Estimating the gradient and higher-order derivatives on quantum hardware

Andrea Mari,[1, 2] Thomas R. Bromley,[1] and Nathan Killoran[1]

[1]*Xanadu, Toronto, ON, M5G 2C8, Canada*
[2]*Unitary Fund, CA, USA*

For a large class of variational quantum circuits, we show how arbitrary-order derivatives can be analytically evaluated in terms of simple *parameter-shift rules*, i.e., by running the same circuit with different shifts of the parameters. As particular cases, we obtain parameter-shift rules for the Hessian of an expectation value and for the metric tensor of a variational state, both of which can be efficiently used to analytically implement second-order optimization algorithms on a quantum computer. We also consider the impact of statistical noise by studying the mean squared error of different derivative estimators. We find that the performance of different estimators and optimizers is intertwined with the values of different hyperparameters, such as a step size or a number of shots. Our findings are supported by several numerical and hardware experiments, including an experimental estimation of the Hessian of a simple variational circuit and an implementation of the Newton optimizer.

I. INTRODUCTION

Many algorithms for near-term quantum computers are based on variational circuits [1–6], i.e., sequences of quantum gates depending on classical parameters that are recursively optimized for solving specific problems. Some of the most important examples are the variational quantum eigensolver (VQE) [1, 7], the quantum approximate optimization algorithm (QAOA) [8], and all variational implementations of quantum machine learning algorithms [2, 3, 6, 9]. In practice, all these cases share the same workflow: a quantum computer is used for measuring one or more expectation values which are classically combined into some cost function to be minimized. For example, the cost function could be the expectation value of some given observable (VQE, QAOA, etc.) or, as typical in quantum machine learning, some function which quantifies the distance between an ideal model and its variational approximation. To minimize the cost function one can use different iterative methods which often require estimation of derivatives of expectation values with respect to the variational parameters of the circuit. The analytic gradient of a quantum circuit can be experimentally evaluated using the so called *parameter-shift rule* [10–13], which allows implementation of all gradient-based optimizers such as gradient descent.

In this work we explore a variety of derivative estimators for quantum circuits, seeking to understand and map out the effectiveness of these different methods. Our first result is a generalization of the analytic *parameter-shift rule* to derivatives of arbitrary orders including, as particular cases, the Hessian and the Fubini-Study metric tensor [14, 15]. Moreover, we analyze the analytic and finite-difference methods of estimating derivatives from a statistical perspective, taking into account the uncertainty which is unavoidable in any quantum computation involving a finite number of measurement shots.

As direct applications of our findings, we propose and test several second-order optimization methods which can be conveniently implemented using high-order parameter-shift rules. Specifically we focus on the Newton optimizer and the quantum natural gradient optimizer. We also propose a diagonal approximation of the Newton optimizer which has a negligible overhead compared to gradient descent but, nonetheless, is sensitive to the curvature of the cost function with respect to individual parameters.

Our generalization of the parameter-shift rule is complementary to other approaches for evaluating derivatives of quantum circuits. In Ref. [16] it was shown how to evaluate the derivatives which are necessary to implement a VQE algorithm in a quantum chemistry scenario. In Ref. [13] a methodology for replacing direct measurements with indirect ones was proposed, including an experimentally feasible method for estimating the metric tensor. Very recently, different generalizations of the parameter-shift rule to arbitrary gates [17, 18] have been proposed which, in principle, could be directly applied to extend the domain of applicability of our results to arbitrary gates. Moreover it is worthwhile to mention that the same 0 and $\pm\pi/2$ shifts which naturally emerge in our analysis were also used in Ref. [19] for a direct minimization of the cost function with respect to single parameters.

This work is structured as follows. In Sec. II we set the notation and define the derivative tensor. In Sec. III we derive the higher-order parameter-shift rules. In Sec. IV the statistical noise associated to analytic and finite-difference estimators is studied. In Sec. V the implementation of second-order optimizers is considered within the context of parameter-shift rules. Section VI focuses on investigating our findings in simulation and on hardware. We first give practical examples of gradient and Hessian estimation and then we use these quantities to minimize the expectation value of a simple variational circuit[1].

[1] Source code for the numerics and experiments in this paper is available at `https://github.com/XanaduAI/derivatives-of-variational-circuits`.

II. SETTING

Most algorithms designed for near-term quantum computers are based on the variational optimization of quantum circuits [1–6]. A variational quantum circuit acting on n qubits is a sequence of gates depending on a set of classical parameters $\boldsymbol{\theta} = (\theta_1, \theta_2, \dots, \theta_m)$ and producing a family of unitary operations $U(\boldsymbol{\theta})$. Typically, the circuit is applied to a fixed reference state $|0\rangle$ and the expectation value of a Hermitian observable M is measured:

$$f(\boldsymbol{\theta}) = \langle 0|U(\boldsymbol{\theta})^\dagger M U(\boldsymbol{\theta})|0\rangle. \tag{1}$$

The expectation value given in Eq. (1) (or some cost function which depends on it) is usually optimized with respect to the parameters $\boldsymbol{\theta}$. For example, if M is the Hamiltonian of a system, we can approximate the ground state energy by minimizing $f(\boldsymbol{\theta})$. Many optimization methods require to estimate the gradient at each iteration. For more advanced optimizers (e.g., Newton's method), one also needs the Hessian matrix or higher-order derivatives.

In this work we are interested in evaluating derivatives of an arbitrary order d, which we can cast as a tensor with d indices $j_1, j_2, \dots, j_d$ whose elements are the real quantities

$$g_{j_1,j_2,\dots,j_d}(\boldsymbol{\theta}) = \frac{\partial^d f(\boldsymbol{\theta})}{\partial\theta_{j_1}\partial\theta_{j_2}\dots\partial\theta_{j_d}}, \tag{2}$$

where the gradient and the Hessian correspond to the particular cases with $d = 1$ and $d = 2$, respectively.

A typical structure of many variational circuits which can be executed by near-term quantum computers is the following:

$$U(\boldsymbol{\theta}) = V_m U_m(\theta_m) \dots V_2 U_2(\theta)\, V_1 U_1(\theta_1), \tag{3}$$

where V_j are constant arbitrary circuits, while $U_j(\theta_j)$ are "rotation-like" gates, i.e., characterized by a generator H_j such that $H_j^2 = \mathbb{1}$ (involutory matrix) and so:

$$U_j(\theta_j) = e^{-\frac{i}{2}H_j\theta_j} = \cos(\theta_j/2)\mathbb{1} - i\sin(\theta_j/2)H_j. \tag{4}$$

For example, all single-qubit rotations belong to this class. More generally, H_j can be any multi-qubit tensor product of Pauli matrices.

Note that this class can be extended to general gates. A possible method is to decompose a gate into a product of rotation-like gates [17]. Another approach is to decompose an arbitrary generator H_j into a linear combination of Pauli operators and then evaluating derivatives with respect to each term using the stochastic method recently proposed in Ref. [18].

III. PARAMETER-SHIFT RULES

The class of variational quantum circuits specified by Eqs. (3) and (4) are commonly used and there exists a simple parameter-shift rule to evaluate their gradients [10–13]. This rule provides the gradient analytically by evaluating the circuit with fixed shifts of $\pi/2$ in the parameters $\boldsymbol{\theta}$.

We begin this section by showing how the established gradient rule can be generalized to arbitrary parameter shifts and then extend our findings to higher order derivatives such as the Hessian and Fubini-Study metric tensor.

A. First-order derivatives: the gradient

From the previous identity, the unitary conjugation of an arbitrary operator $\hat{K}$ by $U_j(\theta_j)$ can always be reduced to the sum of three terms:

$$\hat{K}(\theta_j) = U_j(\theta_j)^\dagger \hat{K} U_j(\theta_j) = \hat{A} + \hat{B}\cos(\theta_j) + \hat{C}\sin(\theta_j), \tag{5}$$

where $\hat{A}, \hat{B}, \hat{C}$ are operators independent of θ_j and involving only $\hat{K}$ and $\hat{H}_j$. Moreover, from the standard trigonometric addition and subtraction identities, we can deduce that:

$$\frac{d\cos(x)}{dx} = \frac{\cos(x+s) - \cos(x-s)}{2\sin(s)}, \tag{6}$$

$$\frac{d\sin(x)}{dx} = \frac{\sin(x+s) - \sin(x-s)}{2\sin(s)}, \tag{7}$$

which are valid for any $s \neq k\pi,\ k \in \mathbb{Z}$. Differentiating both sides of Eq. (5) and using Eqs. (6) and (7), we obtain a parameter-shift rule which is valid at the operator level:

$$\frac{\partial}{\partial\theta_j}\hat{K}(\theta_j) = \frac{\hat{K}(\theta_j + s) - \hat{K}(\theta_j - s)}{2\sin(s)}. \tag{8}$$

Note that, even if the previous expression looks like a finite-difference approximation, it is actually exact and we can use it to analytically estimate the gradient of expectation values whenever the rotation-like property of Eq. (4) holds. Indeed the operator identity in Eq. (8) can be directly applied into Eq. (1) to evaluate the j_{th} component of the gradient. The result is a family of *parameter-shift rules*:

$$g_j(\boldsymbol{\theta}) = \frac{f(\boldsymbol{\theta} + s\,\mathbf{e}_j) - f(\boldsymbol{\theta} - s\,\mathbf{e}_j)}{2\sin(s)} \tag{9}$$

where $\mathbf{e}_j$ is the unit vector along the θ_j axis.

Note that, in the limit $s \to 0$, $\sin(s)$ can be approximated by s and we recover the central-difference approximation for the first derivative. On the other hand, for

$s = \pi/2$ we obtain the parameter-shift rule already studied in [10–13] and used in quantum software libraries [20–24]. It is important to remark that the formula in Eq. (9) is exact for any choice of s. Strictly speaking s cannot be a multiple of π because of the diverging denominator, however, all the corresponding discontinuities are removable.

