Variational Quantum State Diagonalization

Ryan LaRose,1,2 Arkin Tikku,1,3 Étude O’Neel-Judy,1 Lukasz Cincio,1 and Patrick J. Coles1

1Theoretical Division, Los Alamos National Laboratory, Los Alamos, NM 87545, USA.
2Department of Computational Mathematics, Science, and Engineering & Department of Physics and Astronomy, Michigan State University, East Lansing, MI 48823, USA.
3Department 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 exponential 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 $\rho$ 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 $\rho$. One can then read out approximations of the largest eigenvalues, and the associated eigenvectors, of $\rho$. 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 photoexcitation 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 determining 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 the 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 exponential 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 [14–16], approximate optimization [7], error correction [17], quantum data compression [18], quantum simulation [19], and quantum compiling [20]. 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 [21]. In data science and machine learning, diagonalizing the covariance matrix (which could be encoded in a quantum state [2, 22]) 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 [23].

Classical methods for diagonalization typically scale polynomially in the matrix dimensions [24]. 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 $\rho$, our algorithm is composed of three steps: (1) Train the parameters $\alpha$ of a gate sequence $U_p(\alpha)$ such that $\tilde{\rho} = U_p(\alpha_{\text{opt}})\rho U_p(\alpha_{\text{opt}})^\dagger$ is approximately diagonal, (2) Read out the largest eigenvalues of $\rho$ by measuring in the eigenbasis (i.e., by measuring $\tilde{\rho}$ in the standard basis), and (3) 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 $U_p(\alpha)$ (where $p$ is the number of layers) allows one to obtain a hierarchy of approximations for the eigenvalues and eigenvectors. We therefore think of VQSD as an approximate diagonalization algorithm.

We carefully choose our cost function $C$ to have the following properties: (1) $C$ is faithful (i.e., it vanishes if and only if $\tilde{\rho}$ is diagonal), (2) $C$ is efficiently computable on a quantum computer, (3) $C$ has an operational meaning such that it upper bounds the eigenvalue error (the discrepancy between the inferred and the exact eigenvalues), and (4) $C$ 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 Heisenberg model composed of 8 spins.

Our paper is organized as follows. Section II outlines the VQSD algorithm and presents its implementation. In Sec. III, we discuss the complexity of the VQSD algorithm for particular example states, 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 $\rho$ in quantum form and then output approximations of the $m$-largest eigenvalues and their associated eigenvectors. Here, $m$ will typically be much less than $2^n$, the total number of eigenvectors, 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 [25]. 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 [26]. Specifically, VQSD diagonalizes $\rho$ by variationally updating a parameterized unitary $U_p(\alpha)$ such that

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

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

$$U_p(\alpha) = L_1(\alpha_1)L_2(\alpha_2)\cdots L_p(\alpha_p).$$

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

Let $C$ denote our cost function, defined below in (9), 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 point for the optimization of $p+1$ layers. Next, we argue that $C$ is closely connected to the accuracy of the eigenvalues. Specifically, it gives an upper bound
FIG. 1. Schematic diagram showing the steps of the VQSD algorithm. (a) The quantum state $\rho$ is provided as an input. This state is sent to the parameter optimization loop (b) where a hybrid quantum-classical variational algorithm approximates the diagonalizing unitary $U_p(\alpha_{\text{opt}})$. 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 $z_i$, the frequencies of which provide estimates of the eigenvalues of $\rho$. Along with the optimal unitary $U_p(\alpha_{\text{opt}})$, these bitstrings are sent to the eigenvector preparation circuit (c) to prepare the eigenstates of $\rho$ on a quantum computer. Both the eigenvalues and eigenvectors are the outputs (d) of the VQSD algorithm.

on the eigenvalue error. So while $C$ is not directly a measure of eigenvalue error, one obtains an increasingly tighter upper bound on the eigenvalue error as $C$ decreases (equivalently, as $p$ increases). We introduce the following quantity to quantify eigenvalue error, i.e., the discrepancy between the inferred eigenvalues $\tilde{\lambda}$ and the actual eigenvalues $\lambda$,

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

where $d = 2^n$. Here the vectors $\tilde{\lambda} = (\tilde{\lambda}_1, ..., \tilde{\lambda}_d)$ and $\lambda = (\lambda_1, ..., \lambda_d)$ are both ordered in decreasing order. As proven in Sec. IV A, our cost function upper bounds the eigenvalue error up to a proportionality factor $q$,

$$\Delta(\lambda, \tilde{\lambda}) \leq C/q.$$  

(4)

Because $C$ is non-increasing in $p$, the upper bound in (4) is non-increasing in $p$ and goes to zero if $C$ goes to zero.

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 $\alpha$ and the latter computes $C$ for $U_p(\alpha)$, (2) a readout procedure for approximations of the $m$ largest eigenvalues, which involves running a quantum circuit and then classically 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 $U_p(\alpha)$. Ideally one would like the number of parameters to grow polynomially in both $n$ and $p$, so that the search space is not exponentially large. 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. [27].

