Determining system Hamiltonian from eigenstate measurements without correlation functions

New Journal of Physics
The open access journal at the forefront of physics

Deutsche Physikalische Gesellschaft DPG
IOP Institute of Physics

Published in partnership with: Deutsche Physikalische Gesellschaft and the Institute of Physics

OPEN ACCESS

RECEIVED
23 April 2020

REVISED
27 July 2020

ACCEPTED FOR PUBLICATION
30 July 2020

PUBLISHED
28 August 2020

PAPER

Determining system Hamiltonian from eigenstate measurements without correlation functions

Shi-Yao Hou[1,2,8], Ningping Cao[3,4,8], Sirui Lu[5], Yi Shen[6], Yiu-Tung Poon[7] and Bei Zeng[2]

[1] College of Physics and Electronic Engineering, Center for Computational Sciences, Sichuan Normal University, Chengdu 610068, People's Republic of China
[2] Department of Physics, The Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong, People's Republic of China
[3] Department of Mathematics & Statistics, University of Guelph, Guelph N1G 2W1, Ontario, Canada
[4] Institute for Quantum Computing, University of Waterloo, Waterloo N2L 3G1, Ontario, Canada
[5] Max-Planck-Institut für Quantenoptik, Hans-Kopfermann-Str.1, 85748 Garching, Germany
[6] Department of Statistics and Actuarial Science, University of Waterloo, Waterloo, Ontario, Canada
[7] Department of Mathematics, Iowa State University, Ames, Iowa, IA 50011, United States of America
[8] These authors contributed equally to this work.

E-mail: zengb@ust.hk

Keywords: eigenstate, Hamiltonian determination, gradient based optimization method, no correlation functions

Abstract
Local Hamiltonians arise naturally in physical systems. Despite their seemingly 'simple' local structures, exotic features such as non-local correlations and topological orders exhibit in eigenstates of these systems. Previous studies for recovering local Hamiltonians from measurements on an eigenstate $|\psi\rangle$ require information of nonlocal correlation functions. In this work, we argue that local measurements on $|\psi\rangle$ is enough to recover the Hamiltonian in most of the cases. Specially, we develop an algorithm to demonstrate the observation. Our algorithm is tested numerically for randomly generated local Hamiltonians of different system sizes and returns promising reconstructions with desired accuracy. Additionally, for random generated Hamiltonians (not necessarily local), our algorithm also provides precise estimations.

1. Introduction

The principle of locality, arising naturally in physical systems, states that objects are only affected by their nearby surroundings. Locality is naturally embedded in numerous physical systems characterized by local Hamiltonians, which plays a critical role in various quantum physics topics, such as quantum lattice models [1–3], quantum simulation [4–7], topological quantum computation [8], adiabatic quantum computation [9–11], and quantum Hamiltonian complexity [12–14]. In the past few years, rapidly developing machine learning techniques allow us to study these topics in a new manner [15–20]. Empowered by traditional optimization methods and contemporary machine learning techniques, we map the task of revealing the information encoded in a single eigenstate of a local Hamiltonian to an optimization problem.

For a local Hamiltonian $H = \sum_i c_i A_i$ with A_i being some local operators, it is known that a single (non-degenerate) eigenstate $|\psi\rangle$ can encode the full Hamiltonian H in certain cases [21–23], such as when the expectation value of A_is on $|\psi\rangle$, $a_i = \langle\psi| A_i |\psi\rangle$, are given and further assumptions are satisfied. A simple case is that when $|\psi\rangle$ is the unique ground state of H; thus the corresponding density matrix of $|\psi\rangle$ can be represented in the thermal form as

$$|\psi\rangle\langle\psi| = \frac{e^{-\beta H}}{\mathrm{tr}(e^{-\beta H})} \tag{1}$$

for sufficiently large β. This implies that the Hamiltonian H can be directly related to $|\psi\rangle$, and hence can be determined by the measurement results a_is, using algorithms developed in the literature [24, 25] for

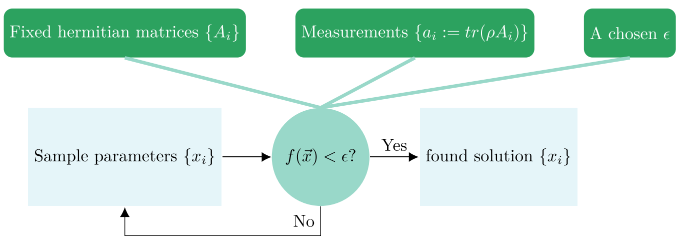

Fixed hermitian matrices $\{A_i\}$
Measurements $\{a_i := tr(\rho A_i)\}$
A chosen ϵ
Sample parameters $\{x_i\}$
$f(\vec{x}) < \epsilon$?
Yes
found solution $\{x_i\}$
No

Figure 1. Diagram of our algorithm: The rounded boxes (the first row) represent known information—the fixed set of Hermitian operators $\{A_i\}$, measurement outcome $\{a_i\}$ and a chosen precision parameter ϵ. The information in hand contributes to the objective function f (equation (6)). The second row demonstrates the procedure of the algorithm: first, sample a set of random parameters $\vec{x} = \{x_i\}$; second, estimate the objective function $f(\vec{x})$ and determine whether $f(\vec{x})$ is smaller then the chosen ϵ; update $\vec{x}$ until it satisfies $f(\vec{x}) < \epsilon$.

practical cases. Because A_is are local operators, the number of parameters of H (i.e., the number of c_is) is only polynomial in terms of system size. We remark that the problem of finding H is also closely related to the problem of determining quantum states from local measurements [15, 16, 26–33], and also has a natural connection to the study of quantum marginal problem [34–37], as well as its bosonic/fermionic version that are called the N-representability problem [34, 38–44].