B. Second-order derivatives: the Hessian

A useful property of Eq. (9) is that it can be iterated to get higher-order derivatives. Applying the same rule twice we get a similar formula for the Hessian:

$$\begin{aligned} g_{j_1,j_2}(\boldsymbol{\theta}) =&[f(\boldsymbol{\theta} + s_1\,\mathbf{e}_{j_1} + s_2\,\mathbf{e}_{j_2}) - f(\boldsymbol{\theta} - s_1\,\mathbf{e}_{j_1} + s_2\,\mathbf{e}_{j_2}) \\ &- f(\boldsymbol{\theta} + s_1\,\mathbf{e}_{j_1} - s_2\,\mathbf{e}_{j_2}) + f(\boldsymbol{\theta} - s_1\,\mathbf{e}_{j_1} - s_2\,\mathbf{e}_{j_2})] \\ &/[4\sin(s_1)\sin(s_2)] \end{aligned} \tag{10}$$

which, for $s1 = s2 = s$, simplifies to:

$$\begin{aligned} g_{j_1,j_2}(\boldsymbol{\theta}) =&[f(\boldsymbol{\theta} + s(\,\mathbf{e}_{j_1} +\,\mathbf{e}_{j_2})) - f(\boldsymbol{\theta} + s(-\mathbf{e}_{j_1} +\,\mathbf{e}_{j_2})) \\ &- f(\boldsymbol{\theta} + s(\,\mathbf{e}_{j_1} -\,\mathbf{e}_{j_2})) + f(\boldsymbol{\theta} - s(\,\mathbf{e}_{j_1} +\,\mathbf{e}_{j_2}))] \\ &/[2\sin(s)]^2. \end{aligned} \tag{11}$$

Also in this case, for $s \to 0$, we get the standard central-difference formula for the Hessian. For $s = \pi/2$, we get an analytic parameter-shift rule which is similar to the gradient formula used in Refs. [10–13], but extended to the Hessian. A formula equivalent to Eq. (11) for the particular case $s = \pi/2$ was recently used in Ref. [25] and Ref. [16]. Particular attention should be paid to the diagonal of the Hessian since, for each element, two shifts are applied to the same parameter θ_j. In this case, two alternative choices for the value of s which appears in Eq. (11) are particularly relevant. For the choice $s = \pi/2$, we get:

$$g_{j,j}(\boldsymbol{\theta}) =[f(\boldsymbol{\theta} + \pi\,\mathbf{e}_j) - f(\boldsymbol{\theta})]/2, \tag{12}$$

where we used that $f(\boldsymbol{\theta} + \pi\,\mathbf{e}_j) = f(\boldsymbol{\theta} - \pi\,\mathbf{e}_j)$. Instead, for $s = \pi/4$, we obtain:

$$g_{j,j}(\boldsymbol{\theta}) =[f(\boldsymbol{\theta} +\,\mathbf{e}_j\pi/2) - 2f(\boldsymbol{\theta}) + f(\boldsymbol{\theta} -\,\mathbf{e}_j\pi/2)]/2. \tag{13}$$

Each of the two previous formulas has alternative advantages. The advantage of Eq. (12) is that it involves only two expectation values, and so it is more direct with respect to Eq. (13), which is instead a linear combination of three terms. On the other hand, the parameter shifts involved in Eq. (13) are only $\pm\pi/2$. This implies that all the elements of the full Hessian matrix can be evaluated using only the same type of $\pm\pi/2$ shifts and this fact could be an experimentally relevant simplification. Moreover, in the typical scenario in which one has already evaluated the gradient using the m pairs of shifts $f(\boldsymbol{\theta} \pm \pi/2\,\mathbf{e}_j)$, Eq. (13) allows to evaluate the diagonal of the Hessian with the extra cost of just a single expectation value (i.e., $f(\boldsymbol{\theta})$). In Section V we show how this fact can be conveniently exploited to replace the vanilla gradient descent optimizer with a diagonal approximation of the Newton optimizer, with a negligible computational overhead.

C. Fubini-Study metric tensor

A second-order tensor which plays an important role in quantum information theory is the Fubini-Study metric tensor which, for a pure variational state $|\psi(\boldsymbol{\theta})\rangle$, can be expressed as:

$$F_{j_1,j_2}(\boldsymbol{\theta}) = -\left.\frac{\partial^2}{\partial\theta_{j_1}\partial\theta_{j_2}}|\langle\psi(\boldsymbol{\theta}')|\psi(\boldsymbol{\theta})\rangle|^2\right|_{\theta'=\theta}, \tag{14}$$

and corresponds to the real part of the quantum geometric tensor (see e.g., Appendix A1 of [26] for a detailed derivation). For pure states and up to constant factors, Eq. (14) can also be associated to other tensors such as the quantum Fisher information matrix or the Bures metric tensor [15]. Since in this work we only deal with pure states, we often refer to Eq. (14) simply as "the metric tensor".

Differently from the Hessian, the metric tensor is not linked to a particular observable M but it is instead a geometric property of a variational quantum state, which in our setting is simply $|\psi(\boldsymbol{\theta})\rangle = U(\boldsymbol{\theta})|0\rangle$. This tensor plays a crucial role in the implementation of the quantum natural gradient optimizer [26] and in the variational quantum simulation of imaginary-time evolution [27].

Now we can make a useful observation: the metric tensor in Eq. (14) can actually be seen as the Hessian of the expectation value $f(\boldsymbol{\theta})$ defined in equation Eq. (1), for the particular observable $M(\boldsymbol{\theta}') = U(\boldsymbol{\theta}')|0\rangle\langle 0|U(\boldsymbol{\theta}')$. Therefore all the previous theoretical machinery that we have derived for the Hessian applies also to the metric tensor and we get the corresponding parameter-shift rule which is simply the same as Eq. (11) where each expectation value is:

$$f(\boldsymbol{\theta}) = |\langle\psi(\boldsymbol{\theta}')|\psi(\boldsymbol{\theta})\rangle|^2. \tag{15}$$

The quantity $|\langle\psi(\boldsymbol{\theta}')|\psi(\boldsymbol{\theta})\rangle|^2$ is the survival probability of the state $|0\rangle$ after the application of the circuit $U(\boldsymbol{\theta}')U(\boldsymbol{\theta})$. This probability can be easily estimated with near-term quantum computers either with a swap-test, or more simply, as the probability of obtaining the $00\ldots0$ bit string after measuring the state $U(\boldsymbol{\theta}')U(\boldsymbol{\theta})|0\rangle$ in the computational basis.

Replacing Eq. (15) into Eq. (11), setting $s = \pi/2$ and $\theta' = \theta$, we get the explicit parameter-shift rule for the metric tensor:

$$\begin{aligned} F_{j_1,j_2}(\boldsymbol{\theta}) =& \frac{1}{4}\Big[|\langle\psi(\boldsymbol{\theta})|\psi(\boldsymbol{\theta}+(\mathbf{e}_{j_1}+\mathbf{e}_{j_2})\pi/2)\rangle|^2 \\ & - |\langle\psi(\boldsymbol{\theta})|\psi(\boldsymbol{\theta}+(\mathbf{e}_{j_1}-\mathbf{e}_{j_2})\pi/2)\rangle|^2 \\ & - |\langle\psi(\boldsymbol{\theta})|\psi(\boldsymbol{\theta}+(-\mathbf{e}_{j_1}+\mathbf{e}_{j_2})\pi/2)\rangle|^2 \\ & + |\langle\psi(\boldsymbol{\theta})|\psi(\boldsymbol{\theta}-(\mathbf{e}_{j_1}+\mathbf{e}_{j_2})\pi/2)\rangle|^2 \Big]. \end{aligned} \quad (16)$$

As we discussed for the Hessian, also in this case the formula for the diagonal elements can be simplified in two alternative ways. The first formula, corresponding to Eq. (12), is:

$$F_{j,j}(\boldsymbol{\theta}) = \frac{1}{2}\Big[|\langle\psi(\boldsymbol{\theta})|\psi(\boldsymbol{\theta}+\pi\,\mathbf{e}_j)\rangle|^2 - 1\Big]. \quad (17)$$

The second equivalent formula, corresponding to Eq. (13), is:

$$F_{j,j}(\boldsymbol{\theta}) = |\langle\psi(\boldsymbol{\theta})|\psi(\boldsymbol{\theta}+\mathbf{e}_j\pi/2)\rangle|^2 - 1\,, \quad (18)$$

where we used that $|\langle\psi(\boldsymbol{\theta})|\psi(\boldsymbol{\theta}+\mathbf{e}_j\pi/2)\rangle|^2 = |\langle\psi(\boldsymbol{\theta})|\psi(\boldsymbol{\theta}-\mathbf{e}_j\pi/2)\rangle|^2$. We also comment that an efficient method for evaluating diagonal blocks of the metric tensor was proposed in [26]. Moreover, an experimentally feasible methodology for measuring the metric tensor was proposed in [13].

D. Arbitrary-order derivatives

The same iterative approach can be used to evaluate derivatives of arbitrary order. In this case, for simplicity, we set $s = \pi/2$ and we introduce the multi-parameter shift vectors:

$$\mathbf{k}_{\pm j_1,\pm j_2,\ldots,\pm j_d} = \frac{\pi}{2}(\pm\,\mathbf{e}_{j_1} \pm\,\mathbf{e}_{j_2} \cdots \pm\,\mathbf{e}_{j_d}), \quad (19)$$

where $j_1, j_2, \ldots, j_d$ are the same d indices which appear also in the derivative tensor defined in Eq. (2). These vectors represent all the shifts in parameter space that are generated by iterating the parameter-shift rule of Eq. (9) for $j \in \{j_1, j_2, \ldots, j_d\}$. We explicitly introduce the set of all shifts which are related to a derivative of order d:

$$S_{j_1,j_2,\ldots,j_d} = \{\mathbf{k}_{\pm j_1,\pm j_2,\ldots,\pm j_d},\ \forall \text{ choices of signs}\}. \quad (20)$$

For example, for the Hessian formula given in Eq. (11) evaluated as $s = \pi/2$ there are 4 possible shifts: $S_{j_1,j_2} = \{\mathbf{k}_{\pm j_1,\pm j_2}\}$.

With this notation, the derivative tensor of order d defined in Eq. (2), can be expressed in terms of the following generalized parameter-shift rule:

$$g_{j_1,j_2,\ldots,j_d}(\boldsymbol{\theta}) = \frac{1}{2^d} \sum_{\mathbf{k}\in S_{j_1,j_2,\ldots,j_d}} \mathcal{P}(\mathbf{k})\, f(\boldsymbol{\theta}+\mathbf{k}). \quad (21)$$

where $\mathcal{P}(\mathbf{k})$ is the parity of a shift vector, i.e., the parity of negative indices in Eq. (19).

How many expectation values are necessary to evaluate the d-order derivative defined in Eq. (2)? This is given by the number of elements in the previous sum, which is $|S_{j_1,j_2,\ldots,j_d}| = 2^d$. When some of the indices are repeated, the number of shifts can be further reduced but we neglect this fact for the moment. Taking into account that the derivative tensor is symmetric with respect to permutations of the indices, the number of distinct elements of a tensor of order d is given by the combinations of d indices sampled with replacement from the set of m variational parameters, corresponding to the multiset coefficient $\binom{m+d-1}{d}$. Therefore, the total number of expectation values to evaluate a derivative tensor of order d is bounded by:

$$\#\text{ expectation values} \le 2^d \binom{m+d-1}{d} = \mathcal{O}(m^d), \quad (22)$$

where in the last step we assumed $m >> d$. In the opposite regime $d >> m$, each angle parameter θ_j can be subject to multiple shifts. However, because of the periodicity of $f(\boldsymbol{\theta})$, for each θ_j we have at most 4 shifts: $0, \pm\pi/2, \pi$. Moreover, from Eq. (5) it is easy to check that a π shift can be expressed as a linear combination of the others:

$$f(\boldsymbol{\theta} \pm \pi\,\mathbf{e}_j) = f(\boldsymbol{\theta}+\mathbf{e}_j\pi/2) + f(\boldsymbol{\theta}-\mathbf{e}_j\pi/2) - f(\boldsymbol{\theta}), \quad (23)$$

which means that only 3 shifts for each parameter are actually enough: the trivial 0 shift and the two $\pm\pi/2$ shifts.