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

$$\alpha_{\text{opt}} := \arg\min_{\alpha} C(U_p(\alpha)).$$

(5)

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. (5), \( C(U_p(\alpha)) \) quantifies how far the state \( \tilde{\rho}_p(\alpha) \) 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 [28]. We aim for a cost that is efficiently computable on a quantum computer, and hence we consider a cost that can be expressed in terms of purifications.

The layered structure ansatz for the diagonalizing unitary \( U_p(\alpha) \). Each layer \( L_i, i = 1,\ldots,p \), consists of a set of optimization parameters \( \alpha_i \). Increasing the number of layers \( p \) yields a better approximation to the eigenvalues and eigenvectors of the state \( \rho \).

(b) The two-qubit gate structure 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 \( G_1 \) wraps around from top to bottom.

3. Eigenvalue Readout

After finding the optimal diagonalizing unitary \( U_p(\alpha_{\text{opt}}) \), one can use it to readout approximations of the eigenvalues of \( \rho \). Figure 1(c) shows the circuit for this readout. One prepares a single copy of \( \rho \) and then acts with \( U_p(\alpha_{\text{opt}}) \) to prepare \( \tilde{\rho}_p(\alpha_{\text{opt}}) \). Measuring in the standard basis \( \{|z\rangle\} \), where \( z = z_1z_2\ldots z_n \) is a bitstring of length \( n \), gives a set of probabilities \( \{\lambda_z\} \) with

\[
\lambda_z = \langle z|\tilde{\rho}_p(\alpha_{\text{opt}})|z\rangle.
\]

We take the \( \lambda_z \) as the inferred eigenvalues of \( \rho \). We emphasize that the \( \lambda_z \) are the diagonal elements, not the eigenvalues, of \( \tilde{\rho}_p(\alpha_{\text{opt}}) \). Note that the squared error between the inferred eigenvalue and the corresponding true eigenvalue is bounded by the final cost that one obtained in the parameter optimization loop. That is,

\[
(\lambda_z - \hat{\lambda}_z)^2 \leq C/q.
\]

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

\[
\hat{\lambda}_z = f_z/N_{\text{readout}}
\]
gives an estimate for \( \tilde{\lambda}_z \). The statistical deviation of \( \tilde{\lambda}_z^{\text{est}} \) from \( \tilde{\lambda}_z \) goes with \( 1/\sqrt{N_{\text{readout}}} \). The relative error \( \epsilon_z \) (i.e., the ratio of the statistical error on \( \tilde{\lambda}_z^{\text{est}} \) to the value of \( \tilde{\lambda}_z^{\text{est}} \)) then goes as

\[
\epsilon_z = \frac{1}{\sqrt{N_{\text{readout}}\tilde{\lambda}_z^{\text{est}}}} = \frac{\sqrt{N_{\text{readout}}}}{f_z}.
\]

This implies that events \( z \) with higher frequency \( f_z \) have lower relative error. In other words, the larger the inferred eigenvalue \( \tilde{\lambda}_z \), 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 \( N_{\text{readout}} \) and a threshold for the relative error, denoted \( \epsilon_{\text{max}} \). This error threshold \( \epsilon_{\text{max}} \) will then determine \( m \), i.e., how many of the largest eigenvalues that get precisely characterized. So \( m = m(N_{\text{readout}}, \epsilon_{\text{max}}, \{ \tilde{\lambda}_z \}) \) is a function of \( N_{\text{readout}} \), \( \epsilon_{\text{max}} \), and the set of inferred eigenvalues \( \{ \tilde{\lambda}_z \} \). Precisely, we take \( m = |\tilde{\lambda}^{\text{est}}| \) as the cardinality of the following set:

\[
\tilde{\lambda}^{\text{est}} = \{ \tilde{\lambda}_z^{\text{est}} : \epsilon_z \leq \epsilon_{\text{max}} \},
\]

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. (14). Let \( Z = \{ z : \tilde{\lambda}_z^{\text{est}} \in \tilde{\lambda}^{\text{est}} \} \) be the set of bitstrings \( z \) associated with the eigenvalues in \( \tilde{\lambda}^{\text{est}} \). (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 \( z \).) For each \( z \in Z \), one can prepare the following state, which we take as the inferred eigenvector associated with our estimate of the inferred eigenvalue \( \lambda_z^{\text{est}} \),

\[
|\tilde{v}_z\rangle = U_p(\alpha_{\text{opt}})^\dagger |z\rangle
= U_p(\alpha_{\text{opt}})^\dagger (X^{z_1} \otimes \cdots \otimes X^{z_n}) |0\rangle.
\]

The circuit for preparing this state is shown in Fig. 1(d). As noted in (16), one first prepares \( |z\rangle \) by acting with \( X \) operators raised to the appropriate powers, and then one acts with \( U_p(\alpha_{\text{opt}})^\dagger \) to rotate from the standard basis to the inferred eigenbasis.