For a wavefunction $|\psi\rangle$ that is an eigenstate (i.e. not necessarily a ground state), one interesting situation is related to the eigenstate thermalization hypothesis (ETH) [45–49]. When the ETH is satisfied, the reduced density matrix of a pure and finite energy density eigenstate for a small subsystem becomes asymptotically equal to a thermal reduced density matrix [21]. In other words, equation (1) will hold for some eigenstate $|\psi\rangle$ of the system in this case, and one can use a similar algorithm [24, 25] to find H from a_is, as in the case of ground states. Another situation previously discussed is that if the two-point correlation functions $\langle\psi| A_iA_j |\psi\rangle$ are known, one can reproduce H without satisfying ETH [22, 23]. Once the two-point correlation functions are known, one can again use an algorithm to recover H from the correlation functions, for the case of ground states. However, in practice, the nonlocal correlation functions $\langle\psi| A_iA_j |\psi\rangle$ are not easy to obtain [50].

In this paper, we answer a simple but significant question: can we determine a local Hamiltonian (c_is) from only the local information (a_is) of any one of the eigenstates ($|\psi\rangle$) without further assumptions. Obviously, there are cases that an eigenstate cannot determine the system Hamiltonian, such as the product state and the eigenstates of a frustration-free Hamiltonians. However, for a randomly chosen physical system for which the Hamiltonian has no special structures, we show that only the knowledge of a_is is sufficient to determine H.

Based on the available information—a set of possible Hermitian operators $\{A_i\}$ for the system Hamiltonian and the measurement outcomes $\{a_i\}$, we formulate a positive-semidefinite function $f(\vec{x})$ (equation (6)), with the $f(\vec{c}) = 0$ for the desired $\vec{c}$. The problem of finding the exact Hamiltonian converts to an unconstrained optimization problem of finding $\vec{x}$ that minimizing f. A small real number ϵ is chosen to control the precision of the result. We update the sampled $\vec{x}$ until $f(\vec{x})$ is less than ϵ. Figure 1 depicts the procedure of our algorithm.

We test the algorithm in two scenarios: one with randomly generated operator A_is acting on the whole system, and one with random local operators A_is. Our algorithm almost perfectly reproduces c_is, that is, the average fidelities for both scenarios are close to 1. Since the algorithm recovers Hamiltonians with almost perfect fidelities based on the reconstructed Hamiltonian H (i.e. c_is), one can also recover the eigenvalue of $|\psi\rangle$ from c_is and a_is, and the wave function itself from the eigenvectors of Hamiltonian H. In the case when A_is are local operators, our method can find c_is from only local measurement results, hence shed light on the correlation structures of eigenstates of local Hamiltonians.

2. Algorithm

We start to discuss our method in a general situation, where the Hamiltonian H can be expressed in terms of a set of known Hermitian operators $\{A_1, A_2, \ldots, A_m\}$: $H = \sum c_iA_i$ with $(c_1, c_2, \ldots, c_m) = \vec{c}$. For an eigenstate $|\psi\rangle$ of H with unknown eigenvalue λ, which satisfies $H|\psi\rangle = \lambda|\psi\rangle$, we can denote the measurement results as $a_i = \langle\psi|A_i|\psi\rangle$. With only knowing the measurement results a_is, our goal is to find the coefficients $\vec{c}$ to determine H.

We observe that, even if $|\psi\rangle$ is not the ground state of H, it can be the ground state of another Hamiltonian $\tilde{H}^2$ with $\tilde{H}$ given by

$$\tilde{H} = H - \lambda I = \sum_i c_i (A_i - a_i I), \tag{2}$$

since

$$\langle\psi| H |\psi\rangle = \lambda = \sum_i c_i \langle\psi| A_i |\psi\rangle = \sum_i c_i a_i. \tag{3}$$

Then the density matrix of $|\psi\rangle$ (which is in fact of rank 1) can be written in the form of a thermal state:

$$\rho(\vec{c}) = |\psi\rangle\langle\psi| = \frac{\mathrm{e}^{-\beta\tilde{H}^2}}{\mathrm{tr}(\mathrm{e}^{-\beta\tilde{H}^2})} \tag{4}$$

for sufficiently large β, satisfying

$$\mathrm{tr}(A_i\rho(\vec{c})) = a_i, \quad \mathrm{tr}(\tilde{H}^2\rho(\vec{c})) = 0. \tag{5}$$

With these conditions in mind, we are ready to reformulate our task as an optimization problem with the following objective function:

$$f(\vec{x}) = \sum_{i=1}^{m} [\mathrm{tr}(A_i\rho(\vec{x})) - a_i]^2 + \mathrm{tr}(\tilde{H}^2\rho(\vec{x})), \tag{6}$$

where $\vec{x}$ is the estimation of $\vec{c}$, and $\vec{x} = \vec{c}$ when $f(\vec{x})$ is minimized.