From this simple counting argument we get another upper bound:

$$\#\text{ expectation values} \le 3^m, \quad (24)$$

which is valid for any d. This implies an interesting fact: from a finite number of 3^m combinations of variational parameters $\boldsymbol{\theta}$ we can Taylor-expand $f(\boldsymbol{\theta})$ up to an arbitrary order and so we can actually evaluate $f(\boldsymbol{\theta})$ for all possible values of $\boldsymbol{\theta}$.

This fact may seem quite surprising, however, it simply reflects the trigonometric structure of rotation-like gates. Indeed from Eq. (5), we see that $\hat{K}(\theta_j)$ is a linear combination of 3 terms. Similarly, a full circuit with m parameters can be expressed as a linear combination of 3^m operators. Therefore, a generic expectation value $f(\boldsymbol{\theta})$ is a linear combination of different trigonometric functions weighted by 3^m scalar coefficients, a fact which was already noticed in Ref. [19]. In principle, these coefficients could be fully determined by evaluating $f(\boldsymbol{\theta})$ at 3^m different values of $\boldsymbol{\theta}$. In this work we focused on the different combinations of the 3 specific shifts of $0, \pm\pi/2$ for each angle θ_j. However, one could also obtain similar results using any other choice of 3 non-trivial shifts for each angle.

This concludes the first part of this work, in which we derived several exact analytical results. However, if we really want to apply Eqs. (9), (11) and (21) in a quantum experiment, we have to take into account that each expectation value on the r.h.s of these equations can be measured only up to a finite precision and so the derivative on the l.h.s. can be estimated only up to a finite error.

IV. STATISTICAL ESTIMATION OF DERIVATIVES

The expectation value in Eq. (1) is a theoretical quantity. In a real quantum computation with N measurement shots, we can only estimate $f(\boldsymbol{\theta})$ up to a finite statistical uncertainty, i.e., what we actually measure is

$$\hat{f}(\boldsymbol{\theta}) = f(\boldsymbol{\theta}) + \hat{\epsilon}, \tag{25}$$

where $\hat{\epsilon}$ is a zero-mean random variable with variance $\sigma^2 = \sigma_0^2/N$, and σ_0^2 is the variance for a single shot which depends on the specific details of the circuit and of the observable. The aim of this section is to take into account the statistical noise which appears in Eq. (25). We consider this problem within the theoretical framework of statistical inference and we study different *derivative estimators* for the quantities defined in Eq. (2). In particular we are interested in two main classes of estimators: one based on a finite-shot version of the analytic parameter-shift rules derived in the previous section and one based on a standard finite-difference approximation.

More precisely the analytic estimator for a derivative tensor of arbitrary order d is:

$$\hat{g}^{(s=\pi/2)}_{j_1,j_2,\ldots,j_d} = \frac{1}{2^d} \sum_{\mathbf{k}\in S_{j_1,j_2,\ldots,j_d}} \mathcal{P}(\mathbf{k})\, \hat{f}(\boldsymbol{\theta}+\mathbf{k}), \tag{26}$$

where we simply placed Eq. (25) into (21).

We can already make a preliminary comparison by considering the central-difference approximation which has a very similar structure but involves an infinitesimal step size h instead of $s=\pi/2$. In order to simplify the comparison we express it using the same notation introduced in Eqs. (19) and (20):

$$\hat{g}^{(h)}_{j_1,j_2,\ldots,j_d} = \frac{1}{(2h)^d} \sum_{\mathbf{k}\in S_{j_1,j_2,\ldots,j_d}} \mathcal{P}(\mathbf{k})\, \hat{f}(\boldsymbol{\theta}+\mathbf{k}\,2h/\pi). \tag{27}$$

It is evident that the only difference introduced in Eq. (27), compared to (26), is that shifts have a step size h and that the full estimator is scaled by h^{-d}. This directly implies that, for $h<1$, the statistical variance of the central-difference estimator is amplified by a factor of h^{-2d}, which is a strong limitation with respect to the analytic estimator. A more detailed analysis of the statistical error characterizing the analytic and the finite-difference methods will be presented later for the particular case of the gradient ($d=1$).

A. Quantifying the error of a statistical estimator

In general terms, our aim is to find a good estimator $\hat{g}_{j_1,\ldots,j_d}(\boldsymbol{\theta})$ which only depends on a finite number of measurement shots and which should approximate the derivative tensor $g_{j_1,\ldots,j_d}(\boldsymbol{\theta})$ defined in Eq. (2) well. As a figure of merit for the performance of an estimator we can use its *mean squared error* (MSE) with respect to the true value which is

$$\Delta(\hat{g}_{j_1,\ldots,j_d}) = \mathbb{E}[(\hat{g}_{j_1,\ldots,j_d} - g_{j_1,\ldots,j_d})^2], \tag{28}$$

where $\mathbb{E}(\cdot)$ is the (classical) average over the statistical distribution of the measurement outcomes. The performance of the estimator can also be measured with respect to the full tensor in terms of the total error,

$$\Delta(\hat{\boldsymbol{g}}) = \mathbb{E}\left[\|\hat{\boldsymbol{g}} - \boldsymbol{g}\|^2\right] = \sum_{j_1,\ldots,j_d} \Delta(\hat{g}_{j_1,\ldots,j_d}), \tag{29}$$

which is simply the sum of the single-element errors.

Before considering specific derivative estimators, it is useful to remind also about the notions of *bias* and *variance*:

$$\mathrm{Bias}(\hat{g}_{j_1,\ldots,j_d}) := \mathbb{E}(\hat{g}_{j_1,\ldots,j_d}) - g_{j_1,\ldots,j_d} \tag{30}$$

$$\mathrm{Var}(\hat{g}_{j_1,\ldots,j_d}) := \mathbb{E}(\hat{g}^2_{j_1,\ldots,j_d}) - \mathbb{E}(\hat{g}_{j_1,\ldots,j_d})^2. \tag{31}$$

These two quantities correspond to different errors: the bias represents a constant error which remains present even in the limit of an infinite number of shots $N\to\infty$, while the variance represents statistical fluctuations which are due to a finite N. It is well known that both effects can increase the MSE, and indeed we have

$$\Delta(\hat{g}_{j_1,\ldots,j_d}) = \mathrm{Var}(\hat{g}_{j_1,\ldots,j_d}) + \mathrm{Bias}(\hat{g}_{j_1,\ldots,j_d})^2. \tag{32}$$

In the next subsections we focus on the estimation of the gradient, but a similar analysis can be extended to the Hessian and higher-order derivatives.

B. Finite-difference gradient estimator

A general way of approximating the gradient is to use a finite-difference estimator, which requires to experimentally measure expectation values for slightly different values of the parameters. The most common finite-difference estimators are the *forward-difference* and the *central-difference*, both well studied in classical numerical analysis [28]. In this work we focus mainly on the central-difference estimator because it has the same structure of to the parameter-shift rules derived in the previous section and so we can easily make a comparison.

Given a fixed step size $h > 0$, the symmetric *finite-difference estimator* for the $j_{\rm th}$ element of the gradient can be defined as:

$$\hat{g}_j^{(h)} = \frac{\hat{f}(\theta_j + h) - \hat{f}(\theta_j - h)}{2h} \\ = \frac{f(\theta_j + h) - f(\theta_j - h)}{2h} + \frac{\hat{\epsilon}_+ - \hat{\epsilon}_-}{2h}, \tag{33}$$

where we used Eq. (25) and $\hat{\epsilon}_\pm$ is the statistical noise associated to $\hat{f}(\theta_j \pm h)$. In the right-hand-side of the previous equation we have the sum of two terms which are the finite-difference approximation and the statistical noise, respectively. Each term introduces a different kind of error: one is linked to finite step size h, the other is linked to the finite number of shots N. Such errors correspond to the bias and to the variance of the estimator discussed in the previous subsection. Indeed, from Eqs. (30) and (31), we have:

$$\mathrm{Bias}(\hat{g}_j^{(h)}) = \frac{f(\theta_j + h) - f(\theta_j - h)}{2h} - g_j \\ = \frac{f_3 h^2}{3!} + O(h^3), \tag{34}$$

$$\mathrm{Var}(\hat{g}_j^{(h)}) = \frac{\sigma_0^2(\theta_j + h) + \sigma_0^2(\theta_j - h)}{4Nh^2} \tag{35}$$

$$\approx \frac{\sigma_0^2}{2Nh^2}, \tag{36}$$

where, in Eq. (34), we used the Taylor-series approximation with $f_3 = \partial^3 f(\theta_j)/\partial\theta_j^3$, and $\sigma_0^2(\theta_j \pm h)$ is the single-shot variance evaluated at $\theta_j \pm h$. The step from Eq. (35) to Eq. (36) is justified only if the following assumption holds.

Assumption 1: The variance of the measured observable depends weakly on the parameter shift, such that $\sigma_0^2(\theta_j + x) + \sigma_0^2(\theta_j - x) \simeq 2\sigma_0^2$ for any value of x.

The previous assumption is usually a quite good approximation, however one can easily construct counter-examples in which this is violated for large values of the shift. For this reason, whenever a subsequent result depends on Assumption 1, it will be explicitly stressed.

It is evident that the terms in Eqs. (34) and (36) have opposite scaling with respect to the step size: for small h the variance diverges, while for large h the bias dominates. This trade-off implies that there must exist an optimal choice of h.

Substituting Eqs. (34) and (36) into Eq. (32) we get, up to $\mathcal{O}(h^6)$ corrections and within the validity of Assumption 1, the MSE for an arbitrary step size:

$$\Delta(\hat{g}_j^{(h)}) \simeq \frac{\sigma_0^2}{2Nh^2} + \frac{f_3^2 h^4}{36}. \tag{37}$$

Imposing the derivative with respect to h to be zero and assuming $f_3 \neq 0$, we get the optimal step size h^* and the optimal error:

$$h^* = \left(\frac{9\sigma_0^2}{f_3^2 N}\right)^{\frac{1}{6}} \propto N^{-\frac{1}{6}} \simeq N^{-0.167}, \tag{38}$$

$$\Delta(\hat{g}_j^{(h^*)}) = \frac{3}{2}\frac{\sigma_0^2}{2N}(h^*)^{-2} = \frac{3}{2}\left[\frac{\sigma_0^2}{2N}\right]^{2/3}\left[\frac{f_3^2}{18}\right]^{1/3} \\ \propto N^{-2/3} \simeq N^{-0.667}, \tag{39}$$

which are valid only for sufficiently large N (i.e., for sufficiently small h^*). When the number of shots is small, the optimal step h^* can become so large that both the Taylor approximation and Assumption 1 may not be valid anymore, so that we could observe deviations from the predictions of Eqs. (37), (38) and (39).

From a practical point of view, it is not straightforward to determine the optimal step h^* since it depends on the third derivative f_3. The situation is significantly simpler for rotation-like gates where, from Eq. (5), we have that the third derivative is equal to the first and so f_3 can be replaced by g_j. However, as we are going to see in the next subsections, in this case it is more convenient to use the parameter-shift estimator.