Naturally, the inferred eigenvector \( |\tilde{v}_z\rangle \) will be an approximation of the true eigenvector \( |v_z\rangle \) of \( \rho \). Unlike the case of eigenvalue error, we do not have a simple bound that relates eigenvector error (i.e., how far \( |\tilde{v}_z\rangle \) is from \( |v_z\rangle \)) to our cost function. Nevertheless, it is clear that the eigenvector error must vanish as our cost function goes to zero.

Once they are in quantum form, each inferred eigenvector \( |\tilde{v}_z\rangle \) 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 \( \langle \tilde{v}_z | M | \tilde{v}_z \rangle \) 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 8-spin ground state of the Heisenberg model.

1. One-Qubit State

We now discuss the results of applying VQSD to the one-qubit plus state \( |+\rangle\langle+| \) on the 8Q-Agate quantum computer provided by Rigetti. Because the problem size is small (\( n = 1 \)), we set \( q = 1 \) in the cost function (9). Since \( \rho \) is a pure state, the cost function is

\[
C(U_p(\alpha)) = C_1(U_p(\alpha)) = 1 - \text{Tr}(\tilde{\rho}(\hat{p})^2).
\]

For \( U_p(\alpha) \), 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 number of iterations in the parameter optimization loop, and the dashed curves show the inferred eigenvalues of \( \rho \) at each iteration. 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-Agate quantum computer, Rigetti’s noisy simulator, and a noiseless simulator. Here, we varied the angle \( \alpha \) in the diagonalizing unitary \( U(\alpha) = R_z(\pi/2)R_x(\alpha) \) 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-Agate quantum computer 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 one-dimensional Heisenberg model, whose Hamiltonian is

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

(18)

with $S^{(j)} = (1/2)(\sigma_{x}^{(j)} \hat{x} + \sigma_{y}^{(j)} \hat{y} + \sigma_{z}^{(j)} \hat{z})$ and periodic boundary conditions, $S^{(n+1)} = S^{(1)}$. Performing entanglement spectroscopy on the ground state $|\psi\rangle_{AB}$ involves diagonalizing the reduced state $\rho = \text{Tr}_{B}|\psi\rangle_{AB}\langle\psi|$. Here we consider a total of 8 spins ($n = 8$). 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 $\rho$ 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 shows that our VQSD implementation is getting roughly the correct values for both the eigenvalues and their quantum numbers, particularly for large eigenvalues. Resolving not only the eigenvalues but also their quantum numbers is important for entanglement spectroscopy [21], and clearly VQSD can do this.
III. DISCUSSION

A. General Complexity Remarks

We emphasize that VQSD is meant for states $\rho$ 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 $N_{\text{readout}}$ polynomial in $n$), then very few eigenvalues would get characterized with the desired precision.

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 $\rho$. (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 efficient with respect to (C1) does not guarantee that (C2) is efficient.

B. 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 $\rho$ 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, $\lambda_{\text{min}} \approx 1$ and $\epsilon_2 \approx 1/\sqrt{N_{\text{readout}}}$ for this eigenvalue. This implies that $N_{\text{readout}}$ 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

$$\rho = \sum_k p_k |b_k^{(1)}\rangle\langle b_k^{(1)}| \otimes \cdots \otimes |b_k^{(n)}\rangle\langle b_k^{(n)}|$$

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 $\{p_k\}$ 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 $N_{\text{readout}}$ 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 $\rho$ as its input, and $\rho$ must have been prepared on a quantum computer. Let $V$ be the unitary that was used to prepare $|\psi\rangle = V|0\rangle$ 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 adjoining each gate, can be used to diagonalize $\rho$. Because $V$ is efficiently implementable, then $V^\dagger$ is also efficiently implementable. Hence, $\rho$ 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 search space dimensionality 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.

C. 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 $\rho$ being a low-entropy state.

We performed a simple implementation of qPCA to get a sense for how it compares to VQSD (see Appendix 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-Agate quantum computer). The circuit that we implemented applied one controlled-exponential-swap gate (in order to approximately exponentiate ρ, as discussed in [2]). 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.

D. 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 ρ = |ψ⟩⟨ψ|, then one can learn the unitary U(α) that maps |ψ⟩ to a standard basis state. Inverting this unitary allows one to map a standard basis state (and hence the state |0⟩⊗n) to the state |ψ⟩, which is known as state preparation. Hence, if one is given |ψ⟩ in quantum form, then VQSD can potentially find a short-depth circuit that approximately prepares |ψ⟩. (We will explore the special case of variational state preparation in future work.)

