Variational Quantum State Diagonalization

Ryan LaRose,[1, 2] Arkin Tikku,[1, 3] Étude O'Neel-Judy,[1] Lukasz Cincio,[1] and Patrick J. Coles[1]

[1] *Theoretical Division, Los Alamos National Laboratory, Los Alamos, NM 87545, USA.*
[2] *Department of Computational Mathematics, Science,*
and Engineering & Department of Physics and Astronomy,
Michigan State University, East Lansing, MI 48823, USA.
[3] *Department of Physics, Blackett Laboratory, Imperial College London,*
Prince Consort Road, London SW7 2AZ, United Kingdom

Variational hybrid quantum-classical algorithms are promising candidates for near-term implementation on quantum computers. In these algorithms, a quantum computer evaluates the cost of a gate sequence (with speedup over classical cost evaluation), and a classical computer uses this information to adjust the parameters of the gate sequence. Here we present such an algorithm for quantum state diagonalization. State diagonalization has applications in condensed matter physics (e.g., entanglement spectroscopy) as well as in machine learning (e.g., principal component analysis). For a quantum state ρ and gate sequence U, our cost function quantifies how far $U\rho U^\dagger$ is from being diagonal. We introduce novel short-depth quantum circuits to quantify our cost. Minimizing this cost returns a gate sequence that approximately diagonalizes ρ. One can then read out approximations of the largest eigenvalues, and the associated eigenvectors, of ρ. As a proof-of-principle, we implement our algorithm on Rigetti's quantum computer to diagonalize one-qubit states and on a simulator to find the entanglement spectrum of the Heisenberg model ground state.

I. INTRODUCTION

The future applications of quantum computers, assuming that large-scale, fault-tolerant versions will eventually be realized, are manifold. From a mathematical perspective, applications include number theory [1], linear algebra [2–4], differential equations [5, 6], and optimization [7]. From a physical perspective, applications include electronic structure determination [8, 9] for molecules and materials and real-time simulation of quantum dynamical processes [10] such as protein folding and photo-excitation events. Naturally, some of these applications are more long-term than others. Factoring and solving linear systems of equations are typically viewed as longer term applications due to their high resource requirements. On the other hand, approximate optimization and the determination of electronic structure may be nearer term applications, and could even serve as demonstrations of quantum supremacy in the near future [11, 12].

A major aspect of quantum algorithms research is to make applications of interest more near term by reducing quantum resource requirements including qubit count, circuit depth, numbers of gates, and numbers of measurements. A powerful strategy for this purpose is algorithm hybridization, where a fully quantum algorithm is turned into a hybrid quantum-classical algorithm [13]. The benefit of hybridization is two-fold, both reducing the resources (hence allowing implementation on smaller hardware) as well as increasing accuracy (by outsourcing calculations to "error-free" classical computers).

Variational hybrid algorithms are a class of quantum-classical algorithms that involve minimizing a cost function that depends on the parameters of a quantum gate sequence. Cost evaluation occurs on the quantum computer, with speedup over classical cost evaluation, and the classical computer uses this cost information to adjust the parameters of the gate sequence. Variational hybrid algorithms have been proposed for Hamiltonian ground state and excited state preparation [8, 14, 15], approximate optimization [7], error correction [16], quantum data compression [17, 18], quantum simulation [19, 20], and quantum compiling [21]. A key feature of such algorithms is their near-term relevance, since only the subroutine of cost evaluation occurs on the quantum computer, while the optimization procedure is entirely classical, and hence standard classical optimization tools can be employed.

In this work, we consider the application of diagonalizing quantum states. In condensed matter physics, diagonalizing states is useful for identifying properties of topological quantum phases—a field known as entanglement spectroscopy [22]. In data science and machine learning, diagonalizing the covariance matrix (which could be encoded in a quantum state [2, 23]) is frequently employed for principal component analysis (PCA). PCA identifies features that capture the largest variance in one's data and hence allows for dimensionality reduction [24].

Classical methods for diagonalization typically scale polynomially in the matrix dimension [25]. Similarly, the number of measurements required for quantum state tomography—a general method for fully characterizing a quantum state—scales polynomially in the dimension. Interestingly, Lloyd et al. proposed a quantum algorithm for diagonalizing quantum states that can potentially perform exponentially faster than these methods [2]. Namely, their algorithm, called quantum principal component analysis (qPCA), gives an exponential speedup for low-rank matrices. qPCA employs quantum phase estimation combined with density matrix exponentiation. These subroutines require a significant number of qubits and gates, making qPCA difficult to implement

in the near term, despite its long-term promise.

Here, we propose a variational hybrid algorithm for quantum state diagonalization. For a given state ρ, our algorithm is composed of three steps: (i) Train the parameters $\boldsymbol{\alpha}$ of a gate sequence $U_p(\boldsymbol{\alpha})$ such that $\tilde{\rho} = U_p(\boldsymbol{\alpha}_{\text{opt}})\rho U_p(\boldsymbol{\alpha}_{\text{opt}})^\dagger$ is approximately diagonal, where $\boldsymbol{\alpha}_{\text{opt}}$ is the optimal value of $\boldsymbol{\alpha}$ obtained (ii) Read out the largest eigenvalues of ρ by measuring in the eigenbasis (i.e., by measuring $\tilde{\rho}$ in the standard basis), and (iii) Prepare the eigenvectors associated with the largest eigenvalues. We call this the variational quantum state diagonalization (VQSD) algorithm. VQSD is a near-term algorithm with the same practical benefits as other variational hybrid algorithms. Employing a layered ansatz for $U_p(\boldsymbol{\alpha})$ (where p is the number of layers) allows one to obtain a hierarchy of approximations for the eigevalues and eigenvectors. We therefore think of VQSD as an approximate diagonalization algorithm.

We carefully choose our cost function C to have the following properties: (i) C is faithful (i.e, it vanishes if and only if $\tilde{\rho}$ is diagonal), (ii) C is efficiently computable on a quantum computer, (iii) C has operational meanings such that it upper bounds the eigenvalue and eigenvector error (see Sec. II A), and (iv) C scales well for training purposes in the sense that its gradient does not vanish exponentially in the number of qubits. The precise definition of C is given in Sec. II A and involves a difference of purities for different states. To compute C, we introduce novel short-depth quantum circuits that likely have applications outside the context of VQSD.

To illustrate our method, we implement VQSD on Rigetti's 8-qubit quantum computer. We successfully diagonalize one-qubit pure states using this quantum computer. To highlight future applications (when larger quantum computers are made available), we implement VQSD on a simulator to perform entanglement spectroscopy on the ground state of the one-dimensional (1D) Heisenberg model composed of 12 spins.

Our paper is organized as follows. Section II outlines the VQSD algorithm and presents its implementation. In Sec. III, we give a comparison to the qPCA algorithm, and we elaborate on future applications. Section IV presents our methods for quantifying diagonalization and for optimizing our cost function.

II. RESULTS

A. The VQSD Algorithm

1. Overall Structure

Figure 1 shows the structure of the VQSD algorithm. The goal of VQSD is to take, as its input, an n-qubit density matrix ρ given as a quantum state and then output approximations of the m-largest eigenvalues and their associated eigenvectors. Here, m will typically be much less than 2^n, the matrix dimension of ρ, although the user is

free to increase m with increased algorithmic complexity (discussed below). The outputted eigenvalues will be in classical form, i.e., will be stored on a classical computer. In contrast, the outputted eigenvectors will be in quantum form, i.e., will be prepared on a quantum computer. This is necessary because the eigenvectors would have 2^n entries if they were stored on a classical computer, which is intractable for large n. Nevertheless, one can characterize important aspects of these eigenvectors with a polynomial number of measurements on the quantum computer.

Similar to classical eigensolvers, the VQSD algorithm is an approximate or iterative diagonalization algorithm. Classical eigenvalue algorithms are necessarily iterative, not exact [26]. Iterative algorithms are useful in that they allow for a trade-off between run-time and accuracy. Higher degrees of accuracy can be achieved at the cost of more iterations (equivalently, longer run-time), or short run-time can be achieved at the cost of lower accuracy. This flexibility is desirable in that it allows the user of the algorithm to dictate the quality of the solutions found.

The iterative feature of VQSD arises via a layered ansatz for the diagonalizing unitary. This idea similarly appears in other variational hybrid algorithms, such as the Quantum Approximate Optimization Algorithm [7]. Specifically, VQSD diagonalizes ρ by variationally updating a parameterized unitary $U_p(\boldsymbol{\alpha})$ such that

$$\tilde{\rho}_p(\boldsymbol{\alpha}) := U_p(\boldsymbol{\alpha})\rho U_p^\dagger(\boldsymbol{\alpha}) \tag{1}$$

is (approximately) diagonal at the optimal value $\boldsymbol{\alpha}_{\text{opt}}$. (For brevity we often write $\tilde{\rho}$ for $\tilde{\rho}_p(\boldsymbol{\alpha})$.) We assume a layered ansatz of the form

$$U_p(\boldsymbol{\alpha}) = L_1(\boldsymbol{\alpha}_1)L_2(\boldsymbol{\alpha}_2)\cdots L_p(\boldsymbol{\alpha}_p). \tag{2}$$

Here, p is a hyperparameter that sets the number of layers $L_i(\boldsymbol{\alpha}_i)$, and each $\boldsymbol{\alpha}_i$ is a set of optimization parameters that corresponds to internal gate angles within the layer. The parameter $\boldsymbol{\alpha}$ in (1) refers to the collection of all $\boldsymbol{\alpha}_i$ for $i = 1, ..., p$. Once the optimization procedure is finished and returns the optimal parameters $\boldsymbol{\alpha}_{\text{opt}}$, one can then run a particular quantum circuit (shown in Fig. 1(c) and discussed below) N_{readout} times to approximately determine the eigenvalues of ρ. The precision (i.e, the number of significant digits) of each eigenvalue increases with N_{readout} and with the eigenvalue's magnitude. Hence for small N_{readout} only the largest eigenvalues of ρ will be precisely characterized, so there is a connection between N_{readout} and how many eigenvalues, m, are determined. The hyperparameter p is a refinement parameter, meaning that the accuracy of the eigensystem (eigenvalues and eigenvectors) typically increases as p increases. We formalize this argument as follows.

Let C denote our cost function, defined below in (10), which we are trying to minimize. In general, the cost C will be non-increasing (i.e., will either decrease or stay constant) in p. One can ensure that this is true by taking the optimal parameters learned for p layers as the starting
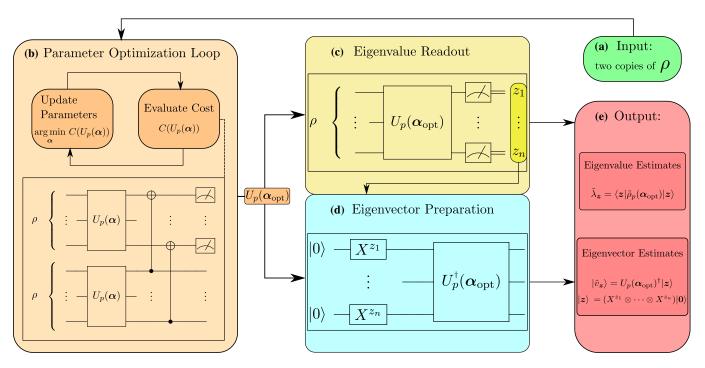
FIG. 1. Schematic diagram showing the steps of the VQSD algorithm. (a) Two copies of quantum state ρ are provided as an input. These states are sent to the parameter optimization loop (b) where a hybrid quantum-classical variational algorithm approximates the diagonalizing unitary $U_p(\boldsymbol{\alpha}_{\mathrm{opt}})$. Here, p is a hyperparameter that dictates the quality of solution found. This optimal unitary is sent to the eigenvalue readout circuit (c) to obtain bitstrings $\boldsymbol{z}$, the frequencies of which provide estimates of the eigenvalues of ρ. Along with the optimal unitary $U_p(\boldsymbol{\alpha}_{\mathrm{opt}})$, these bitstrings are sent to the eigenvector preparation circuit (c) to prepare the eigenstates of ρ on a quantum computer. Both the eigenvalues and eigenvectors are the outputs (d) of the VQSD algorithm.

point for the optimization of $p+1$ layers and by setting $\boldsymbol{\alpha}_{p+1}$ such that $L_{p+1}(\boldsymbol{\alpha}_{p+1})$ is an identity. This strategy also avoids barren plateaus [27, 28] and helps to mitigate the problem of local minima, as we discuss in Appendix C of Supplementary Material (SM).

Next, we argue that C is closely connected to the accuracy of the eigensystem. Specifically, it gives an upper bound on the eigensystem error. Hence, one obtains an increasingly tighter upper bound on the eigensystem error as C decreases (equivalently, as p increases). To quantify eigenvalue error, we define

$$\Delta_\lambda := \sum_{i=1}^{d} (\lambda_i - \tilde{\lambda}_i)^2 \,, \qquad (3)$$

where $d = 2^n$, and $\{\lambda_i\}$ and $\{\tilde{\lambda}_i\}$ are the true and inferred eigenvalues, respectively. Here, i is an index that orders the eigenvalues in decreasing order, i.e., $\lambda_i \geqslant \lambda_{i+1}$ and $\tilde{\lambda}_i \geqslant \tilde{\lambda}_{i+1}$ for all $i \in \{1, ..., d-1\}$. To quantify eigenvector error, we define

$$\Delta_v := \sum_{i=1}^{d} \langle \delta_i | \delta_i \rangle \,, \quad \text{with } |\delta_i\rangle = \rho|\tilde{v}_i\rangle - \tilde{\lambda}_i|\tilde{v}_i\rangle = \Pi_i^\perp \rho|\tilde{v}_i\rangle \,. \qquad (4)$$

Here, $|\tilde{v}_i\rangle$ is the inferred eigenvector associated with $\tilde{\lambda}_i$, and $\Pi_i^\perp = \mathbb{1} - |\tilde{v}_i\rangle\langle\tilde{v}_i|$ is the projector onto the subspace

orthogonal to $|\tilde{v}_i\rangle$. Hence, $|\delta_i\rangle$ is a vector whose norm quantifies the component of $\rho|\tilde{v}_i\rangle$ that is orthogonal to $|\tilde{v}_i\rangle$, or in other words, how far $|\tilde{v}_i\rangle$ is from being an eigenvector of ρ.