Notice that the first term of $f(\vec{x})$ is minimized by $\mathrm{tr}(A_i\rho(\vec{c})) = a_i$, which guarantees that the state $\rho(\vec{c})$ obtained is the state that produces the desired measurement outcomes on A_is. However, there may be many (thermal) states which also yield such outcomes. By simply minimizing this first term, optimization algorithms tend to return thermal states $\rho(\vec{c})$ with nonzero entropy, which is not the eigenstate of H (with entropy zero) that we are willing to find. In order to fix this issue and ensure that the optimization returning a (nearly) rank 1 state $\rho(\vec{c})$, we add the second term, which is only zero when $\rho(\vec{c})$ is the ground state of $\tilde{H}^2$, hence an eigenstate of H. Combining these two terms together, we make sure that when the minimum value of Hamiltonian $f(\vec{x})$ is reached, we will obtain a $\rho(\vec{c})$ corresponding to measurement outcomes a_i, and at the same time an eigenstate of H. In practice, we set up a parameter ϵ, such that with $f(\vec{x}) < \epsilon$, we find a result with high fidelity to the desired value of $\{c_i\}$.

For the convenience of numerical implementation, we let $\tilde{H}_\beta = \sqrt{\beta}\tilde{H}$, then the thermal state $\rho(\vec{c})$ becomes

$$\rho(\vec{c}) = \frac{\mathrm{e}^{-\tilde{H}_\beta^2}}{\mathrm{tr}(\mathrm{e}^{-\tilde{H}_\beta^2})}. \tag{7}$$

Consequently, the objective function equation (6) can be rewritten in an equivalent form

$$f(\vec{x}) = \sum_{i=1}^{m} [\mathrm{tr}\,(A_i \cdot \rho(\vec{x})] - a_i)^2 + \mathrm{tr}(\tilde{H}_\beta^2 \cdot \rho(\vec{x})). \tag{8}$$

We aim to solve for $f(\vec{x}) = 0$ by minimizing $f(\vec{x})$. In practice, we terminate our iterations when $f(\vec{c})$ is smaller than a fixed small value ϵ. The corresponding optimization result is denoted as $\vec{c}_{\mathrm{opt}}$. As we reformed the objective function by using $\tilde{H}_\beta$, the result is actually $\vec{c}_{\mathrm{opt}} = \sqrt{\beta}\cdot\vec{c}$.

Theoretically, we need β to be 'infinity' to pick up the ground state of $\tilde{H}_\beta^2$. In practice, however, we only require that β is some 'large number'. What is more, to pick up the ground state of $\tilde{H}_\beta^2$ for $\rho(\vec{c})$, what really matters is in fact the gap between the first excited state and the ground state of $\tilde{H}_\beta^2$. Therefore, we just simply set $\beta = 1$ and let the optimizer automatically amplify the energy gap during iteration, when $\rho(\vec{c})$ is approaching the desired state. A more detailed discussion regarding the choice of β can be found in appendix A.

Since all the constraints are written in equation (8), minimizing $f(\vec{c})$ is an unconstrained minimization problem. There are plenty of standard algorithms for this task. In our setting, computing the second-order derivative information of $f(\vec{c})$ is quite complicated and expensive. Therefore, instead of using Newton method which requires the Hessian matrix of the second derivatives of the objective function, we choose the quasi-Newton method with BFGS-formula to approximate the Hessians [51–54]. The MATLAB function FMINUNC, which uses the quasi-Newton algorithm, is capable of realizing this algorithm starting from an initial random guess of c_is. When the quasi-Newton algorithm fails (converges into a local minimum), we start with a different set of random initial value c_is and optimize again until we obtain a good enough solution.

The BFGS algorithm is a typical optimization algorithm which requires gradients of the objective function on c_is. The form of the objective function $f(\vec{c})$ is so complicated such that computing the gradient is a difficult task. To solve this issue, we borrow the methods of computational graph and auto differentiation from machine learning. The computational graph is shown in figure 10, in which we show the intermediate functions and variables. Mathematically, the final gradients could be calculated via chain rules. However, since some of the intermediate variables are matrices and complex-valued, the automatic differentiation toolboxes, which deal with real variables, can not be applied directly. To obtain the gradients, we have to careful handle the derivation of the intermediate functions, especially those with matrices as their variables. More details about the our gradient method can be found in appendix B.

We use Matlab to implement our algorithm, and the detailed implementation can be found in appendix C.

3. Results

In this section, we test our algorithm in three steps as follows. First, we randomly generate several A_is and c_is, hence the Hamiltonian H_{rd} and its eigenstates. Second, for each Hamiltonian, we randomly choose one eigenstate $|\psi\rangle$, therefore we have $a_i = \langle\psi|A_i|\psi\rangle$ and $\rho = |\psi\rangle\langle\psi|$. Hereby we can run our algorithm to find the Hamiltonian H_{al}. Comparing H_{al} and H_{rd}, we then know that how well the approach works. The algorithm has been tested for two scenarios: A_i being the generic operator and local operator.

To compare H_{al} and H_{rd}, we need a measure to characterize the similarity, or *distance* between these two Hamiltonians. The metric we used here is the following fidelity as discussed in [55]:

$$f(H_{\mathrm{al}}, H_{\mathrm{rd}}) = \frac{\mathrm{tr}\ H_{\mathrm{al}}H_{\mathrm{rd}}}{\sqrt{\mathrm{Tr}\ H_{\mathrm{al}}^2}\sqrt{\mathrm{Tr}\ H_{\mathrm{rd}}^2}}. \tag{9}$$

