.21 to simplify the cycle in the variance computation above, also replacing the previous application of Lemma 5.15 with the more general Lemma 5.21:
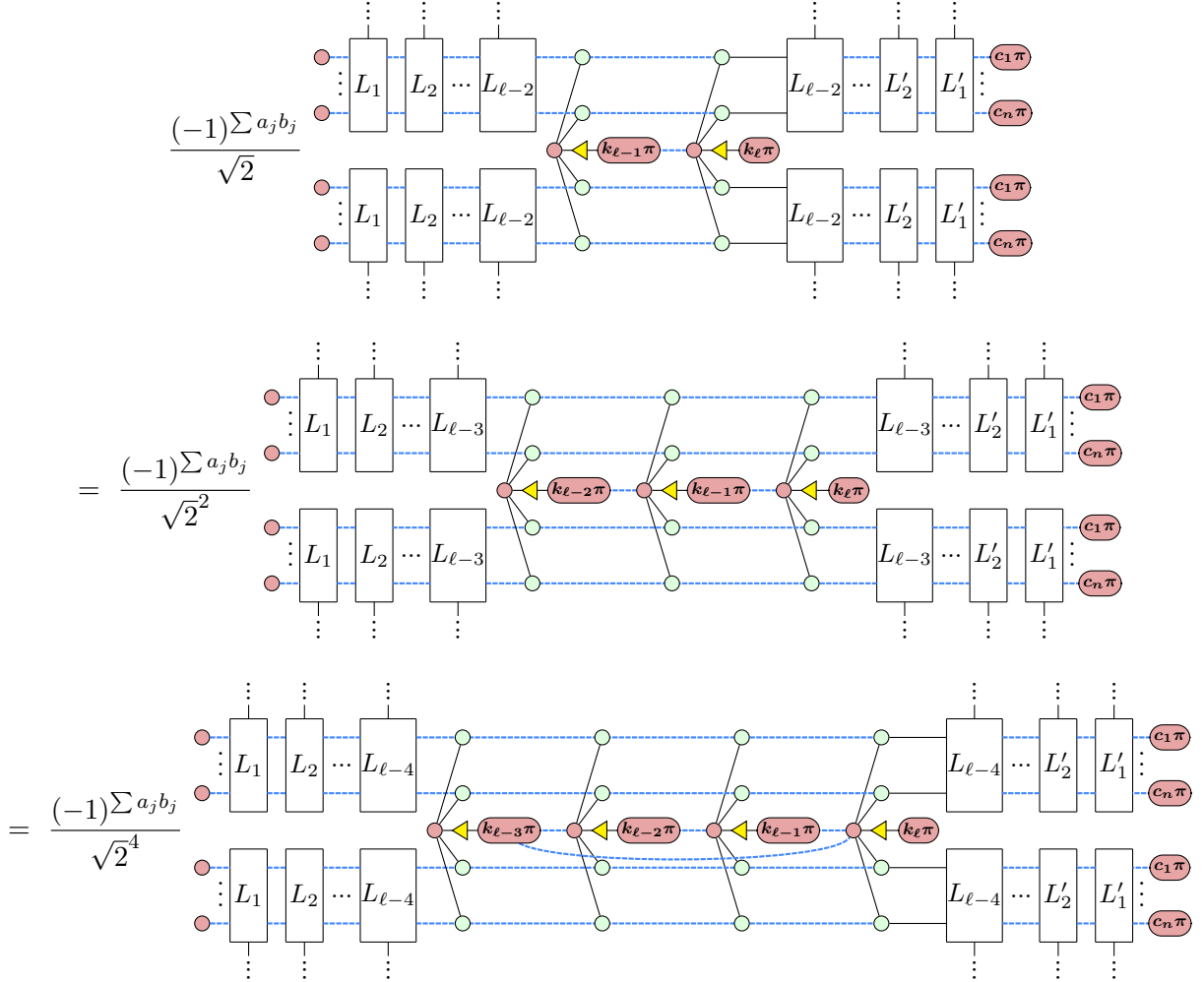
Iterating this process for all layers yields the diagram shown in Figure 5.8 which we evaluate using a recursive strategy. However, we only sketch the proof here, fixing $i = 1$ and skipping over some details. We refer to Appendix B for the full derivation.

One can show that $\mathrm{Var}\left(\frac{\partial \langle H \rangle}{\partial \theta_1}\right)$ is only non-zero if $c_1 = c_2 = ... = c_n$. Furthermore, it turns out that the value of the c's only effects the sign of the scalar represented by the diagram. Therefore we will ignore them here. Also note that by definition
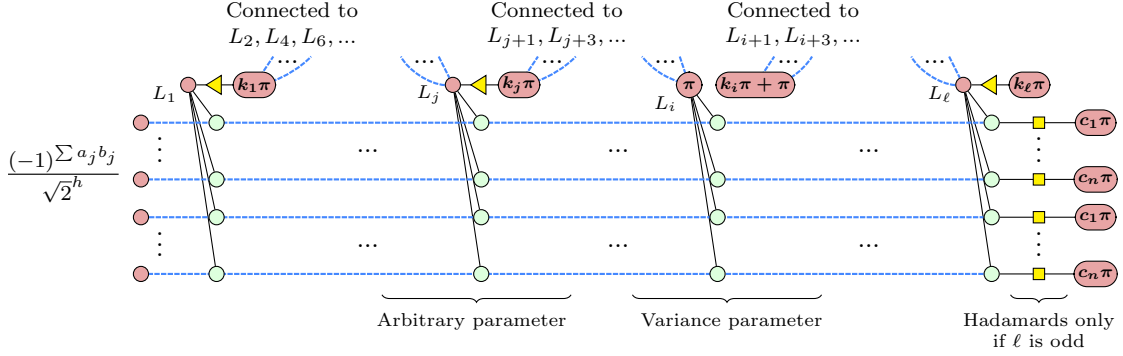
Figure 5.8: Diagram for $\mathrm{Var}\left(\frac{\partial \langle H \rangle}{\partial \theta_i}\right)$ where h is the number of Hadamard wires connecting the red spiders.

we have $k_i = k_j$ if i and j have the same parity. Thus, we define the following shorthands:

$$k_o := k_1 = k_3 = k_5 = \ldots \qquad k_e := k_2 = k_4 = k_6 = \ldots$$

This means that the diagram in Figure 5.8 only depends on the numbers k_e, k_o, and ℓ. We can show that it satisfies the following recurrence relation.

Lemma 5.22. *Let $V_\ell(k_e, k_o, c)$ denote the diagram in Figure 5.8 and let $\ell > 1$ be odd. Then*

$$V_1(k_e, 0) = 0 \qquad\qquad V_1(k_e, 1) = \frac{1}{2}$$

$$V_\ell(0, 0) = 0 \qquad\qquad V_\ell(0, 1) = \frac{1}{4}(3V_{\ell-2}(0, 1) - V_{\ell-2}(1, 0))$$

$$V_\ell(1, 1) = V_\ell(0, 1) \qquad\qquad V_\ell(1, 0) = \frac{1}{2}(V_{\ell-2}(1, 0) - V_{\ell-2}(0, 1))$$

Proof. See equations (B.7), (B.14), (B.15), and (B.16) in Appendix B. $\qquad\square$

Similarly, we can obtain recursive equations for even ℓ. This yields a recursive algorithm for computing $\mathrm{Var}\left(\frac{\partial \langle H \rangle}{\partial \theta_1}\right)$. However, it is also possible to derive a closed-form solution:

Lemma 5.23. *For odd ℓ we have*

$$V_\ell(1,1) = V_\ell(0,1) = \frac{2 \cdot 4^{\lfloor \ell/2 \rfloor} + 1}{6 \cdot 4^{\lfloor \ell/2 \rfloor}} \qquad\qquad V_\ell(1,0) = \frac{4^{\lfloor \ell/2 \rfloor} - 1}{3 \cdot 4^{\lfloor \ell/2 \rfloor}}.$$

For even ℓ we have

$$V_\ell(1,1) = V_\ell(1,0) = \frac{2^\ell - 1}{3 \cdot 2^\ell} \qquad\qquad V_\ell(0,1) = \frac{2 \cdot 4^{\ell/2-1} + 1}{6 \cdot 4^{\ell/2-1}}$$

Proof. See Lemma B.1 and Corollary B.2 in Appendix B. $\qquad\qquad\qquad\qquad\square$

As a result, we get the following formula for the variance:

Theorem 5.24.

$$Var\left(\frac{\partial \langle H \rangle}{\partial \theta_1}\right) = \begin{cases} V_\ell(\sum a_j, \sum b_j) & \text{if } \ell \text{ is even and } a_1 = ... = a_n \\ V_\ell(\sum b_j, \sum a_j) & \text{if } \ell \text{ is odd and } b_1 = ... = b_n \\ 0 & \text{otherwise.} \end{cases}$$

Proof. See Theorem B.3 in Appendix B. $\qquad\qquad\qquad\qquad\qquad\qquad\qquad\square$

Corollary 5.25. *Either $Var\left(\frac{\partial \langle H \rangle}{\partial \theta_1}\right) = 0$ or $Var\left(\frac{\partial \langle H \rangle}{\partial \theta_1}\right) \to \frac{1}{3}$ for $\ell \to \infty$. In particular, this proves Hypothesis 5.5.*

Proof. This follows from the fact that all terms in Lemma 5.23 converge to $\frac{1}{3}$ for $\ell \to \infty$. See Corollary B.4 in Appendix B for the full details. $\qquad\qquad\square$

Corollary 5.26. *For example, in the case $H = Z^{\otimes n}$ we have $a_j = 1$ and $b_j = 0$ for all j such that*

$$Var\left(\frac{\partial \langle H \rangle}{\partial \theta_1}\right) = \begin{cases} \frac{2^\ell - 1}{3 \cdot 2^\ell} & \text{if } \ell \text{ is even.} \\[2ex] \frac{2 \cdot 4^{\lfloor \ell/2 \rfloor} + 1}{6 \cdot 4^{\lfloor \ell/2 \rfloor}} & \text{if } \ell \text{ is odd} \end{cases}$$

Note that this exactly matches the numerical values from Figure 5.7.

5.5 Barren Plateau Mitigation Techniques

After identifying barren plateaus in a variety of ansätze, we want to close this chapter with a brief discussion of how to avoid them. Crucially, the barren plateau analysis in this chapter relied on the assumption made in Section 5.1 that the parameters θ_i are uniformly and independently sampled from $[-\pi, \pi]$. This means that our results, as well the ones in the literature like McLean et al. [10], no longer apply if one chooses a different parameter initialisation.

To this end, different initialisation strategies have been proposed with the goal of avoiding barren plateaus: For example, Grant et al. [67] choose parameters such that the circuit turns into a sequence of shallow blocks that each evaluate to the identity, thus reducing the effective circuit depth. Kulshrestha and Safro [68] experimentally show that initialisation with the Beta distribution reduces the prevalence of barren plateaus.