In machine learning, PCA is a common 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[XXT] with the largest eigenvalues, where E denotes expectation value. Thus PCA is reduced to diagonalizing a positive-semidefinite matrix, E[XXT]. Hence VQSD can perform this task provided one has access to QRAM [22] to prepare the covariance matrix as a quantum state. PCA can be used to reduce the dimension of X as well as to filter out noise in data. It is a subroutine in multidimensional scaling—a machine learning algorithm that reconstructs an approximate “map” of vectors x; given only pairwise distances ||x_i - x_j||^2 between them—as well as the isomap algorithm [31] in manifold learning. By generalizing the Euclidean inner product to a positive semi-definite kernel, nonlinear (kernel) PCA can be used on data that is not linearly separable.

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 [14] 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 [32]. Extracting both the eigenvalues and eigenvectors is useful for entanglement spectroscopy [32], and we illustrated this capability of VQSD in Fig. 4.

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 (6) 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_HS(A, B) := Tr ((A - B)†(A - B)) denote the Hilbert-Schmidt distance. Then we can write

\begin{align}
C_1 &= \min_{\sigma \in \mathcal{D}} D_{HS}(\hat{\rho}, \sigma) \\
&= D_{HS}(\hat{\rho}, \mathcal{Z}(\hat{\rho})) \\
&= \sum_{z, z' \neq z} |\langle z | \hat{\rho} | z' \rangle|^2.
\end{align}

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

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

\begin{align}
\Delta(\lambda, \hat{\lambda}) &= \lambda - \hat{\lambda} - \hat{\lambda} \cdot \hat{\lambda} - 2\lambda \cdot \hat{\lambda} \quad (23) \\
C_1 &= \lambda \cdot \hat{\lambda} - \hat{\lambda} \cdot \hat{\lambda} \\
&= \Delta(\lambda, \hat{\lambda}) + 2(\lambda \cdot \hat{\lambda} - \hat{\lambda} \cdot \hat{\lambda}). \\
\end{align}

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

\begin{equation}
\lambda \cdot \hat{\lambda} \geq \hat{\lambda} \cdot \hat{\lambda}. \quad (26)
\end{equation}

Hence from (25) and (26) we obtain the bound

\begin{equation}
\Delta(\lambda, \hat{\lambda}) \leq C_1. \quad (27)
\end{equation}

For computational purposes, we use the difference of purities interpretation of \( C_1 \) given in (6). The Tr(\( \rho^2 \)) term is independent of \( U_p(\alpha) \). 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 $\rho$, using the identity

$$\text{Tr}(\rho^2) = \text{Tr}((\rho \otimes \rho)S).$$

(28)

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 [33, 34]. 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 an ancilla.

Similarly, the $\text{Tr}(Z(\tilde{\rho})^2)$ 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. [35]. 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 $\sigma$ and $\tau$. For our application we will set $\sigma = \tau = \tilde{\rho}$, although we discuss the general case below.

Let $\sigma$ and $\tau$ be states on the $n$-qubit systems $A$ and $B$, respectively. Let $\omega_{AB} = \sigma \otimes \tau$ denote the initial state. The action of the CNOTs in Fig. 5(b) gives the state

$$\omega'_{AB} = \sum_{z,z'} X^z \sigma X^{z'} \otimes |z\rangle\langle z| z'\rangle\langle z'|,$$

(29)

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

$$\omega'_A = \sum_z \tau_{z,z} X^z \sigma X^z,$$

(30)

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

$$\langle 0 | \omega'_A | 0 \rangle = \sum_z \tau_{z,z} \langle 0 | X^z \sigma X^z | 0 \rangle = \sum_z \tau_{z,z} \sigma_{z,z},$$

(31)

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

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

2. $C_2$ and the PDIP Test

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

$$C_2 = \frac{1}{n} \sum_{j=1}^n D_{\text{HS}}(\tilde{\rho}, Z_j(\tilde{\rho})).$$

(32)

where $Z_j(\cdot) = \sum_i (|i\rangle\langle i| \otimes 1_{\not= j}) (|i\rangle\langle i| \otimes 1_{\not= j})$. Naturally, one would expect that $C_2 \leq C_1$, 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:

$$C_2 = C_1 - \frac{1}{n} \sum_{j=1}^n \min_{\sigma \in D} D_{\text{HS}}(Z_j(\tilde{\rho}), \sigma).$$

(33)

However, $C_1$ and $C_2$ vanish under precisely the same conditions, as noted in Eq. (8). One can see this as follows. The condition $\tilde{\rho} = Z_j(\tilde{\rho})$ for all $j$ corresponds to $C_2 = 0$. This implies that $\tilde{\rho} = (Z_1 \otimes \cdots \otimes Z_n)(\tilde{\rho}) = Z(\tilde{\rho})$, which follows by repeatedly acting with local dephasing channels. Thus, $C_2$ is faithful, vanishing iff $\tilde{\rho}$ is diagonal.