To see the meaning of this metric, notice that $\mathrm{tr}H_{\mathrm{al}}H_{\mathrm{rd}}$ is the inner product of the two Hamiltonians, while $\sqrt{\mathrm{Tr}\ H_{al}^2}$ and $\sqrt{\mathrm{Tr}\ H_{rd}^2}$ are the two normalization constants. Therefore, $f(H_{\mathrm{al}}, H_{\mathrm{rd}}) \in [0, 1]$. If H_{al} and H_{rd} describe the same system up to a constant $b \in \mathbb{R}$, then $f(H_{\mathrm{al}}, H_{\mathrm{rd}}) = 1$. Smaller values of $f(H_{\mathrm{al}}, H_{\mathrm{rd}})$ indicate that the two Hamiltonians are far apart. Moreover, notice that in our settings the Hamiltonians are represented by vectors $\vec{c}$ and $\vec{c}'$ in m-dimensional real space. If the chosen A_is are normalized and orthogonal, which means

$$\mathrm{tr}(A_iA_j) = d\delta_{ij}, \tag{10}$$

where d is the normalization constant given by the system dimension (e.g. for Pauli matrices, $d = 2$), this fidelity definition is exactly the cosine loss function of $\vec{c}$ and $\vec{c}\,'$, where $\vec{c}\,'$ is generated from our algorithm, that is, for normalized A_is,

$$f(H_{\mathrm{al}}, H_{\mathrm{rd}}) = \frac{\mathrm{Tr}\ H_{\mathrm{al}}H_{\mathrm{rd}}}{\sqrt{\mathrm{Tr}\ H_{\mathrm{al}}^2}\sqrt{\mathrm{Tr}\ H_{\mathrm{rd}}^2}} = \frac{\vec{c}\cdot\vec{c}\,'}{||\vec{c}||\,||\vec{c}\,'||}, \tag{11}$$

where $||\vec{c}||$ means the two-norm of the vector $\vec{c}$.

3.1. Results with general operators

When implementing our method on generic operators, we randomly generate three Hermitian operators A_is and fix them. We also create several real constants c_i randomly, then assemble A_is and c_is into Hamiltonians

$$H = c_1A_1 + c_2A_2 + c_3A_3. \tag{12}$$

After diagonalizing H, we choose one eigenvector $|\psi\rangle$ and calculate the expectation values $a_i = \langle\psi|A_i|\psi\rangle$.

For n-qubit systems, the dimension of the system is $d = 2^n$. We tested the cases for $n = 4, 5, 6$ and 7. For each n, we generate 200 data points. Each test set is $\{c_i, A_i, a_i | i = 1, 2, 3\}$, where $c_i \in (0, 1)$ for $i = 1, 2, 3$ and A_is are randomly generated Hermitian matrices of dimension $d = 2^n$ [56]. With this data, $H_{\mathrm{rd}} = \sum_i c_iA_i$ is obtained. Diagonalizing H_{rd} and randomly choosing one eigenvector $|\psi\rangle$, we obtain $\{a_1, a_2, a_3\}$. We show the results for applying our algorithm to all cases in figure 2.

We find that the final fidelities are larger than 99.8% for all tested cases. Although the fidelity slightly decreases with the increasing number of qubits, the lowest fidelity (seven-qubit case) is still higher than 99.8%. Because the eigenvector $|\psi\rangle$ is randomly chosen, our method is not dependent on the energy level of eigenstates.

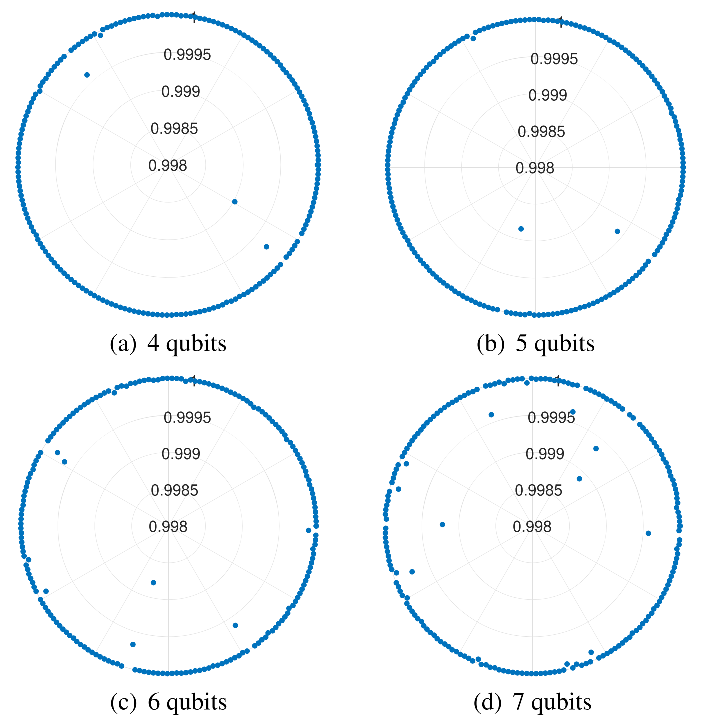


Figure 2. Results for generic operators. Each dot represents the fidelity f of a test data point. (a) 4 qubits. (b) 5 qubits. (c) 6 qubits. (d) 7 qubits.