Apart from initialisation strategies, various other techniques have been proposed: Rad et al. [69] use Bayesian learning to find a promising regions in the parameter space which are then explored using local optimisers. Sack et al. [70] introduce a new learning scheme that adapts the learning rate when a barren plateau is detected. Liu et al. [71] propose a novel ansatz family where barren plateaus can be mitigated and Skolik et al. [72] use quantum circuit learning to find ansätze that avoid barren plateaus. Patti et al. [73] discuss a variety of techniques including the addition of noise, reducing entanglement and partitioning the qubit registers depending on the cost function.

Chapter 6

Discussion

6.1 Summary of Results

Gradient Recipes

We have refined the diagrammatic differentiation technique by Wang and Yeung [19] for the special case of parametrised quantum circuits and used it to give diagrammatic proofs of parameter shift rules given by Schuld et al. [7] and Anselmetti et al. [8]. Furthermore, we derived a novel $2n$-term shift rule for gates that can be represented with n parametrised spiders. We also discussed the optimality of shift rules, proving an open conjecture by Anselmetti et al. [8] by deriving a no-go theorem ruling out shift rules with less than four terms for all gates whose Hermitian generators have eigenvalues of shape $-\lambda, 0, \lambda$.

Barren Plateaus

We investigated both empirical and formal methods to detect barren plateaus in ansätze using the variance computation framework laid out by Wang and Yeung [19]. For the empirical analysis, we developed a tool that automatically computes $\mathrm{Var}\left(\frac{\partial \langle H \rangle}{\partial \theta_i}\right)$ which can be used to diagnose barren plateaus without the user having to perform any calculations or mathematical reasoning. Using this tool, we investigate several

ansätze studied by Sim et al. [20] and empirically concluded that even at a single layer they likely all have barren plateaus.

To showcase the analytical barren plateau analysis powered by ZX, we formally proved this claim for three of the Sim ansätze. Furthermore, we analysed a range of IQP ansätze, in particular showing that a single layer of the ansatz used by the quantum natural language processing library `lambeq` [21] has barren plateaus when measuring in the computational basis. Additionally, we proved that one of the IQP ansätze does not have barren plateaus, with $\mathrm{Var}\left(\frac{\partial\langle H\rangle}{\partial\theta_i}\right)$ converging to a constant independent of the number of qubits n when the number of layers ℓ goes to infinity.

6.2 Discussion and Future Work

Gradient Recipes

One of the initial motivations for using the ZX calculus to study gradient recipes was the hope that the graphical representation of derivatives might make it easier to discover new recipes that go beyond parameter shift rules. However, as we have mentioned in Section 4.3, it proved to be harder than expected to find decompositions of the differentiation gadget that actually yield unitaries when applied to gates.

However, the diagrammatic approach proved to be very fruitful for the analysis of parameter shift rules. Originally, Schuld et al. [7] and Anselmetti et al. [8] arrived at their shift rules by observing that the Hermitian generators for the gates they consider satisfy $H^2 = I$ and $H^3 = H$, respectively. From this, they derived systems of equations that yielded the shift rules. While our approach also involved systems of equations, we arrived at and solved them in a completely different way. Concretely, we used a diagrammatic approach to find systems of equations that characterise valid shift rules which turned out to be easily solvable using a discrete sine transform. The benefit of this hybrid approach involving both graphical and algebraic techniques is that it applies to a wider range of gates. This allowed us to generalise to the $2n$-term

shift rule, whereas Schuld et al.'s and Anselmetti et al.'s approach only works for gates that satisfy specific eigenvalue constraints. Wierichs et al. [9] obtained their generalised shift rule by expressing the expectation value $\langle H \rangle$ in terms of a discrete Fourier transform (DFT) which is closely related to the discrete sine transform. This might suggest a possible connection between our diagrammatically obtained system of equations and Wierichs et al.'s DFT reconstruction of the expectation value which would be interesting to investigate in the future.

Another interesting question that showed up at multiples points in our work is the relationship between eigenvalues of a parametrised unitary and the minimum number of parametrised spiders required to implement the unitary in ZX. To the best of our knowledge, this question has not been investigated before. We have given a general upper bound, and a lower bound for the special case of eigenvalues $-\lambda$, 0, λ. Those bounds were close enough to derive existing parameter shift rules and prove Anselmetti et al.'s conjecture [8]. However, it would be interesting to investigate whether there are tighter bounds and if there is a deeper relationship between parametrised spiders and eigenvalues. Besides being interesting in its own right, this could potentially unify our generalised shift rule with the one given Wierichs et al. [9]. Orthogonally, proving lower bounds could lead to general optimality results for shift rules, generalising our no-go theorem to arbitrary parametrised unitaries.

Barren Plateaus

We have presented both empirical and analytical methods to detect barren plateaus in ansätze. Our numerical tool can be used to quickly check a specific combination of ansatz and Hamiltonian for barren plateaus. While the tool cannot formally *prove* or *disprove* the existence of barren plateaus, it gives a good indication of the behaviour of an ansatz for practically used circuit dimensions. The potential future use case we envision for software like this is as a part of the QML practitioners' toolbox for evaluating the suitability of ansätze for QML tasks. For example, exponentially

vanishing variance curves like in Figure 5.3 could indicate that experimentation with different initialisation strategies might be warranted. Additionally, the numerical data is very useful for gaining confidence in theoretical results. In particular, it gave us confidence that the intricate formula we derived for the variance of the multi-layer IQP_1 ansatz is correct.

While all the experiments in this thesis run relatively quickly, we have some ideas how to further improve the performance of our tool. For example, we could experiment with decomposing the triangle directly instead of using the representation with four $\frac{\pi}{4}$ spiders. In that case, we would no longer benefit from the efficient magic state and cat decompositions used in [65]. However, we could search for alternative, more efficient decompositions of groups of triangles. Here, it might also help that the triangles in Theorem 5.1 are embedded in a regular, known structure that is possibly easier to decompose.

Moving beyond the methods and looking at the concrete data we obtained, it is surprising that all Sim ansätze we considered seem to already have barren plateaus for a single layer. The results by McClean et al. [10] for example only apply if the ansatz has enough layers to approximate a 2-design. Sim et al. showed that most of their ansätze only gain their expressive power when adding more layers. But our results indicate that the low expressivity of a single layer already suffices to produce barren plateaus. The same seems to be true for the IQP circuit used in `lambeq` [21]. In fact, the only ansatz we considered that does not have barren plateaus is the very simple IQP_1. This might suggest that the expressiveness vs. trainability trade-off described by Holmes et al. [11] is already significant at fairly low expressive powers with mildly expressible circuits already having poor trainability. Analysing more circuits in this way is needed to gain a better understanding of this relationship in the future.

The fact that barren plateaus already appear in a single layer actually simplified our analysis. First, running the numerical experiments for multi-layer circuits is

more expensive.[1] Secondly, we have already seen for the example of IQP_1 that the formal analysis of multi-layer ansätze requires significantly more work. Nonetheless, it would be very interesting to expand on this work and analyse ansätze where the barren plateau phenomenon only appears after adding enough layers. One example of this would be *non-local* cost functions as discussed in [12].

Finally, the biggest limitation of the current ZXW-based barren plateau analysis is the fact that Theorem 5.1 only applies if each parameter occurs exactly once in the ZX representation of the ansatz. In particular, this excludes all anätze that use controlled rotations. We have discussed a small caveat to this in Section 5.4.6 where the analysis is possible in certain special cases where the Hamiltonian cancels out additional parameter occurrences. However, in the general case, such ansätze cannot be handled by our method. The difficulty of adding support for this differs between our numerical and our analytical approach. Both would require extending or generalising the integration results by Wang and Yeung [19]. But while the numerical tool could, in principle (barring performance concerns), work with any diagram that represents the variance, in order to prove results by hand we need a diagram that is amenable to manual rewriting and reasoning. Thus, finding such representations for the gradient variance of circuits with multiple parameter occurrences would serve to significantly generalise the results presented in this work.

---

[1]Note that for the ansätze we considered, adding a qubit adds $O(1)$ parametrised spiders while a new layers adds $O(n)$ new parametrised spiders.

Appendix A

Constructing the Ancilla State

In Section 4.3, we discussed a gradient recipe that prepares ancillae in the state $\pm\alpha$ ◀. In this appendix, we explain how to prepare this state on a quantum device. First, not that because of the (**pcy**) rule, it actually suffices to prepare the state ○ ◀. As it turns out, this state is an equal superposition of basis states:

$$\text{○} \blacktriangleleft \overset{(2.5)}{=} \text{○} \blacktriangleleft + \boxed{\pi} \blacktriangleleft \overset{(2.16)}{=} \text{○} + \boxed{\pi} + \boxed{\pi}$$

To construct this, we define the following gate:

Definition A.1. *Let* $D(p) := R_Y(2\arccos(\sqrt{p}))$ *for* $0 \le p \le 1$.

Lemma A.2. *This gate satisfies* $D(p)|0\rangle = \sqrt{p}|0\rangle + \sqrt{1-p}|1\rangle$.

Proof. In general, we have

$$R_Y(\alpha)|0\rangle = \begin{pmatrix} \cos(\frac{\alpha}{2}) & -\sin(\frac{\alpha}{2}) \\ \sin(\frac{\alpha}{2}) & \cos(\frac{\alpha}{2}) \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} \cos(\frac{\alpha}{2}) \\ \sin(\frac{\alpha}{2}) \end{pmatrix}$$