Writing $C_2$ in terms of purities, as in (7), shows how it can be computed on a quantum computer. As in the case of $C_1$, the first term in (7) is computed with the Destructive Swap Test. For the second term in (7), each purity $\text{Tr}(Z_j(\tilde{\rho})^2)$ 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 circuit is shown in Fig. 5(c) for the general case of feeding in two distinct states $\sigma$ and $\tau$ with the goal of computing the inner product between $Z_j(\sigma)$ and $Z_j(\tau)$. For generality we let $l$, with $0 \leq l \leq n$, denote the number of qubits being locally dephased for this computation. If $l > 0$, we define $j = (j_1, \ldots, j_l)$ 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 $j$ and $\sigma$ are being locally dephased for this computation. If $j'\neq j$, we let $z = (z_1, \ldots, z_n)$ be such that $z_j \neq z_j'$, and obtaining the all-zeros outcome would leave systems $A_jB_{j'}$ in two distinct states $A_jB_{j'}$ and $B_{j'}A_j$, respectively. Measuring system $A_jB_{j'}$ in the (unnormalized) conditional state:

$$\text{Tr}((|z\rangle\langle z| \otimes \mathbb{I} )\tau),$$

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

$$\text{Tr}_{A_j}(|0\rangle\langle 0| \otimes \mathbb{I})\omega_{AB_{j'}} = \sum_z \sigma^z_j \otimes \tau^z_{j'},$$

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

$$\text{Tr}_{A_j}(|0\rangle\langle 0| \otimes \mathbb{I})\omega_{AB_{j'}} = \sum_z \sigma^z_j \otimes \tau^z_{j'},$$

where $\sigma^z_j := \text{Tr}_{A_j}(|z\rangle\langle z| \otimes \mathbb{I})\sigma$ and $\tau^z_{j'} := \text{Tr}_{B_{j'}}(|z\rangle\langle z| \otimes \mathbb{I})\tau$. Finally, computing the expectation value for the swap operator (via the Destructive Swap Test) on the state in (35) gives

$$\sum_z \text{Tr}(\sigma^z_j \otimes \tau^z_{j'})S = \sum_z \text{Tr}(\sigma^z_j \otimes \tau^z_{j'}) = \text{Tr}(Z_j(\sigma)Z_j(\tau)).$$

The last equality can be verified by noting that $Z_j(\sigma) = \sum_z |z\rangle\langle z| \otimes \sigma^z_j$, and $Z_j(\tau) = \sum_z |z\rangle\langle z| \otimes \tau^z_{j'}$. Specializing (36) to $\sigma = \tau = \hat{\rho}$ gives the quantity $\text{Tr}(Z_j(\hat{\rho})^2)$.

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 (9).

As noted previously, $C_1$ has an operational meaning as an upper bound on eigenvalue error, whereas $C_2$ does not share this interpretation. In addition, the circuit for computing $C_1$ is more efficient than that for $C_2$. The circuit for computing the second term in $C_1$, shown in Fig. 5(b) has a gate depth of one, with $n$ CNOT gates, $n$ measurements, and no classical post-processing. The circuit for computing the second term in $C_2$, shown in Fig. 5(c) has a gate depth of two, with $n$ CNOT gates, $2n-1$ measurements, and classical post-processing whose complexity scales linearly in $n$. So in every aspect (gate depth, gate count, number of measurements, and classical post-processing), the circuit for computing $C_1$ is less complex than that for $C_2$. Hence, 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 $\rho = |0\rangle\langle 0| \otimes \cdots \otimes |0\rangle\langle 0|$ is a tensor product of $n$ copies of the $|0\rangle\langle 0|$ state. 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 $\theta$. For this example, $C_1$ is

$$C_1(\theta) = 1 - \text{Tr}(Z(\hat{\rho})^2) = 1 - x(\theta)^n$$  (37)

where $x(\theta) = \text{Tr}(Z(R_X(\theta)|0\rangle\langle 0| R_X(\theta)^\dagger)^2) = (1 + \cos^2 \theta)/2$. If $\theta$ is not an integer multiple of $\pi$, 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 $\theta$ except for multiples of $\pi$. Hence, for large $n$, it becomes difficult to train the unitary $U(\theta)$ because the gradient vanishes for most $\theta$. 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, the function $C_2$ does not suffer from this issue. For the example in the previous paragraph, we have

$$C_2(\theta) = 1 - x(\theta),$$  (38)

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 the parameter $\theta$. 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 $\rho$ that have low rank or low entropy. The second term in (7), which is the term that provides the variability with $\alpha$, does not vanish even for large $n$, since (as shown in Appendix D):

$$\text{Tr}(Z_j(\hat{\rho})^2) \geq 2^{-H(\rho)-1} \geq \frac{1}{2r}.$$  (39)

Here, $H(\rho) = -\text{Tr}(\rho \log_2 \rho)$ is the von Neumann entropy of $\rho$, and $r$ is the rank of $\rho$. So as long as $\rho$ 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 $n$.