For the sake of completeness we comment that, by repeating the same analysis for the forward difference estimator $\hat{G}_j^{(h)} = [\hat{f}(\theta_j + h) - \hat{f}(\theta_j)]/h$, one gets similar scaling laws but with different exponents:

$$\Delta(\hat{G}_j^{(h)}) \simeq \frac{\sigma_0^2}{2Nh^2} + \frac{f_2^2 h^2}{4}, \tag{40}$$

$$h^* = \left(\frac{2\sigma_0^2}{f_2^2 N}\right)^{\frac{1}{4}} \propto N^{-0.25}, \tag{41}$$

$$\Delta(\hat{G}_j^{(h^*)}) = \frac{\sigma_0^2}{N}(h^*)^{-2} = \left(\frac{\sigma_0^2 f_2^2}{2N}\right)^{\frac{1}{2}} \propto N^{-0.5}\ , \tag{42}$$

which are valid as long as the approximations in the Taylor truncation and in Assumption 1 are justified.

It is notable that a similar trade-off for the choice of the step-size h was studied also in the field of classical numerical analysis (see e.g., [28] [29]). In a classical computer the error term $\hat{\epsilon}$ in Eqs. (25) and (33) is the *round-off error* (or *machine epsilon*) associated with the finite-precision representation of real numbers. Optimizing the value of h was particularly useful in the early days of classical computing, because of the limited memory and computational resources of that time. Nowadays, with current quantum computers, we are in a similar situation: the statistical uncertainty of quantum measurements and the limited number measurement shots give rise to a *quantum round-off error*, which is usually several orders of magnitude larger than the classical counterpart. For this reason, it is likely that many old problems of classical numerical analysis will become relevant again in the context of quantum computing.

C. Parameter-shift gradient estimators

In Section III we derived, for a large class of variational circuits, an exact formula for the gradient which is given in Eq. (9). For a finite number of shots, we can define the corresponding *parameter-shift estimator*:

$$\hat{g}_j^{(s)} = \frac{\hat{f}(\theta_j + s) - \hat{f}(\theta_j - s)}{2\sin(s)} = g_j + \frac{\hat{\epsilon}_+ - \hat{\epsilon}_-}{2\sin(s)}, \tag{43}$$

where $\hat{\epsilon}_\pm$ is the statistical noise associated to the measurement of $\hat{f}(\theta_j \pm s)$. Differently from the finite-difference estimator $\hat{g}_j^{(h)}$ presented in Eq. (33), $\hat{g}_j^{(s)}$ is unbiased because in this case there is no finite-step error since Eq. (9) is exact. Specifically, we have:

$$\mathrm{Bias}(\hat{g}_j^{(s)}) = 0, \tag{44}$$

$$\mathrm{Var}(\hat{g}_j^{(s)}) = \frac{\sigma_0^2(\theta_j + s) + \sigma_0^2(\theta_j - s)}{4N\sin(s)^2} \tag{45}$$

$$\approx \frac{\sigma_0^2}{2N\sin(s)^2}. \tag{46}$$

The step from Eq. (45) to Eq. (46) is justified only if Assumption 1 is valid.

The MSE of the partial-shift estimator is only due to the statistical noise and, if Assumption 1 applies, this is approximated by:

$$\Delta(\hat{g}_j^{(s)}) = \mathrm{Var}(\hat{g}_j^{(s)}) \approx \frac{\sigma_0^2}{2N\sin(s)^2}, \tag{47}$$

1. *The parameter-shift rule with maximum shift ($s = \pi/2$)*

The simple expression for the MSE (Eq. (47)) implies that, under the validity of Assumption 1, the optimal shift s^* is the one which maximizes the denominator of Eq. (47), i.e., $s^* = \pi/2$. This corresponds to the parameter-shift rule already studied in the literature [10–13].

The explicit definition of the estimator and its MSE are:

$$\hat{g}_j^{(s=\pi/2)} = \frac{\hat{f}(\theta_j + \pi/2) - \hat{f}(\theta_j - \pi/2)}{2}$$

$$= g_j + \frac{\hat{\epsilon}_+ - \hat{\epsilon}_-}{2}. \tag{48}$$

$$\Delta(\hat{g}_j^{(s=\pi/2)}) = \mathrm{Var}(\hat{g}_j^{(s=\pi/2)})$$

$$= \frac{\sigma_0^2(\theta_j + \pi/2) + \sigma_0^2(\theta_j - \pi/2)}{2N} \tag{49}$$

$$\simeq \frac{\sigma_0^2}{2N}. \tag{50}$$

2. *Scaled parameter-shift gradient estimator*

A simple way of further reducing the statistical error below the value of Eq. (45) is to define a *scaled parameter-shift estimator*, which is the same as Eq. (43) but scaled by a multiplicative constant $\lambda \in [0, 1]$:

$$\hat{g}_j^{(\lambda,\, s)} = \lambda \hat{g}_j^{(s)} = \lambda g_j + \lambda \frac{\hat{\epsilon}_+ - \hat{\epsilon}_-}{2\sin(s)}. \tag{51}$$

The effect of the scaling is to reduce the variance by a factor of λ^2. However it has a cost: the new estimator is not unbiased anymore. Indeed, we have:

$$\mathrm{Bias}(\hat{g}_j^{(\lambda,\, s)}) = (\lambda - 1) g_j, \tag{52}$$

$$\mathrm{Var}(\hat{g}_j^{(\lambda,\, s)}) = \lambda^2 \mathrm{Var}(\hat{g}_j^{(s)}). \tag{53}$$

and so, from Eq. (32), its MSE is:

$$\Delta(\hat{g}_j^{(\lambda,\, s)}) = \lambda^2 \mathrm{Var}(\hat{g}_j^{(s)}) + (\lambda - 1)^2 g_j^2. \tag{54}$$

The error is quadratic with respect to λ and is minimized by

$$\lambda^* = \frac{1}{1 + \frac{\mathrm{Var}(\hat{g}_j^{(s)})}{g_j^2}}, \tag{55}$$

corresponding to the MSE

$$\Delta(\hat{g}_j^{(\lambda^*,\, s)}) = \lambda^* \Delta(\hat{g}_j^{(\lambda=1,\, s)}) = \lambda^* \mathrm{Var}(\hat{g}_j^{(s)}), \tag{56}$$

or, equivalently, to

$$\Delta(\hat{g}_j^{(\lambda^*,\, s)}) = (1 - \lambda^*) g_j^2. \tag{57}$$

Eq. (56) shows that the scaled estimator is always more accurate than the simple parameter-shift estimator of Eq. (48). The last equation (57) is also interesting since it implies that the relative MSE of $\hat{g}_j^{(\lambda^*,\, s)}$ is always less than 1, for any amount of statistical noise and so for any N.

We envisage that this estimator could potentially be helpful when faced with the so-called barren plateau phenomenon [30], according to which the typical magnitude of the gradient of a random circuit decays exponentially with respect to the number of qubits. When the gradient is much smaller than the statistical estimation error ($\mathrm{Var}(\hat{g}_j^{(s)}) \gg g_j$), the optimal scaling factor λ^* can be much smaller than the vanilla value of 1. So, in this kind of noise-dominated regime, the scaled estimator can be particularly advantageous if we wish to reduce the mean squared error. Note also that from a practical point of view, since λ^* is a constant scalar, it can be simply absorbed into a re-normalization of the learning rate in most machine learning optimizers. This observation could shed some light on the practical applicability of machine learning algorithms even when the gradient is so small that its estimation is dominated by statistical noise.

D. Comparison between analytic and finite-difference gradient estimators

Even if some generalization approaches have been proposed [17, 18], it is important to stress that all the parameter-shift methods can be directly applied only to a class of circuits (those involving involutory generators), while the operating regime of the finite-difference method is more general. However, in all those cases in which both approaches could be applied, which is the optimal one?

To give an answer to this question, it is important to look for a fair comparison and to choose a reasonable figure of merit. Since the parameter-shift rule involves two symmetric shifts, it is reasonable to compare it to the symmetric-difference estimator and not to the forward-difference one. As figure of merit, we chose the MSE between the estimated value and the true value of the gradient, i.e., Eqs. (28) and (29).

As a first step, we are going to make a comparison between $\hat{g}_j^{(h)}$ and $\hat{g}_j^{(s)}$. Eventually, we will take into consideration also the scaled parameter-shift estimator $\hat{g}_j^{(\lambda,\, s)}$ defined in Eq.(51), which will be shown to outperform the other estimators, assuming the optimal weighting parameter is known.

1. *Small step limit ($h \to 0$)*

In the small step limit $h \to 0$, we notice that the finite-difference estimator (Eq. (33)) is approximately equal to the parameter-shift estimator (Eq. (43)) with $s = h$. Indeed, in this limit, $\sin(h) \simeq h$ and so

$$\hat{g}_j^{(h)} \xrightarrow{h\to 0} \hat{g}_j^{(s=h)}. \tag{58}$$

We have already shown that, if the noise is independent of the variational parameter, the MSE for the parameter-shift estimator is minimized when $s = \pi/2$ (see Eq. (50)). So, in the small h limit, we have

$$\Delta(\hat{g}_j^{(h)}) \xrightarrow{h\to 0} \Delta(\hat{g}_j^{(s=h)}) > \Delta(\hat{g}_j^{(s=\pi/2)}), \tag{59}$$

where the last inequality is valid only under the validity of Assumption 1. In this limit, for $s = \pi/2$ we get a large improvement of the error since we have $\Delta(\hat{g}_j^{(h)})/\Delta(\hat{g}_j^{(s=\pi/2)}) \propto 1/h^2$.

2. *Finite step case with $h \le 1$*

For a non-infinitesimal $h \le 1$, we have a similar result. Indeed, for all $h \le 1$, there always exists an s_h such that $\sin(s_h) = h$. And so:

$$\begin{aligned}\Delta(\hat{g}_j^{(h)}) &= \Delta(\hat{g}_j^{(s=s_h)}) + \mathrm{Bias}(\hat{g}_j^{(h)})^2 \\ &\ge \Delta(\hat{g}_j^{(s=s_h)}),\end{aligned} \tag{60}$$

where in the first equality we used Eq. (32), the fact that $\hat{g}_j^{(h)}$ and $\hat{g}_j^{(s_h)}$ have equal variance, and that $\hat{g}_j^{(s_h)}$ is unbiased.

So, also in this case, the parameter-shift rule has a smaller error with respect to the finite-difference method. Furthermore, if Assumption 1 is valid, the estimator $g_j^{(s=\pi/2)}$ becomes optimal since $\Delta(\hat{g}_j^{(s=s_h)}) > \Delta(\hat{g}_j^{(s=\pi/2)})$.