Setting $\alpha = 2\arccos(\sqrt{p})$, we get

$$D(p)|0\rangle = \begin{pmatrix} \sqrt{p} \\ \sqrt{1-p} \end{pmatrix} = \sqrt{p}|0\rangle + \sqrt{1-p}|1\rangle. \qquad \square$$
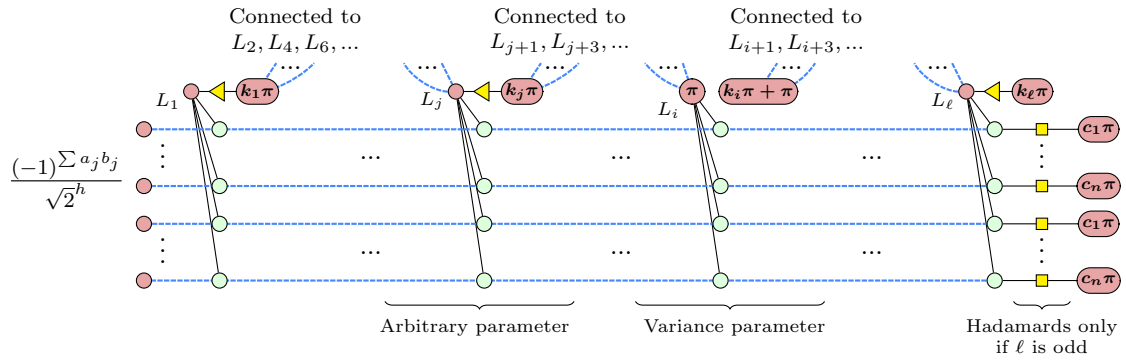
Using $D(p)$, we can construct the state as follows:

$$\begin{array}{c} \text{(circuit)} \\ \overset{\text{(Lem. A.2)}}{=} \sqrt{\frac{2}{3}}\ \text{(circuit)} \ + \ \sqrt{\frac{1}{3}}\ \text{(circuit)} \end{array}$$

$$= \ \sqrt{\frac{2}{3}}\ \text{(circuit)} \ + \ \sqrt{\frac{1}{3}}\ \text{(circuit)} \overset{\text{(Lem. A.2)}}{=} \sqrt{\frac{2}{3}}\sqrt{\frac{1}{2}}\ \text{(circuit)} \ + \ \sqrt{\frac{2}{3}}\sqrt{\frac{1}{2}}\ \text{(circuit)} \ + \ \sqrt{\frac{1}{3}}\ \text{(circuit)}$$

$$= \ \sqrt{\frac{1}{3}}|00\rangle + \sqrt{\frac{1}{3}}|01\rangle + \sqrt{\frac{1}{3}}|10\rangle \ = \ \sqrt{\frac{1}{3}}\ \text{(circuit)}$$

Appendix B

Details on Recursive Contraction

Here, we give the full details for the recursive contraction of Figure 5.8. For the reader's convenience, we restate the diagram:

B.1 Deriving the Recurrence Relation

To contract Figure 5.8, we introduce the notation

$$k_e := k_2 = k_4 = ... = k_\ell = b_1 + ... + b_n$$
$$k_o := k_1 = k_3 = ... = k_{\ell-1} = a_1 + ... + a_n$$

and write $VE_\ell(k_e, k_o, c_1, ..., c_n)$ and $VO_\ell(k_e, k_o, c_1, ..., c_n)$ for the diagram for $\mathrm{Var}\left(\frac{\partial\langle H\rangle}{\partial\theta_i}\right)$ for even and odd ℓ respectively, excluding the factor $(-1)^{\sum a_j b_j}$. Furthermore, we write $\bar{x}$ for the negation of a Boolean variable, i.e. $\bar{0} = 1$ and $\bar{1} = 0$.

Out goal is to find recursive formulas to compute VE_ℓ and VO_ℓ. For the base case, consider VE_0:

$$VE_0(k_e, k_o, c_1, ..., c_n) \;=\; \begin{gathered}\includegraphics{}\end{gathered} \;=\; \begin{cases} 1 & \text{if } c_1 = ... = c_n = 0 \\ 0 & \text{otherwise.} \end{cases} \tag{B.1}$$

Note that the term becomes zero if the c_j are not all the same. Hence, from now on we can ignore all terms where this is the case and simplify the notation to $VO_\ell(k_e, k_o, c)$ and $VE_\ell(k_e, k_o, c)$. Now, we proceed recursively:

- If $i = \ell$ and ℓ is odd then

$$VO_\ell(k_e, k_o, c) \;=\; \underbrace{\frac{1}{\sqrt{2}^h}}_{\substack{\text{Scalar for} \\ \text{part up to } \ell}} \cdot \underbrace{\frac{1}{\sqrt{2}^{\lfloor \ell/2 \rfloor}}}_{\substack{\text{Scalar for Had.} \\ \text{wires to } L_2, L_4, ...}} \cdot \begin{gathered}\includegraphics{}\end{gathered}$$

$$\overset{(2.14)}{=} \sum_{x \in \{0,1\}} \frac{1}{\sqrt{2}^h} \cdot \frac{1}{\sqrt{2}^{\lfloor \ell/2 \rfloor}} \cdot \frac{1}{2} \cdot (-1)^x \; \begin{gathered}\includegraphics{}\end{gathered}$$

$$\overset{(cc)}{=} \sum_{x \in \{0,1\}} \frac{1}{\sqrt{2}^h} \cdot \frac{1}{2} \cdot (-1)^x \; \begin{gathered}\includegraphics{}\end{gathered}$$

$$\overset{(sf)}{=} \sum_{x \in \{0,1\}} \frac{1}{\sqrt{2}^h} \cdot \frac{1}{2} \cdot (-1)^x \; \underbrace{\begin{gathered}\includegraphics{}\end{gathered}}_{= VE_{\ell-1}(k_e + x, k_o, c + x)}$$

$$= \begin{cases} 0 & \text{if } k_o = 0 \\ \frac{1}{2}\left(VE_{\ell-1}(k_e, 1, c) - VE_{\ell-1}(\bar{k}_e, 1, \bar{c})\right) & \text{if } k_o = 1 \end{cases} \tag{B.2}$$

- If $i = \ell$ and ℓ is even then

$$VE_\ell(k_e, k_o, c) \;=\; \frac{1}{\sqrt{2}^h} \cdot \frac{1}{\sqrt{2}^{\ell/2}} \quad \text{(diagram)}$$

$$\overset{(2.14)}{=} \sum_{x\in\{0,1\}} \frac{1}{\sqrt{2}^h} \cdot \frac{1}{\sqrt{2}^{\ell/2}} \cdot \frac{1}{2} \cdot (-1)^x \quad \text{(diagram)}$$

$$\overset{(cc,cp)}{=} \sum_{x\in\{0,1\}} \frac{1}{\sqrt{2}^h} \cdot \frac{1}{2} \cdot (-1)^x \quad \text{(diagram)}$$

$$=\; \begin{cases} 0 & \text{if } k_e = 0 \\ \frac{1}{2}\left(VO_{\ell-1}(1, k_o, c) - VO_{\ell-1}(1, \overline{k}_o, c)\right) & \text{if } k_e = 1 \end{cases} \tag{B.3}$$

- If $i \neq \ell$ and ℓ is odd then

$$VO_\ell(k_e, k_o, c) \;=\; \frac{1}{\sqrt{2}^h} \cdot \frac{1}{\sqrt{2}^{\lfloor \ell/2 \rfloor}} \quad \text{(diagram)}$$

$$\overset{(2.14)}{=} \sum_{x\in\{0,1\}} \frac{1}{\sqrt{2}^h} \cdot \frac{1}{\sqrt{2}^{\lfloor \ell/2 \rfloor}} \cdot \frac{1}{2} \quad \text{(diagram)}$$

$$\overset{(cc,sf)}{=} \sum_{x\in\{0,1\}} \frac{1}{\sqrt{2}^h} \cdot \frac{1}{2} \quad \text{(diagram)}$$

$$=\; \begin{cases} VE_{\ell-1}(k_e, 0, c) & \text{if } k_o = 0 \\ \frac{1}{2}\left(VE_{\ell-1}(k_e, 1, c) - VE_{\ell-1}(\overline{k}_e, 1, \overline{c})\right) & \text{if } k_o = 1 \end{cases} \tag{B.4}$$

- If $i \neq \ell$ and ℓ is even then

$$VE_\ell(k_e, k_o, c) \;=\; \frac{1}{\sqrt{2}^h} \cdot \frac{1}{\sqrt{2}^{\ell/2}} \quad [\text{diagram}]$$

$$\overset{(2.14)}{=}\; \sum_{x \in \{0,1\}} \frac{1}{\sqrt{2}^h} \cdot \frac{1}{\sqrt{2}^{\ell/2}} \cdot \frac{1}{2} \quad [\text{diagram}]$$

$$\overset{(cc,cp)}{=}\; \sum_{x \in \{0,1\}} \frac{1}{\sqrt{2}^h} \cdot \frac{1}{2} \quad [\text{diagram}]$$

$$=\; \begin{cases} VO_{\ell-1}(0, k_o, c) & \text{if } k_e = 0 \\ \frac{1}{2}(VO_{\ell-1}(1, k_o, c) - VO_{\ell-1}(1, \overline{k}_o, c)) & \text{if } k_e = 1 \end{cases} \tag{B.5}$$

This yields an recursive algorithm for computing $\mathrm{Var}\left(\frac{\partial \langle H \rangle}{\partial \theta_i}\right)$. One interesting thing to note is that

$$VO_\ell(0,0,c) \overset{(B.4)}{=} VE_{\ell-1}(0,0,c) \overset{(B.5)}{=} VO_{\ell-2}(0,0,c) = \ldots$$
$$= \begin{cases} VO_i(0,0,c) & \text{if } i \text{ is odd} \\ VE_i(0,0,c) & \text{if } i \text{ is even} \end{cases} \overset{(B.2,B.3)}{=} 0. \tag{B.6}$$

Furthermore, for $i \neq \ell, \ell-1$ we can derive

$$VO_\ell(0,0,c) \overset{(B.6)}{=} VO_{\ell-2}(0,0,c) \tag{B.7}$$

$$VO_\ell(0,1,c) \overset{(B.4,B.5)}{=} \frac{1}{4}(2\,VO_{\ell-2}(0,1,c) + VO_{\ell-2}(1,0,\overline{c}) - VO_{\ell-2}(1,1,\overline{c})) \tag{B.8}$$

$$VO_\ell(1,0,c) \overset{(B.4,B.5)}{=} \frac{1}{2}(VO_{\ell-2}(1,0,c) - VO_{\ell-2}(1,1,c)) \tag{B.9}$$

$$VO_\ell(1,1,c) \overset{(B.4,B.5)}{=} \frac{1}{4}(-2\,VO_{\ell-2}(0,1,\overline{c}) - VO_{\ell-2}(1,0,c) + VO_{\ell-2}(1,1,c)) \tag{B.10}$$

B.2 Solving the Recurrence Relation

When considering concrete values of i, we can derive closed-form solutions for $\text{Var}\left(\frac{\partial \langle H \rangle}{\partial \theta_i}\right)$. We discuss the case $i = 1$. First, note that

$$
VO_1(k_e, k_o, c) \stackrel{(B.3)}{=} \begin{cases} 0 & \text{if } k_o = 0 \\ \frac{1}{2}(VE_0(k_e, 0, c) - VE_0(\bar{k}_e, 0, \bar{c})) & \text{if } k_o = 1 \end{cases}
$$
$$
\stackrel{(B.1)}{=} \begin{cases} 0 & \text{if } k_o = 0 \\ \frac{1}{2} \cdot (-1)^c & \text{if } k_o = 1 \end{cases} \tag{B.11}
$$

As a consequence, we have

$$
VE_\ell(k_e, k_o, \bar{c}) = -VE_\ell(k_e, k_o, c) \tag{B.12}
$$
$$
VO_\ell(k_e, k_o, \bar{c}) = -VO_\ell(k_e, k_o, c). \tag{B.13}
$$