Due to these issues, we proposed a weighted average of $C_1$ and $C_2$ for our cost $C$ in (9). This allows one to weight $C_1$ moreso for small $n$ and $C_2$ moreso for large $n$. 
B. Optimization Methods

Finding $\alpha_{\text{opt}}$ in (5) is a major component of VQSD. While many works have benchmarked classical optimization algorithms (e.g., Ref. [36]), the particular case of optimization for variational hybrid algorithms [37] is limited and needs further work [38]. Both gradient-based and gradient-free methods are possible, but gradient-based methods may not work as well with noisy data. Additionally, Ref. [39] 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 B), we found that the Powell optimization algorithm [40] 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 [41].

ACKNOWLEDGMENTS

We thank Rigetti for providing access to their quantum computer. The views expressed in this paper are those of the authors and do not reflect those of Rigetti. RL, EO, and AT acknowledge support from the U.S. Department of Energy through a quantum computing program sponsored by the LANL Information Science & Technology Institute. RL acknowledges support from an Engineering Distinguished Fellowship through Michigan State University. LC was supported by the U.S. Department of Energy through the J. Robert Oppenheimer fellowship. PJJC was supported by the LANL ASC Beyond Moore’s Law project.

[1] Peter W Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM review, 41(2):303–332, 1999.
[2] Seth Lloyd, Masoud Mohseni, and Patrick Rebentrost. Quantum principal component analysis. Nature Physics, 10(9):631–633.
[3] Aram W Harrow, Avinatan Hassidim, and Seth Lloyd. Quantum algorithm for linear systems of equations. Physical review letters, 103(15):150502, 2009.
[4] Patrick Rebentrost, Adrian Steffens, Iman Marvian, and Seth Lloyd. Quantum singular-value decomposition of non-sparse low-rank matrices. Physical review A, 97(1):012327, 2018.
[5] Sarah K Leyton and Tobias J Osborne. A quantum algorithm to solve nonlinear differential equations. arXiv preprint arXiv:0812.4423, 2008.
[6] Dominic W Berry. High-order quantum algorithm for solving linear differential equations. Journal of Physics A: Mathematical and Theoretical, 47(10):105301, 2014.
[7] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. A quantum approximate optimization algorithm. arXiv preprint arXiv:1411.4028, 2014.
[8] Alberto Peruzzo, Jarrod McClean, Peter Shadbolt, Man-Hong Yung, Xiao-Qi Zhou, Peter J Love, Alán Aspuru-Guzik, and Jeremy L. O’Brien. A variational eigenvalue solver on a photonic quantum processor. Nature communications, 5:4213, 2014.
[9] Abhinav Kandala, Antonio Mezzacapo, Kristan Temme, Maika Takita, Markus Brink, Jerry M Chow, and Jay M Gambetta. Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets. Nature, 549(7671):242, 2017.
[10] Dominic W Berry, Andrew M Childs, Richard Cleve, Robin Kothari, and Rolando D Somma. Simulating hamiltonian dynamics with a truncated taylor series. Physical review letters, 114(9):090502, 2015.
[11] John Preskill. Quantum computing and the entanglement frontier. arXiv:1203.5813, 2012.
[12] Aram W Harrow and Ashley Montanaro. Quantum computational supremacy. Nature, 549(7671):203, 2017.
[13] S. Bravyi, G. Smith, and J. A. Smolin. Trading classical and quantum computational resources. Physical Review X, 6(2):021043, 2016.
[14] Alberto Peruzzo, Jarrod McClean, Peter Shadbolt, Man-Hong Yung, Xiao-Qi Zhou, Peter J. Love, Alán Aspuru-Guzik, and Jeremy L. O’Brien. A variational eigenvalue solver on a photonic quantum processor. Nature communications, 5:4213, 2014.
[15] O. Higgott, D. Wang, and S. Brierley. Variational Quantum Computation of Excited States. arXiv:1805.08129, May 2018.
[16] S. Endo, T. Jones, S. McArdle, X. Yuan, and S. Benjamin. Variational quantum algorithms for discovering Hamiltonian spectra. arXiv:1806.05707, June 2018.
[17] Peter D Johnson, Jonathan Romero, Jonathan Olson, Yudong Cao, and Alán Aspuru-Guzik. QVECTOR: an algorithm for device-tailored quantum error correction. arXiv:1711.02249, 2017.
[18] Amir Khoshaman, Walter Vinci, Brandon Denis, Evgeny Andriyash, and Mohammad H. Amin. Quantum variational autoencoder. arXiv:1802.05779 [quant-ph, stat], Feb 2018. arXiv: 1802.05779.
[19] Ying Li and Simon C Benjamin. Efficient variational quantum simulator incorporating active error minimization. Physical Review X, 7(2):021050, 2017.
[20] Sumeet Khatri, Ryan LaRose, Alexander Poremba, Lukasz Cincio, Andrew T. Sornborger, and Patrick J. Coles. Quantum-assisted quantum compiling. arXiv:1807.00800 [quant-ph], Jul 2018. arXiv: 1807.00800.
[21] Hui Li and F Duncan M Haldane. Entanglement spectrum as a generalization of entanglement entropy: Iden-
tification of topological order in non-abelian fractional quantum hall effect states. *Physical Review Letters*, 101(1):010504, 2008.

