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 $\rho = |+\rangle\langle+| = H|0\rangle\langle 0|H$ 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 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 [1]. We explore optimization methods further in Appendix E.

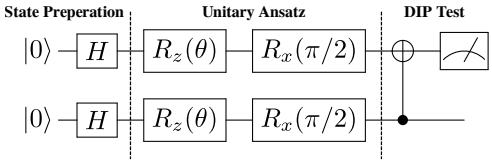

FIG. S.1. Circuit used to implement VQSD for $\rho = |+\rangle\langle+|$ on Rigetti's 8Q-Agave quantum computer. Vertical dashed lines separate the circuit into logical components.

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	$\min(C)$	$\min(\tilde{\lambda}_{\boldsymbol{z}}^{\text{est}})$	$\max(\tilde{\lambda}_{\boldsymbol{z}}^{\text{est}})$
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.

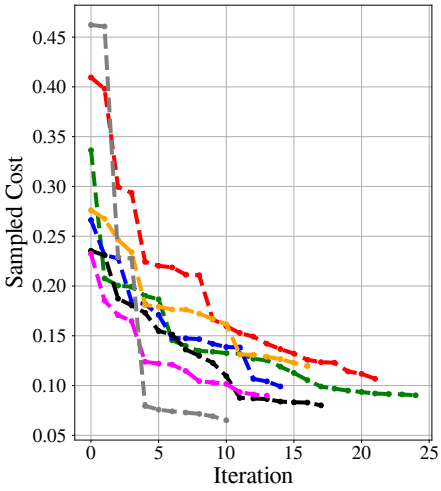

FIG. S.2. Cost vs iteration for all attempts of VQSD on Rigetti's 8Q-Agave computer for diagonalizing the plus state $\rho = |+\rangle\langle+|$. Each of the seven curves represents a different independent run. Each run starts at a random initial angle and uses the Powell optimization algorithm to minimize the cost.

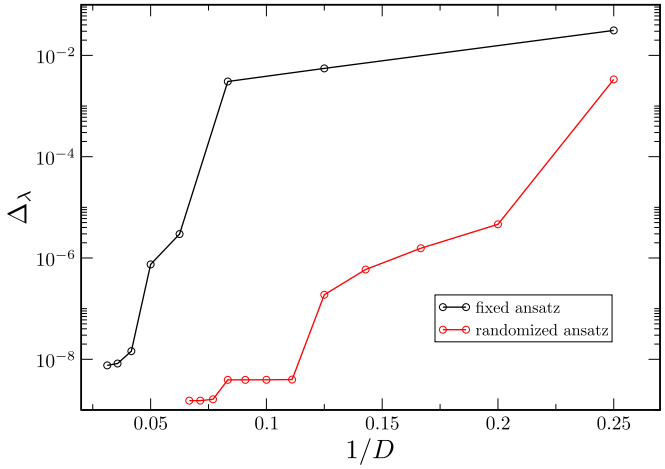

FIG. S.3. Comparison of two approaches to obtaining the diagonalizing unitary $U(\boldsymbol{\alpha})$: (i) based on a fixed layered ansatz shown in Fig. 2 in the main text (black line) and (ii) based on random updates to the structure of $U(\boldsymbol{\alpha})$ (red line). The plot shows eigenvalue 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 [2].

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 [2] 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