Therefore,

$$
VO_\ell(1, 1, c) \stackrel{(B.10)}{=} \frac{1}{4}(-2\,VO_{\ell-2}(0, 1, \bar{c}) - VO_{\ell-2}(1, 0, c) + VO_{\ell-2}(1, 1, c))
$$
$$
\stackrel{(B.13)}{=} \frac{1}{4}(2\,VO_{\ell-2}(0, 1, c) + VO_{\ell-2}(1, 0, \bar{c}) - VO_{\ell-2}(1, 1, \bar{c}))
$$
$$
\stackrel{(B.8)}{=} VO_\ell(0, 1) \tag{B.14}
$$

$$
VO_\ell(0, 1, c) \stackrel{(B.8, B.14)}{=} \frac{1}{4}(3\,VO_{\ell-2}(0, 1, c) - VO_{\ell-2}(1, 0, c)) \tag{B.15}
$$
$$
VO_\ell(1, 0, c) \stackrel{(B.8, B.14)}{=} \frac{1}{2}(VO_{\ell-2}(1, 0, c) - VO_{\ell-2}(0, 1, c)). \tag{B.16}
$$

Now, we just need to derive a closed form for this recurrence relation:

Lemma B.1. *VO has the following closed-form representation:*

$$
VO_{2l+1}(0, 1, c) = (-1)^c \cdot \frac{2 \cdot 4^l + 1}{6 \cdot 4^l} \qquad VO_{2l+1}(1, 0, c) = (-1)^{1-c} \cdot \frac{4^l - 1}{3 \cdot 4^l}
$$

Proof. By induction on l:

- We have $VO_1(0,1,c) \stackrel{(B.11)}{=} (-1)^c \cdot \frac{1}{2} = (-1)^c \cdot \frac{2 \cdot 4^0 + 1}{6 \cdot 4^0}$ and $VO_1(1,0,c) \stackrel{(B.11)}{=} 0 = (-1)^{1-c} \cdot \frac{4^0 - 1}{3 \cdot 4^0}$.

- We have

$$
\begin{aligned}
VO_{2l+3}(0,1,c) \ \stackrel{(B.15)}{=} \ & \frac{1}{4} \left(3\, VO_{2l+1}(0,1,c) - VO_{2l+1}(1,0,c) \right) \\
\stackrel{(\mathrm{IH})}{=} \ & \frac{1}{4} \left(3 \cdot (-1)^c \cdot \frac{2 \cdot 4^l + 1}{6 \cdot 4^l} - (-1)^{1-c} \cdot \frac{4^l - 1}{3 \cdot 4^l} \right) \\
= \ & (-1)^c \cdot \frac{1}{4} \left(\frac{2 \cdot 4^l + 1}{6 \cdot 4^l} + \frac{4^l - 1}{3 \cdot 4^l} \right) \\
= \ & (-1)^c \cdot \frac{1}{4} \cdot \frac{8 \cdot 4^l - 1}{6\dot{4}^l} \\
= \ & (-1)^c \cdot \frac{2 \cdot 4^{l+1} + 1}{6 \cdot 4^{l+1}}
\end{aligned}
$$

$$
\begin{aligned}
VO_{2l+3}(1,0,c) \ \stackrel{(B.9)}{=} \ & \frac{1}{2} \left(VO_{2l+1}(1,0,c) - VO_{2l+1}(0,1,c) \right) \\
\stackrel{(\mathrm{IH})}{=} \ & \frac{1}{2} \left((-1)^{1-c} \cdot \frac{4^l - 1}{3 \cdot 4^l} - (-1)^c \cdot \frac{2 \cdot 4^l + 1}{6 \cdot 4^l} \right) \\
= \ & (-1)^{1-c} \cdot \frac{1}{2} \left(\frac{4^l - 1}{3 \cdot 4^l} + \frac{2 \cdot 4^l + 1}{6 \cdot 4^l} \right) \\
= \ & (-1)^{1-c} \cdot \frac{1}{2} \cdot \frac{4 \cdot 4^l - 1}{6 \cdot 4^l} \\
= \ & (-1)^{1-c} \cdot \frac{4^{l+1} - 1}{3 \cdot 4^{l+1}} \qquad\qquad \square
\end{aligned}
$$

Corollary B.2. *VE has the following closed-form representation:*

$$
\begin{aligned}
VE_{2l}(0,1,c) \ & \stackrel{(B.5)}{=} \ VO_{2l-1}(0,1,c) \ \stackrel{(Lem.\ B.1)}{=} \ (-1)^c \cdot \frac{2 \cdot 4^{l-1} + 1}{6 \cdot 4^{l-1}} \\
VE_{2l}(1,0,c) \ & \stackrel{(B.4)}{=} \ VO_{2l+1}(1,0,c) \ \stackrel{(Lem.\ B.1)}{=} \ (-1)^{1-c} \cdot \frac{4^l - 1}{3 \cdot 4^l}
\end{aligned}
$$

$$
\begin{aligned}
VE_{2l}(1,1,c) \ \stackrel{(B.5)}{=} \ & \frac{1}{2} \left(VO_{2l-1}(0,1,c) - VO_{2l-1}(1,0,c) \right) \\
\stackrel{(Lem.B.1)}{=} \ & \frac{1}{2} (-1)^c \left(\frac{2 \cdot 4^{l-1} + 1}{6 \cdot 4^{l-1}} + \frac{4^{l-1} - 1}{3 \cdot 4^{l-1}} \right) \\
= \ & (-1)^c \frac{4^l - 1}{3 \cdot 4^l}
\end{aligned}
$$

Now, recalling the definition of k_e, k_o, and c, we finally have

Theorem B.3. *We have*

$$Var\left(\frac{\partial\langle H\rangle}{\partial\theta_1}\right) = \begin{cases} (-1)^{\sum a_j b_j} \cdot VE_\ell(\sum a_j, \sum b_j, a_1) & \text{if } \ell \text{ is even and } a_1 = ... = a_n \\ (-1)^{\sum a_j b_j} \cdot VO_\ell(\sum b_j, \sum a_j, b_1) & \text{if } \ell \text{ is odd and } b_1 = ... = b_n \\ 0 & \text{otherwise.} \end{cases}$$

Corollary B.4. *Either $Var\left(\frac{\partial\langle H\rangle}{\partial\theta_1}\right) = 0$ or $Var\left(\frac{\partial\langle H\rangle}{\partial\theta_1}\right) \to \frac{1}{3}$ for $\ell \to \infty$.*

Proof. This essentially follows from the fact that all terms in Lemma B.1 and Corollary B.2 converge to $\pm\frac{1}{3}$. To be precise, we can show that the negation always cancels out by considering the different cases: Suppose ℓ is odd and $b_1 = ... = b_n = 0$. Then

$$\text{Var}\left(\frac{\partial\langle H\rangle}{\partial\theta_1}\right) = VO_\ell(0, \sum a_j, 0) \overset{(B.7,\text{Lem. } B.1)}{=} \begin{cases} 0 & \text{if } \sum a_j = 0 \\ \frac{2\cdot 4^{\lfloor\ell/2\rfloor}+1}{6\cdot 4^{\lfloor\ell/2\rfloor}} \to \frac{1}{3} & \text{if } \sum a_j = 1 \end{cases}$$

If $b_1 = ... = b_n = 1$, then we have $\sum b_j = 1$ since we assume that n is odd. Thus,

$$\text{Var}\left(\frac{\partial\langle H\rangle}{\partial\theta_1}\right) = VO_\ell(1, \sum a_j, 1) \overset{(B.13,\text{Lem. } B.1)}{=} \begin{cases} \frac{4^{\lfloor\ell/2\rfloor}-1}{3\cdot 4^{\lfloor\ell/2\rfloor}} \to \frac{1}{3} & \text{if } \sum a_j = 0 \\ \frac{2\cdot 4^{\lfloor\ell/2\rfloor}+1}{6\cdot 4^{\lfloor\ell/2\rfloor}} \to \frac{1}{3} & \text{if } \sum a_j = 1 \end{cases}$$

We do not get a negation in the second case since $\sum a_j = 1$ implies that $(-1)^{\sum a_j b_j} = 1$. The case for even ℓ and $a_1 = ... = a_n$ is symmetric. Otherwise, $Var\left(\frac{\partial\langle H\rangle}{\partial\theta_1}\right) = 0$. $\square$

Appendix C

Additional Lemmas and Proofs

Lemma C.1. *For all $x \in \{0, 1\}$, we have*

$$\text{(C.1)}$$

Proof.

□

Lemma C.2.

$$\text{(C.2)}$$

Proof.

□

Proof of Proposition 4.4. We first prove than we cannot represent the $CR_Z(\theta)$ gate using less than two parametrised spiders. Suppose we had

$$CR_Z(\theta) = e^{ig(\theta)} \; \boxed{D} \text{ with } f(\theta)$$

But then

$$\frac{\partial}{\partial\theta} CR_Z(\theta) \overset{(3.1)}{=} ig'(\theta)e^{ig(\theta)} \boxed{D}^{f(\theta)} + e^{ig(\theta)} i f'(\theta) \boxed{D}^{\pi, f(\theta)}$$

$$\overset{(cp)}{=} ig'(\theta)CR_Z(\theta) + e^{ig(\theta)} i f'(\theta) e^{if(\theta)} \boxed{D}^{\pi} \tag{C.3}$$

Note that the diagram on the right-hand side no longer depends on θ. Since $\frac{\partial}{\partial\theta} CR_Z(\theta) = \mathrm{diag}(0, 0, -\frac{i\theta}{2}e^{-i\frac{\theta}{2}}, \frac{i\theta}{2}e^{i\frac{\theta}{2}})$ is diagonal, we must also have

$$\boxed{D}^{\pi} = \mathrm{diag}(a, b, c, d)$$

for some constants $a, b, c, d \in \mathbb{C}$. By comparing the diagonal entries to (C.3), we get the equations

$$g'(\theta) + ae^{i(f(\theta)+g(\theta))}f'(\theta) = 0 \tag{C.4}$$

$$g'(\theta) + be^{i(f(\theta)+g(\theta))}f'(\theta) = 0 \tag{C.5}$$

$$g'(\theta)e^{-i\frac{\theta}{2}} + ce^{i(f(\theta)+g(\theta))}f'(\theta) = -\frac{\theta}{2}e^{-i\frac{\theta}{2}} \tag{C.6}$$

$$g'(\theta)e^{i\frac{\theta}{2}} + de^{i(f(\theta)+g(\theta))}f'(\theta) = \frac{\theta}{2}e^{i\frac{\theta}{2}} \tag{C.7}$$

Since (C.4) implies $g'(\theta) = -ae^{i(f(\theta)+g(\theta))}f'(\theta)$, we can rewrite (C.6) and (C.7) to

$$e^{i(f(\theta)+g(\theta))}f'(\theta) \cdot (c - ae^{-i\frac{\theta}{2}}) = -\frac{\theta}{2}e^{-i\frac{\theta}{2}}$$

$$e^{i(f(\theta)+g(\theta))}f'(\theta) \cdot (d - ae^{i\frac{\theta}{2}}) = \frac{\theta}{2}e^{i\frac{\theta}{2}}$$

In particular, this implies that

$$c - ae^{-i\frac{\theta}{2}} = -d + ae^{i\frac{\theta}{2}}$$

for all θ, which is clearly not possible for constant a, b, c.