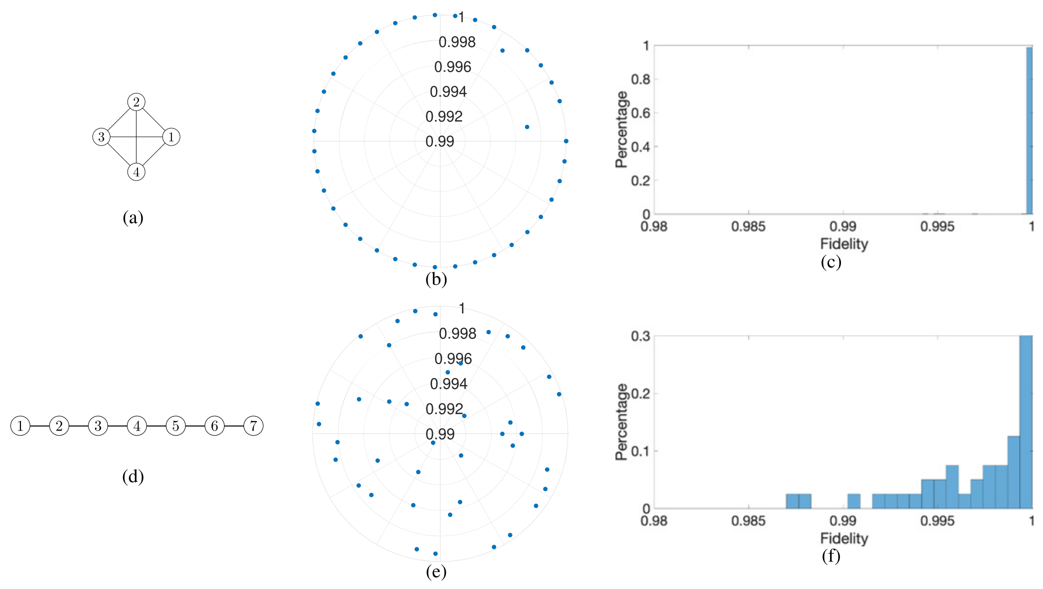


Figure 3. Configurations and results of tested local Hamiltonians (a) structure of a four-qubit full connected lattice. (b) The fidelity f between random four-qubit fully connected lattice Hamiltonians H_{rd} and Hamiltonian obtained in our algorithm H_{al}. (c) The fidelity distribution of four-qubit systems. (d) Structure of a seven-qubit chain lattice. (e) The fidelity f between seven-qubit two-local Hamiltonians H_{rd} and Hamiltonian estimation H_{al} of our algorithm. (f) The fidelity distribution of our seven-qubit systems.

3.2. Results with local operators

In this section, we report our results on the systems with a two-local interaction structure. We tested two different structures, shown in figures 3(a) and (d). Each circle represents a qubit on a lattice, and each line represents an interaction between the connected two qubits.

The fully-connected four-qubit system is shown in figure 3(a). The Hamiltonian can be written as

$$H = \sum_{1\leqslant j,i\leqslant 4} \sum_{i<j} c_{ij} A_{ij}, \qquad (13)$$

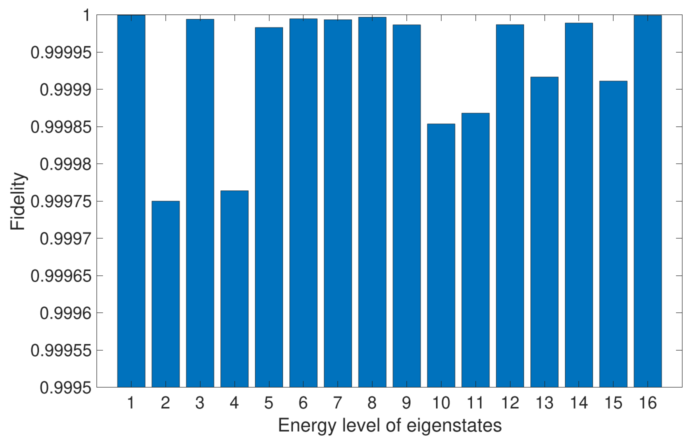


Figure 4. The average fidelities f by applying our algorithm to different energy levels of eigenstates of the four-qubit case. The average fidelities for different energy levels of eigenstates are all higher than 99.97%.

where c_{ij}s are real parameters and A_{ij}s are random generated two-local operators. One eigenstate out of 16 is randomly chosen as the state $|\psi\rangle$. Our algorithm has been tested on 800 such Hamiltonians.

We then analyze the seven-qubit chain model shown in figure 3(d). Similarly, we can write the Hamiltonian as

$$H = \sum_{1\leqslant i\leqslant 6} c_{i,i+1}A_{i,i+1}, \tag{14}$$

where $c_{i,i+1}$s and $A_{i,i+1}$s are the parameters and two-local interactions. We randomly generate 40 such Hamiltonians and applied our algorithm.

The results of these two two-local Hamiltonians are shown in figure 3. Our algorithm recovered Hamiltonians with high fidelities for both cases. The average fidelity for our four-qubit (seven-qubit) system is 99.99% (99.73%). As the dimension of the system increases, the fidelity between H_{rd} and H_{al} slightly decreased. The histogram of the fidelities shows that, for most data points, the fidelities are very close to 1. Our algorithm almost perfectly recovered these two-local Hamiltonian from measurement outcomes of a randomly picked eigenvector.

We examine the effectiveness of our algorithm according to different eigenvectors of the same Hamiltonians. Figure 4 demonstrates average fidelities between H_{rd} and H_{al} of four-qubit case by energy level of eigenstates. These average fidelities are higher than 99.9% for all 16 eigenvectors. Hence, the effectiveness of our algorithm is independent of energy levels.

4. Further analysis of the algorithm

In this section, we analyze the error tolerance and the performance of the algorithm, based on the results of our numerical experiments.

4.1. Error tolerance analysis

The numerical tests in previous sections deal with noiseless theoretical data. In practical scenarios, however, data is always noisy. Here we provide analysis of error tolerance for our algorithm.