As proven in Sec. IV A, our cost function upper bounds the eigenvalue and eigenvector error up to a proportionality factor β,

$$\Delta_\lambda \leqslant \beta C \,, \quad \text{and} \quad \Delta_v \leqslant \beta C \,. \qquad (5)$$

Because C is non-increasing in p, the upper bound in (5) is non-increasing in p and goes to zero if C goes to zero.

We remark that Δ_v can be interpreted as a weighted eigenvector error, where eigenvectors with larger eigenvalues are weighted more heavily in the sum. This is a useful feature since it implies that lowering the cost C will force the eigenvectors with the largest eigenvalues to be highly accurate. In many applications, such eigenvectors are precisely the ones of interest. (See Sec. II B 2 for an illustration of this feature.)

The various steps in the VQSD algorithm are shown schematically in Fig. 1. There are essentially three main steps: (1) an optimization loop that minimizes the cost C via back-and-forth communication between a classical and quantum computer, where the former adjusts $\boldsymbol{\alpha}$ and the latter computes C for $U_p(\boldsymbol{\alpha})$, (2) a readout procedure for approximations of the m largest eigenvalues, which involves running a quantum circuit and then classically
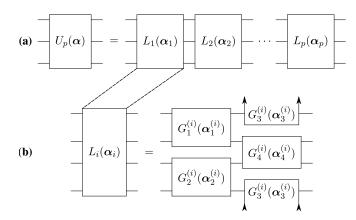
FIG. 2. **(a)** Layered ansatz for the diagonalizing unitary $U_p(\boldsymbol{\alpha})$. Each layer L_i, $i = 1, ..., p$, consists of a set of optimization parameters $\boldsymbol{\alpha}_i$. **(b)** The two-qubit gate ansatz for the ith layer, shown on four qubits. Here we impose periodic boundary conditions on the top/bottom edge of the circuit so that G_3 wraps around from top to bottom. Appendix B of SM discusses an alternative approach to the construction of $U_p(\boldsymbol{\alpha})$, in which the ansatz is modified during the optimization process.

analyzing the statistics, and (3) a preparation procedure to prepare approximations of the eigenvectors associated with the m largest eigenvalues. In the following subsections, we elaborate on each of these procedures.

2. Parameter Optimization Loop

Naturally, there are many ways to parameterize $U_p(\boldsymbol{\alpha})$. Ideally one would like the number of parameters to grow at most polynomially in both n and p. Figure 2 presents an example ansatz that satisfies this condition. Each layer L_i is broken down into layers of two-body gates that can be performed in parallel. These two-body gates can be further broken down into parameterized one-body gates, for example, with the construction in Ref. [29]. We discuss a different approach to parameterize $U_p(\boldsymbol{\alpha})$ in Appendix B of SM.

For a given ansatz, such as the one in Fig. 2, parameter optimization involves evaluating the cost C on a quantum computer for an initial choice of parameters and then modifying the parameters on a classical computer in an iterative feedback loop. The goal is to find

$$\boldsymbol{\alpha}_{\rm opt} := \arg\min_{\boldsymbol{\alpha}} C(U_p(\boldsymbol{\alpha})) . \qquad (6)$$

The classical optimization routine used for updating the parameters can involve either gradient-free or gradient-based methods. In Sec. IV B, we explore this further and discuss our optimization methods.

In Eq. (6), $C(U_p(\boldsymbol{\alpha}))$ quantifies how far the state $\tilde{\rho}_p(\boldsymbol{\alpha})$ is from being diagonal. There are many ways to define such a cost function, and in fact there is an entire field of research on *coherence measures* that has introduced

various such quantities [30]. We aim for a cost that is efficiently computable with a quantum-classical system, and hence we consider a cost that can be expressed in terms of purities. (It is well known that a quantum computer can find the purity $\mathrm{Tr}(\sigma^2)$ of an n-qubit state σ with complexity scaling only linearly in n, an exponential speedup over classical computation [31, 32].) Two such cost functions, whose individual merits we discuss in Sec. IV A, are

$$C_1(U_p(\boldsymbol{\alpha})) = \mathrm{Tr}(\rho^2) - \mathrm{Tr}(\mathcal{Z}(\tilde{\rho})^2) , \qquad (7)$$

$$C_2(U_p(\boldsymbol{\alpha})) = \mathrm{Tr}(\rho^2) - \frac{1}{n} \sum_{j=1}^{n} \mathrm{Tr}(\mathcal{Z}_j(\tilde{\rho})^2) . \qquad (8)$$

Here, $\mathcal{Z}$ and $\mathcal{Z}_j$ are quantum channels that dephase (i.e., destroy the off-diagonal elements) in the global standard basis and in the local standard basis on qubit j, respectively. Importantly, the two functions vanish under the same conditions:

$$C_1(U_p(\boldsymbol{\alpha})) = 0 \iff C_2(U_p(\boldsymbol{\alpha})) = 0 \iff \tilde{\rho} = \mathcal{Z}(\tilde{\rho}) . \qquad (9)$$

So the global minima of C_1 and C_2 coincide and correspond precisely to unitaries $U_p(\boldsymbol{\alpha})$ that diagonalize ρ (i.e., unitaries such that $\tilde{\rho}$ is diagonal).

As elaborated in Sec. IV A, C_1 has operational meanings: it bounds our eigenvalue error, $C_1 \geqslant \Delta_\lambda$, and it is equivalent to our eigenvector error, $C_1 = \Delta_v$. However, its landscape tends to be insensitive to changes in $U_p(\boldsymbol{\alpha})$ for large n. In contrast, we are not aware of a direct operational meaning for C_2, aside from its bound on C_1 given by $C_2 \geqslant (1/n)C_1$. However, the landscape for C_2 is more sensitive to changes in $U_p(\boldsymbol{\alpha})$, making it useful for training $U_p(\boldsymbol{\alpha})$ when n is large. Due to these contrasting merits of C_1 and C_2, we define our overall cost function C as a weighted average of these two functions

$$C(U_p(\boldsymbol{\alpha})) = qC_1(U_p(\boldsymbol{\alpha})) + (1 - q)C_2(U_p(\boldsymbol{\alpha})) , \qquad (10)$$

where $q \in [0, 1]$ is a free parameter that allows one to tailor the VQSD method to the scale of one's problem. For small n, one can set $q \approx 1$ since the landscape for C_1 is not too flat for small n, and, as noted above, C_1 is an operationally relevant quantity. For large n, one can set q to be small since the landscape for C_2 will provide the gradient needed to train $U_p(\boldsymbol{\alpha})$. The overall cost maintains the operational meaning in (5) with

$$\beta = n/(1 + q(n - 1)) . \qquad (11)$$

Appendix D illustrates the advantages of training with different values of q.

Computing C amounts to evaluating the purities of various quantum states on a quantum computer and then doing some simple classical post-processing that scales linearly in n. This can be seen from Eqns. (7) and (8). The first term, $\mathrm{Tr}(\rho^2)$, in C_1 and C_2 is independent of

$U_p(\boldsymbol{\alpha})$. Hence, $\text{Tr}(\rho^2)$ can be evaluated outside of the optimization loop in Fig. 1 using the Destructive Swap Test (see Sec. IV A for the circuit diagram). Inside the loop, we only need to compute $\text{Tr}(\mathcal{Z}(\tilde{\rho})^2)$ and $\text{Tr}(\mathcal{Z}_j(\tilde{\rho})^2)$ for all j. Each of these terms are computed by first preparing two copies of $\tilde{\rho}$ and then implementing quantum circuits whose depths are constant in n. For example, the circuit for computing $\text{Tr}(\mathcal{Z}(\tilde{\rho})^2)$ is shown in Fig. 1(b), and surprisingly it has a depth of only one gate. We call it the Diagonalized Inner Product (DIP) Test. The circuit for computing $\text{Tr}(\mathcal{Z}_j(\tilde{\rho})^2)$ is similar, and we call it the Partially Diagonalized Inner Product (PDIP) Test. We elaborate on both of these circuits in Sec. IV A.

3. Eigenvalue Readout

After finding the optimal diagonalizing unitary $U_p(\boldsymbol{\alpha}_{\text{opt}})$, one can use it to readout approximations of the eigenvalues of ρ. Figure 1(c) shows the circuit for this readout. One prepares a single copy of ρ and then acts with $U_p(\boldsymbol{\alpha}_{\text{opt}})$ to prepare $\tilde{\rho}_p(\boldsymbol{\alpha}_{\text{opt}})$. Measuring in the standard basis $\{|\boldsymbol{z}\rangle\}$, where $\boldsymbol{z} = z_1 z_2 ... z_n$ is a bitstring of length n, gives a set of probabilities $\{\tilde{\lambda}_{\boldsymbol{z}}\}$ with

$$\tilde{\lambda}_{\boldsymbol{z}} = \langle \boldsymbol{z} | \tilde{\rho}_p(\boldsymbol{\alpha}_{\text{opt}}) | \boldsymbol{z} \rangle. \tag{12}$$

We take the $\tilde{\lambda}_{\boldsymbol{z}}$ as the inferred eigenvalues of ρ. We emphasize that the $\tilde{\lambda}_{\boldsymbol{z}}$ are the diagonal elements, not the eigenvalues, of $\tilde{\rho}_p(\boldsymbol{\alpha}_{\text{opt}})$.

Each run of the circuit in Fig. 1(c) generates a bitstring $\boldsymbol{z}$ corresponding to the measurement outcomes. If one obtains $\boldsymbol{z}$ with frequency $f_{\boldsymbol{z}}$ for N_{readout} total runs, then

$$\tilde{\lambda}_{\boldsymbol{z}}^{\text{est}} = f_{\boldsymbol{z}} / N_{\text{readout}} \tag{13}$$

gives an estimate for $\tilde{\lambda}_{\boldsymbol{z}}$. The statistical deviation of $\tilde{\lambda}_{\boldsymbol{z}}^{\text{est}}$ from $\tilde{\lambda}_{\boldsymbol{z}}$ goes with $1/\sqrt{N_{\text{readout}}}$. The relative error $\epsilon_{\boldsymbol{z}}$ (i.e., the ratio of the statistical error on $\tilde{\lambda}_{\boldsymbol{z}}^{\text{est}}$ to the value of $\tilde{\lambda}_{\boldsymbol{z}}^{\text{est}}$) then goes as

$$\epsilon_{\boldsymbol{z}} = \frac{1}{\sqrt{N_{\text{readout}}} \tilde{\lambda}_{\boldsymbol{z}}^{\text{est}}} = \frac{\sqrt{N_{\text{readout}}}}{f_{\boldsymbol{z}}}. \tag{14}$$

This implies that events $\boldsymbol{z}$ with higher frequency $f_{\boldsymbol{z}}$ have lower relative error. In other words, the larger the inferred eigenvalue $\tilde{\lambda}_{\boldsymbol{z}}$, the lower the relative error, and hence the more precisely it is determined from the experiment. When running VQSD, one can pre-decide on the desired values of N_{readout} and a threshold for the relative error, denoted $\epsilon_{\max}$. This error threshold $\epsilon_{\max}$ will then determine m, i.e., how many of the largest eigenvalues that get precisely characterized. So $m = m(N_{\text{readout}}, \epsilon_{\max}, \{\tilde{\lambda}_{\boldsymbol{z}}\})$ is a function of N_{readout}, $\epsilon_{\max}$, and the set of inferred eigenvalues $\{\tilde{\lambda}_{\boldsymbol{z}}\}$. Precisely, we take $m = |\tilde{\boldsymbol{\lambda}}^{\text{est}}|$ as the cardinality of the following set:

$$\tilde{\boldsymbol{\lambda}}^{\text{est}} = \{\tilde{\lambda}_{\boldsymbol{z}}^{\text{est}} : \epsilon_{\boldsymbol{z}} \leqslant \epsilon_{\max}\}, \tag{15}$$

which is the set of inferred eigenvalues that were estimated with the desired precision.

4. Eigenvector Preparation

The final step of VQSD is to prepare the eigenvectors associated with the m-largest eigenvalues, i.e., the eigenvalues in the set in Eq. (15). Let $\boldsymbol{Z} = \{\boldsymbol{z} : \tilde{\lambda}_{\boldsymbol{z}}^{\text{est}} \in \tilde{\boldsymbol{\lambda}}^{\text{est}}\}$ be the set of bitstrings $\boldsymbol{z}$ associated with the eigenvalues in $\tilde{\boldsymbol{\lambda}}^{\text{est}}$. (Note that these bitstrings are obtained directly from the measurement outcomes of the circuit in Fig. 1(c), i.e., the outcomes become the bitstring $\boldsymbol{z}$.) For each $\boldsymbol{z} \in \boldsymbol{Z}$, one can prepare the following state, which we take as the inferred eigenvector associated with our estimate of the inferred eigenvalue $\tilde{\lambda}_{\boldsymbol{z}}^{\text{est}}$,

$$|\tilde{v}_{\boldsymbol{z}}\rangle = U_p(\boldsymbol{\alpha}_{\text{opt}})^{\dagger} |\boldsymbol{z}\rangle \tag{16}$$

$$= U_p(\boldsymbol{\alpha}_{\text{opt}})^{\dagger} (X^{z_1} \otimes \cdots \otimes X^{z_n}) |\boldsymbol{0}\rangle. \tag{17}$$

The circuit for preparing this state is shown in Fig. 1(d). As noted in (17), one first prepares $|\boldsymbol{z}\rangle$ by acting with X operators raised to the appropriate powers, and then one acts with $U_p(\boldsymbol{\alpha}_{\text{opt}})^{\dagger}$ to rotate from the standard basis to the inferred eigenbasis.

Once they are prepared on the quantum computer, each inferred eigenvector $|\tilde{v}_{\boldsymbol{z}}\rangle$ can be characterized by measuring expectation values of interest. That is, important physical features such as energy or entanglement (e.g., entanglement witnesses) are associated with some Hermitian observable M, and one can evaluate the expectation value $\langle \tilde{v}_{\boldsymbol{z}} | M | \tilde{v}_{\boldsymbol{z}} \rangle$ to learn about these features.