Thus, we can conclude that we cannot represent $CR_Z(\theta)$ with less than two parametrised spiders. Now, suppose there were some other unitary $U(\theta) = e^{i\theta H}$ whose Hermitian generator has eigenvalues $-\lambda, 0, \lambda$ that can be implemented with a single parametrised spider:

However, by Lemma 4.15 this would immediately yield a one-spider representation on $CR_Z(\theta)$ via

which is not possible. □

Proof of Lemma 4.6. First, note that

$$\text{(C.8)}$$

On the other hand, we have

such that

$$-\!\!\boxed{\alpha}\,\boxed{-\alpha}\!-\;\;-\;\;-\!\!\boxed{-\alpha}\,\boxed{\alpha}\!-\;=\;(e^{i\alpha}-e^{-i\alpha})|0\rangle\langle 1|+(e^{-i\alpha}-e^{i\alpha})|1\rangle\langle 0|$$
$$=(e^{i\alpha}-e^{-i\alpha})(|0\rangle\langle 1|-|1\rangle\langle 0|)$$
$$\overset{(C.8)}{=}\;2i\sin(\alpha)\;-\!\!\boxed{\pi}\,\boxed{\pi}\!-\;.\hspace{2cm}\square$$

Proof of Lemma 5.2. By comparing the action on the computational basis:
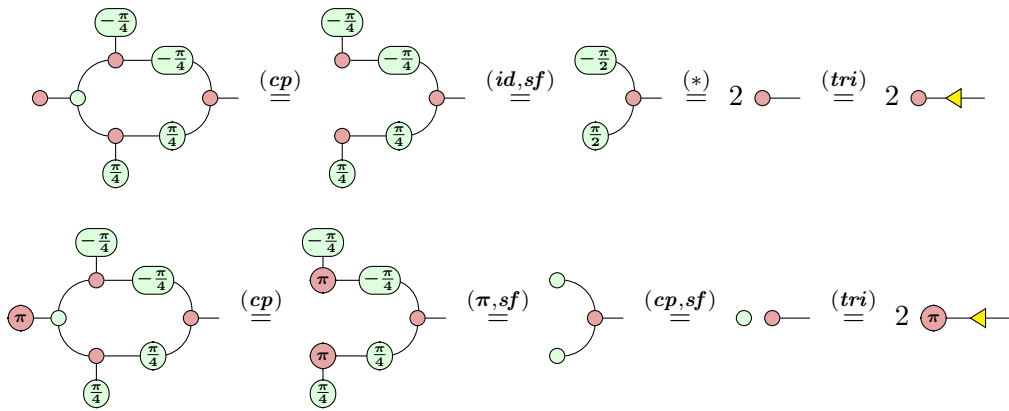
where the step $(*)$ follows again from plugging in the computational basis:

$$\square$$
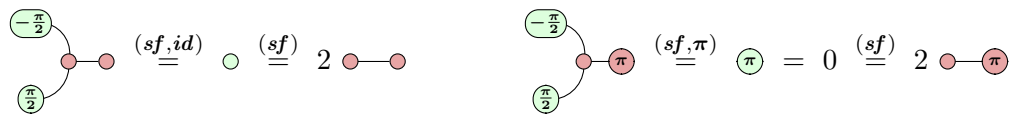
Proof of Lemma 5.6.

$\square$
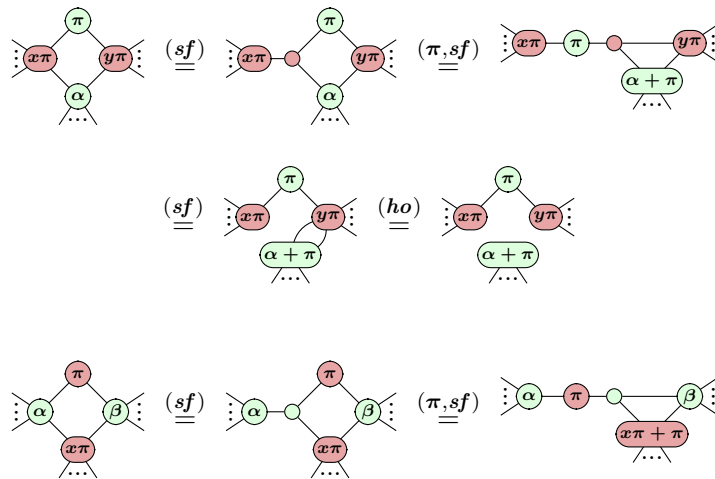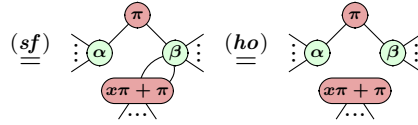
Proof of Lemma 5.7.
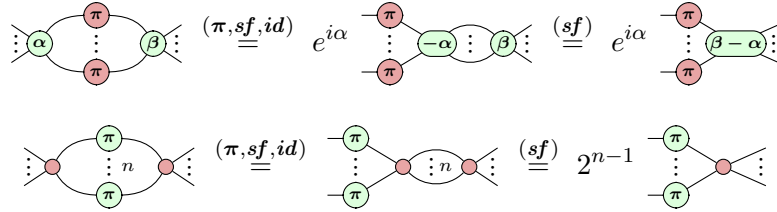
Note that the scalar 2^{n-1} appears in the last step according to Lemma 2.13. $\square$

Proof of Lemma 5.8. We prove the three equations separately:

- We have
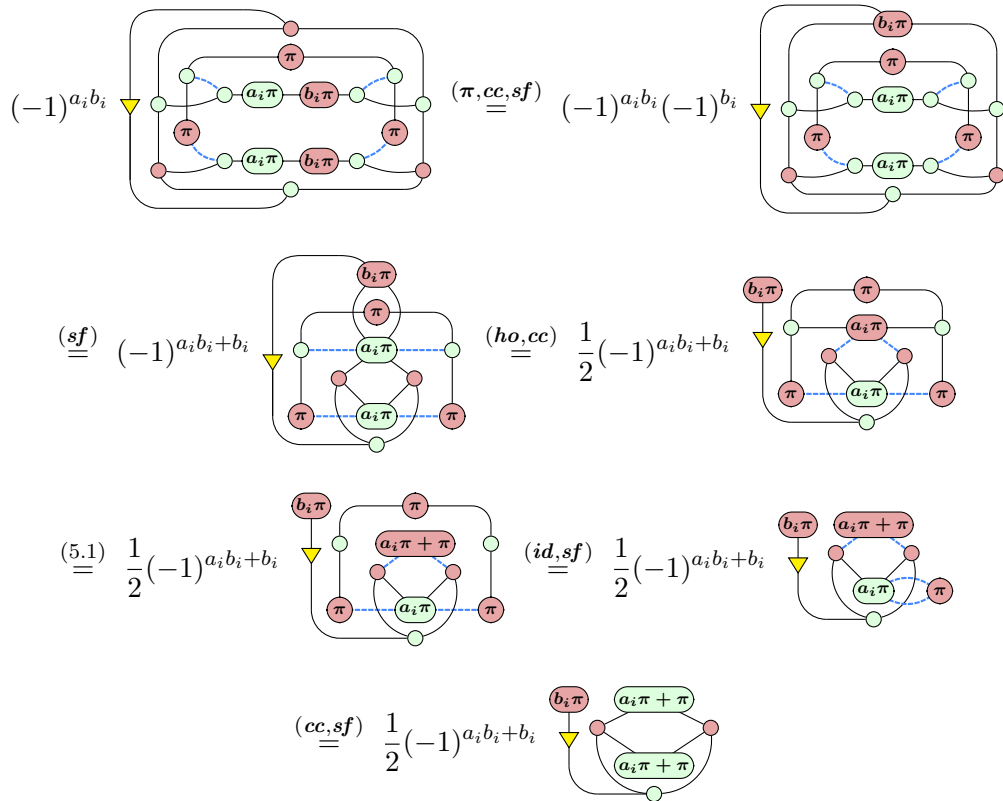
If $b_i = 0$, we get

$$\frac{1}{2} \quad \overset{(\boldsymbol{tri},\boldsymbol{sf})}{=} \quad \frac{1}{2} \quad \overset{(\boldsymbol{id},\underline{\boldsymbol{sf}},\boldsymbol{ho})}{=} \quad \frac{1}{2}\ \frac{\boxed{a_i\pi+\pi}}{\boxed{a_i\pi+\pi}} \ = \ 2a_i.$$

If $b_i = 1$, we get

$$\frac{1}{2}(-1)^{a_i+1} \qquad \overset{(\boldsymbol{tri},\underline{\boldsymbol{cp}},\boldsymbol{sf})}{=} \quad \frac{1}{2}(-1)^{a_i+1}$$

$$\overset{(\boldsymbol{\pi},\underline{\boldsymbol{sf}},\boldsymbol{id})}{=} \quad \frac{1}{2}(-1)^{a_i+1}(-1)^{a_i+1} \ \circ \ = \ 1.$$

- We have

$$(-1)^{a_ib_i} \qquad \overset{(\boldsymbol{\pi},\underline{\boldsymbol{cc}},\boldsymbol{sf})}{=} \quad (-1)^{a_ib_i}$$

$$\overset{(\underline{\boldsymbol{sf}})}{=} \quad (-1)^{a_ib_i} \qquad \overset{(\boldsymbol{ho},\underline{\boldsymbol{cp}},\boldsymbol{cc},\boldsymbol{sf})}{=} \quad (-1)^{a_ib_i}(-1)^{a_i}$$

$$\overset{(\underline{\boldsymbol{cc}})}{=} \quad \frac{1}{2}(-1)^{a_ib_i+a_i} \qquad \overset{(\boldsymbol{sf},\boldsymbol{ho})}{=} \quad \frac{1}{2}(-1)^{a_ib_i+a_i}$$

$$\overset{(\boldsymbol{\pi},\boldsymbol{sf})}{=} \quad \frac{1}{2}(-1)^{a_ib_i+a_i} \qquad = \quad \begin{cases} 0 & \text{if } b_i = 0 \\ 1 & \text{if } b_i = 1 \end{cases}$$

- We have

$$(-1)^{a_i b_i}$$

$$\stackrel{(\boldsymbol{\pi},\boldsymbol{cc},\boldsymbol{sf})}{=} (-1)^{a_i b_i}$$ $$\stackrel{(\boldsymbol{sf},\boldsymbol{ho})}{=} (-1)^{a_i b_i}$$