3. *Finite step case with $h > 1$*

If $h > 1$, the problem is non-trivial. In this case, the variance of the finite-difference estimator is smaller than the variance of the parameter-shift estimator for all values of s, i.e., $\mathrm{Var}(\hat{g}_j^{(h)}) \le \mathrm{Var}(\hat{g}_j^{(s)})$. On the other hand, since h is large, the bias of $\hat{g}_j^{(h)}$ can be large while $\hat{g}_j^{(s)}$ is unbiased. The trade-off between the two effects determines which method minimizes the MSE. Moreover we should also consider that that Taylor approximation which we used for the error analysis of the finite-difference estimator is likely to be broken for $h > 1$.

It is however intuitive that when the statistical noise is extremely large, the finite-difference formula in Eq. (33) for $h > 1$ has a large denominator $2h$ which is able to attenuate the statistical fluctuations more than the parameter-shift denominator (which is equal to 2). So we may ask if, by simply adding a multiplicative constant $\lambda \in [0, 1]$ to the parameter-shift estimator we could always obtain a smaller error with respect to the finite-difference method for all values of h and for any value of the statistical noise. This is indeed true, as we are going to show in the following subsection.

4. *The scaled parameter-shift estimator is always optimal*

We have just compared the error scaling of the analytic and the finite-difference estimators. In this subsection, we instead compare both methods with respect to the *scaled parameter-shift estimator* defined in Eq. (51). In particular we are going to show that $\hat{g}_j^{(\lambda^*,\, s)}$, where λ^* is the optimal scaling parameter given in Eq. (55), always has a smaller MSE when compared to $\hat{g}_j^{(h)}$ or $\hat{g}_j^{(s)}$.

Indeed, from Eq. (56) and since $\lambda^* \le 1$, we directly have:

$$\Delta(\hat{g}_j^{(s)}) \ge \lambda^* \Delta(\hat{g}_j^{(s)}) = \Delta(\hat{g}_j^{(\lambda^*,\, s)}). \tag{61}$$

Moreover, after recognizing that $\hat{g}_j^{(h)} = \lambda_h \hat{g}_j^{(s=h)}$ with $\lambda_h = \sin(h)/h < 1$, we have:

$$\Delta(\hat{g}_j^{(h)}) = \Delta(\lambda_h \hat{g}_j^{(s=h)}) = \Delta(\hat{g}_j^{(\lambda=\lambda_h,\, s)}) \ge \Delta(\hat{g}_j^{(\lambda^*,\, s)}). \tag{62}$$

Therefore, the scaled parameter-shift estimator $\hat{g}_j^{(\lambda^*, s)}$ is characterized by a smaller mean squared error when compared to both the finite-difference estimator $\hat{g}_j^{(h)}$ and the (non-scaled) parameter-shift estimator $\hat{g}_j^{(s)}$.

We stress that this result can be straightforwardly generalized also to high-order derivative estimators beyond the gradient, by simply replacing the constant $\lambda_h = \sin(h)/h$ which appears in Eq. (62) with $\lambda_h = (\sin(h)/h)^d$, where d is the order of the derivative tensor. Moreover, we also highlight that this result is independent on the validity of Assumption 1. However, if Assumption 1 holds, we can deduce the additional fact that the scaled parameter-shift estimator with $s = \pi/2$ is always the optimal choice since it minimizes the mean squared error.

V. FIRST- AND SECOND-ORDER OPTIMIZATION

In many variational algorithms and in quantum machine learning problems, a loss function (which may depend nonlinearly on one or more expectation values) is minimized with respect to the circuit parameters $\boldsymbol{\theta}$. For simplicity, in this section we assume that our goal is to minimize a single expectation value $f(\boldsymbol{\theta})$, associated to an observable M as described in Eq. (1). More general scenarios can be treated in a similar way, with only some classical overhead due to the application of the chain rule for evaluating derivatives of composite functions.

A. Gradient descent optimizer

One of the most common optimizers is *gradient descent* (GD). According to this algorithm, the parameters are first initialized with an initial (usually random) guess $\boldsymbol{\theta}^{(0)}$ and then, a sequence of updated configurations $\boldsymbol{\theta}^{(1)}, \boldsymbol{\theta}^{(2)}, \dots, \boldsymbol{\theta}^{(T)}$ is recursively generated, hopefully converging to a minimum of $f(\boldsymbol{\theta})$. At each iteration, the GD update rule is:

$$\boldsymbol{\theta}^{(t)} = \boldsymbol{\theta}^{(t-1)} - \eta \nabla f(\boldsymbol{\theta}^{(t-1)}), \tag{63}$$

where $\eta > 0$ is a real hyper-parameter usually called the *learning rate*. Note that this is essentially the same algorithm in both the classical and quantum settings, so the only part of the problem which is intrinsically “quantum” is the estimation of the gradient. Indeed, GD is the default optimizer used in most quantum machine learning software frameworks [20–24]. In the previous sections of this work, we have studied different methods of estimating the quantum gradient, including the parameter-shift rule [10–13].

B. Newton optimizer

The generalization of the parameter-shift rule to higher-order derivatives that we introduced in Eq. (21) and, in particular the analytic evaluation of the Hessian and the Fubini-study metric tensor that we have discussed in the first part of this work, can be directly exploited to implement several second-order optimizers. Such methods are similar to GD but in place of the first-order update rule of Eq. (63) they have more advanced iteration rules, which take into account also some information about the curvature of the objective function (or of the quantum state) with respect to the variational parameters $\boldsymbol{\theta}$.

For example, the update rule of the Newton optimizer (see, e.g., [28]) is:

$$\boldsymbol{\theta}^{(t)} = \boldsymbol{\theta}^{(t-1)} - \eta [\text{Hess} f(\boldsymbol{\theta}^{(t-1)})]^{-1} \nabla f(\boldsymbol{\theta}^{(t-1)}), \tag{64}$$

where $[\text{Hess} f(\boldsymbol{\theta})]^{-1}$ is the inverse of the Hessian matrix, which can be estimated using the associated parameter-shift rule presented in Eq. (11). Since the Hessian is an $m \times m$ matrix, at each iteration, the number of expectation values which need to be estimated scales as m^2.

C. Diagonal Newton optimizer

The computational cost of this method can be significantly reduced if one approximates the full Hessian matrix with its diagonal part, such that the update rule for the j_{th} element of $\boldsymbol{\theta}^{(t)}$, simplifies to:

$$\boldsymbol{\theta}_j^{(t)} = \boldsymbol{\theta}_j^{(t-1)} - \eta g_{j,j}^{-1}(\boldsymbol{\theta}^{(t-1)}) g_j(\boldsymbol{\theta}^{(t-1)}), \tag{65}$$

where we used notation of the derivative tensor of Eq. (2).

The computational cost of this optimizer, which was originally introduced in classical machine learning [31], has a linear scaling with respect to m and so it can be a competitive alternative to GD. Moreover, the computational overhead of this method with respect to the analytic GD optimizer is negligible. Indeed, according to Eq. (13), the diagonal of the Hessian can be evaluated with the same $\pm\pi/2$ shifts which would be measured anyway to estimate for the gradient.

D. Quantum natural gradient optimizer

A different second-order optimizer can be obtained from the concept of quantum natural gradient [26]. The corresponding update rule is very similar to Newton's method, but here the Hessian is replaced by the Fubini-study metric tensor defined in Eq. (14):

$$\boldsymbol{\theta}^{(t)} = \boldsymbol{\theta}^{(t-1)} - \eta [F(\boldsymbol{\theta}^{(t-1)})]^{-1} \nabla f(\boldsymbol{\theta}^{(t-1)}). \tag{66}$$

As we have shown in Section III C, all the elements of metric tensor $F_{i,j}$ can be analytically estimated with the parameter-shift rule derived in Eq. (16).

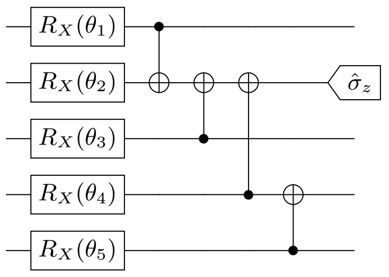


FIG. 1. The variational circuit run on simulator and hardware to investigate the findings in this paper.

E. Regularization of second-order methods

All the previous second-order methods share the same structure of the update rule $\boldsymbol{\theta}^{(t)} = \boldsymbol{\theta}^{(t-1)} - \eta A^{-1}\nabla f(\boldsymbol{\theta}^{(t-1)})$, where A can be the Hessian, the metric tensor or some diagonal approximation. In the field of classical optimization, it is well known that, if A is not positive-definite, the optimizer can converge to a maximum instead of a minimum [28]. Moreover, if one or more eigenvalues are close to zero, the norm of the inverse matrix can be so large that the algorithm becomes unstable. Therefore it is a common practice to regularize A in such a way to make it sufficiently positive. For example one can replace A with $A' = A + \epsilon\mathbb{1}$, for some positive parameter $\epsilon > 0$, which corresponds to shifting all the eigenvalues by $+\epsilon$. Alternatively, one can regularize each eigenvalue λ_k of A by replacing it with $\lambda'_k = \max(\lambda_k, \epsilon)$, without changing the associated eigenvector. In our experiments we used the previous method, however, from additional numerical simulations (see Appendix A, Sec. A 5) we noticed that the second method can be much more efficient since it only perturbs A in the necessary subspace and keeps A invariant whenever it is sufficiently positive.

Finally we mention the approach which was recently theoretically proposed in Ref. [25], where the inverse of the maximum eigenvalue of the Hessian is used as a learning rate. In the notation of our work, this approach could be interpreted as a particular regularization method in which A is replaced by $A' = \max(\lambda_k)\mathbb{1}$.

VI. NUMERICAL AND HARDWARE EXPERIMENTS

This section contains details of experiments run on simulators and hardware. We use the PennyLane software library [20] to construct a simple variational quantum circuit and access its expectation value and gradient. The circuit is evaluated using PennyLane's built-in simulator as well as on hardware by using integration with the IBM Quantum Experience to execute on the `ibmq_valencia` and `ibmq_burlington` five-qubit chips.

The circuit has five parameters $\boldsymbol{\theta} = (\theta_1, \theta_2, \theta_3, \theta_4, \theta_5)$, as shown in Fig. 1. It consists of a collection of single qubit Pauli-X rotations, an entangling block, and a measurement of the Pauli-Z observable on the second qubit. The circuit is chosen to provide a non-trivial gradient with a low depth and the entangling block is set to match the topology of the hardware devices. We use a fixed set of parameters $\boldsymbol{\theta}$, which is detailed in Appendix A along with the resultant analytic expectation value and gradient.

A. Estimating the gradient

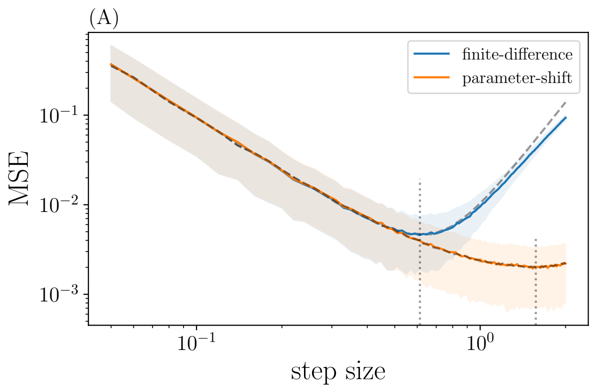
We now investigate the performance of gradient estimators in the finite-shot setting, as discussed in Sec. IV. Figure 2 (A) shows how the MSE $\Delta(\hat{\boldsymbol{g}})$ scales with step size for the finite-difference and parameter-shift estimators when expectation values are estimated on a simulator with 10^3 shots. The MSE is calculated over 10^3 repetitions and seen to coincide closely with the theoretical predictions provided earlier. Note that the finite-difference curve starts to diverge from the predicted behavior for large step sizes due to the Taylor-series approximation breaking down in Eq. (34).