B. Implementations

Here we present our implementations of VQSD, first for a one-qubit state on a cloud quantum computer to show that it is amenable to currently available hardware. Then, to illustrate the scaling to larger, more interesting problems, we implement VQSD on a simulator for the 12-spin ground state of the Heisenberg model. See Appendices A and B of SM for further details. The code used to generate some of the examples presented here and in SM can be accessed from [33].

1. One-Qubit State

We now discuss the results of applying VQSD to the one-qubit plus state $\rho = |+\rangle\langle+|$ on the 8Q-Agave quantum computer provided by Rigetti [34]. Because the problem size is small ($n = 1$), we set $q = 1$ in the cost function (10). Since ρ is a pure state, the cost function is

$$C(U_p(\boldsymbol{\alpha})) = C_1(U_p(\boldsymbol{\alpha})) = 1 - \text{Tr}(\mathcal{Z}(\tilde{\rho})^2). \tag{18}$$

For $U_p(\boldsymbol{\alpha})$, we take $p = 1$, for which the layered ansatz becomes an arbitrary single qubit rotation.

The results of VQSD for this state are shown in Fig. 3. In Fig. 3(a), the solid curve shows the cost versus the
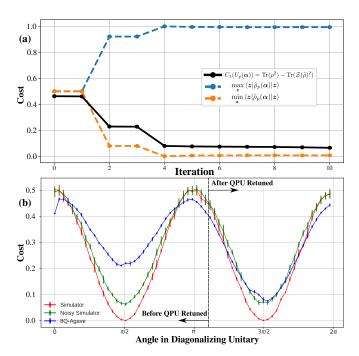
FIG. 3. The VQSD algorithm run on Rigetti's 8Q-Agave quantum computer for $\rho = |+\rangle\langle+|$. (a) A representative run of the parameter optimization loop, using the Powell optimization algorithm (see Sec. IV B for details and Appendix A for data from additional runs). Cost versus iteration is shown by the black solid line. The dotted lines show the two inferred eigenvalues. After four iterations, the inferred eigenvalues approach $\{0, 1\}$, as required for a pure state. (b) The cost landscape on a noiseless simulator, Rigetti's noisy simulator, and Rigetti's quantum computer. Error bars show the standard deviation (due to finite sampling) of multiple runs. The local minima occur roughly at the theoretically predicted values of $\pi/2$ and $3\pi/2$. During data collection for this plot, the 8Q-Agave quantum computer retuned, after which its cost landscape closely matched that of the noisy simulator.

number of iterations in the parameter optimization loop, and the dashed curves show the inferred eigenvalues of ρ at each iteration. Here we used the Powell optimization algorithm, see Section IV B for more details. As can be seen, the cost decreases to a small value near zero and the eigenvalue estimates simultaneously converge to the correct values of zero and one. Hence, VQSD successfully diagonalized this state.

Figure 3(b) shows the landscape of the optimization problem on Rigetti's 8Q-Agave quantum computer, Rigetti's noisy simulator, and a noiseless simulator. Here, we varied the angle α in the diagonalizing unitary $U(\alpha) = R_x(\pi/2)R_z(\alpha)$ and computed the cost at each value of this angle. The landscape on the quantum computer has local minima near the optimal angles $\alpha = \pi/2, 3\pi/2$ but the cost is not zero. This explains why we obtain the correct eigenvalues even though the cost is nonzero in Fig. 3(a). The nonzero cost can be due to a combination of decoherence, gate infidelity, and measurement error. As shown in Fig. 3(b), the 8Q-Agave quantum computer
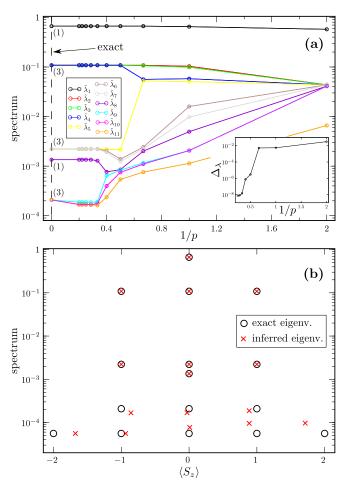

FIG. 4. Implementing VQSD with a simulator for the ground state of the 1D Heisenberg model, diagonalizing a 4-spin subsystem of a chain of 8 spins. We chose $q = 1$ for the cost in (10) and employed a gradient-based method to find $\boldsymbol{\alpha}_{\mathrm{opt}}$. (a) Largest inferred eigenvalues $\tilde{\lambda}_j$ versus $1/p$, where p is the number of layers in our ansatz, which in this example takes half-integer values corresponding to fractions of layers shown in Fig. 2. The exact eigenvalues are shown on the y-axis (along $1/p = 0$ line) with their degeneracy indicated in parentheses. One can see the largest eigenvalues converge to their correct values, including the correct degeneracies. Inset: overall eigenvalue error Δ_λ versus $1/p$. (b) Largest inferred eigenvalues resolved by the inferred $\langle S_z \rangle$ quantum number of their associated eigenvector, for $p = 5$. The inferred data points (red X's) roughly agree with the theoretical values (black circles), particularly for the largest eigenvalues. Appendix B of SM discusses Heisenberg chain of 12 spins.

retuned during our data collection, and after this retuning, the landscape of the quantum computer matched that of the noisy simulator significantly better.

2. Heisenberg Model Ground State

While current noise levels of quantum hardware limit our implementations of VQSD to small problem sizes,

we can explore larger problem sizes on a simulator. An important application of VQSD is to study the entanglement in condensed matter systems, and we highlight this application in the following example.

Let us consider the ground state of the 1D Heisenberg model, the Hamiltonian of which is

$$H = \sum_{j=1}^{2n} \boldsymbol{S}^{(j)} \cdot \boldsymbol{S}^{(j+1)}, \qquad (19)$$

with $\boldsymbol{S}^{(j)} = (1/2)(\sigma_x^{(j)}\hat{x} + \sigma_y^{(j)}\hat{y} + \sigma_z^{(j)}\hat{z})$ and periodic boundary conditions, $\boldsymbol{S}^{(2n+1)} = \boldsymbol{S}^{(1)}$. Performing entanglement spectroscopy on the ground state $|\psi\rangle_{AB}$ involves diagonalizing the reduced state $\rho = \text{Tr}_B(|\psi\rangle\langle\psi|_{AB})$. Here we consider a total of 8 spins ($2n = 8$). We take A to be a subset of 4 nearest-neighbor spins, and B is the complement of A.

The results of applying VQSD to the 4-spin reduced state ρ via a simulator are shown in Fig. 4. Panel (a) plots the inferred eigenvalues versus the number of layers p in our ansatz (see Fig. 2). One can see that the inferred eigenvalues converge to their theoretical values as p increases. Panel (b) plots the inferred eigenvalues resolved by their associated quantum numbers (z-component of total spin). This plot illustrates the feature we noted previously that minimizing our cost will first result in minimizing the eigenvector error for those eigenvectors with the largest eigenvalues. Overall our VQSD implementation returned roughly the correct values for both the eigenvalues and their quantum numbers. Resolving not only the eigenvalues but also their quantum numbers is important for entanglement spectroscopy [22], and clearly VQSD can do this.

In Appendix B of SM we discuss an alternative approach employing a variable ansatz for $U_p(\boldsymbol{\alpha})$, and we present results of applying this approach to a 6-qubit reduced state of the 12-qubit ground state of the Heisenberg model.

III. DISCUSSION

We emphasize that VQSD is meant for states ρ that have either low rank or possibly high rank but low entropy $H(\rho) = -\text{Tr}(\rho \log \rho)$. This is because the eigenvalue readout step of VQSD would be exponentially complex for states with high entropy. In other words, for high entropy states, if one efficiently implemented the eigenvalue readout step (with N_{readout} polynomial in n), then very few eigenvalues would get characterized with the desired precision. In Appendix F of SM we discuss the complexity of VQSD for particular example states.

Examples of states for which VQSD is expected to be efficient include density matrices computed from ground states of 1D, local, gapped Hamiltonians. Also, thermal states of some 1D systems in a many-body localized phase at low enough temperature are expected to be diagonalizable by VQSD. These states have rapidly decaying spectra and are eigendecomposed into states obeying a 1D area law [35–37]. This means that every eigenstate can be prepared by a constant depth circuit in alternating ansatz form [36], and hence VQSD will be able to diagonalize it.

A. Comparison to Literature

Diagonalizing quantum states with classical methods would require exponentially large memory to store the density matrix, and the matrix operations needed for diagonalization would be exponentially costly. VQSD avoids both of these scaling issues.

Another quantum algorithm that extracts the eigenvalues and eigenvectors of a quantum state is qPCA [2]. Similar to VQSD, qPCA has the potential for exponential speedup over classical diagonalization for particular classes of quantum states. Like VQSD, the speedup in qPCA is contingent on ρ being a low-entropy state.

We performed a simple implementation of qPCA to get a sense for how it compares to VQSD, see Appendix G in SM for details. In particular, just like we did for Fig. 3, we considered the one-qubit plus state $\rho = |+\rangle\langle+|$. We implemented qPCA for this state on Rigetti's noisy simulator (whose noise is meant to mimic that of their 8Q-Agave quantum computer). The circuit that we implemented applied one controlled-exponential-swap gate (in order to approximately exponentiate ρ, as discussed in [2]). We employed a machine-learning approach [38] to compile the controlled-exponential-swap gate into a novel short-depth gate sequence (see Appendix G in SM). With this circuit we inferred the two eigenvalues of ρ to be approximately 0.8 and 0.2. Hence, for this simple example, it appears that qPCA gave eigenvalues that were slightly off from the true values of 1 and 0, while VQSD was able to obtain the correct eigenvalues, as discussed in Fig. 3.

B. Future Applications

Finally we discuss various applications of VQSD.

As noted in Ref. [2], one application of quantum state diagonalization is benchmarking of quantum noise processes, i.e., quantum process tomography. Here one prepares the Choi state by sending half of a maximally entangled state through the process of interest. One can apply VQSD to the resulting Choi state to learn about the noise process, which may be particular useful for benchmarking near-term quantum computers.

A special case of VQSD is variational state preparation. That is, if one applies VQSD to a pure state $\rho = |\psi\rangle\langle\psi|$, then one can learn the unitary $U(\boldsymbol{\alpha})$ that maps $|\psi\rangle$ to a standard basis state. Inverting this unitary allows one to map a standard basis state (and hence the state $|0\rangle^{\otimes n}$) to the state $|\psi\rangle$, which is known as state preparation. Hence, if one is given $|\psi\rangle$ in quantum form, then VQSD can potentially find a short-depth circuit

that approximately prepares $|\psi\rangle$. Variational quantum compiling algorithms that were very recently proposed [21, 39] may also be used for this same purpose, and hence it would be interesting to compare VQSD to these algorithms for this special case. Additionally, in this special case one could use VQSD and these other algorithms as an error mitigation tool, i.e., to find a short-depth state preparation that achieves higher accuracy than the original state preparation.

In machine learning, PCA is a subroutine in supervised and unsupervised learning algorithms and also has many direct applications. PCA inputs a data matrix X and finds a new basis such that the variance is maximal along the new basis vectors. One can show that this amounts to finding the eigenvectors of the covariance matrix $E[XX^T]$ with the largest eigenvalues, where E denotes expectation value. Thus PCA involves diagonalizing a positive-semidefinite matrix, $E[XX^T]$. Hence VQSD can perform this task provided one has access to QRAM [23] to prepare the covariance matrix as a quantum state. PCA can reduce the dimension of X as well as filter out noise in data. In addition, nonlinear (kernel) PCA can be used on data that is not linearly separable. Very recent work by Tang [40] suggests that classical algorithms could be improved for PCA of low-rank matrices, and potentially obtain similar scaling as qPCA and VQSD. Hence future work is needed to compare these different approaches to PCA.

Perhaps the most important near-term application of VQSD is to study condensed matter physics. In particular, we propose that one can apply the variational quantum eigensolver [8] to prepare the ground state of a many-body system, and then one can follow this with the VQSD algorithm to characterize the entanglement in this state. Ultimately this approach could elucidate key properties of condensed matter phases. In particular, VQSD allows for entanglement spectroscopy, which has direct application to the identification of topological order [22]. Extracting both the eigenvalues and eigenvectors is useful for entanglement spectroscopy [22], and we illustrated this capability of VQSD in Fig. 4. Finally, an interesting future research direction is to check how the discrepancies in preparation of multiple copies affect the performance of the diagonalization.

IV. METHODS

A. Diagonalization Test Circuits

Here we elaborate on the cost functions C_1 and C_2 and present short-depth quantum circuits to compute them.

1. C_1 and the DIP Test

The function C_1 defined in (7) has several intuitive interpretations. These interpretations make it clear that C_1 quantifies how far a state is from being diagonal. In particular, let $D_{\mathrm{HS}}(A, B) := \mathrm{Tr}\left((A - B)^\dagger(A - B)\right)$ denote the Hilbert-Schmidt distance. Then we can write

$$C_1 = \min_{\sigma \in \mathcal{D}} D_{\mathrm{HS}}(\tilde{\rho}, \sigma) \qquad (20)$$

$$= D_{\mathrm{HS}}(\tilde{\rho}, \mathcal{Z}(\tilde{\rho})) \qquad (21)$$

$$= \sum_{\boldsymbol{z}, \boldsymbol{z}' \neq \boldsymbol{z}} |\langle \boldsymbol{z}|\tilde{\rho}|\boldsymbol{z}'\rangle|^2 . \qquad (22)$$

In other words, C_1 is (1) the minimum distance between $\tilde{\rho}$ and the set of diagonal states $\mathcal{D}$, (2) the distance from $\tilde{\rho}$ to $\mathcal{Z}(\tilde{\rho})$, and (3) the sum of the absolute squares of the off-diagonal elements of $\tilde{\rho}$.