If $b_i = 0$, we get

 $$\stackrel{(\boldsymbol{tri},\boldsymbol{sf})}{=}$$ $$\stackrel{(\boldsymbol{id},\boldsymbol{sf},\boldsymbol{ho})}{=}$$

$$\stackrel{(\boldsymbol{cc})}{=}$$ $$\stackrel{(\boldsymbol{sf},\boldsymbol{ho})}{=}$$ $$\stackrel{(\boldsymbol{cp},\boldsymbol{tri},\boldsymbol{sf})}{=}$$ $$= \begin{cases} 4 & \text{if } a_i = 0 \\ 2 & \text{if } a_i = 1 \end{cases}$$

If $b_i = 1$, we get

$$(-1)^{a_i}$$ $$\stackrel{(\boldsymbol{tri},\boldsymbol{cp},\boldsymbol{sf})}{=}$$

$$\stackrel{(\boldsymbol{cp},\boldsymbol{cc},\boldsymbol{sf})}{=} (-1)^{a_i}(-1)^{a_i}$$ $$\stackrel{(\boldsymbol{cc})}{=} \frac{1}{2}$$
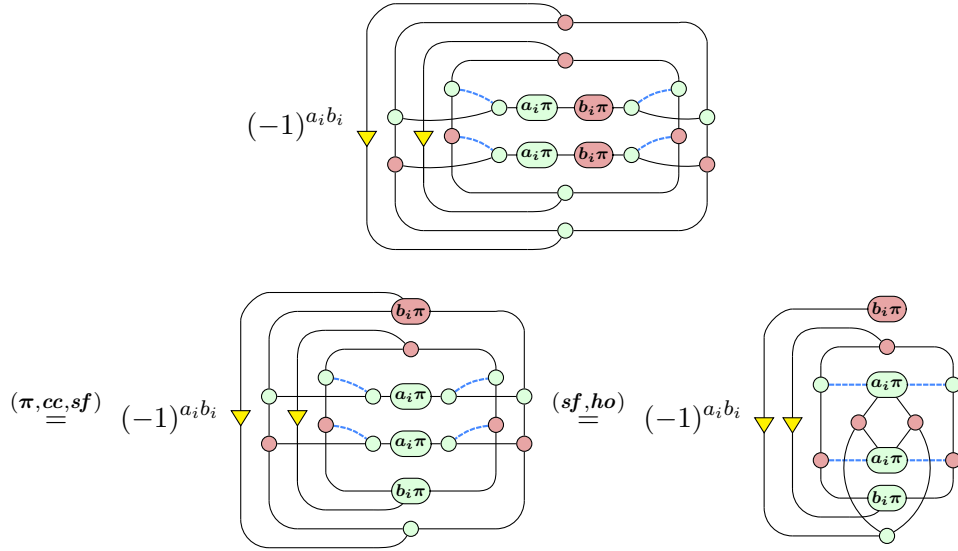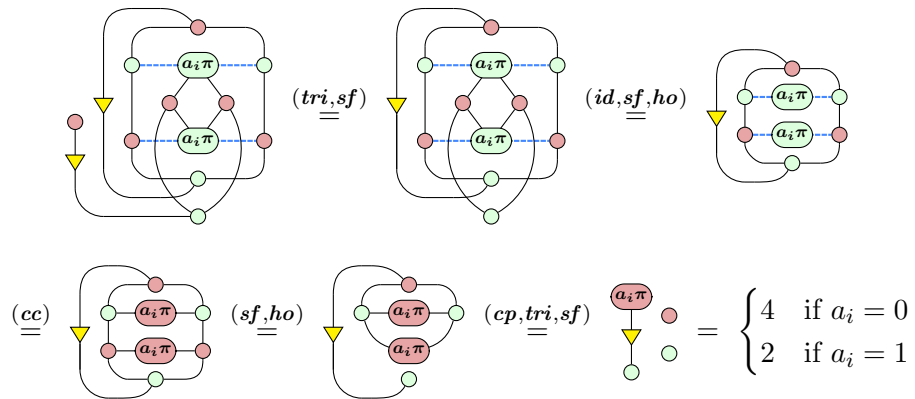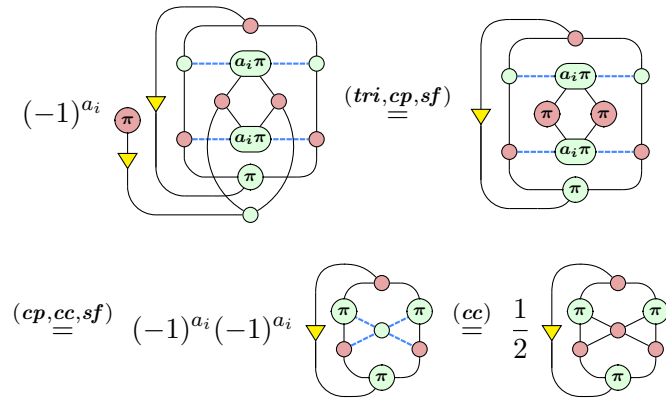
$$\overset{(\boldsymbol{sf},\boldsymbol{ho})}{=} \frac{1}{2}\ \ \overset{(\boldsymbol{cp},\boldsymbol{sf})}{=} \frac{1}{2}\ \ = \ 1.$$

$\square$

Proof of Lemma 5.13. We prove both cases separately:

- We have

If $a_i = 0$, the diagram becomes 0. If $a_i = 1$, we get

- We have

$$\overset{(\boldsymbol{cp},\underline{\boldsymbol{cc}},\boldsymbol{sf})}{=} \quad 2(-1)^{a_i b_i}$$

If $a_i = 0$, we get

$$2 \quad \overset{(\boldsymbol{tri},\underline{\boldsymbol{sf}},\boldsymbol{cc})}{=} \quad 2$$

$$\overset{(\boldsymbol{cp},\boldsymbol{sf})}{=} \quad 2 \quad \overset{(\boldsymbol{sf},\boldsymbol{ho})}{=} \quad 4$$

If $a_i = 1$, we get

$$2(-1)^{b_i} \quad \overset{(\boldsymbol{tri},\boldsymbol{cp})}{=} \quad 2$$

$$\overset{(\boldsymbol{cc},\boldsymbol{sf})}{=} \quad \overset{(5.2)}{=} \quad 2$$

□

Proof of Lemma 5.15.