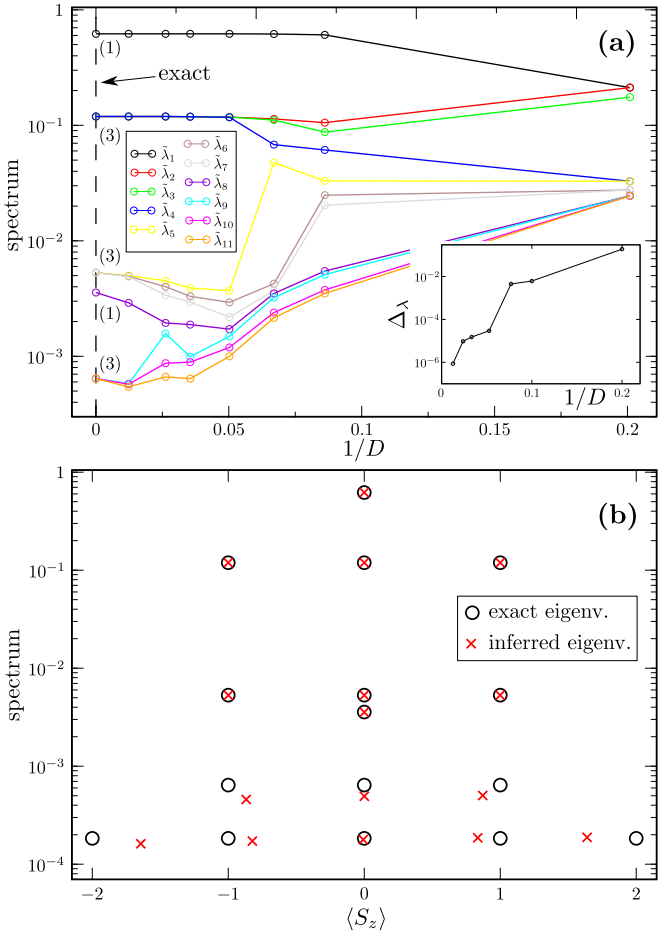

FIG. S.4. VQSD applied to the ground state of the Heisenberg model. Here we consider a 6-qubit reduced state ρ of the 12-qubit ground state. **(a)** Largest inferred eigenvalues $\tilde{\lambda}_j$ of ρ as a function of $1/D$, where D is the total number of gates in the diagonalizing unitary $U(\boldsymbol{\alpha})$. 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.

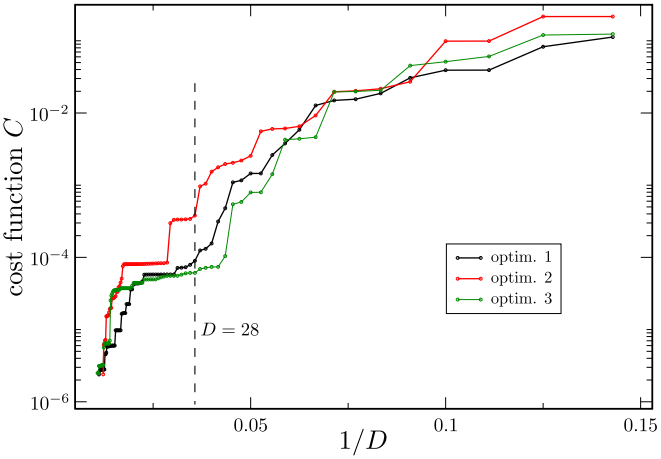

FIG. S.5. Cost function C versus $1/D$ for three independent optimization runs. Here, D is the total number of gates in the diagonalizing unitary $U_D(\boldsymbol{\alpha})$. Every optimization run got stuck at local minimum at some point during the minimization but thanks to the growth of the ansatz for $U_D(\boldsymbol{\alpha})$ described in the text, the predefined small value of C was eventually attained. The data was obtained for a 6-qubit reduced state of the 12-qubit ground state of the Heisenberg model.

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 [3].

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 [4], Constrained Optimization BY Linear Approximation (COBYLA) [5], Bound Optimization BY Quadratic Approximation (BOBYQA) [6], Nelder-Mead [7], Broyden-Fletcher-Goldfarb-Shanno (BFGS) [8], and conjugate gradient (CG) [8]. 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 [9] and in Ref. [10].