C_1 can also be written as the eigenvector error in (4) as follows. For an inferred eigenvector $|\tilde{v}_{\boldsymbol{z}}\rangle$, we define $|\delta_{\boldsymbol{z}}\rangle = \rho|\tilde{v}_{\boldsymbol{z}}\rangle - \tilde{\lambda}_{\boldsymbol{z}}|\tilde{v}_{\boldsymbol{z}}\rangle$ and write the eigenvector error as

$$\langle \delta_{\boldsymbol{z}}|\delta_{\boldsymbol{z}}\rangle = \langle \tilde{v}_{\boldsymbol{z}}|\rho^2|\tilde{v}_{\boldsymbol{z}}\rangle + \tilde{\lambda}_{\boldsymbol{z}}^2 - 2\tilde{\lambda}_{\boldsymbol{z}}\langle \tilde{v}_{\boldsymbol{z}}|\rho|\tilde{v}_{\boldsymbol{z}}\rangle \qquad (23)$$

$$= \langle \tilde{v}_{\boldsymbol{z}}|\rho^2|\tilde{v}_{\boldsymbol{z}}\rangle - \tilde{\lambda}_{\boldsymbol{z}}^2 , \qquad (24)$$

since $\langle \tilde{v}_{\boldsymbol{z}}|\rho|\tilde{v}_{\boldsymbol{z}}\rangle = \tilde{\lambda}_{\boldsymbol{z}}$. Summing over all $\boldsymbol{z}$ gives

$$\Delta_v = \sum_{\boldsymbol{z}}\langle \delta_{\boldsymbol{z}}|\delta_{\boldsymbol{z}}\rangle = \sum_{\boldsymbol{z}}\langle \tilde{v}_{\boldsymbol{z}}|\rho^2|\tilde{v}_{\boldsymbol{z}}\rangle - \tilde{\lambda}_{\boldsymbol{z}}^2 \qquad (25)$$

$$= \mathrm{Tr}(\rho^2) - \mathrm{Tr}(\mathcal{Z}(\tilde{\rho})^2) = C_1 , \qquad (26)$$

which proves the bound in (5) for $q = 1$.

In addition, C_1 bounds the eigenvalue error defined in (3). Let $\tilde{\boldsymbol{\lambda}} = (\tilde{\lambda}_1, ..., \tilde{\lambda}_d)$ and $\boldsymbol{\lambda} = (\lambda_1, ..., \lambda_d)$ denote the inferred and actual eigenvalues of ρ, respectively, both arranged in decreasing order. In this notation we have

$$\Delta_\lambda = \boldsymbol{\lambda} \cdot \boldsymbol{\lambda} + \tilde{\boldsymbol{\lambda}} \cdot \tilde{\boldsymbol{\lambda}} - 2\boldsymbol{\lambda} \cdot \tilde{\boldsymbol{\lambda}} \qquad (27)$$

$$C_1 = \boldsymbol{\lambda} \cdot \boldsymbol{\lambda} - \tilde{\boldsymbol{\lambda}} \cdot \tilde{\boldsymbol{\lambda}} \qquad (28)$$

$$= \Delta_\lambda + 2(\boldsymbol{\lambda} \cdot \tilde{\boldsymbol{\lambda}} - \tilde{\boldsymbol{\lambda}} \cdot \tilde{\boldsymbol{\lambda}}) . \qquad (29)$$

Since the eigenvalues of a density matrix majorize its diagonal elements, $\boldsymbol{\lambda} \succ \tilde{\boldsymbol{\lambda}}$, and the dot product with an ordered vector is a Schur convex function, we have

$$\boldsymbol{\lambda} \cdot \tilde{\boldsymbol{\lambda}} \geqslant \tilde{\boldsymbol{\lambda}} \cdot \tilde{\boldsymbol{\lambda}} . \qquad (30)$$

Hence from (29) and (30) we obtain the bound

$$\Delta_\lambda \leqslant C_1 , \qquad (31)$$

which corresponds to the bound in (5) for the special case of $q = 1$.

For computational purposes, we use the difference of purities interpretation of C_1 given in (7). The $\mathrm{Tr}(\rho^2)$ term is independent of $U_p(\boldsymbol{\alpha})$. Hence it only needs to be evaluated once, outside of the parameter optimization loop. It can be computed via the expectation value of the swap operator S on two copies of ρ, using the identity

$$\mathrm{Tr}(\rho^2) = \mathrm{Tr}((\rho \otimes \rho)S) . \qquad (32)$$
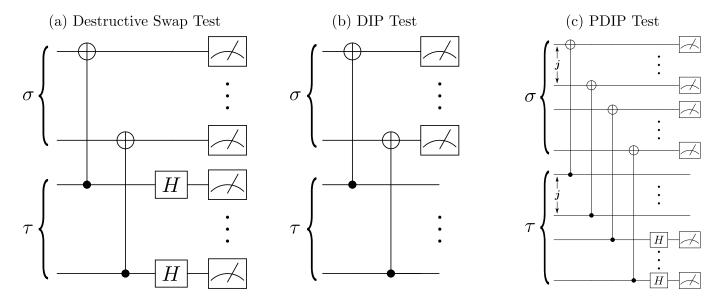
FIG. 5. Diagonalization test circuits used in VQSD. (a) The Destructive Swap Test computes $\mathrm{Tr}(\sigma\tau)$ via a depth-two circuit. (b) The Diagonalized Inner Product (DIP) Test computes $\mathrm{Tr}(\mathcal{Z}(\sigma)\mathcal{Z}(\tau))$ via a depth-one circuit. (c) The Partially Diagonalized Inner Product (PDIP) Test computes $\mathrm{Tr}(\mathcal{Z}_{\boldsymbol{j}}(\sigma)\mathcal{Z}_{\boldsymbol{j}}(\tau))$ via a depth-two circuit, for a particular set of qubits $\boldsymbol{j}$. While the DIP test requires no postprocessing, the postprocessing for the Destructive Swap Test and the Partial DIP Test scales linearly in n.

This expectation value is found with a depth-two quantum circuit that essentially corresponds to a Bell-basis measurement, with classical post-processing that scales linearly in the number of qubits [38, 41]. This is shown in Fig. 5(a). We call this procedure the Destructive Swap Test, since it is like the Swap Test, but the measurement occurs on the original systems instead of on an ancilla.

Similarly, the $\mathrm{Tr}(\mathcal{Z}(\tilde{\rho})^2)$ term could be evaluated by first dephasing $\tilde{\rho}$ and then performing the Destructive Swap Test, which would involve a depth-three quantum circuit with linear classical post-processing. This approach was noted in Ref. [42]. However, there exists a simpler circuit, which we call the Diagonalized Inner Product (DIP) Test. The DIP Test involves a depth-one quantum circuit with no classical post-processing. An abstract version of this circuit is shown in Fig. 5(b), for two states σ and τ. The proof that this circuit computes $\mathrm{Tr}(\mathcal{Z}(\sigma)\mathcal{Z}(\tau))$ is given in Appendix H of SM. For our application we will set $\sigma = \tau = \tilde{\rho}$, for which this circuit gives $\mathrm{Tr}(\mathcal{Z}(\tilde{\rho})^2)$.

In summary, C_1 is efficiently computed by using the Destructive Swap Test for the $\mathrm{Tr}(\rho^2)$ term and the DIP Test for the $\mathrm{Tr}(\mathcal{Z}(\tilde{\rho})^2)$ term.

2. C_2 and the PDIP Test

Like C_1, C_2 can also be rewritten in terms of of the Hilbert-Schmidt distance. Namely, C_2 is the average distance of $\tilde{\rho}$ to each locally-dephased state $\mathcal{Z}_j(\tilde{\rho})$:

$$C_2 = \frac{1}{n}\sum_{j=1}^{n} D_{\mathrm{HS}}(\tilde{\rho}, \mathcal{Z}_j(\tilde{\rho})). \qquad (33)$$

where $\mathcal{Z}_j(\cdot) = \sum_z (|z\rangle\langle z|_j \otimes \mathbb{1}_{k\neq j})(\cdot)(|z\rangle\langle z|_j \otimes \mathbb{1}_{k\neq j})$. Naturally, one would expect that $C_2 \leqslant C_1$, since $\tilde{\rho}$ should be closer to each locally dephased state than to the fully dephased state. Indeed this is true and can be seen from:

$$C_2 = C_1 - \frac{1}{n}\sum_{j=1}^{n} \min_{\sigma \in \mathcal{D}} D_{\mathrm{HS}}(\mathcal{Z}_j(\tilde{\rho}), \sigma). \qquad (34)$$

However, C_1 and C_2 vanish under precisely the same conditions, as noted in Eq. (9). One can see this by noting that C_2 also upper bounds $(1/n)C_1$ and hence we have

$$C_2 \leqslant C_1 \leqslant nC_2. \qquad (35)$$

Combining the upper bound in (35) with the relations in (26) and (31) gives the bounds in (5) with β defined in (11). The upper bound in (35) is proved as follows. Let $\boldsymbol{z} = z_1...z_n$ and $\boldsymbol{z'} = z'_1...z'_n$ be n-dimensional bitstrings. Let $\mathcal{S}$ be the set of all pairs $(\boldsymbol{z}, \boldsymbol{z'})$ such that $\boldsymbol{z} \neq \boldsymbol{z'}$, and let $\mathcal{S}_j$ be the set of all pairs $(\boldsymbol{z}, \boldsymbol{z'})$ such that $z_j \neq z'_j$. Then we have $C_1 = \sum_{(\boldsymbol{z},\boldsymbol{z'})\in\mathcal{S}} |\langle\boldsymbol{z}|\tilde{\rho}|\boldsymbol{z'}\rangle|^2$, and

$$nC_2 = \sum_{j=1}^{n} \sum_{(\boldsymbol{z},\boldsymbol{z'})\in\mathcal{S}_j} |\langle\boldsymbol{z}|\tilde{\rho}|\boldsymbol{z'}\rangle|^2 \qquad (36)$$

$$\geqslant \sum_{(\boldsymbol{z},\boldsymbol{z'})\in\mathcal{S}^U} |\langle\boldsymbol{z}|\tilde{\rho}|\boldsymbol{z'}\rangle|^2 = C_1, \qquad (37)$$

where $\mathcal{S}^U = \bigcup_{j=1}^{n} \mathcal{S}_j$ is the union of all the $\mathcal{S}_j$ sets. The inequality in (37) arises from the fact that the $\mathcal{S}_j$ sets have non-trivial intersection with each other, and hence we throw some terms away when only considering the union $\mathcal{S}^U$. The last equality follows from the fact that

$\mathcal{S}^U = \mathcal{S}$, i.e, the set of all bitstring pairs that differ from each other ($\mathcal{S}$) corresponds to the set of all bitstring pairs that differ for at least one element ($\mathcal{S}^U$).

Writing C_2 in terms of purities, as in (8), shows how it can be computed on a quantum computer. As in the case of C_1, the first term in (8) is computed with the Destructive Swap Test. For the second term in (8), each purity $\text{Tr}(\mathcal{Z}_j(\tilde{\rho})^2)$ could also be evaluated with the Destructive Swap Test, by first locally dephasing the appropriate qubit. However, we present a slightly improved circuit to compute these purities that we call the Partially Diagonalized Inner Product (PDIP) Test. The PDIP Test is shown in Fig. 5(c) for the general case of feeding in two distinct states σ and τ with the goal of computing the inner product between $\mathcal{Z}_j(\sigma)$ and $\mathcal{Z}_j(\tau)$. For generality we let l, with $0 \leqslant l \leqslant n$, denote the number of qubits being locally dephased for this computation. If $l > 0$, we define $\boldsymbol{j} = (j_1, \ldots, j_l)$ as a vector of indices that indicates which qubits are being locally dephased. The PDIP Test is a hybrid of the Destructive Swap Test and the DIP Test, corresponding to the former when $l = 0$ and the latter when $l = n$. Hence, it generalizes both the Destructive Swap Test and the DIP Test. Namely, the PDIP Test performs the DIP Test on the qubits appearing in $\boldsymbol{j}$ and performs the Destructive Swap Test on the qubits not appearing in $\boldsymbol{j}$. The proof that the PDIP Test computes $\text{Tr}(\mathcal{Z}_{\boldsymbol{j}}(\sigma)\mathcal{Z}_{\boldsymbol{j}}(\tau))$, and hence $\text{Tr}(\mathcal{Z}_{\boldsymbol{j}}(\tilde{\rho})^2)$ when $\sigma = \tau = \tilde{\rho}$, is given in Appendix H of SM.

3. C_1 versus C_2

Here we discuss the contrasting merits of the functions C_1 and C_2, hence motivating our cost definition in (10).

As noted previously, C_2 does not have an operational meaning like C_1. In addition, the circuit for computing C_1 is more efficient than that for C_2. The circuit in Fig. 5(b) for computing the second term in C_1 has a gate depth of one, with n CNOT gates, n measurements, and no classical post-processing. The circuit in Fig. 5(c) for computing the second term in C_2 has a gate depth of two, with n CNOT gates, $n-1$ Hadamard gates, $2n-1$ measurements, and classical post-processing whose complexity scales linearly in n. So in every aspect, the circuit for computing C_1 is less complex than that for C_2. This implies that C_1 can be computed with greater accuracy than C_2 on a noisy quantum computer.

On the other hand, consider how the landscape for C_1 and C_2 scale with n. As a simple example, suppose $\rho = |0\rangle\langle 0| \otimes \cdots \otimes |0\rangle\langle 0|$. Suppose one takes a single parameter ansatz for U, such that $U(\theta) = R_X(\theta) \otimes \cdots \otimes R_X(\theta)$, where $R_X(\theta)$ is a rotation about the X-axis of the Bloch sphere by angle θ. For this example,

$$C_1(\theta) = 1 - \text{Tr}(\mathcal{Z}(\tilde{\rho})^2) = 1 - x(\theta)^n \quad (38)$$

where $x(\theta) = \text{Tr}(\mathcal{Z}(R_X(\theta)|0\rangle\langle 0|R_X(\theta)^\dagger)^2) = (1 + \cos^2\theta)/2$. If θ is not an integer multiple of π, then $x(\theta) < 1$, and $x(\theta)^n$ will be exponentially suppressed

for large n. In other words, for large n, the landscape for $x(\theta)^n$ becomes similar to that of a delta function: it is zero for all θ except for multiples of π. Hence, for large n, it becomes difficult to train the unitary $U(\theta)$ because the gradient vanishes for most θ. This is just an illustrative example, but this issue is general. Generally speaking, for large n, the function C_1 has a sharp gradient near its global minima, and the gradient vanishes when one is far away from these minima. Ultimately this limits C_1's utility as a training function for large n.