It is hence clear from the figure that the parameter-shift method with $s = \pi/2$ is the best performing estimator in this setting. We investigate the relative performance of the two gradient estimators further using simulations in Appendix A, where we see that the parameter-shift method has the lower MSE for $N > 50$ shots. For low shot numbers, the scaled parameter-shift method mentioned in Sec. IV C 2 can also be used.

Figure 2 (B) compares the gradient estimators using the `ibmq_valencia` device with 10^3 shots per expectation value. The MSE is calculated over 16 repetitions due to limited availability of the hardware. We observe a similar qualitative behavior to the simulator-based results in Fig. 2 (A) but with the MSE offset upward by roughly one order of magnitude, likely due to systematic errors in the device.

B. Estimating the Hessian

We use the results presented in Sec. III B to measure the Hessian matrix using the parameter-shift method. The Hessian of the circuit in Fig. 1 is a (5×5)-dimensional symmetric matrix with 10 off-diagonal terms and 5 along the diagonal. The parameter-shift method with shift $s = \pi/2$ was used to estimate the off-diagonal terms, requiring measurement of 40 expectation values. The diagonal terms were estimated with $s = \pi/4$ which required measuring 11 expectation values. The exact value of the Hessian matrix is provided in Appendix A, which also includes a simulator-based comparison of the MSE for the Hessian when estimated using the finite-difference and parameter-shift methods. The exact Hessian matrix consists of a 4×4 block of non-zero terms and a final row and column of zeros due to the expectation value being



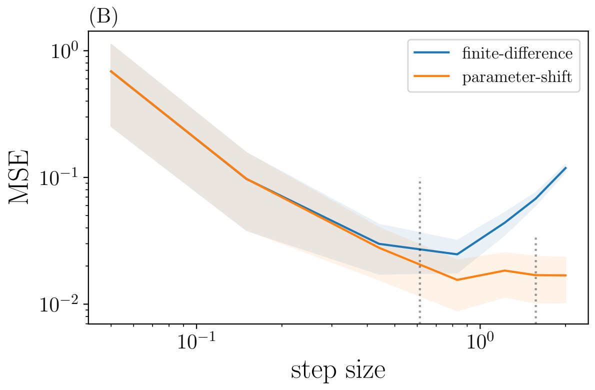
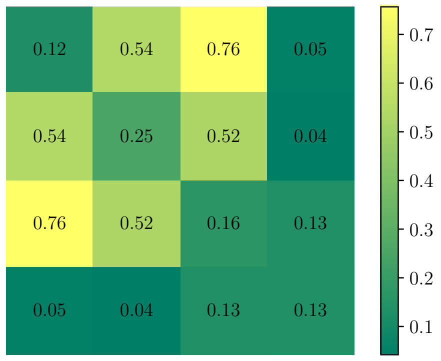


FIG. 2. Mean squared error (MSE) for different choices of step size when estimating the gradient using the finite-difference and parameter-shift estimators with 10^3 shots used to evaluate expectation values. The solid lines show the MSE while the shaded regions illustrate the range of values within one standard deviation. Plot (A) compares the estimators when run on a simulator, while Plot (B) compares the estimators when run on `ibmq_valencia`. Additional gray lines are calculated from theoretical predictions: the dashed lines show the expected behavior of the MSE and the dotted vertical lines show the expected optimal choice of step size: $h^* = 0.613$ and $s^* = \pi/2$.



0.12	0.54	0.76	0.05
0.54	0.25	0.52	0.04
0.76	0.52	0.16	0.13
0.05	0.04	0.13	0.13

FIG. 3. Relative error for using the parameter-shift estimator to approximate the Hessian matrix on the `ibmq_valencia` hardware device.

independent of θ_5. We estimated the Hessian matrix on hardware using `ibmq_valencia` with $N = 10^3$. Figure 3 shows the relative error for the 4×4 block, while the maximum squared error for second derivatives including the θ_5 term was 8.12×10^{-4}.

C. Optimization

The GD, Newton and diagonal Newton optimizers discussed in Sec. V are now used to minimize the expectation value $f(\boldsymbol{\theta})$ of the circuit in Fig. 1 when run on hardware using the `ibmq_valencia` and `ibmq_burlington` devices. We select θ_1 and θ_2 as trainable parameters and leave θ_3, θ_4, and θ_5 fixed, as well as choosing a constant learning rate of $\eta = 0.4$, as detailed in Appendix A.

A comparison of the performance of the GD optimizer when using different shot numbers N to evaluate expectation values is provided in row A of Fig. 4. Plot (A1) shows the cost function $f(\boldsymbol{\theta})$ evolving with the total number of circuit evaluations on the device, while Plot (A2) illustrates the path taken by the optimizer through the θ_1-θ_2 space. The starting point of the optimizer is $(\theta_1, \theta_2) = (0.1, 0.15)$ and an exact GD optimizer tends to the value $(\theta_1, \theta_2) = (0, \pi)$.

These plots highlight some interesting features of optimization on hardware. We see that the minimum can be approached even with a low shot number of $N = 10$, requiring orders of magnitude fewer circuit evaluations from the device than the $N = 100$ and $N = 1000$ cases. However, the $N = 10$ case is noisy and oscillates around the minimum point. This suggests that an adaptive shot number N may be useful in practice with a low shot number initially and an increase in N as the optimizer approaches a minimum, as has been discussed in recent works [32–34]. It is also important to note that the optimizer is able to approach the expected minimum even though the cost function available on the device is noisy, indicating that the direction of the gradient is still an accessible signal. Nevertheless, the gradient direction is clearly still prone to error: note that in Fig. 4 the hardware-based paths are different from the ideal simulated path.

Row B of Fig. 4 shows the result of using the GD, Newton and diagonal Newton optimizers on hardware with $N = 10^3$ and with the same starting point of $(\theta_1, \theta_2) = (0.1, 0.15)$. The Newton optimizer requires evaluation of a (2×2)-dimensional Hessian at each iteration step which is realised with 9 expectation value measurements, while the diagonal Newton optimizer requires 5 expectation value measurements per iteration step. This contrasts with the GD optimizer which requires 4 evaluations per step. These differences are factored into Plot (B1) of Fig. 4, which plots the cost function $f(\boldsymbol{\theta})$ in terms of the total number of circuit evaluations taken on the device.

We observe a comparable performance between each of the optimizers, but it is important to note that the learn-

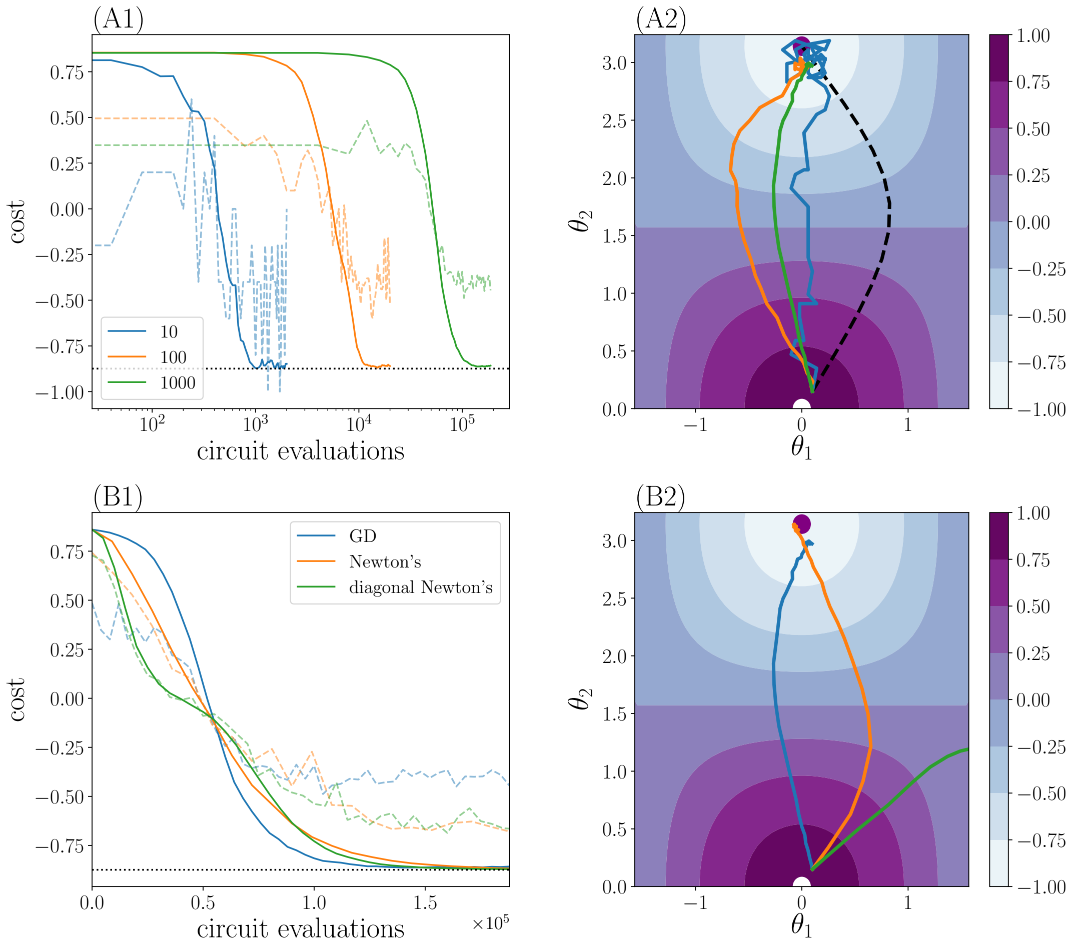


FIG. 4. A comparison of optimizers with circuit evaluation performed on the `ibmq_burlington` and `ibmq_valencia` hardware devices. The first row of plots (experiment A) focuses on the gradient descent (GD) optimizer for a varying number of shots N per expectation value measurement, while the second row of plots (experiment B) compares different optimizers discussed in Sec. V when using $N = 1000$. Plots (A1) and (B1) show the cost function $f(\boldsymbol{\theta})$ in terms of the total number of circuit evaluations on the device and Plots (A2) and (B2) show the path taken by the optimizer through the θ_1-θ_2 space. In (A1) and (B1), the dashed lines give the cost function evaluated on hardware and the solid lines give the cost function evaluated on an exact simulator with the same $\boldsymbol{\theta}$. The dotted line gives the analytic minimum of -0.874. In (A2) and (B2), the contrasting white and purple circles highlight the maximum and minimum, while the dashed line in (A2) is the path taken by the GD optimizer with access to exact expectation values.

ing rate hyperparameter and the regularization method can change the performance of each optimizer. One may also wonder why, for a large number of circuit evaluations, GD seems to be equivalent or even better than second-order methods. The supplementary theoretical analysis reported in Appendix A suggests that this is strongly related to the choice of the regularization method (see in particular Sec. A 5).