As an example, we consider a four-qubit system with Hamiltonian $H = \sum_{i=1}^{3} c_i A_i$ where A_i's are random generated 16 by 16 Hermitian operators. Choose one eigenstate $|\psi\rangle$ of H, the noiseless measurements of the eigenstate are denoted as $\{a_i | a_i = \langle\psi| A_i |\psi\rangle\}$. Noises used here are randomly drawn from normal distributions $\gamma \sim \mathcal{N}(0, \sigma^2)$. Adding the generated noise γ to measurements a_i, the noisy data $a_{i,m}$ follows the normal distribution $\mathcal{N}(a_i, \sigma^2)$.

We can control how noisy $a_{i,m}$ is by changing the standard deviation σ. Clearly, the noisiness of data is relative to the magnitude of true values, thus σ/a_i can be used as an indicator. We test ten different σ's of which σ/a_i's ranging from 0.01 (1%) to 0.1 (10%). For each σ, 1000 data points have been generated.

The Hamiltonians attained by our algorithm with these noisy data $a_{i,m}$ are denoted as H_{err}. The fidelity between each pair of the true Hamiltonian H and H_{err} is shown in figure 5. Though increasing the noise causes the fidelities of a few points to significantly decrease, all median fidelities of different σ are above 99.8%. Even for $\sigma/a_i = 0.1$ (added 10% of noise), only 2.1% of data points have fidelities lower than

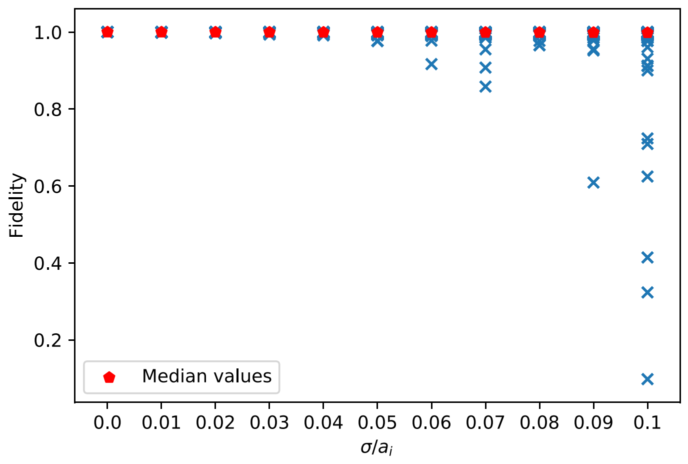


Figure 5. Fidelities of noisy data: Each σ/a_i (σ) has 1000 test points. Blue cross marks represent the fidelities between Hamiltonians recovered with noisy data H_{err} and real Hamiltonians H . The red pentagons are median values of fidelities for each σ/a_i.

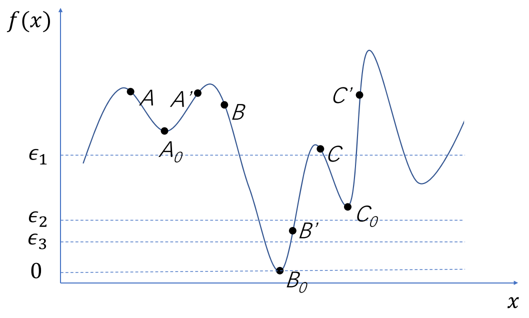


Figure 6. The schematic diagram for demonstrate the influences of local minima. A_0 and C_0 are local minima of the object function $f(x)$. B_0 is the global minimum. A slightly larger ϵ may cause the iteration sticks in the local minima.

99.0%. In other words, singular points that have significant low fidelities are rare. This demonstrates that the performance of our algorithm is stable.

4.2. Performance analysis

The convergence and time cost of our method are analyzed in this subsection. We study the relationship between the halting condition ϵ and the convergence of our algorithm. We also observed that the time cost tends to differ for each Hamiltonian configuration.

The parameter ϵ, which is the halting condition of the algorithm, affects the convergence of our algorithm. It is the consequence of the object function $f(\vec{x})$ has many local minima. In the schematic diagram figure 6, if the initial point is chosen as A or A', the gradient method will return the value A_0, which is a local minimum (so as B or B' to B_0, and C or C' to C_0).

When ϵ is greater then $\|f(A_0) - f(B_0)\|$ or $\|f(C_0) - f(B_0)\|$, the algorithm may recognize A_0 or C_0 as the optimal solution instead of finding the true global minimum B_0. An appropriate choice of ϵ is necessary to eliminate certain local minima.

The objective function equation (6) used in our method has a high dimension and complicated landscape. Its properties are also subject to the particular class of H that we work with. More analysis could be done in terms of finding an appropriate ϵ. Empirically, we choose ϵ as 10^{-6} for all our numerical experiments. It is numerically proved to be applicable as shown in figure 7. The figure depicts the relation between ϵ and the output fidelity of the three-qubit general operator's case. Chosen $\epsilon \leqslant 10^{-3}$ grantees the average fidelity almost equals to 1.

By setting the halting parameter ϵ to 10^{-6}, we can now discuss the time cost for each numerical experiment. The total time cost t for optimization depends on the number of trials n for each numerical example and the time t_0 of each trial. The initial values are chosen randomly, n is different from case to case.