In contrast, C_2 does not suffer from this issue. For the example in the previous paragraph,

$$C_2(\theta) = 1 - x(\theta), \quad (39)$$

which is independent of n. So for this example the gradient of C_2 does not vanish as n increases, and hence C_2 can be used to train θ. More generally, the landscape of C_2 is less barren than that of C_1 for large n. We can argue this, particularly, for states ρ that have low rank or low entropy. The second term in (8), which is the term that provides the variability with $\boldsymbol{\alpha}$, does not vanish even for large n, since (as shown in Appendix I of SM):

$$\text{Tr}(\mathcal{Z}_j(\tilde{\rho})^2) \geqslant 2^{-H(\rho)-1} \geqslant \frac{1}{2r}. \quad (40)$$

Here, $H(\rho) = -\text{Tr}(\rho \log_2 \rho)$ is the von Neumann entropy, and r is the rank of ρ. So as long as ρ is low entropy or low rank, then the second term in C_2 will not vanish. Note that a similar bound does not exist for second term in C_1, which does tend to vanish for large n.

B. Optimization Methods

Finding $\boldsymbol{\alpha}_{\text{opt}}$ in (6) is a major component of VQSD. While many works have benchmarked classical optimization algorithms (e.g., Ref. [43]), the particular case of optimization for variational hybrid algorithms [44] is limited and needs further work [45]. Both gradient-based and gradient-free methods are possible, but gradient-based methods may not work as well with noisy data. Additionally, Ref. [27] notes that gradients of a large class of circuit ansatze vanish when the number of parameters becomes large. These and other issues (e.g., sensitivity to initial conditions, number of function evaluations) should be considered when choosing an optimization method.

In our preliminary numerical analyses (see Appendix E in SM), we found that the Powell optimization algorithm [46] performed the best on both quantum computer and simulator implementations of VQSD. This derivative-free algorithm uses a bi-directional search along each parameter using Brent's method. Our studies showed that Powell's method performed the best in terms of convergence, sensitivity to initial conditions, and number of correct solutions found. The implementation of Powell's algorithm used in this paper can be found in the open-source Python package SciPy Optimize [47]. Finally, Appendix C of SM shows how our layered ansatz

for $U_p(\boldsymbol{\alpha})$ as well as proper initialization of $U_p(\boldsymbol{\alpha})$ helps in mitigating the problem of local minima.

DATA AVAILABILITY

Data generated and analyzed during current study are available from the corresponding author upon reasonable request.

CODE AVAILABILITY

The code used to generate some of the examples presented here and in Supplementary Material can be accessed from [33].

ACKNOWLEDGMENTS

AUTHOR CONTRIBUTION

RL, AT and LC implemented the algorithms and performed numerical analysis. LC and PJC designed the project. PJC proposed the cost function and proved the analytical results. RL, AT, ÉO-J, LC, and PJC contributed to data analysis, as well as writing and editing the final manuscript.

COMPETING INTERESTS

The authors declare no competing interests.

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

[2] S. Lloyd, M. Mohseni, and P. Rebentrost, "Quantum principal component analysis," Nature Physics **10**, 631–633.

[3] A. W. Harrow, A. Hassidim, and S. Lloyd, "Quantum algorithm for linear systems of equations," Physical Review Letters **103**, 150502 (2009).

[4] P. Rebentrost, A. Steffens, I. Marvian, and S. Lloyd, "Quantum singular-value decomposition of nonsparse low-rank matrices," Physical Review A **97**, 012327 (2018).

[5] S. K. Leyton and T. J. Osborne, "A quantum algorithm to solve nonlinear differential equations," arXiv:0812.4423 (2008).

[6] D. W. Berry, "High-order quantum algorithm for solving linear differential equations," Journal of Physics A: Mathematical and Theoretical **47**, 105301 (2014).

[7] E. Farhi, J. Goldstone, and S. Gutmann, "A quantum approximate optimization algorithm," arXiv:1411.4028 (2014).

[8] A. Peruzzo, J. McClean, P. Shadbolt, M.-H. Yung, X.-Q. Zhou, P. J. Love, A. Aspuru-Guzik, and J. L. O'Brien, "A variational eigenvalue solver on a photonic quantum processor," Nature Communications **5**, 4213 (2014).

[9] A. Kandala, A. Mezzacapo, K. Temme, M. Takita, M. Brink, J. M. Chow, and J. M. Gambetta, "Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets," Nature **549**, 242 (2017).

[10] D. W. Berry, A. M. Childs, R. Cleve, R. Kothari, and R. D. Somma, "Simulating Hamiltonian dynamics with a truncated taylor series," Physical Review Letters **114**, 090502 (2015).

[11] J. Preskill, "Quantum computing and the entanglement frontier," arXiv:1203.5813 (2012).

[12] A. W. Harrow and A. Montanaro, "Quantum computational supremacy," Nature **549**, 203 (2017).

[13] S. Bravyi, G. Smith, and J. A. Smolin, "Trading classical and quantum computational resources," Physical Review X **6**, 021043 (2016).

[14] O. Higgott, D. Wang, and S. Brierley, "Variational Quantum Computation of Excited States," arXiv:1805.08138 (2018).

[15] T. Jones, S. Endo, S. McArdle, X. Yuan, and S. C. Benjamin, "Variational quantum algorithms for discovering Hamiltonian spectra," Physical Review A **99**, 062304 (2019).

[16] P. D. Johnson, J. Romero, J. Olson, Y. Cao, and A. Aspuru-Guzik, "QVECTOR: an algorithm for device-tailored quantum error correction," arXiv:1711.02249 (2017).

[17] J. Romero, J. P. Olson, and A. Aspuru-Guzik, "Quantum autoencoders for efficient compression of quantum data," Quantum Science and Technology **2**, 045001 (2017).

[18] A. Khoshaman, W. Vinci, B. Denis, E. Andriyash, and M. H. Amin, "Quantum variational autoencoder," Quantum Science and Technology **4**, 014001 (2018).

[19] Y. Li and S. C. Benjamin, "Efficient variational quantum simulator incorporating active error minimization," Physical Review X **7**, 021050 (2017).

[20] C. Kokail, C. Maier, R. van Bijnen, T. Brydges, M. K. Joshi, P. Jurcevic, C. A. Muschik, P. Silvi, R. Blatt, C. F. Roos, and P. Zoller, "Self-verifying variational quantum simulation of the lattice Schwinger model," Nature **569**, 355 (2019).

[21] S. Khatri, R. LaRose, A. Poremba, L. Cincio, A. T. Sornborger, and P. J. Coles, "Quantum-assisted quantum compiling," Quantum **3**, 140 (2019).

[22] H. Li and F. D. M. Haldane, "Entanglement spectrum as a generalization of entanglement entropy: Identification of topological order in non-abelian fractional quantum Hall effect states," Physical Review Letters **101**, 010504 (2008).

[23] V. Giovannetti, S. Lloyd, and L. Maccone, "Quantum random access memory," Physical Review Letters **100**, 160501 (2008).

[24] K. Pearson, "On lines and planes of closest fit to systems of points in space," The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science **2**, 559–572 (1901).

[25] L. N. Trefethen and D. Bau, *Numerical Linear Algebra* (SIAM, Philadelphia, PA, 1997).

[26] This can be seen by noting that computing eigenvalues is equivalent to computing roots of a polynomial equation (namely the characteristic polynomial of the matrix) and that no closed-form solution exists for the roots of general polynomials of degree greater than or equal to five [25].

[27] J. R. McClean, S. Boixo, V. N. Smelyanskiy, R. Babbush, and H. Neven, "Barren plateaus in quantum neural network training landscapes," Nature Communications **9**, 4812 (2018).

[28] E. Grant, L. Wossnig, M. Ostaszewski, and M. Benedetti, "An initialization strategy for addressing barren plateaus in parametrized quantum circuits," arXiv:1903.05076 (2019).

[29] F. Vatan and C. Williams, "Optimal quantum circuits for general two-qubit gates," Physical Review A **69**, 032315 (2004).

[30] T. Baumgratz, M. Cramer, and M. B. Plenio, "Quantifying coherence," Physical Review Letters **113**, 140401 (2014).

[31] H. Buhrman, R. Cleve, J. Watrous, and R. De Wolf, "Quantum fingerprinting," Physical Review Letters **87**, 167902 (2001).

[32] D. Gottesman and I. Chuang, "Quantum digital signatures," quant-ph/0105032 (2001).

[33] "VQSD source code," https://github.com/rmlarose/vqsd.

[34] R. S. Smith, M. J. Curtis, and W. J. Zeng, "A Practical Quantum Instruction Set Architecture," arXiv:1608.03355 (2016).

[35] M. B. Hastings, "An area law for one-dimensional quantum systems," Journal of Statistical Mechanics: Theory and Experiment **2007**, 08024 (2007).

[36] B. Bauer and C. Nayak, "Area laws in a many-body localized state and its implications for topological order," Journal of Statistical Mechanics: Theory and Experiment **2013**, 09005 (2013).

[37] T. Grover, "Certain General Constraints on the Many-Body Localization Transition," arXiv:1405.1471 (2014).

[38] L. Cincio, Y. Subaşı, A. T. Sornborger, and P. J. Coles, "Learning the quantum algorithm for state overlap," New Journal of Physics **20**, 113022 (2018).

[39] T. Jones and S. C. Benjamin, "Quantum compilation and circuit optimisation via energy dissipation," arXiv:1811.03147 (2018).

[40] E. Tang, "Quantum-inspired classical algorithms for principal component analysis and supervised clustering," arXiv:1811.00414 (2018).

[41] J. C. Garcia-Escartin and P. Chamorro-Posada, "Swap test and Hong-Ou-Mandel effect are equivalent," Physical Review A **87**, 052330 (2013).

[42] G. Smith, J. A. Smolin, X. Yuan, Q. Zhao, D. Girolami, and X. Ma, "Quantifying coherence and entanglement via simple measurements," arXiv:1707.09928 (2017).

[43] L. M. Rios and N. V. Sahinidis, "Derivative-free optimization: a review of algorithms and comparison of software implementations," Journal of Global Optimization **56**, 1247–1293 (2013).

[44] G. G. Guerreschi and M. Smelyanskiy, "Practical optimization for hybrid quantum-classical algorithms," arXiv:1701.01450 (2017).

[45] J. R. McClean, J. Romero, R. Babbush, and A. Aspuru-Guzik, "The theory of variational hybrid quantum-classical algorithms," New Journal of Physics **18**, 023023 (2016).

[46] M. J. D. Powell, "A fast algorithm for nonlinearly constrained optimization calculations," in *Numerical Analysis*, Lecture Notes in Mathematics, edited by G. A.Editor Watson (Springer Berlin Heidelberg, 1978) pp. 144–157.

[47] "Scipy optimization and root finding," https://docs.scipy.org/doc/scipy/reference/optimize.html (2018).

[48] M. J. D. Powell, "Direct search algorithms for optimization calculations," Acta Numerica **7**, 287–336 (1998).

[49] M. J. D. Powell, "The BOBYQA algorithm for bound constrained optimization without derivatives," (2009).

[50] F. Gao and L. Han, "Implementing the Nelder-Mead simplex algorithm with adaptive parameters," Computational Optimization and Applications **51**, 259–277 (2012).

[51] J. Nocedal and S. Wright, *Numerical Optimization*, 2nd ed., Springer Series in Operations Research and Financial Engineering (Springer-Verlag, 2006).

[52] C. Cartis, J. Fiala, B. Marteau, and L. Roberts, "Improving the Flexibility and Robustness of Model-Based Derivative-Free Optimization Solvers," arXiv:1804.00154 (2018).

Supplementary Material for
"Variational Quantum State Diagonalization"

Appendix A: Details on VQSD Implementations

Here we provide further details on our implementations of VQSD in Sec. II B. This includes further details about the optimization parameters as well as additional statistics for our runs on ns on the quantum computer.

1. Optimization Parameters

First, we discuss our implementation on a quantum computer (data shown in Fig. 3). Figure S.1 displays the circuit used for this implementation. This circuit is logically divided into three sections. First, we prepare two copies of the plus state $\rho = |+\rangle\langle+| = H|0\rangle\langle0|H$ by doing a Hadamard gate H on each qubit. Next, we implement one layer of a unitary ansatz, namely $U(\theta) = R_x(\pi/2)R_z(\theta)$. This ansatz was chosen because each gate can be natively implemented on Rigetti's quantum computer. To simplify the search space, we restricted to one parameter instead of a universal one-qubit unitary. Last, we implement the DIP Test circuit, described in Fig. 5, which here consists of only one CNOT gate and one measurement.

For the parameter optimization loop, we used the Powell algorithm mentioned in Sec. IV B. This algorithm found the minimum cost in less than ten objective function evaluations on average. Each objective function evaluation (i.e., call to the quantum computer) sampled from 10,000 runs of the circuit in Fig. S.1. As can be seen in Fig. 3(b), 10,000 runs was sufficient to accurately estimate the cost function (10) with small variance. Because the problem size was small, we took $q = 1$ in (10), which provided adequate variability in the cost landscape.

Because of the noise levels in current quantum computers, we limited VQSD implementations on quantum hardware to only one-qubit states. Noise affects the computation in multiple areas. For example, in state preparation, qubit-specific errors can cause the two copies of ρ to actually be different states. Subsequent gate errors (notably two-qubit gates), decoherence, and measurement errors prevent the cost from reaching zero even though the optimal value of θ is obtained. The effect of these various noise sources, and in particular the effect of discrepancies in preparation of two copies of ρ, will be important to study in future work.

Next, we discuss our VQSD implementation on a simulator (data shown in Fig. 4). For this implementation we again chose $q = 1$ in our cost function. Because of the larger problem size (diagonalizing a 4-qubit state), we employed multiple layers in our ansatz, up to $p = 5$. The simulator directly calculated the measurement probability distribution in the DIP Test, as opposed to determining the desired probability via sampling. This al-
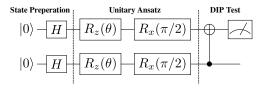

FIG. S.1. Circuit used to implement VQSD for $\rho = |+\rangle\langle+|$ on Rigetti's 8Q-Agave quantum computer. Vertical dashed lines separate the circuit into logical components.

lowed us to use a gradient-based method to optimize our cost function, reducing the overall runtime of the optimization. Hence, our simulator implementation for the Heisenberg model demonstrated a future application of VQSD while alleviating the optimization bottleneck that is present for all variational quantum algorithms on large problem sizes, an area that needs further research [45]. We explore optimization methods further in Appendix E.