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

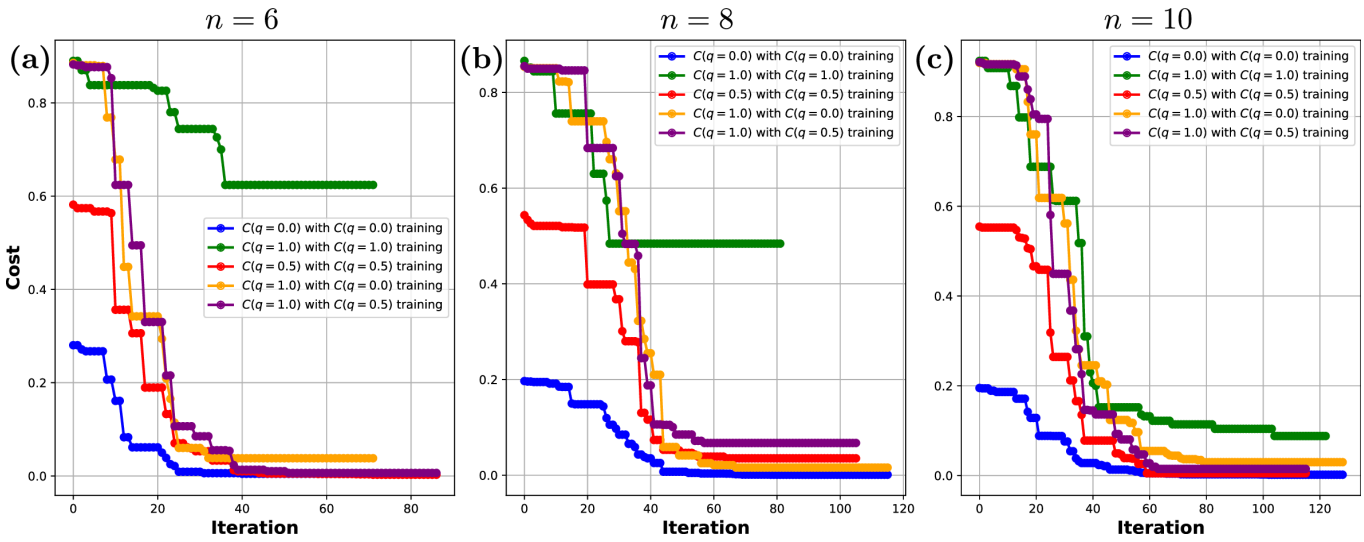

FIG. S.6. Cost versus iteration for different values of q, when ρ is a tensor product of pure states on n qubits. Here we consider (a) $n = 6$, (b) $n = 8$, and (a) $n = 10$. We employed the COBYLA optimization method for training (see Appendix E for discussion of this method). For each call to the quantum 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:

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

Here, the state preparation unitary is

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

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 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-

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.

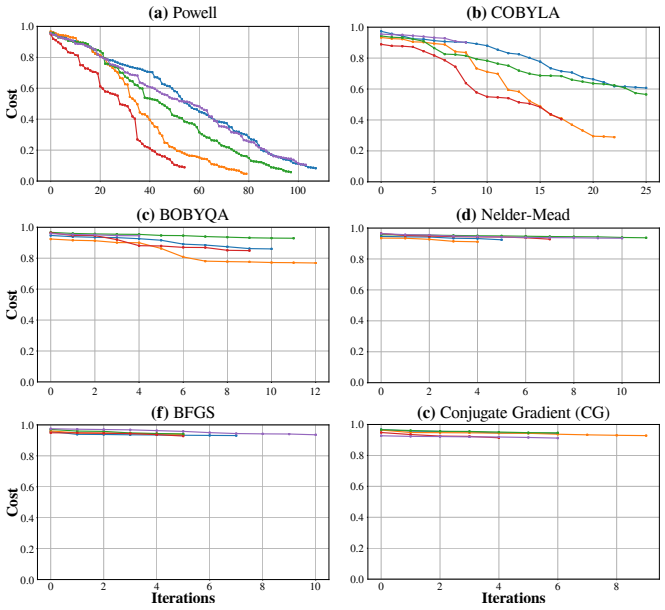

FIG. S.7. Optimization tests on six-qubit product states in the VQSD algorithm. Each plot shows a different optimization algorithm (described in main text) and curves on each plot show optimization attempts with different (random) initial conditions. Cost refers to the C_1 cost function ($q = 1$ in (10)), 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 efficient 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, $\{t, 2t, 2^2t, ..., 2^x t\}$, as part of the phase-estimation algorithm. Here we define $t_{\max} := 2^x t$.

Ref. [11] 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

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

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 $C_{V(t)}$.