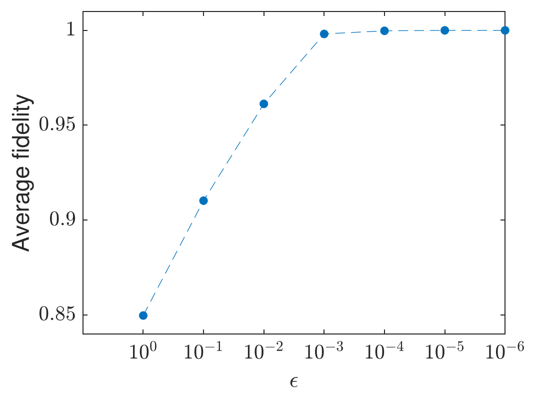


Figure 7. The impact of ϵ on the final fidelity (the three-qubit general operator scenario). For each ϵ, we conduct 50 numerical experiments, then take the average of outcome fidelities. The vertical axis is the average fidelity and the horizontal axis is the value of ϵ.

Table 1. Time consumed of each trial and the number of trials. For general operator cases and the 4-qubit local operators case, we used 200 examples to achieve the averages. For 7-qubit local operators, The result is obtained from 40 examples.

	General operators				Local operators	
Number of qubits	4	5	6	7	4	7
Time for single trial t_0(s)	0.0487	0.0518	0.0539	0.0579	0.2246	7.938
Number of trials $\bar{n}$	32.36	42.34	45.89	161.6	54.84	539.4
Average time cost ($\bar{t}$) (s)	1.576	2.193	2.377	2.4735	12.32	4282

We consider the average number of trials $\bar{n}$ for each Hamiltonian class. Let t_0 be the duration of a single trial which is mainly depending on the Hamiltonian configuration and number of qubits. The total average time $\bar{t}$ could be estimated as

$$\bar{t} = \bar{n}t_0. \tag{15}$$

We test our algorithm on a workstation with Intel i7-8700K and 32 GB RAM, the results are listed in table 1. The table demonstrates that the time cost does not grow rapidly for the general operator cases, while it changes dramatically with the system size for the local operator cases. The $\bar{t}$ of seven-qubit general operator instance is almost 2 times more than $\bar{t}$ of the four-qubit general operator case. On the contrast, the average time cost $\bar{t}$ of seven-qubit lattice is almost 350 times more than the $\bar{t}$ of four-qubit lattice.

5. Comparison with the correlation matrix method

In reference [22], a method is proposed to recover Hamiltonians from correlation functions. Their method works as follows: with an eigenstate $|\psi\rangle$, the Hamiltonian of the system is defined as

$$H = \sum_i c_i L_i, \tag{16}$$

where L_is are normalized and orthogonal Hermitian matrices (e.g. the Pauli matrices). They defined a correlation matrix, of which the matrix elements are

$$M_{ij} = \frac{1}{2}\langle\psi|\{L_i, L_j\}|\psi\rangle - \langle\psi|L_i|\psi\rangle\langle\psi|L_j|\psi\rangle, \tag{17}$$

where $\{L_i, L_j\} = L_iL_j + L_jL_i$ is the anti-commutator. Then diagonalize matrix M and find the eigenvector $\vec{\omega} = (\omega_1, \omega_2, \ldots, \omega_i)^T$ corresponding to the eigenvalue 0. The Hamiltonian

$$H_{\text{cor}} = \sum \omega_i L_i \tag{18}$$

is the desired Hamiltonian of the system H. We refer to this algorithm as the correlation matrix method (CMM).

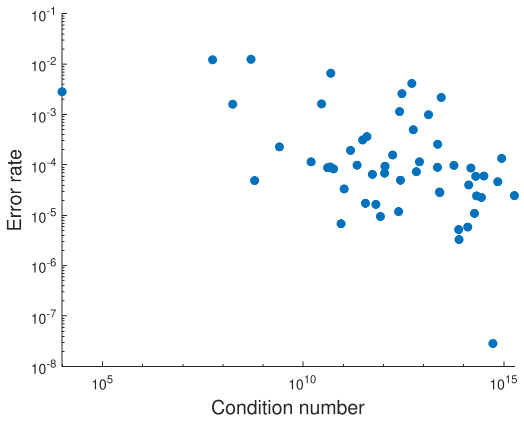


Figure 8. The impact of condition number on our algorithm. The x axis is the condition number. The y axis is the error rate $r(H_{\text{rd}}, H_{\text{al}}) = 1 - f(H_{\text{rd}}, H_{\text{al}})$.

We conduct numerical experiments to compare the performance of CMM and our method. We calculate such four-qubit Hamiltonians

$$H = \sum_{i=1}^{4} a_i \sigma_z^i + \sum_{i=1}^{3} b_i \sigma_x^i \sigma_x^{i+1}, \tag{19}$$

where σ_k^i is the kth Pauli matrix acting on the ith qubit. It turns out that CMM and our algorithm both render good estimations. The results of CMM possess greater accuracy–the error rate, defined as $r(H_{\text{rd}}, H_{\text{al}}) = 1 - f(H_{\text{rd}}, H_{\text{al}})$, is less than 10^{-10}. Error rates of our results range from 10^{-4} to 10^{-2}.