2. Additional statistics for the Quantum Computer Implementation

Here, we present statistics for several runs of the VQSD implementation run on Rigetti's 8Q-Agave quantum computer. One example plot of cost vs. iteration for diagonalizing the plus state $\rho = |+\rangle\langle+|$ is shown in Figure 3(a). Here, we present all data collected for this implementation of VQSD, shown in Figure S.2. The following table displays the final costs achieved as well the associated inferred eigenvalues.

VQSD Run	$\min(C)$	$\min(\widetilde{\lambda_{\boldsymbol{z}}^{\text{est}}})$	$\max(\widetilde{\lambda_{\boldsymbol{z}}^{\text{est}}})$
1	0.107	0.000	1.000
2	0.090	0.142	0.858
3	0.099	0.054	0.946
4	0.120	0.079	0.921
5	0.080	0.061	0.939
6	0.090	0.210	0.790
7	0.65	0.001	0.999
Avg.	0.093	**0.078**	**0.922**
Std.	0.016	0.070	0.070

TABLE I. Minimum cost and eigenvalues achieved after performing the parameter optimization loop for seven independent runs of VQSD for the example discussed in Sec. II B. The final two rows show average values and standard deviation across all runs.
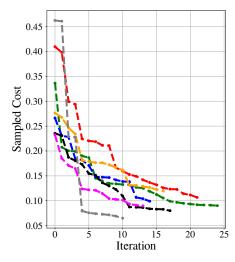
FIG. S.2. Cost vs iteration for all attempts of VQSD on Rigetti's 8Q-Agave computer for diagonalizing the plus state $\rho = |+\rangle\langle+|$. Each of the seven curves represents a different independent run. Each run starts at a random initial angle and uses the Powell optimization algorithm to minimize the cost.
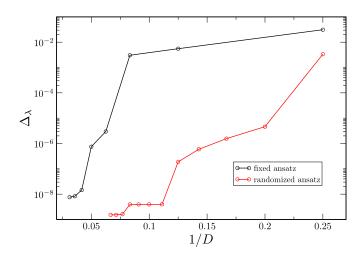

FIG. S.3. Comparison of two approaches to obtaining the diagonalizing unitary $U(\boldsymbol{\alpha})$: (i) based on a fixed layered ansatz shown in Fig. 2 in the main text (black line) and (ii) based on random updates to the structure of $U(\boldsymbol{\alpha})$ (red line). The plot shows eigenvalue error Δ_λ versus $1/D$, where D is the number of gates in $U(\boldsymbol{\alpha})$. For the same D, the second approach found a more optimal gate sequence.

Appendix B: Alternative Ansatz and the Heisenberg Model Ground State

In this Appendix, we describe a modification of the layered ansatz discussed in Section II A. Figure 2 in the main text shows an example of a layered ansatz in which every layer has the same, fixed structure consisting of alternating two-qubit gates acting on nearest-neighbor qubits. The modified approach presented here may be useful in situations where there is no natural choice of the structure of the layered ansatz.

Here, instead of working with a fixed structure for the diagonalizing unitary $U(\boldsymbol{\alpha})$, we allow it to vary during the optimization process. The algorithm used to update the structure of $U(\boldsymbol{\alpha})$ is probabilistic and resembles the one presented in [38].

In the examples studied here, the initial $U(\boldsymbol{\alpha})$ consists of a small number of random two-qubit gates with random supports (i.e. the qubits on which a gate acts). An optimization step involves minimizing the cost function by changing parameters $\boldsymbol{\alpha}$ as well as a small random change to the structure of $U(\boldsymbol{\alpha})$. This change to the structure typically amounts to a random modification of support for a limited number of gates. The new structure is accepted or rejected following the usual simulated annealing schemes. We refer the reader to Section II D of [38] for further details on the optimization method.

The gate sequence representing $U(\boldsymbol{\alpha})$ is allowed to grow. If the algorithm described above cannot minimize the cost function for a specified number of iterations, an identity gate (spanned by new variational parameters) is randomly added to $U(\boldsymbol{\alpha})$. This step is similar in spirit to adding a layer to $U(\boldsymbol{\alpha})$ as discussed in Section II A of the main text.

We compared the current method with the one based on the layered ansatz and found that it produced diagonalizing circuits involving significantly fewer gates. Figure S.3 shows the eigenvalue error Δ_λ, defined in Eq. (3), as a function of $1/D$, where D is the total number of gates of $U(\boldsymbol{\alpha})$. Here, VQSD is used to diagonalize a 4-qubit reduced state of the ground state of the one-dimensional Heisenberg model defined on 8 qubits, see Eq. (19). For every number of gates D, the current algorithm outperforms the one based on the fixed, layered ansatz. It finds a sequence of gates that results in a smaller eigenvalue error Δ_λ.

Finally, we use the current optimization approach to find the spectrum of a 6-qubit reduced state ρ of the 12-qubit ground state of a one-dimensional Heisenberg model. The results of performing VQSD on ρ are shown in Fig. S.4. Panel (a) shows the convergence of the 11 largest inferred eigenvalues $\tilde{\lambda}_j$ of ρ to their exact values. We can see that the quality of the inferred eigenvalues increases quickly with the number of gates D used in the diagonalizing unitary $U(\boldsymbol{\alpha})$. In panel (b), we show the dominant part of the spectrum of ρ resolved in the z-component of the total spin. The results show that VQSD could be used to accurately obtain the dominant part of the spectrum of the density matrix together with the associated quantum numbers.

Appendix C: Optimization and local minima

In this Appendix we describe a strategy to avoid local minima that is used in the optimization algorithms throughout the paper and detailed in Appendix B. We
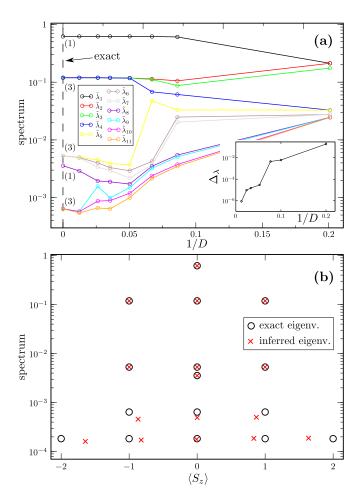
FIG. S.4. VQSD applied to the ground state of the Heisenberg model. Here we consider a 6-qubit reduced state ρ of the 12-qubit ground state. **(a)** Largest inferred eigenvalues $\tilde{\lambda}_j$ of ρ as a function of $1/D$, where D is the total number of gates in the diagonalizing unitary $U(\boldsymbol{\alpha})$. The inferred eigenvalues converge to their exact values shown along the $1/D = 0$ line recovering the correct degeneracy. Inset: Eigenvalue error Δ_λ as a function of $1/D$. **(b)** The largest inferred eigenvalues $\tilde{\lambda}_j$ of ρ resolved in the $\langle S_z \rangle$ quantum number. We find very good agreement between the inferred eigenvalues (red crosses) and the exact ones (black circles), especially for large eigenvalues. The data was obtained for $D = 150$ gates.

adapt the optimization involved in the diagonalization of the 6-qubit density matrix described in Appendix B as an illustrative example.

We note that the classical optimization problem associated with VQSD is potentially very difficult one. In the example studied in Appendix B the diagonalizing unitary consisted of 150 two-qubit gates. This means that in order to find that unitary one has to optimize over at least $150 \cdot 13$ continuous parameters (every two-qubit gate is spanned by 15 parameters, but there is some reduction in the total number of parameters when two consecutive gates have overlapping supports). Initiated randomly, off-the-shelf techniques will most likely return suboptimal solution due to the presence of multiple local minima
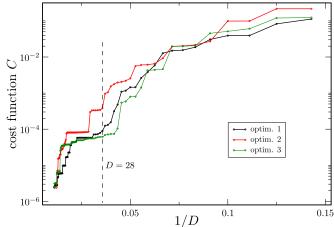
and the rough cost function landscape.

Let $U_D(\boldsymbol{\alpha})$ denote a diagonalizing unitary that is built by D two-qubit gates parametrized by $\boldsymbol{\alpha}$. Our optimization method begins with a shallow circuit consisting of few gates only. Since there is only a small number of variational parameters, the local minimum is quickly attained. After this initial step, the circuit that implements the unitary $U_D(\boldsymbol{\alpha})$ is grown by adding an identity gate (either randomly as discussed in this Appendix or by means of a layer of identity gates as presented in the main text). This additional gate contains new variational parameters that are initiated such that the unitary $U_D(\boldsymbol{\alpha}) = U_{D+1}(\boldsymbol{\alpha})$ and hence the value of the cost function are not changed. After the gate was added, the unitary $U_{D+1}(\boldsymbol{\alpha})$ contains more variational parameters which allows for further minimization of the cost function. In summary, the optimization of a deeper circuit $U_{D+1}(\boldsymbol{\alpha})$ is initialized by previously obtained $U_D(\boldsymbol{\alpha})$ as opposed to random initialization. What is more, even if the unitary $U_D(\boldsymbol{\alpha})$ was not the most optimal one for a given D, the growth of the circuit allows the algorithm to escape the local minimum and eventually find the global one, as illustrated by an example below and shown in Fig. S.5. For a similar discussion, see [28].

To clarify the above analysis, let us consider an example of diagonalizing a 6-qubit reduced state of the 12-qubit ground state of the Heisenberg model, see Appendix B for comparison. Figure S.5 shows the value of the cost function C as a function of $1/D$ for three independent optimization runs. Each optimization was initialized randomly and we applied the same optimization scheme described above to each of them. We see

that despite getting stuck in local minima, every optimization run managed to minimize the cost function to the predefined small value (which was set to $2 \cdot 10^{-6}$ in this example). For instance, at $D = 28$, optimization run no. 2 clearly returns suboptimal solution (optimization run no. 3 gives lower cost function by a factor of 6) but after adding several identity gates, it manages to escape the local minimum and continue towards the global one.

Appendix D: Optimization runs with various q values

In this Appendix we present some numerical results for training our overall cost function for various values of q. Recall from Eq. (10) that q is the weighting parameter that weights the contributions of C_1 and C_2 in the overall cost, as follows:

$$C(U_p(\boldsymbol{\alpha})) = qC_1(U_p(\boldsymbol{\alpha})) + (1-q)C_2(U_p(\boldsymbol{\alpha})), \quad \text{(D1)}$$

where

$$C_1(U_p(\boldsymbol{\alpha})) = \mathrm{Tr}(\rho^2) - \mathrm{Tr}(\mathcal{Z}(\tilde{\rho})^2), \quad \text{(D2)}$$

$$C_2(U_p(\boldsymbol{\alpha})) = \mathrm{Tr}(\rho^2) - \frac{1}{n}\sum_{j=1}^{n} \mathrm{Tr}(\mathcal{Z}_j(\tilde{\rho})^2). \quad \text{(D3)}$$

As argued in Section II, C_1 is operationally meaningful, while C_2 has a landscape that is more amendable to training when n is large. In particular, one expects that for large n, the gradient of C_1 is sharp near the global minima but vanishes exponentially in n away from these minima. In contrast, the gradient of C_2 is not expected to exponentially vanish as n increases, even away from the minima.

Here, we numerically study the performance for different q values for a simple example where ρ is a tensor product of qubit pure states. Namely, we choose $\rho = \bigotimes_{j=1}^{n} V_j|0\rangle\langle 0|V_j^\dagger$, where $V_j = R_X(\theta_j)$ with θ_j randomly chosen. Such tensor product states are diagonalizable by a single layer ansatz: $U(\boldsymbol{\alpha}) = \bigotimes_{j=1}^{n} R_X(\alpha_j)$. We consider three different problem sizes: $n = 6$, 8, and 10. Figure S.6 shows our numerical results.

Directly training the C_1 cost (corresponding to $q = 1$) sometimes fails to find the global minimum. One can see this in Fig. S.6, where the green curve fails to fully reach zero cost. In contrast, the red and blue curves in Fig. S.6, which correspond to $q = 0.5$ and $q = 0$ respectively, approximately go to zero for large iterations.

Even more interesting are the purple and yellow curves, which respectively correspond to evaluating the C_1 cost at the angles $\boldsymbol{\alpha}$ obtained from training the $q = 0.5$ and $q = 0$ costs. It is remarkable that both the purple and yellow curves perform better (i.e., achieve lower values) than the green curve. This implies that one can indirectly train the C_1 cost by training the $q = 0.5$ or $q = 0$ costs, and this indirect training performs better than directly training C_1. Since C_1 is operationally meaningful, this indirect training with $q < 1$ is performing better in an operationally meaningful way.

We expect that direct training of C_1 will perform worse as n increases, due to the exponential vanishing of the gradient of C_1. The particular runs shown in Fig. S.6 do not show this trend, although this can be explained by the fact that the gradient of C_1 depends significantly on the initial values of the $\boldsymbol{\alpha}$, and indeed we saw large variability in the performance of the green curve even for a fixed n. Nevertheless, it is worth noting that we were always able to directly train C_1 (i.e., to make the green curve go to zero) for $n < 6$, which is consistent with our expectations.

Overall, Fig. S.6 provides numerical justification of the definition of our cost function as a weighted average, as in Eq. (10). Namely, it shows that there is an advantage to choosing $q < 1$.

Appendix E: Comparison of Optimization Methods

As emphasized previously, numerical optimization plays a key role in all variational hybrid algorithms, and further research in optimization methods is needed. In VQSD, the accuracy of the inferred eigenvalues are closely tied to the performance of the optimization algorithm used in the parameter optimization loop. This issue becomes increasingly important as one goes to large problem sizes (large n), where the number of parameters in the diagonalizing unitary becomes large.