$$\overset{(\boldsymbol{sf})}{=} \quad \overset{(\boldsymbol{sc})}{=} \quad \overset{(\boldsymbol{cp},\boldsymbol{sf})}{=}$$
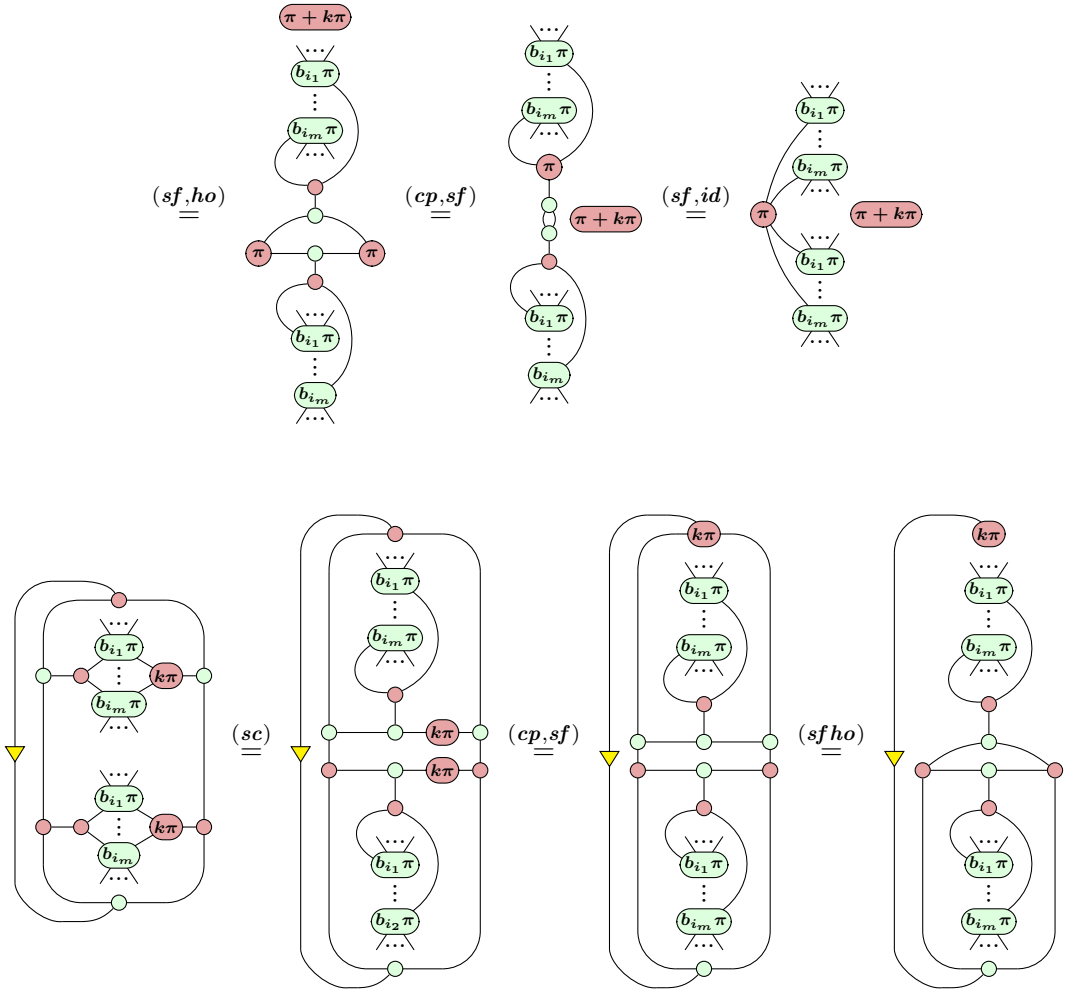
If $k = 0$, we get
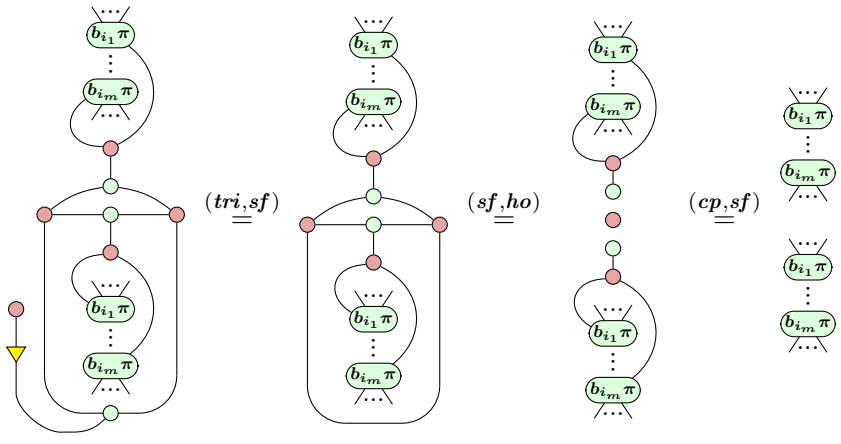
If $k = 1$, we get
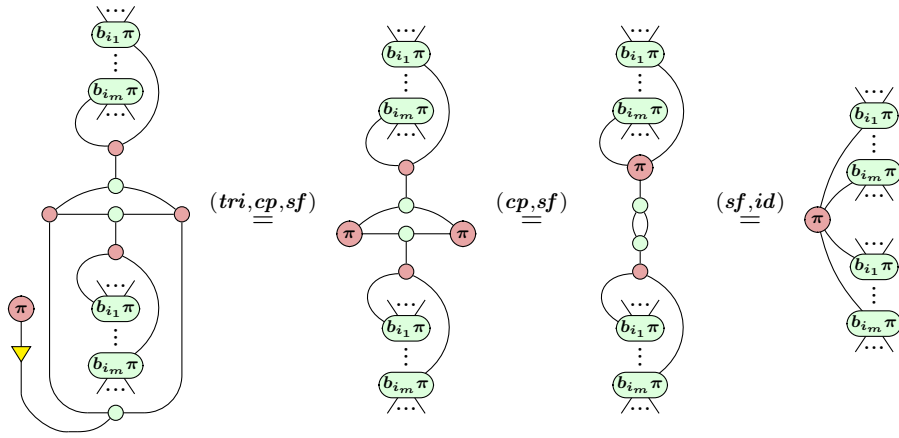
Proof of Lemma 5.21. We have
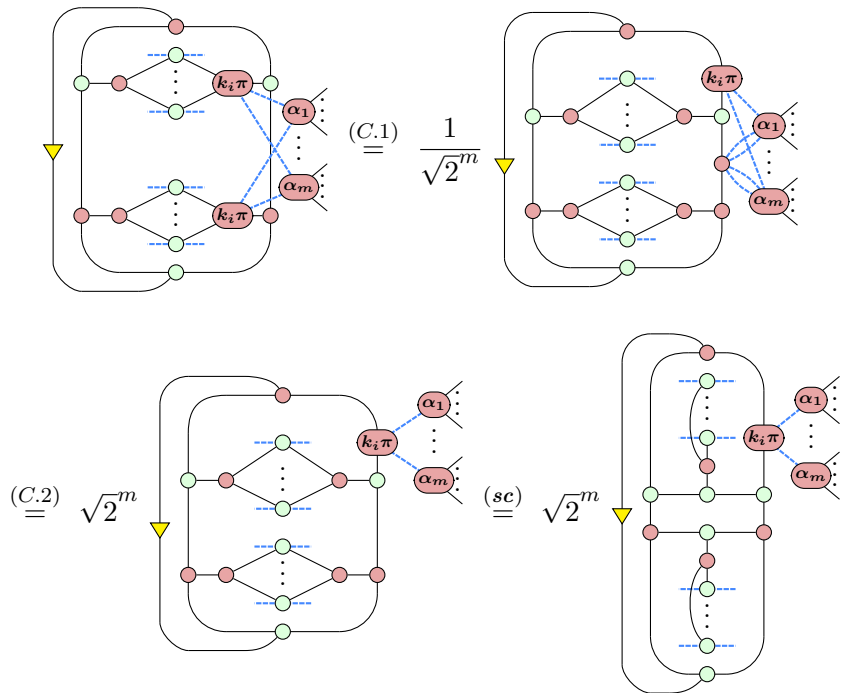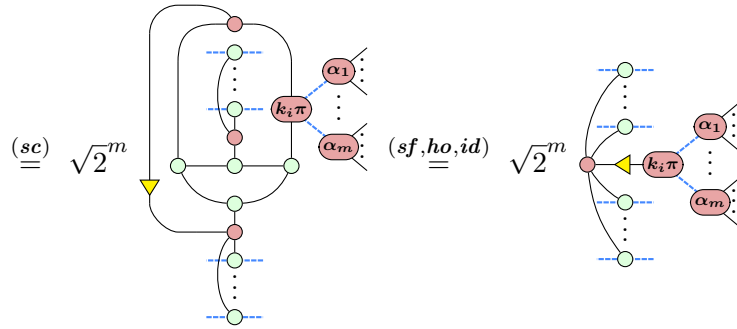
For the second cycle, we have
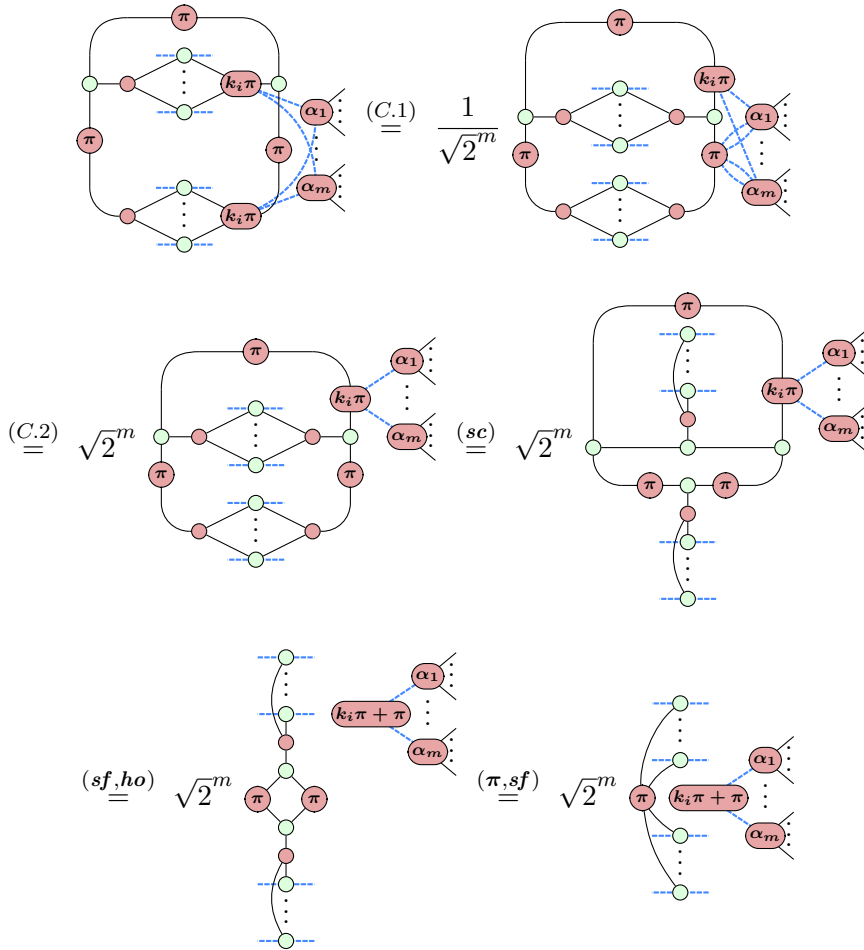
Bibliography

[1] Lov K Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 212–219, 1996.

[2] Peter W Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM review*, 41(2):303–332, 1999.

[3] Michael J Bremner, Ashley Montanaro, and Dan J Shepherd. Achieving quantum supremacy with sparse and noisy commuting quantum computations. *Quantum*, 1:8, 2017.

[4] Jacob Biamonte, Peter Wittek, Nicola Pancotti, Patrick Rebentrost, Nathan Wiebe, and Seth Lloyd. Quantum machine learning. *Nature*, 549(7671):195–202, 2017.

[5] Aram W Harrow and John C Napp. Low-depth gradient measurements can improve convergence in variational hybrid quantum-classical algorithms. *Physical Review Letters*, 126(14):140502, 2021.

[6] Kosuke Mitarai, Makoto Negoro, Masahiro Kitagawa, and Keisuke Fujii. Quantum circuit learning. *Physical Review A*, 98(3):032309, 2018.

[7] Maria Schuld, Ville Bergholm, Christian Gogolin, Josh Izaac, and Nathan Killoran. Evaluating analytic gradients on quantum hardware. *Physical Review A*, 99(3):032331, 2019.

[8] Gian-Luca R Anselmetti, David Wierichs, Christian Gogolin, and Robert M Parrish. Local, expressive, quantum-number-preserving VQE ansätze for fermionic systems. *New Journal of Physics*, 23(11):113010, 2021.

[9] David Wierichs, Josh Izaac, Cody Wang, and Cedric Yen-Yu Lin. General parameter-shift rules for quantum gradients. *Quantum*, 6:677, 2022.

[10] Jarrod R McClean, Sergio Boixo, Vadim N Smelyanskiy, Ryan Babbush, and Hartmut Neven. Barren plateaus in quantum neural network training landscapes. *Nature communications*, 9(1):1–6, 2018.

[11] Zoë Holmes, Kunal Sharma, Marco Cerezo, and Patrick J Coles. Connecting ansatz expressibility to gradient magnitudes and barren plateaus. *PRX Quantum*, 3(1):010313, 2022.

[12] Marco Cerezo, Akira Sone, Tyler Volkoff, Lukasz Cincio, and Patrick J Coles. Cost function dependent barren plateaus in shallow parametrized quantum circuits. *Nature communications*, 12(1):1–12, 2021.

[13] Chen Zhao and Xiao-Shan Gao. Analyzing the barren plateau phenomenon in training quantum neural networks with the ZX-calculus. *Quantum*, 5:466, 2021.

[14] Bob Coecke and Ross Duncan. Interacting quantum observables. In *International Colloquium on Automata, Languages, and Programming*, pages 298–310. Springer, 2008.

[15] Ross Duncan, Aleks Kissinger, Simon Perdrix, and John Van De Wetering. Graph-theoretic simplification of quantum circuits with the ZX-calculus. *Quantum*, 4:279, 2020.

[16] Arianne Meijer-van de Griend and Ross Duncan. Architecture-aware synthesis of phase polynomials for NISQ devices. *arXiv preprint arXiv:2004.06052*, 2020.

[17] Aleks Kissinger and John van de Wetering. Simulating quantum circuits with

ZX-calculus reduced stabiliser decompositions. *Quantum Science and Technology*, 2022.

[18] Razin Shaikh, Quanlong Wang, and Richie Yeung. How to sum and exponentiate hamiltonians in ZXW calculus. *Quantum Physics and Logic*, 2022.

[19] Quanlong Wang and Richie Yeung. Differentiating and integrating ZX diagrams. *arXiv preprint arXiv:2201.13250*, 2022.