The paths taken by each optimizer are drawn in Plot (B2) of Fig. 4. This highlights the key qualitative difference between the optimizers, which take paths based on the different information they have available, such as the local curvature information for the optimizers using the Hessian. Note that the diagonal Newton optimizer tended to the $(\pi, 0)$ minimum when evaluated on hardware. This behaviour was due to the presence of noise in the device: all optimizers tend to the $(0, \pi)$ minimum with access to an ideal simulator, as shown in Appendix A.

The main scope of our experiments was to show the practical feasibility of GD in the regime of high statistical noise and to give a proof-of-principle demonstration of second-order methods on real hardware. A quantitative and detailed benchmarking of the different optimizers is left for a future work. On this subject, we mention the recent theoretical analysis reported in Ref. [35] and Ref. [25].

VII. CONCLUSIONS

We have derived a generalization of the parameter-shift rule for evaluating derivatives of arbitrary order. We have studied how such derivatives are affected by the statistical noise which is due to a finite number of experimental measurements, testing our predictions with numerical simulations and real quantum experiments. We have also theoretically proposed and experimentally tested how the generalized parameter-shift rules can be used to efficiently implement second-order optimization methods on near-term quantum computers.

The results presented in this work could pave the way to interesting future research directions. For example, the strong similarity between the classical *round-off* error the quantum statistical error, which naturally emerged in our analysis of derivative estimators, is probably worth being further explored and generalized. Moreover, the analytic second-order optimizers, which in this work we tested with simple examples, could be subject to a more systematic benchmarking analysis in order to understand their competitive potential with respect to first-order methods.

ACKNOWLEDGMENTS

We thank Josh Izaac for his help with numerical and hardware experiments. We also thank Maria Schuld for useful comments. This project was supported under DARPA project HR00112090015, Backpropagating Through Quantum Computers.

[1] Alberto Peruzzo, Jarrod McClean, Peter Shadbolt, Man-Hong Yung, Xiao-Qi Zhou, Peter J Love, Alán Aspuru-Guzik, and Jeremy L O'Brien. A variational eigenvalue solver on a photonic quantum processor. *Nature Communications*, 5:4213, 2014.

[2] Maria Schuld, Alex Bocharov, Krysta M Svore, and Nathan Wiebe. Circuit-centric quantum classifiers. *Physical Review A*, 101(3):032308, 2020.

[3] Alejandro Perdomo-Ortiz, Marcello Benedetti, John Realpe-Gómez, and Rupak Biswas. Opportunities and challenges for quantum-assisted machine learning in near-term quantum computers. *Quantum Science and Technology*, 3(3):030502, 2018.

[4] Jarrod R McClean, Jonathan Romero, Ryan Babbush, and Alán Aspuru-Guzik. The theory of variational hybrid quantum-classical algorithms. *New Journal of Physics*, 18(2):023023, 2016.

[5] Sukin Sim, Peter D Johnson, and Alan Aspuru-Guzik. Expressibility and entangling capability of parameterized quantum circuits for hybrid quantum-classical algorithms. *arXiv preprint arXiv:1905.10876*, 2019.

[6] Nathan Killoran, Thomas R Bromley, Juan Miguel Arrazola, Maria Schuld, Nicolás Quesada, and Seth Lloyd. Continuous-variable quantum neural networks. *Physical Review Research*, 1(3):033063, 2019.

[7] 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.

[8] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. A quantum approximate optimization algorithm. *arXiv preprint arXiv:1411.4028*, 2014.

[9] Marcello Benedetti, Erika Lloyd, Stefan Sack, and Mattia Fiorentini. Parameterized quantum circuits as machine learning models. *Quantum Science and Technology*, 4(4):043001, 2019.

[10] Jun Li, Xiaodong Yang, Xinhua Peng, and Chang-Pu Sun. Hybrid quantum-classical approach to quantum optimal control. *Physical Review Letters*, 118(15):150503, 2017.

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

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

[13] Kosuke Mitarai and Keisuke Fujii. Methodology for replacing indirect measurements with direct measurements. *Physical Review Research*, 1(1):013006, 2019.

[14] Ran Cheng. Quantum geometric tensor (Fubini-Study metric) in simple quantum system: a pedagogical introduction. *arXiv preprint arXiv:1012.1337*, 2010.

[15] Jing Liu, Haidong Yuan, Xiao-Ming Lu, and Xiaoguang Wang. Quantum Fisher information matrix and multiparameter estimation. *Journal of Physics A: Mathematical and Theoretical*, 53(2):023001, 2019.

[16] Kosuke Mitarai, Yuya O Nakagawa, and Wataru Mizukami. Theory of analytical energy derivatives for the variational quantum eigensolver. *Physical Review Research*, 2(1):013129, 2020.

[17] Gavin E Crooks. Gradients of parameterized quantum gates using the parameter-shift rule and gate decomposition. *arXiv preprint arXiv:1905.13311*, 2019.

[18] Leonardo Banchi and Gavin E Crooks. Measuring analytic gradients of general quantum evolution with the stochastic parameter shift rule. *arXiv preprint arXiv:2005.10299*, 2020.

[19] Mateusz Ostaszewski, Edward Grant, and Marcello Benedetti. Quantum circuit structure learning. *arXiv preprint arXiv:1905.09692*, 2019.

[20] Ville Bergholm, Josh Izaac, Maria Schuld, Christian Gogolin, and Nathan Killoran. PennyLane: Automatic differentiation of hybrid quantum-classical computations. *arXiv preprint arXiv:1811.04968*, 2018.

[21] Michael Broughton, Guillaume Verdon, Trevor McCourt, Antonio J Martinez, Jae Hyeon Yoo, Sergei V Isakov, Philip Massey, Murphy Yuezhen Niu, Ramin Halavati, Evan Peters, et al. TensorFlow Quantum: A software framework for quantum machine learning. *arXiv preprint arXiv:2003.02989*, 2020.

[22] Qulacs. `https://github.com/qulacs/qulacs`, 2018.

[23] Tequila. `https://github.com/aspuru-guzik-group/tequila`, 2020.

[24] Xiu-Zhe Luo, Jin-Guo Liu, Pan Zhang, and Lei Wang. Yao.jl: Extensible, efficient framework for quantum algorithm design. *arXiv preprint arXiv:1912.10877*, 2019.

[25] Patrick Huembeli and Alexandre Dauphin. Characterizing the loss landscape of variational quantum circuits. *arXiv preprint arXiv:2008.02785*, 2020.

[26] James Stokes, Josh Izaac, Nathan Killoran, and Giuseppe Carleo. Quantum natural gradient. *Quantum*, 4:269, 2020.

[27] Sam McArdle, Tyson Jones, Suguru Endo, Ying Li, Simon C Benjamin, and Xiao Yuan. Variational ansatz-based quantum simulation of imaginary time evolution. *npj Quantum Information*, 5(1):1–6, 2019.

[28] Philip E Gill, Walter Murray, and Margaret H Wright. *Practical optimization.* SIAM, 2019.

[29] Ravishankar Mathur. *An analytical approach to computing step sizes for finite-difference derivatives.* PhD thesis, The University of Texas at Austin, 2012.

[30] 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):4812, 2018.

[31] Sue Becker, Yann Le Cun, et al. Improving the convergence of back-propagation learning with second order methods. In *Proceedings of the 1988 connectionist models summer school*, pages 29–37, 1988.

[32] Ryan Sweke, Frederik Wilde, Johannes Meyer, Maria Schuld, Paul K Fährmann, Barthélémy Meynard-Piganeau, and Jens Eisert. Stochastic gradient descent for hybrid quantum-classical optimization. *arXiv preprint arXiv:1910.01155*, 2019.

[33] Andrew Arrasmith, Lukasz Cincio, Rolando D Somma, and Patrick J Coles. Operator sampling for shot-frugal optimization in variational algorithms. *arXiv preprint arXiv:2004.06252*, 2020.

[34] Jonas M Kübler, Andrew Arrasmith, Lukasz Cincio, and Patrick J Coles. An adaptive optimizer for measurement-frugal variational algorithms. *Quantum*, 4:263, 2020.

[35] David Wierichs, Christian Gogolin, and Michael Kastoryano. Avoiding local minima in variational quantum eigensolvers with the natural gradient optimizer. *arXiv preprint arXiv:2004.14666*, 2020.

[36] Milton Abramowitz and Irene Stegun. Handbook of mathematical functions. *US Department of Commerce*, 10, 1972.

Appendix A: Numerics and hardware experiments: further details

1. Problem setting

We use a fixed set of parameters generated from the uniform distribution over the interval $[0, 2\pi]$,

$$\boldsymbol{\theta} = (2.739, 0.163, 3.454, 2.735, 2.641) \tag{A1}$$

Using an exact simulator, the expectation value, its gradient, and its Hessian can be evaluated as

$$f(\boldsymbol{\theta}) = -0.794, \tag{A2}$$

$$\boldsymbol{g}(\boldsymbol{\theta}) = (-0.338, 0.130, 0.256, -0.342, 0), \tag{A3}$$

$$H(\boldsymbol{\theta}) = \begin{pmatrix} 0.794 & 0.055 & 0.109 & -0.145 & 0 \\ 0.055 & 0.794 & -0.042 & 0.056 & 0 \\ 0.109 & -0.042 & 0.794 & 0.110 & 0 \\ -0.145 & 0.056 & 0.110 & 0.794 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{A4}$$

Note that the final element of the gradient vector is exactly zero. Simulations were carried out in this paper using the `default.qubit` device in PennyLane with a finite number of shots. Hardware evaluation was performed using the `qiskit.ibmq` device in the PennyLane-Qiskit plugin.

2. Estimating the gradient

Figure 5 shows the optimal step size for both the finite-difference and parameter-shift gradient estimators over a range of shot numbers N used to measure expectation values on simulator. In both cases, the gradient was estimated over a fixed set of candidate choices for the step size and the one with the smallest MSE was selected. The plots also contain dashed lines illustrating the predicted value of the optimal step size. For small N, the optimal step size for the finite-difference estimator begins to deviate from the prediction. The reason for this deviation is that, for small N, the optimal step size h^* is so large that the Taylor expansion around $h \simeq 0$ is not valid anymore. Figure 6 shows the MSE for the gradient estimators when using their optimal step sizes over a range of shot numbers N. In both Figs. 5 and 6, the MSE is calculated using 1000 repetitions.