Here, we compare the performance of six different optimization algorithms when used inside the parameter optimization loop of VQSD. These include Powell's algorithm [46], Constrained Optimization BY Linear Approximation (COBYLA) [48], Bound Optimization BY Quadratic Approximation (BOBYQA) [49], Nelder-Mead [50], Broyden-Fletcher-Goldfarb-Shanno (BFGS) [51], and conjugate gradient (CG) [51]. As mentioned in the main text, Powell's algorithm is a derivative-free optimizer that uses a bi-directional search along each parameter. The COBYLA and BOBYQA algorithms are both *trust region* or *restricted-step* methods, which approximate the objective function by a model function. The region where this model function is a good approximation is known as the trust region, and at each step the optimizer attempts to expand the trust region. The Nelder-Mead algorithm is a simplex method useful for derivative-free optimization with smooth objective functions. Lastly, the BFGS and CG algorithms are both gradient-based. The BFGS method is a quasi-Newton method that uses first derivatives only, and the CG method uses a nonlinear conjugate gradient descent. The implementations used in our study can be found in the open-source Python package SciPy Optimize [47] and in Ref. [52].

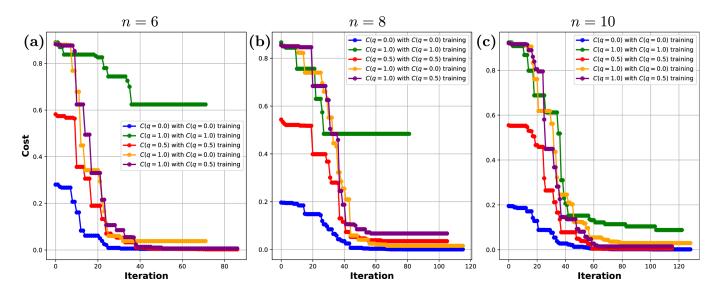For this study, we take the input state ρ to be a six-

FIG. S.6. Cost versus iteration for different values of q, when ρ is a tensor product of pure states on n qubits. Here we consider (a) $n = 6$, (b) $n = 8$, and (a) $n = 10$. We employed the COBYLA optimization method for training (see Appendix E for discussion of this method). For each call to the quantum simulator (i.e., classical simulator of a quantum computer), we took 500 shots for statistics. The green, red, and blue curves respectively correspond to directly training the cost with $q = 1$, $q = 0.5$, and $q = 0$. The purple and yellow curves respectively correspond to evaluating the $q = 1$ cost for the angles $\boldsymbol{\alpha}$ obtained by training the $q = 0.5$ and $q = 0$ costs.

qubit pure product state:

$$\rho = \bigotimes_{j=1}^{6} |\psi_j\rangle\langle\psi_j|, \quad \text{where} \quad |\psi_j\rangle = V_j|0\rangle. \quad (E1)$$

Here, the state preparation unitary is

$$V_j = R_x(\alpha_x^{(j)})R_y(\alpha_y^{(j)})R_z(\alpha_z^{(j)}) \quad (E2)$$

where the angles $(\alpha_x^{(j)}, \alpha_y^{(j)}, \alpha_z^{(j)})$ are randomly chosen.

Using each algorithm, we attempt to minimize the cost by adjusting 36 parameters in one layer of the unitary ansatz in Fig. 2. For fairness of comparison, only the objective function and initial starting point were input to each algorithm, i.e., no special options such as constraints, bounds, or other information was provided. The results of this study are shown in Fig. S.7 and Table II.

Figure S.7 shows cost versus iteration for each of the six algorithms. Here, we define one iteration by a call to the objective function in which the cost decreases. In particular, the number of iterations is different than the number of cost function evaluations (see Table II), which is not set a priori but rather determined by the optimizer. Plotting cost per each function evaluation would essentially produce a noisy curve since the optimizer is trying many values for the parameters. Instead, we only plot the cost for each parameter update in which the cost decreases. Both the number of iterations, function evaluations, and overall runtime are important features of the optimizer.

In this study, as well as others, we found that the Powell optimization algorithm provides the best performance in terms of lowest minimum cost achieved, sensitivity to

Alg.	Powell	COBYLA	BOBYQA	Nelder-Mead	BFGS	CG
r.r.	13.20	1	2.32	23.65	3.83	2.89
f.ev.	4474	341	518	7212	1016	1045

TABLE II. Relative average run-times (r.r.) and absolute number of function evaluations (f.ev.) of each optimization algorithm (Alg.) used for the data obtained in Fig. S.7. For example, BOBYQA took 2.32 times as long to run on average than COBYLA, which took the least time to run out of all algorithms. Absolute run-times depend on a variety of factors and computer performance. For reference, the COBYLA algorithm takes approximately one minute for this problem on a laptop computer. The number of cost function evaluations used (related to run-time but also dependent on the method used by the optimizer) is shown in the second row.

initial conditions, and fraction of correct solutions found. The trust-region algorithms COBYLA and BOBYQA were the next best methods. In particular, although the Powell algorithm consistently obtained lower minimum costs, the COBYLA method ran thirteen times faster on average (see Table II). Indeed, both trust region methods provided the shortest runtime. The gradient-based methods BFGS and CG had comparable run-times but were unable to find any minima. Similarly, the Nelder-Mead simplex algorithm was unable to find any minima. This method also had the longest average run-time of all algorithms tested.

This preliminary analysis suggests that the Powell algorithm is the best method for VQSD. For other variational quantum algorithms, this may not necessarily be the case. In particular, we emphasize that the op-
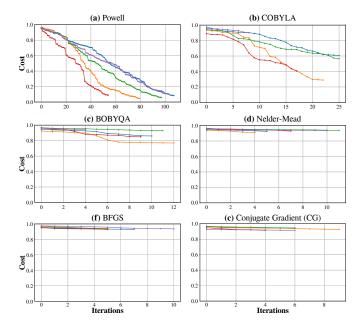
FIG. S.7. Optimization tests on six-qubit product states in the VQSD algorithm. Each plot shows a different optimization algorithm (described in main text) and curves on each plot show optimization attempts with different (random) initial conditions. Cost refers to the C_1 cost function ($q = 1$ in (10)), and each iteration is defined by a decrease in the cost function. As can be seen, the Powell algorithm is the most robust to initial conditions and provides the largest number of solved problem instances.

timization landscape is determined by both the unitary ansatz and the cost function definition, which may vary drastically in different algorithms. While we found that gradient-based methods did not perform well for VQSD, they may work well for other applications. Additionally, optimizers that we have not considered here may also provide better performance. We leave these questions to further work.

Appendix F: Complexity for particular examples

1. General Complexity Remarks

In what follows we discuss some simple examples of states to which one might apply VQSD. There are several aspects of complexity to keep in mind when considering these examples, including:

(C1) The gate complexity of the unitary that diagonalizes ρ. (It is worth remarking that approximate diagonalization might be achieved with a less complex unitary than exact diagonalization.)

(C2) The complexity of searching through the search space to find the diagonalizing unitary.

(C3) The statistical complexity associated with reading out the eigenvalues.

Naturally, (C1) is related to (C2). However, being effi-

cient with respect to (C1) does not guarantee that (C2) is efficient.

2. Example States

In the simplest case, suppose $\rho = |\psi_1\rangle\langle\psi_1| \otimes \cdots \otimes |\psi_n\rangle\langle\psi_n|$ is a tensor product of pure states. This state can be diagonalized by a depth-one circuit $U = U_1 \otimes \cdots \otimes U_n$ composed of n one-qubit gates (all done in parallel). Each U_j diagonalizes the associated $|\psi_j\rangle\langle\psi_j|$ state. Searching for this unitary within our ansatz can be done by setting $p = 1$, i.e., with a single layer L_1 shown in Fig. 2. A single layer is enough to find the unitary that exactly diagonalizes ρ in this case. Hence, for this example, both complexities (C1) and (C2) are efficient. Finally, note that the eigenvalue readout, (C3), is efficient because there is only one non-zero eigenvalue. Hence, $\tilde{\lambda}_{\boldsymbol{z}}^{\text{est}} \approx 1$ and $\epsilon_{\boldsymbol{z}} \approx 1/\sqrt{N_{\text{readout}}}$ for this eigenvalue. This implies that N_{readout} can be chosen to be constant, independent of n, in order to accurately characterize this eigenvalue.

A generalization of product states are classically correlated states, which have the form

$$\rho = \sum_{\boldsymbol{z}} p_{\boldsymbol{z}} |b_{z_1}^{(1)}\rangle\langle b_{z_1}^{(1)}| \otimes \cdots \otimes |b_{z_n}^{(n)}\rangle\langle b_{z_n}^{(n)}| \qquad \text{(F1)}$$

where $\{|b_0^{(j)}\rangle, |b_1^{(j)}\rangle\}$ form an orthonormal basis for qubit j. Like product states, classically correlated states can be diagonalized with a depth-one circuit composed of one-body unitaries. Hence (C1) and (C2) are efficient for such states. However, the complexity of eigenvalue readout depends on the $\{p_{\boldsymbol{z}}\}$ distribution; if it is high entropy then eigenvalue readout can scale exponentially.

Finally, we consider pure states of the form $\rho = |\psi\rangle\langle\psi|$. For such states, eigenvalue readout (C3) is efficient because N_{readout} can be chosen to be independent of n, as we noted earlier for the example of pure product states.

Next we argue that the gate complexity of the diagonalizing unitary, (C1), is efficient. The argument is simply that VQSD takes the state ρ as its input, and ρ must have been prepared on a quantum computer. Let V be the unitary that was used to prepare $|\psi\rangle = V|\mathbf{0}\rangle$ on the quantum computer. For large n, V must have been efficient to implement, otherwise the state $|\psi\rangle$ could not have been prepared. Note that $V^\dagger$, which is constructed from V by reversing the order of the gates and adjointing each gate, can be used to diagonalize ρ. Because V is efficiently implementable, then $V^\dagger$ is also efficiently implementable. Hence, ρ can be efficiently diagonalized. A subtlety is that one must compile $V^\dagger$ into one's ansatz, such as the ansatz in Fig. 2. Fortunately, the overhead needed to compile $V^\dagger$ into our ansatz grows (at worst) only linearly in n. An explicit construction for compiling $V^\dagger$ into our ansatz is as follows. Any one-qubit gate directly translates without overhead into our ansatz, while any two-qubit gate can be compiled using a linear number of swap gates to make the qubits of interest to be nearest

neighbors, then performing the desired two-qubit gate, and finally using a linear number of swap gates to move the qubits back to their original positions.

Let us now consider the complexity (C2) of searching for U. Since there are a linear number of parameters in each layer, and p needs only to grow polynomially in n, then the total number of parameters grows only polynomially in n. But this does not guarantee that we can efficiently minimize the cost function, since the landscape is non-convex. In general, search complexity for problems such as this remains an open problem. Hence, we cannot make a general statement about (C2) for pure states.

Appendix G: Implementation of qPCA

In the main text we compared VQSD to the qPCA algorithm. Here we give further details on our implementation of qPCA. Let us first give an overview of qPCA.

1. Overview of qPCA

The qPCA algorithm exploits two primitives: quantum phase estimation and density matrix exponentiation. Combining these two primitives allows one to estimate the eigenvalues and prepare the eigenvectors of a state ρ.

Density matrix exponentiation refers to generating the unitary $V(t) = e^{-i\rho t}$ for a given state ρ and arbitrary time t. For qPCA, one actually needs to apply the controlled-$V(t)$ gate ($C_{V(t)}$). Namely, in qPCA, the $C_{V(t)}$ gate must be applied for a set of times, $\{t, 2t, 2^2 t, ..., 2^x t\}$, as part of the phase-estimation algorithm. Here we define $t_{\max} := 2^x t$.

Ref. [2] noted that $V(t)$ can be approximated with a sequence of k exponential swap operations between a target state σ and k copies of ρ. That is, let S_{JK} be the swap operator between systems J and K, and let σ and $\rho^{\otimes k}$ be states on systems A and $B = B_1...B_r$, respectively. Then one performs the transformation

$$\tau_{AB} = \sigma \otimes (\rho^{\otimes k}) \quad \rightarrow \quad \tau'_{AB} = W(\sigma \otimes (\rho^{\otimes k}))W^\dagger, \tag{G1}$$

where

$$W = U_{AB_k} \cdots U_{AB_1}, \quad \text{and} \quad U_{JK} = e^{-iS_{JK}\Delta t}. \tag{G2}$$

The resulting reduced state is

$$\tau'_A = \text{Tr}_B(\tau'_{AB}) \approx V(t)\rho V(t)^\dagger \tag{G3}$$

where $t = k\Delta t$. Finally, by turning each U_{JK} in (G2) into a controlled operation:

$$C_{U_{JK}} = |0\rangle\langle 0| \otimes \mathbb{1} + |1\rangle\langle 1| \otimes e^{-iS_{JK}\Delta t}, \tag{G4}$$

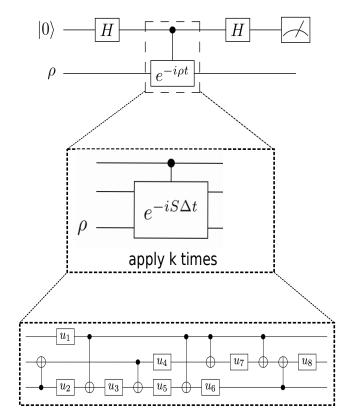and hence making W controlled, one can then construct an approximation of $C_{V(t)}$.

FIG. S.8. Circuit for our qPCA implementation. Here, the eigenvalues of a one-qubit pure state ρ are estimated to a single digit of precision. We use k copies of ρ to approximate $C_{V(t)}$ by applying the controlled-exponential-swap operator k times for a time period $\Delta t = t/k$. The bottom panel shows our compilation of the controlled-exponential-swap gate into one- and two-qubit gates.

If one chooses the input state for quantum phase estimation to be $\rho = \sum_z \lambda_z |v_z\rangle\langle v_z|$ itself, then the final state becomes

$$\sum_z \lambda_z |v_z\rangle\langle v_z| \otimes |\hat{\lambda}_z\rangle\langle\hat{\lambda}_z| \tag{G5}$$

where $\hat{\lambda}_z$ is a binary representation of an estimate of the corresponding eigenvalue λ_z. One can then sample from the state in (G5) to characterize the eigenvalues and eigenvectors.

The approximation of $V(t)$ in (G3) can be done with accuracy ϵ provided that one uses $O(t^2\epsilon^{-1})$ copies of ρ. The time $t_{\max}$ needed for quantum phase estimation to achieve accuracy ϵ is $t_{\max} = O(\epsilon^{-1})$. Hence, with qPCA, the eigenvalues and eigenvectors can be obtained with accuracy ϵ provided that one uses $O(\epsilon^{-3})$ copies of ρ.