[20] Sukin Sim, Peter D Johnson, and Alán Aspuru-Guzik. Expressibility and entangling capability of parameterized quantum circuits for hybrid quantum-classical algorithms. *Advanced Quantum Technologies*, 2(12):1900070, 2019.

[21] Dimitri Kartsaklis, Ian Fan, Richie Yeung, Anna Pearson, Robin Lorenz, Alexis Toumi, Giovanni de Felice, Konstantinos Meichanetzidis, Stephen Clark, and Bob Coecke. lambeq: An efficient high-level python library for quantum NLP. *arXiv preprint arXiv:2110.04236*, 2021.

[22] Marshall H Stone. On one-parameter unitary groups in Hilbert space. *Annals of Mathematics*, pages 643–648, 1932.

[23] Maria Schuld and Nathan Killoran. Quantum machine learning in feature hilbert spaces. *Physical review letters*, 122(4):040504, 2019.

[24] Marcello Benedetti, Delfina Garcia-Pintos, Oscar Perdomo, Vicente Leyton-Ortega, Yunseong Nam, and Alejandro Perdomo-Ortiz. A generative modeling approach for benchmarking and training shallow quantum circuits. *npj Quantum Information*, 5(1):1–9, 2019.

[25] Jin-Guo Liu and Lei Wang. Differentiable learning of quantum circuit born machines. *Physical Review A*, 98(6):062324, 2018.

[26] Seth Lloyd and Christian Weedbrook. Quantum generative adversarial learning. *Physical review letters*, 121(4):040502, 2018.

[27] Abhinav Kandala, Antonio Mezzacapo, Kristan Temme, Maika Takita, Markus

Brink, Jerry M Chow, and Jay M Gambetta. Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets. *Nature*, 549(7671):242–246, 2017.

[28] Yudong Cao, Jonathan Romero, Jonathan P Olson, Matthias Degroote, Peter D Johnson, Mária Kieferová, Ian D Kivlichan, Tim Menke, Borja Peropadre, Nicolas PD Sawaya, et al. Quantum chemistry in the age of quantum computing. *Chemical reviews*, 119(19):10856–10915, 2019.

[29] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. A quantum approximate optimization algorithm. *arXiv preprint arXiv:1411.4028*, 2014.

[30] Konstantinos Meichanetzidis, Stefano Gogioso, Giovanni De Felice, Nicolò Chiappori, Alexis Toumi, and Bob Coecke. Quantum natural language processing on near-term quantum computers. *arXiv preprint arXiv:2005.04147*, 2020.

[31] William Huggins, Piyush Patil, Bradley Mitchell, K Birgitta Whaley, and E Miles Stoudenmire. Towards quantum machine learning with tensor networks. *Quantum Science and technology*, 4(2):024001, 2019.

[32] Xiangjian Qian and Mingpu Qin. From tree tensor network to multiscale entanglement renormalization ansatz. *Physical Review B*, 105(20):205102, 2022.

[33] Y Du, MH Hsieh, T Liu, and D Tao. The expressive power of parameterized quantum circuits. *arXiv preprint arXiv:1810.11922*, 2018.

[34] Xavier Bonet-Monroig, Hao Wang, Diederick Vermetten, Bruno Senjean, Charles Moussa, Thomas Bäck, Vedran Dunjko, and Thomas E O'Brien. Performance comparison of optimization methods on variational quantum algorithms. *arXiv preprint arXiv:2111.13454*, 2021.

[35] James C Spall et al. Multivariate stochastic approximation using a simultaneous perturbation gradient approximation. *IEEE transactions on automatic control*, 37(3):332–341, 1992.

[36] Michael JD Powell. A direct search optimization method that models the objective and constraint functions by linear interpolation. *Advances in optimization and numerical analysis*, pages 51–67, 1994.

[37] Jonas M Kübler, Andrew Arrasmith, Lukasz Cincio, and Patrick J Coles. An adaptive optimizer for measurement-frugal variational algorithms. *Quantum*, 4:263, 2020.

[38] Ken M Nakanishi, Keisuke Fujii, and Synge Todo. Sequential minimal optimization for quantum-classical hybrid algorithms. *Physical Review Research*, 2(4):043158, 2020.

[39] Max Wilson, Rachel Stromswold, Filip Wudarski, Stuart Hadfield, Norm M Tubman, and Eleanor G Rieffel. Optimizing quantum heuristics with meta-learning. *Quantum Machine Intelligence*, 3(1):1–14, 2021.

[40] Anna Choromanska, Mikael Henaff, Michael Mathieu, Gérard Ben Arous, and Yann LeCun. The loss surfaces of multilayer networks. In *Artificial intelligence and statistics*, pages 192–204. PMLR, 2015.

[41] Xuchen You and Xiaodi Wu. Exponentially many local minima in quantum neural networks. In *International Conference on Machine Learning*, pages 12144–12155. PMLR, 2021.

[42] Kang Feng Ng and Quanlong Wang. A universal completion of the ZX-calculus. *arXiv preprint arXiv:1706.09877*, 2017.

[43] Aleks Kissinger and John van de Wetering. Reducing T-count with the ZX-calculus. *arXiv preprint arXiv:1903.10477*, 2019.

[44] Alexander Cowtan, Will Simmons, and Ross Duncan. A generic compilation strategy for the unitary coupled cluster ansatz. *arXiv preprint arXiv:2007.10515*, 2020.

[45] Ross Duncan. A graphical approach to measurement-based quantum computing. *arXiv preprint arXiv:1203.6242*, 2012.

[46] Aleks Kissinger and John van de Wetering. Universal MBQC with generalised parity-phase interactions and Pauli measurements. *Quantum*, 3:134, 2019.

[47] Niel de Beaudrap and Dominic Horsman. The ZX calculus is a language for surface code lattice surgery. *Quantum*, 4:218, 2020.

[48] Quanlong Wang. Algebraic complete axiomatisation of ZX-calculus with a normal form via elementary matrix operations. *arXiv preprint arXiv:2007.13739*, 2020.

[49] Bob Coecke and Aleks Kissinger. *Picturing Quantum Processes: A First Course in Quantum Theory and Diagrammatic Reasoning.* Cambridge University Press, 2017.

[50] Alexander Cowtan, Silas Dilkes, Ross Duncan, Will Simmons, and Seyon Sivarajah. Phase gadget synthesis for shallow circuits. *arXiv preprint arXiv:1906.01734*, 2019.

[51] Aleks Kissinger and John van de Wetering. *Picturing Quantum Software.* 2022.

[52] Richie Yeung. Diagrammatic design and study of ansätze for quantum machine learning. *arXiv preprint arXiv:2011.11073*, 2020.

[53] Alexis Toumi, Richie Yeung, and Giovanni de Felice. Diagrammatic differentiation for quantum machine learning. *arXiv preprint arXiv:2103.07960*, 2021.

[54] William Wernick. Complete sets of logical functions. *Transactions of the American Mathematical Society*, 51:117–132, 1942.

[55] Ville Bergholm, Josh Izaac, Maria Schuld, Christian Gogolin, M Sohaib Alam, Shahnawaz Ahmed, Juan Miguel Arrazola, Carsten Blank, Alain Delgado, Soran Jahangiri, et al. Pennylane: Automatic differentiation of hybrid quantum-classical computations. *arXiv preprint arXiv:1811.04968*, 2018.

[56] Gavin E Crooks. Gradients of parameterized quantum gates using the parameter-shift rule and gate decomposition. *arXiv preprint arXiv:1905.13311*, 2019.

[57] Brooks Foxen, Charles Neill, Andrew Dunsworth, Pedram Roushan, Ben Chiaro, Anthony Megrant, Julian Kelly, Zijun Chen, Kevin Satzinger, Rami Barends, et al. Demonstrating a continuous set of two-qubit gates for near-term quantum algorithms. *Physical Review Letters*, 125(12):120504, 2020.

[58] Oleksandr Kyriienko and Vincent E Elfving. Generalized quantum circuit differentiation rules. *Physical Review A*, 104(5):052417, 2021.

[59] Vladimir Britanak, Patrick C Yip, and Kamisetty Ramamohan Rao. *Discrete cosine and sine transforms: general properties, fast algorithms and integer approximations.* Elsevier, 2010.

[60] Anil K Jain. A sinusoidal family of unitary transforms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (4):356–365, 1979.

[61] Dan Shepherd and Michael J Bremner. Temporally unstructured quantum computation. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 465(2105):1413–1439, 2009.

[62] Michael J Bremner, Ashley Montanaro, and Dan J Shepherd. Average-case complexity versus approximate simulation of commuting quantum computations. *Physical review letters*, 117(8):080501, 2016.

[63] Austin P Lund, Michael J Bremner, and Timothy C Ralph. Quantum sampling problems, boson sampling and quantum supremacy. *npj Quantum Information*, 3(1):1–8, 2017.

[64] Michael J Bremner, Richard Jozsa, and Dan J Shepherd. Classical simulation of commuting quantum computations implies collapse of the polynomial hierarchy.

Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences, 467(2126):459–472, 2011.

[65] Aleks Kissinger, John van de Wetering, and Renaud Vilmart. Classical simulation of quantum circuits with partial and graphical stabiliser decompositions. *arXiv preprint arXiv:2202.09202*, 2022.

[66] John Coates and Ramdorai Sujatha. *Cyclotomic fields and zeta values*. Springer Science & Business Media, 2006.

[67] Edward Grant, Leonard Wossnig, Mateusz Ostaszewski, and Marcello Benedetti. An initialization strategy for addressing barren plateaus in parametrized quantum circuits. *Quantum*, 3:214, 2019.

[68] Ankit Kulshrestha and Ilya Safro. BEINIT: Avoiding barren plateaus in variational quantum algorithms. *arXiv preprint arXiv:2204.13751*, 2022.

[69] Ali Rad, Alireza Seif, and Norbert M Linke. Surviving the barren plateau in variational quantum circuits with bayesian learning initialization. *arXiv preprint arXiv:2203.02464*, 2022.

[70] Stefan H Sack, Raimel A Medina, Alexios A Michailidis, Richard Kueng, and Maksym Serbyn. Avoiding barren plateaus using classical shadows. *PRX Quantum*, 3(2):020365, 2022.

[71] Xia Liu, Geng Liu, Jiaxin Huang, and Xin Wang. Mitigating barren plateaus of variational quantum eigensolvers. *arXiv preprint arXiv:2205.13539*, 2022.

[72] Andrea Skolik, Jarrod R McClean, Masoud Mohseni, Patrick van der Smagt, and Martin Leib. Layerwise learning for quantum neural networks. *Quantum Machine Intelligence*, 3(1):1–11, 2021.

[73] Taylor L Patti, Khadijeh Najafi, Xun Gao, and Susanne F Yelin. Entanglement devised barren plateau mitigation. *Physical Review Research*, 3(3):033090, 2021.