[22] Vittorio Giovannetti, Seth Lloyd, and Lorenzo Maccone. Quantum random access memory. *Physical review letters*, 100(16):160501, 2008.

[23] Karl Pearson. Liii. 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(11):559–572, 1901.

[24] Lloyd N. Trefethen and David Bau. *Numerical Linear Algebra*. SIAM, 1997.

[25] 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 [24].

[26] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. A quantum approximate optimization algorithm. *arXiv:1411.4028*, 2012.

[27] Farrokh Vatan and Colin Williams. Optimal quantum circuits for general two-qubit gates. *Physical Review A*, 69:032315, Mar 2004.

[28] Tillmann Baumgratz, Marcus Cramer, and Martin B Plenio. Quantifying coherence. *Physical review letters*, 113(14):140401, 2014.

[29] Harry Buhrman, Richard Cleve, John Watrous, and Ronald De Wolf. Quantum fingerprinting. *Physical Review Letters*, 87(16):167902, 2001.

[30] Daniel Gottesman and Isaac Chuang. Quantum digital signatures. *quant-ph/0105032*, 2001.

[31] J. B. Tenenbaum, Vin de Silva, and J.C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, Dec 2000.

[32] Hui 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. *Phys. Rev. Lett.*, 101:010504, Jul 2008.

[33] Juan Carlos Garcia-Escartin and Pedro Chamorro-Posada. Swap test and Hong-Ou-Mandel effect are equivalent. *Physical Review A*, 87(5):052330, 2013.

[34] Lukasz Cincio, Yiğit Subaşı, Andrew T Sornborger, and Patrick J Coles. Learning the quantum algorithm for state overlap. *arXiv:1803.04114* [quant-ph], 2018.

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

[36] Jarrod R McClean, Sergio Boixo, Vadim N Smelyanskiy, Ryan Babbush, and Hartmut Neven. Barren plateaus in quantum neural network training landscapes. *arXiv:1803.11173*, 2018.

[37] M. J. D. Powell. A fast algorithm for nonlinearly constrained optimization calculations. In G. A. Editor Watson, editor, *Numerical Analysis*, Lecture Notes in Mathematics, pages 144–157. Springer Berlin Heidelberg, 1978.

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

[39] M. J. D. Powell. Direct search algorithms for optimization calculations. *Acta Numerica*, 7:287–336, 1998.

[40] M. J. D. Powell. The bovyq algorithm for bound constrained optimization without derivatives. 2009.

[41] Fuchang Gao and Lixing Han. Implementing the nelder-mead simplex algorithm with adaptive parameters. *Computational Optimization and Applications*, 51(1):259–277, Jan 2012.

[42] Jorge Nocedal and S. Wright. *Numerical Optimization*. Springer Series in Operations Research and Financial Engineering. Springer-Verlag, 2 edition, 2006.

[43] Coralia Cartis, Jan Fiala, Benjamin Marteau, and Lindon Roberts. Improving the flexibility and robustness of model-based derivative-free optimization solvers. Mar 2018.
FIG. 6. 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.

Appendix A: Details on VQSD Implementations

Here we provide further details on our implementations of VQSD in Sec. II B.

First, we discuss our implementation on a quantum computer (data shown in Fig. 3). Figure 6 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 $U(\theta) = R_z(\theta)R_x(\pi/2)$. 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. 6. As can be seen in Fig. 3(b), 10,000 runs was sufficient to accurately estimate the cost function (9) with small variance. Because the problem size was small, we took $q = 1$ in (9), 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 $\rho$ 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 $\theta$ is obtained.

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 allowed 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 [38].

We explore optimization methods further in Appendix B.

Appendix B: 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 [40], Constrained Optimization BY Linear Approximation (COBYLA) [42], Bound Optimization BY Quadratic Approximation (BOBYQA) [43], Nelder-Mead [44], Broyden-Fletcher-Goldfarb-Shanno (BFGS) [45], and conjugate gradient (CG) [45]. As mentioned in
In this study, as well as others, we found 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 optimization 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 C: 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 \( \rho \).

Density matrix exponentiation refers to generating the unitary \( V(t) = e^{-i\rho t} \) for a given state \( \rho \) 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, \( \{ t, 2t, 2^2t, \ldots, 2^\text{tmax} \} \), as part of the phase-estimation algorithm. Here we define \( \text{tmax} := 2^\text{t} \).