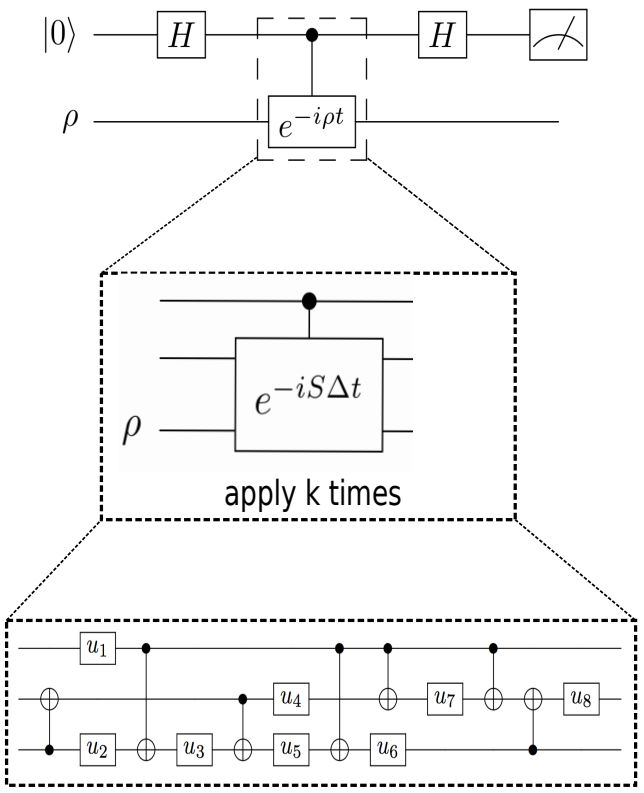

FIG. S.8. Circuit for our qPCA implementation. Here, the eigenvalues of a one-qubit pure state ρ are estimated to a single digit of precision. We use k copies of ρ to approximate $C_{V(t)}$ by applying the controlled-exponential-swap operator k times for a time period $\Delta t = t/k$. The bottom panel 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 $\rho = \sum_{\boldsymbol{z}} \lambda_{\boldsymbol{z}} |v_{\boldsymbol{z}}\rangle\langle v_{\boldsymbol{z}}|$ itself, then the final state becomes

$$\sum_{\boldsymbol{z}} \lambda_{\boldsymbol{z}} |v_{\boldsymbol{z}}\rangle\langle v_{\boldsymbol{z}}| \otimes |\hat{\lambda}_{\boldsymbol{z}}\rangle\langle\hat{\lambda}_{\boldsymbol{z}}| \tag{G5}$$

where $\hat{\lambda}_{\boldsymbol{z}}$ is a binary representation of an estimate of the corresponding eigenvalue $\lambda_{\boldsymbol{z}}$. 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

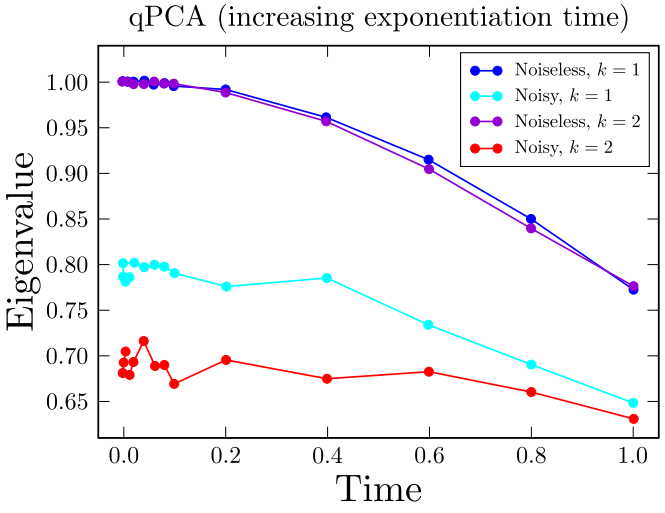

FIG. S.9. The largest inferred eigenvalue for the one-qubit pure state $\rho = |+\rangle\langle+|$ versus application time of unitary $e^{-i\rho t}$, for our implementation of qPCA on Rigetti's noisy and noiseless QVMs. Curves are shown for $k = 1$ and $k = 2$, where k indicates the number of controlled-exponential-swap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

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.

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 arbitary 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 $\mathrm{Tr}(\mathcal{Z}(\sigma)\mathcal{Z}(\tau))$ 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.