3. Estimating the Hessian

Figure 7 illustrates the MSE when the Hessian is estimated using the finite-difference and parameter-shift methods for varying step sizes when expectation values are estimated on a simulator with 10^3 shots and with 10^3 repetitions to calculate the average. The parameter-shift estimator is calculated using Eq. 11 and the finite-difference estimator of the Hessian is taken to be [36]

$$\begin{aligned}\hat{g}^{(h)}_{j_1,j_2}(\boldsymbol{\theta}) =& [f(\boldsymbol{\theta} + h(\mathbf{e}_{j_1} + \mathbf{e}_{j_2})) - f(\boldsymbol{\theta} + h(-\mathbf{e}_{j_1} + \mathbf{e}_{j_2})) \\ & - f(\boldsymbol{\theta} + h(\mathbf{e}_{j_1} - \mathbf{e}_{j_2})) + f(\boldsymbol{\theta} - h(\mathbf{e}_{j_1} + \mathbf{e}_{j_2}))] \\ & /4h^2. \end{aligned} \tag{A5}$$

Figure 8 compares both estimators of the Hessian for varying shot numbers N when each estimator uses the corresponding optimal step size.

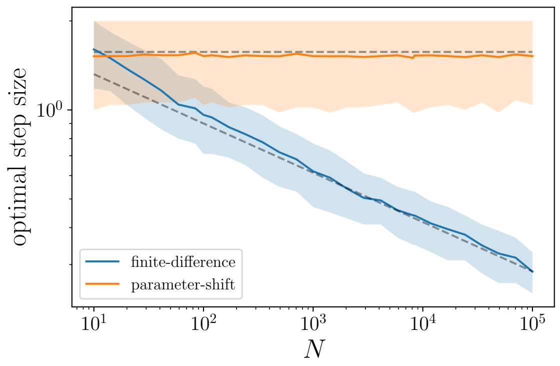


FIG. 5. Optimal step size for the finite-difference and parameter-shift gradient estimators for a range of shot numbers N. The shaded regions display the range of step sizes resulting in an MSE that is within 20% of the minimum observed value, while the solid lines are the medians of these regions. The dashed gray lines are the predicted values for the optimal step size.

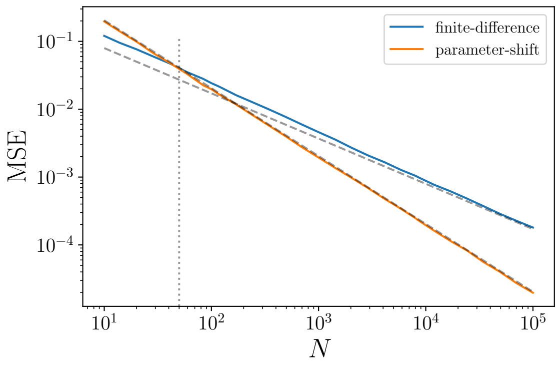


FIG. 6. Mean squared error (MSE) for different choices of shot number N when using the finite-difference and parameter-shift estimators with corresponding optimal choices of step size to estimate the gradient. The solid lines are shown from simulation while the dashed gray lines are the predicted scaling. The dotted vertical line indicates the crossover-point between the two methods at $N \approx 50$ shots.

4. Second-order optimization

We minimize the expectation value $f(\boldsymbol{\theta})$ with $\boldsymbol{\theta} = (\theta_1, \theta_2, 3.454, 2.735, 2.641)$, where θ_1 and θ_2 are trainable parameters. The analytic minimum is equal to

$$\min_{\theta_1,\theta_2} f(\boldsymbol{\theta}) = -0.874. \tag{A6}$$

Extremal points occur alternately when θ_1 and θ_2 are multiples of π.

In the results provided, `ibmq_burlington` was used when investigating the GD optimizer and `ibmq_valencia` was used when investigating the Newton and diagonal Newton optimizers. A learning rate of $\eta = 0.4$ was adopted for all optimizers.

Figure 9 compares the GD, Newton and diagonal Newton optimizers with circuit evaluation on a noise-free simulator.

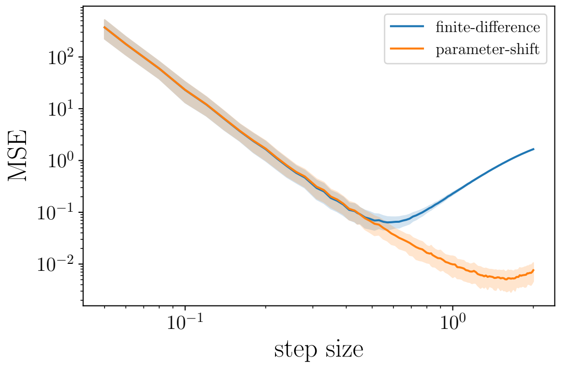


FIG. 7. Mean squared error (MSE) for different choices of step size when estimating the Hessian using the finite-difference and parameter-shift estimators with 10^3 shots used to evaluate expectation values on a simulator. The solid lines show the MSE while the shaded regions illustrate the range of values within one standard deviation.

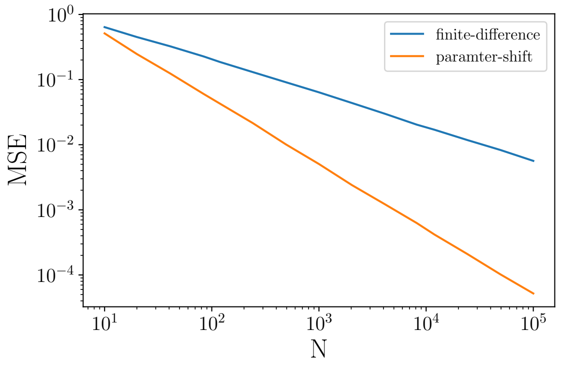


FIG. 8. Mean squared error (MSE) for different choices of shot number N when using the finite-difference and parameter-shift estimators with optimal choices of step size to estimate the Hessian. A power-law fit to each line gives a scaling of $N^{-0.5013}$ and $N^{-1.0008}$ for the finite-difference and parameter-shift methods, respectively.

5. Using a different regularization

As discussed in Sec. V E, one can use different regularization methods. In the previous examples we used $\text{Hess}(f) \to \text{Hess}(f) + \epsilon\mathbb{1}$ with $\epsilon > 0$ which, in the specific cases of Figs. 9 and 4, was set to a relatively large value ($\epsilon = 1$) to compensate for the large negativity of the initial Hessian matrix. In Fig. 10 instead, we numerically simulate the same optimization problem but we use a different regularization method: we replace each eigenvalue λ_k of $\text{Hess}(f)$ with $max(\lambda_k, \epsilon)$, without changing the corresponding eigenvectors. In this case the results demonstrate a clear advantage of second-order optimizers with respect to gradient descent.

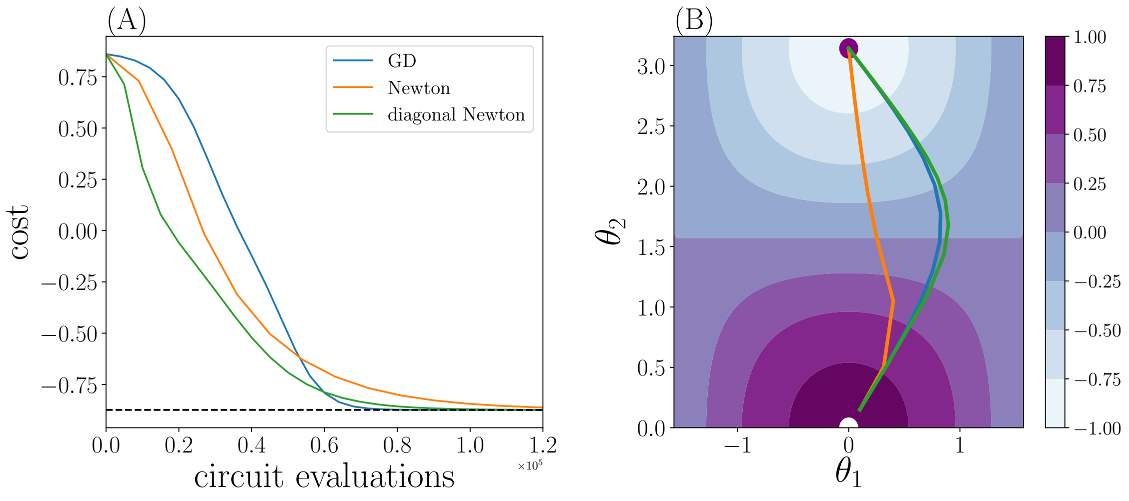


FIG. 9. A comparison of different optimizers discussed in Sec. V with circuit evaluation performed on PennyLane's noise-free `default.qubit` simulator. Plot (A) shows the cost function $f(\boldsymbol{\theta})$ in terms of the total number of circuit evaluations and Plot (B) shows the path taken by the optimizer through the θ_1-θ_2 space. In (A), the dotted line gives the analytic minimum of -0.874 and in (B), the contrasting white and purple circles highlight the maximum and minimum.

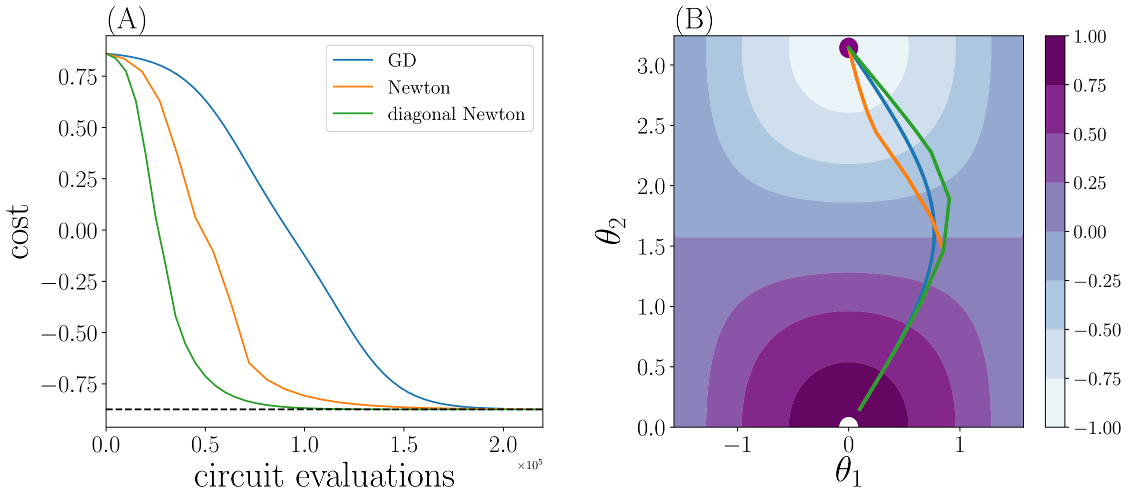


FIG. 10. A comparison of different optimizers discussed in Sec. V with circuit evaluation performed on PennyLane's noise-free `default.qubit` simulator. Plot (A) shows the cost function $f(\boldsymbol{\theta})$ in terms of the total number of circuit evaluations and Plot (B) shows the path taken by the optimizer through the θ_1-θ_2 space. In (A), the dotted line gives the analytic minimum of -0.874 and in (B), the contrasting white and purple circles highlight the maximum and minimum. With respect to the simulation of Fig. 9, in this case a different regularization method is used as discussed in the text.