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

$$\tau_{AB} = \sigma \otimes (\rho^\otimes k) \quad \rightarrow \quad \tau'_{AB} = W(\sigma \otimes (\rho^\otimes k))W^\dagger, \quad (C1)$$

where

$$W = U_{AB_k} \cdots U_{AB_1}, \quad \text{and} \quad U_{JK} = e^{-iS_{JK} \Delta t}. \quad (C2)$$

The resulting reduced state is

$$\tau'_A = \text{Tr}_B(\tau'_{AB}) \approx V(t)\rho V(t)^\dagger \quad (C3)$$

where $t = k\Delta t$. Finally, by turning each $U_{JK}$ in (C2) into a controlled operation:

$$C_{U_{JK}} = |0\rangle\langle 0| \otimes 1 + |1\rangle\langle 1| \otimes e^{-iS_{JK} \Delta t}, \quad (C4)$$

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

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

$$\sum_z \lambda_z |v_z\rangle\langle v_z| \otimes |\hat{\lambda}_z\rangle \langle \hat{\lambda}_z| \quad (C5)$$

where $\hat{\lambda}_z$ is a binary representation of an estimate of the corresponding eigenvalue $\lambda_z$. One can then sample from the state in (C5) to characterize the eigenvalues and eigenvectors.

The approximation of $V(t)$ in (C3) can be done with accuracy $\epsilon$ provided that one uses $O(t^2 \epsilon^{-1})$ copies of $\rho$. The time $t_{\text{max}}$ needed for quantum phase estimation to achieve accuracy $\epsilon$ is $t_{\text{max}} = O(\epsilon^{-1})$. Hence, with qPCA, the eigenvalues and eigenvectors can be obtained with accuracy $\epsilon$ provided that one uses $O(\epsilon^{-3})$ copies of $\rho$.

2. Our Implementation of qPCA

Figure 8 shows our strategy for implementing qPCA on an arbitrary one-qubit state $\rho$. The circuit shown corresponds to the quantum phase estimation algorithm with 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 $C_{V(i)}$ gate, and finally the ancilla is measured in the $x$-basis. The $C_{V(i)}$ is approximated (as discussed above) with $k$ applications of the controlled-exponential-swap gate.

To implement qPCA, the controlled-exponential-swap gate in (C4) must be compiled into one- and two-body gates. For this purpose, we used the machine-learning approach from Ref. [34] to obtain a short-depth gate sequence for controlled-exponential-swap. The gate sequence we obtained is shown in Fig. 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:

$$u_1 = u_5 = u_7 = R_z(-\pi + \Delta t)/2 \quad (C6)$$
$$u_3 = R_z((\pi - \Delta t)/2) \quad (C7)$$
$$u_4 = R_z(\Delta t/2) \quad (C8)$$
$$u_8 = R_z(\pi/2). \quad (C9)$$

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

$$u_2 = \frac{1}{\sqrt{2}} \begin{pmatrix} e^{-i(\pi-\Delta t)/2} & 1 \\ 1 & e^{i(\pi+\Delta t)/2} \end{pmatrix} \quad (C10)$$
$$u_6 = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & e^{-i(\pi+\Delta t)/2} \\ -i & e^{-i\pi/2} \end{pmatrix}. \quad (C11)$$

We implemented the circuit in Fig. 8 using both Rigetti’s noisefree simulator, known as the Quantum Virtual Machine (QVM), as well as their noisy QVM that 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 \( e^{-\mathrm{i}pt} \) was applied was increased.

Figure 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 II 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 \( \rho \) 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. 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. 9 and Table II), due to the doubled number of required CNOT gates relative to \( k = 1 \).

Appendix D: Proof of Eq. (39)

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

\[
\text{Tr}(\hat{Z}_j(\hat{\rho})) = 2^{-H_2(\hat{Z}_j(\hat{\rho}))} \geq 2^{-H(\hat{Z}_j(\hat{\rho}))},
\]

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

\[
H(\hat{Z}_j(\hat{\rho})) = H(\hat{Z}_j(\hat{\rho}_{AB}))
\]
\[
= H(\hat{Z}_j(\hat{\rho}_{AC}))
\]
\[
\leq H(\hat{Z}_j(\hat{\rho}_{A})) + H(\hat{\rho}_{C})
\]

where the inequality in (D4) used the subadditivity of von Neumann entropy. Finally, note that

\[
H(\hat{\rho}_{C}) = H(\hat{\rho}_{AB}) = H(\rho_{AB}) = H(\rho)
\]

and \( H(\hat{Z}_j(\hat{\rho}_{A})) \leq 1 \), which gives

\[
H(\hat{Z}_j(\hat{\rho})) \leq 1 + H(\rho).
\]

Substituting (D6) into (D1) gives

\[
\text{Tr}(\hat{Z}_j(\hat{\rho})) \geq 2^{-1-H(\rho)},
\]

and (39) follows from \( H(\rho) \leq \log_2 r \).