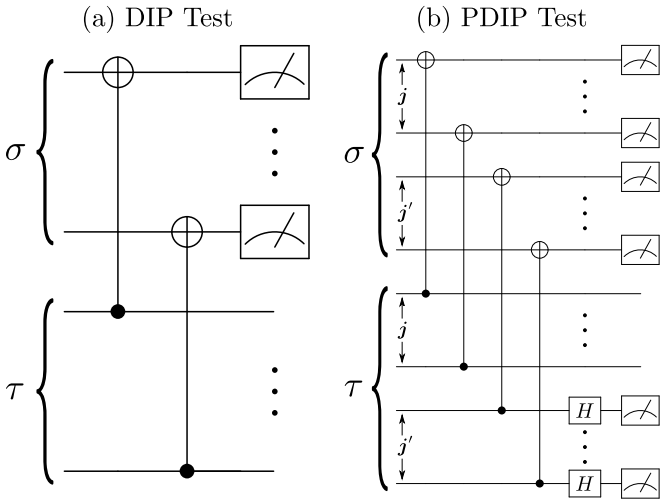

FIG. S.10. Test circuits used to compute the cost function in VQSD. (a) DIP test (b) PDIP test. (These circuits appear in Fig. 5 and are also shown here for the reader's convenience.)

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

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

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

$$\omega'_A = \sum_{\boldsymbol{z}} \tau_{\boldsymbol{z},\boldsymbol{z}} X^{\boldsymbol{z}} \sigma X^{\boldsymbol{z}}\,, \tag{H2}$$

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

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

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, $\mathrm{Tr}(\mathcal{Z}(\sigma)\mathcal{Z}(\tau))$. Note that in the special case where $\sigma = \tau = \tilde{\rho}$, we obtain the sum of the squares of the diagonal elements, $\sum_{\boldsymbol{z}} \tilde{\rho}^2_{\boldsymbol{z},\boldsymbol{z}} = \mathrm{Tr}(\mathcal{Z}(\tilde{\rho})^2)$.

2. PDIP test

We prove that the circuit in Fig. S.10(b) computes $\mathrm{Tr}(\mathcal{Z}_{\boldsymbol{j}}(\sigma)\mathcal{Z}_{\boldsymbol{j}}(\tau))$ 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

$$\omega'_{AB_{\boldsymbol{j'}}} = \sum_{\boldsymbol{z}} (X^{\boldsymbol{z}} \otimes \mathbb{1})\sigma(X^{\boldsymbol{z}} \otimes \mathbb{1}) \otimes \mathrm{Tr}_{B_{\boldsymbol{j}}}((|\boldsymbol{z}\rangle\langle\boldsymbol{z}| \otimes \mathbb{1})\tau), \tag{H4}$$

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:

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

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

$$\sum_{\boldsymbol{z}} \mathrm{Tr}((\sigma^{\boldsymbol{z}}_{\boldsymbol{j'}} \otimes \tau^{\boldsymbol{z}}_{\boldsymbol{j'}})S) = \sum_{\boldsymbol{z}} \mathrm{Tr}(\sigma^{\boldsymbol{z}}_{\boldsymbol{j'}}\tau^{\boldsymbol{z}}_{\boldsymbol{j'}}) = \mathrm{Tr}(\mathcal{Z}_{\boldsymbol{j}}(\sigma)\mathcal{Z}_{\boldsymbol{j}}(\tau))\,. \tag{H6}$$

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 $\mathrm{Tr}(\mathcal{Z}_{\boldsymbol{j}}(\tilde{\rho})^2)$.

Appendix I: Proof of Eq. (40)

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

$$\mathrm{Tr}(\mathcal{Z}_j(\tilde{\rho})^2) = 2^{-H_2(\mathcal{Z}_j(\tilde{\rho}))} \geq 2^{-H(\mathcal{Z}_j(\tilde{\rho}))}\,. \tag{I1}$$

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

$$\begin{aligned} H(\mathcal{Z}_j(\tilde{\rho})) &= H(\mathcal{Z}_j(\tilde{\rho}_{AB})) && \text{(I2)}\\ &= H(\mathcal{Z}_j(\tilde{\rho}_{AC})) && \text{(I3)}\\ &\leq H(\mathcal{Z}_j(\tilde{\rho}_A)) + H(\tilde{\rho}_C) && \text{(I4)} \end{aligned}$$

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

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

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

$$H(\mathcal{Z}_j(\tilde{\rho})) \leq 1 + H(\rho)\,. \tag{I6}$$

Substituting (I6) into (I1) gives

$$\mathrm{Tr}(\mathcal{Z}_j(\tilde{\rho})^2) \geq 2^{-1-H(\rho)}\,, \tag{I7}$$

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

[1] 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).

[2] 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).

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

[4] 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.

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

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

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

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

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

[10] 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).

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