2. Our Implementation of qPCA

Figure S.8 shows our strategy for implementing qPCA on an arbitary one-qubit state ρ. The circuit shown corresponds to the quantum phase estimation algorithm with
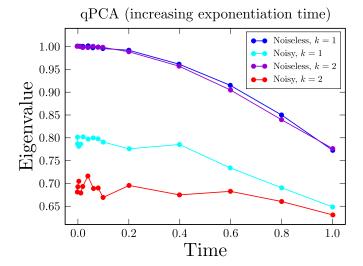
qPCA (increasing exponentiation time)

FIG. S.9. The largest inferred eigenvalue for the one-qubit pure state $\rho = |+\rangle\langle+|$ versus application time of unitary $e^{-i\rho t}$, for our implementation of qPCA on Rigetti's noisy and noiseless QVMs. Curves are shown for $k = 1$ and $k = 2$, where k indicates the number of controlled-exponential-swap operators applied.

one bit of precision (i.e., one ancilla qubit). A Hadamard gate is applied to the ancilla qubit, which then acts as the control system for the $C_{V(t)}$ gate, and finally the ancilla is measured in the x-basis. The $C_{V(t)}$ is approximated (as discussed above) with k applications of the controlled-exponential-swap gate.

To implement qPCA, the controlled-exponential-swap gate in (G4) must be compiled into one- and two-body gates. For this purpose, we used the machine-learning approach from Ref. [38] to obtain a short-depth gate sequence for controlled-exponential-swap. The gate sequence we obtained is shown in Fig. S.8 and involves 7 CNOTs and 8 one-qubit gates. Most of the one-qubit gates are z-rotations and hence are error-free (implemented via a clock change), including the following gates:

$$u_1 = u_7 = R_z(-(\pi + \Delta t)/2) \tag{G6}$$
$$u_3 = R_z((\pi - \Delta t)/2) \tag{G7}$$
$$u_4 = R_z(\Delta t/2) \tag{G8}$$
$$u_5 = R_z((\pi + \Delta t)/2) \tag{G9}$$
$$u_8 = R_z(\pi/2). \tag{G10}$$

The one-qubit gates that are not z-rotations are:

$$u_2 = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ e^{-i(\pi - \Delta t)/2} & e^{i(\pi + \Delta t)/2} \end{pmatrix} \tag{G11}$$

$$u_6 = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & e^{-i(\pi + \Delta t)/2} \\ -i & e^{-i\Delta t/2} \end{pmatrix}. \tag{G12}$$

We implemented the circuit in Fig. S.8 using both Rigetti's noiseless simulator, known as the Quantum Virtual Machine (QVM), as well as their noisy QVM that

QVM	$k = 1$	$k = 2$
noiseless	$\approx \{1, 0\}$	$\approx \{1, 0\}$
noisy	$\approx \{0.8, 0.2\}$	$\approx \{0.7, 0.3\}$

TABLE III. Estimated eigenvalues for the $\rho = |+\rangle\langle+|$ state using qPCA on both the noiseless and the noisy QVMs of Rigetti.

utilizes a noise model of their 8Q-Agave chip. Because the latter is meant to mimic the noise in the 8Q-Agave chip, our qPCA results on the noisy QVM can be compared to our VQSD results on the 8Q-Agave chip in Fig. 3. (We remark that lack of availability prevented us from implementing qPCA on the actual 8Q-Agave chip.)

For our implementation, we chose the one-qubit plus state, $\rho = |+\rangle\langle+|$. Implementations were carried out using both one and two controlled-exponential-swap gates, corresponding to $k = 1$ and $k = 2$. The time t for which the unitary $e^{-i\rho t}$ was applied was increased.

Figure S.9 shows the raw data, i.e., the largest inferred eigenvalue versus t. In each case, small values of t gave more accurate eigenvalues. In the noiseless case, the eigenvalues of $\rho = |+\rangle\langle+|$ were correctly estimated to be $\approx \{1, 0\}$ already for $k = 1$ and consequently also for $k = 2$. In the noisy case, the eigenvalues were estimated to be $\approx \{0.8, 0.2\}$ for $k = 1$ and $\approx \{0.7, 0.3\}$ for $k = 2$, where we have taken the values for small t. Table III summarizes the different cases.

Already for the case of $k = 1$, the required resources of qPCA (3 qubits + 7 CNOT gates) for estimating the eigenvalue of an arbitrary pure one-qubit state ρ are higher than those of the DIP test (2 qubits + 1 CNOT gate) for the same task. Consequently, the DIP test yields more accurate results as can be observed by comparing Fig. 3 to Fig. S.9. Increasing the number of copies to $k = 2$ only decreases the accuracy of the estimation, since the $C_{V(t)}$ gate is already well approximated for short application times t when $k = 1$ in the noiseless case. Thus, increasing the number of copies does not offer any improvement in the noiseless case, but instead leads to poorer estimation performance in the noisy case. This can be seen for the $k = 2$ case (see Fig. S.9 and Table III), due to the doubled number of required CNOT gates relative to $k = 1$.

Appendix H: Circuit derivation

1. DIP test

Here we prove that the circuit in Fig. S.10(a) computes $\text{Tr}(\mathcal{Z}(\sigma)\mathcal{Z}(\tau))$ for any two density matrices σ and τ.

Let σ and τ be states on the n-qubit systems A and B, respectively. Let $\omega_{AB} = \sigma \otimes \tau$ denote the initial state.
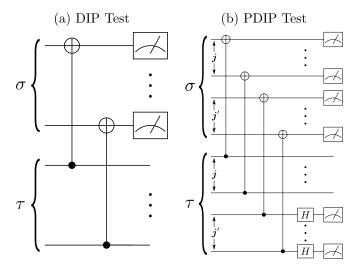
FIG. S.10. Test circuits used to compute the cost function in VQSD. (a) DIP test (b) PDIP test. (These circuits appear in Fig. 5 and are also shown here for the reader's convenience.)

The action of the CNOTs in Fig. S.10(a) gives the state

$$\omega'_{AB} = \sum_{\boldsymbol{z},\boldsymbol{z}'} X^{\boldsymbol{z}} \sigma X^{\boldsymbol{z}'} \otimes |\boldsymbol{z}\rangle\langle\boldsymbol{z}|\tau|\boldsymbol{z}'\rangle\langle\boldsymbol{z}'|, \qquad \text{(H1)}$$

where the notation $X^{\boldsymbol{z}}$ means $X^{z_1} \otimes X^{z_2} \otimes \cdots \otimes X^{z_n}$. Partially tracing over the B system gives

$$\omega'_A = \sum_{\boldsymbol{z}} \tau_{\boldsymbol{z},\boldsymbol{z}} X^{\boldsymbol{z}} \sigma X^{\boldsymbol{z}}, \qquad \text{(H2)}$$

where $\tau_{\boldsymbol{z},\boldsymbol{z}} = \langle\boldsymbol{z}|\tau|\boldsymbol{z}\rangle$. The probability for the all-zeros outcome is then

$$\langle\boldsymbol{0}|\omega'_A|\boldsymbol{0}\rangle = \sum_{\boldsymbol{z}} \tau_{\boldsymbol{z},\boldsymbol{z}}\langle\boldsymbol{0}|X^{\boldsymbol{z}}\sigma X^{\boldsymbol{z}}|\boldsymbol{0}\rangle = \sum_{\boldsymbol{z}} \tau_{\boldsymbol{z},\boldsymbol{z}}\sigma_{\boldsymbol{z},\boldsymbol{z}}, \quad \text{(H3)}$$

which follows because $X^{\boldsymbol{z}}|\boldsymbol{0}\rangle = |\boldsymbol{z}\rangle$. Hence the probability for the all-zeros outcome is precisely the diagonalized inner product, $\text{Tr}(\mathcal{Z}(\sigma)\mathcal{Z}(\tau))$. Note that in the special case where $\sigma = \tau = \tilde{\rho}$, we obtain the sum of the squares of the diagonal elements, $\sum_{\boldsymbol{z}} \tilde{\rho}^2_{\boldsymbol{z},\boldsymbol{z}} = \text{Tr}(\mathcal{Z}(\tilde{\rho})^2)$.

2. PDIP test

We prove that the circuit in Fig. S.10(b) computes $\text{Tr}(\mathcal{Z}_{\boldsymbol{j}}(\sigma)\mathcal{Z}_{\boldsymbol{j}}(\tau))$ for a given set of qubits $\boldsymbol{j}$.

Let $\boldsymbol{j}'$ denote the complement of $\boldsymbol{j}$. Let σ and τ, respectively, be states on the n-qubit systems $A = A_{\boldsymbol{j}\boldsymbol{j}'}$ and $B = B_{\boldsymbol{j}\boldsymbol{j}'}$. The initial state $\omega_{AB} = \sigma \otimes \tau$ evolves, under the action of the CNOTs associated with the DIP Test and then tracing over the control systems, to

$$\omega'_{AB_{\boldsymbol{j}'}} = \sum_{\boldsymbol{z}} (X^{\boldsymbol{z}} \otimes \mathbb{1})\sigma(X^{\boldsymbol{z}} \otimes \mathbb{1}) \otimes \text{Tr}_{B_{\boldsymbol{j}}}((|\boldsymbol{z}\rangle\langle\boldsymbol{z}| \otimes \mathbb{1})\tau),$$

$$\text{(H4)}$$

where $X^{\boldsymbol{z}}$ and $|\boldsymbol{z}\rangle\langle\boldsymbol{z}|$ act non-trivially only on the $\boldsymbol{j}$ subsystems of A and B, respectively. Measuring system $A_{\boldsymbol{j}}$ and obtaining the all-zeros outcome would leave systems $A_{\boldsymbol{j}'}B_{\boldsymbol{j}'}$ in the (unnormalized) conditional state:

$$\text{Tr}_{A_{\boldsymbol{j}}}((|\boldsymbol{0}\rangle\langle\boldsymbol{0}| \otimes \mathbb{1})\omega'_{AB_{\boldsymbol{j}'}}) = \sum_{\boldsymbol{z}} \sigma^{\boldsymbol{z}}_{\boldsymbol{j}'} \otimes \tau^{\boldsymbol{z}}_{\boldsymbol{j}'}, \qquad \text{(H5)}$$

where $\sigma^{\boldsymbol{z}}_{\boldsymbol{j}'} := \text{Tr}_{A_{\boldsymbol{j}}}((|\boldsymbol{z}\rangle\langle\boldsymbol{z}|\otimes\mathbb{1})\sigma)$ and $\tau^{\boldsymbol{z}}_{\boldsymbol{j}'} := \text{Tr}_{B_{\boldsymbol{j}}}((|\boldsymbol{z}\rangle\langle\boldsymbol{z}|\otimes \mathbb{1})\tau)$. Finally, computing the expectation value for the swap operator (via the Destructive Swap Test) on the state in (H5) gives

$$\sum_{\boldsymbol{z}} \text{Tr}((\sigma^{\boldsymbol{z}}_{\boldsymbol{j}'} \otimes \tau^{\boldsymbol{z}}_{\boldsymbol{j}'})S) = \sum_{\boldsymbol{z}} \text{Tr}(\sigma^{\boldsymbol{z}}_{\boldsymbol{j}'}\tau^{\boldsymbol{z}}_{\boldsymbol{j}'}) = \text{Tr}(\mathcal{Z}_{\boldsymbol{j}}(\sigma)\mathcal{Z}_{\boldsymbol{j}}(\tau)).$$

$$\text{(H6)}$$

The last equality can be verified by noting that $\mathcal{Z}_{\boldsymbol{j}}(\sigma) = \sum_{\boldsymbol{z}} |\boldsymbol{z}\rangle\langle\boldsymbol{z}| \otimes \sigma^{\boldsymbol{z}}_{\boldsymbol{j}'}$ and $\mathcal{Z}_{\boldsymbol{j}}(\tau) = \sum_{\boldsymbol{z}} |\boldsymbol{z}\rangle\langle\boldsymbol{z}| \otimes \tau^{\boldsymbol{z}}_{\boldsymbol{j}'}$. Specializing (H6) to $\sigma = \tau = \tilde{\rho}$ gives the quantity $\text{Tr}(\mathcal{Z}_{\boldsymbol{j}}(\tilde{\rho})^2)$.

Appendix I: Proof of Eq. (40)

Let $H_2(\sigma) = -\log_2[\text{Tr}(\sigma^2)]$ be the Renyi entropy of order two. Then, noting that $H_2(\sigma) \leqslant H(\sigma)$, we have

$$\text{Tr}(\mathcal{Z}_j(\tilde{\rho})^2) = 2^{-H_2(\mathcal{Z}_j(\tilde{\rho}))} \geqslant 2^{-H(\mathcal{Z}_j(\tilde{\rho}))}. \qquad \text{(I1)}$$

Next, let A denote qubit j, and let B denote all the other qubits. This allows us to write $\rho = \rho_{AB}$ and $\tilde{\rho} = \tilde{\rho}_{AB}$. Let C be a purifying system such that ρ_{ABC} and $\tilde{\rho}_{ABC}$ are both pure states. Then we have

$$H(\mathcal{Z}_j(\tilde{\rho})) = H(\mathcal{Z}_j(\tilde{\rho}_{AB})) \qquad \text{(I2)}$$
$$= H(\mathcal{Z}_j(\tilde{\rho}_{AC})) \qquad \text{(I3)}$$
$$\leqslant H(\mathcal{Z}_j(\tilde{\rho}_A)) + H(\tilde{\rho}_C) \qquad \text{(I4)}$$

where the inequality in (I4) used the subadditivity of von Neumann entropy. Finally, note that

$$H(\tilde{\rho}_C) = H(\tilde{\rho}_{AB}) = H(\rho_{AB}) = H(\rho) \qquad \text{(I5)}$$

and $H(\mathcal{Z}_j(\tilde{\rho}_A)) \leqslant 1$, which gives

$$H(\mathcal{Z}_j(\tilde{\rho})) \leqslant 1 + H(\rho). \qquad \text{(I6)}$$

Substituting (I6) into (I1) gives

$$\text{Tr}(\mathcal{Z}_j(\tilde{\rho})^2) \geqslant 2^{-1-H(\rho)}, \qquad \text{(I7)}$$

and (40) follows from $H(\rho) \leqslant \log_2 r$.