CMM is deterministic, more accurate as well as faster than our method. It only involves diagonalization of matrix M, of which the dimension is polynomial of the number of qubits for local systems. CMM also provides a criteria to decide whether the eigenstate is uniquely determining the Hamiltonian. That is, if M only has one eigenvalue equals to 0, then the Hamiltonian is uniquely determined. The advantage of CMM arises from more restrictions and more information required for applying it: (1) All L_is in CMM are orthogonal, while in our tests the corresponding A_is are randomly generated; (2) CMM requires non-local correlation functions. The matrix element includes the term

$$\langle \psi | \{L_i, L_j\} | \psi \rangle, \tag{20}$$

where $\{L_i, L_j\} = L_i L_j + L_j L_i$. Although L_is may be local, $L_i L_j$ for all is and js could be global. Therefore, as indicated in [22], if only partial knowledge of the system is available, the question that whether the Hamiltonian can be reconstructed remains unclear.

Another problem of applying CMM is similar to the problem of the halting parameter ϵ in our method. The recovery depends on the existence of one eigenvalue equals to 0. However, the equivalence of two numbers in a computer is different from in theory. Even when working with noiseless theoretical data, the finite length data storage (e.g. the precision of double float-point format is 10^{-16}) and data processing can introduce certain errors, not to mention the noisy data from real quantum devices. Namely, determine whether a number is equal to 0 is up to a certain precision. Therefore, one needs to set a threshold δ to determine equivalence before applying CMM. How to appropriately chose a δ is empirical and case dependent. From this perspective, both methods are similarly complicated.

We are also willing to check the consistency of CMM and our methods. Lacking a systemic way of choosing the threshold δ, we need to seek another way to bridge these two methods. From the numerical experiments, we find in CMM that the lowest absolute eigenvalue (denotes as λ_0) of M is about 10^{-16} to 10^{-18} and the second-lowest absolute eigenvalue (denotes as λ_1) ranges from 10^{-2} to 10^{-12}. If λ_1 is too small, it would be hard to tell whether there exists one or more 0 eigenvalues, that is, the farther λ_0 and λ_1 are, the more accurate the CMM's outcome is. It will be evidence of consistency if we can witness the same tendency from our method. We, therefore, define the *condition number* as the ratio of the second-lowest absolute eigenvalue to the lowest absolute eigenvalue (λ_1/λ_0). We use the estimations provided by our method to construct the correlation matrix M, obtain the λ_0 and λ_1 from M, then calculate the condition number λ_1/λ_0. The results are shown in figure 8. We find, generally, the larger the condition number, the better our algorithm performs (higher fidelity). It is consistent with the behavior of CMM.

In summary, for all the tests performed, CMM and our method have similar behavior in terms of the condition number, which shows that both method return the desired results (for reconstruction the desired

Hamiltonian with high fidelity). Regarding algorithm performance, when all sorts of correlations are available, the CMM is numerically more efficient and accurate. In contrast, our algorithm uses less information, therefore can be used in much wider situations, especially when certain global correlation information is hard to obtain in practice.

6. Discussion

In this work, we discuss the problem of reconstructing a system Hamiltonian $H = \sum_i c_i A_i$ using only measurement data $a_i = \langle\psi| A_i |\psi\rangle$ of one eigenstate $|\psi\rangle$ of H by reformulating the task as an unconstrained optimization problem of some target function of c_is. We numerically tested our method for both randomly generated A_is and also the case that A_is are random local operators. In both cases, we obtain good fidelities of the reconstructed H_{al}. Our results are somewhat surprising: only local measurements on one eigenstate are enough to determine the system Hamiltonian for generic cases, no further assumptions are needed. Though it is beautiful theoretically, our algorithm is not scalable since the calculations of exponentiation and gradient of matrices become expensive when the system size gets large. Improvements can be made by more efficiently approximating them.

We also remark that, in the sense that our method almost perfectly recovered the Hamiltonian H, the information encoded in the Hamiltonian H, such as the eigenstate $|\psi\rangle$ itself (though described exponentially many parameters in system size) can also in principle be revealed. This builds a bridge from our study to quantum tomography and other related topics of local Hamiltonians. Empowered by traditional optimization methods and machine learning techniques, our algorithm could be applied in various quantum physics problems, such as quantum simulation, quantum lattice models, adiabatic quantum computation, etc.

Our discussion also raises many new questions. For instance, a straightforward one is whether other methods can be used for the reconstruction problem, and their efficiency/stability compared to the method we have used in this work. One may also wonder how the information of ‘being an eigenstate’ helps to determine a quantum state locally, which is generically only the case for ground states, and how this information could be related to help quantum state tomography in a more general setting.

Acknowledgment

We thank Feihao Zhang, Lei Wang, Shilin Huang, and Dawei Lu for helpful discussions. S-YH is supported by National Natural Science Foundation of China under Grant No. 11847154. NC is supported by NSERC.

Appendix A. The value of β

It is easy to observe that $\tilde{H}^2$ and $\tilde{H}^2_\beta$ have the same eigenstates. The constant β only contributes a constant factor to the magnitude of eigenvalues, i.e., the eigenenergies of the given system. Theoretically, a thermal state tends to the ground state of the given Hamiltonian if and only if the temperature is zero (or, for a numerical algorithm, close to zero), which means $\beta = \frac{1}{kT}$ goes to infinite. Numerically, the β only needs to be a sufficiently large positive number.

Let us denote the ith eigenvalue of $\tilde{H}^2_\beta$ as E_i and the energy gaps as $\Delta_i = E_{i+1} - E_i$. From the definition of $\tilde{H}^2_\beta$, the ground energy $E_{\mathrm{g}} = E_1$ is always 0. As we observed, during the optimization process, the eigenenergies, as well as the energy gaps of the Hamiltonian, gets larger. From figure 9(a), we can see the energy gaps grow as the optimization goes. The corresponding β is a finite sufficient large number.

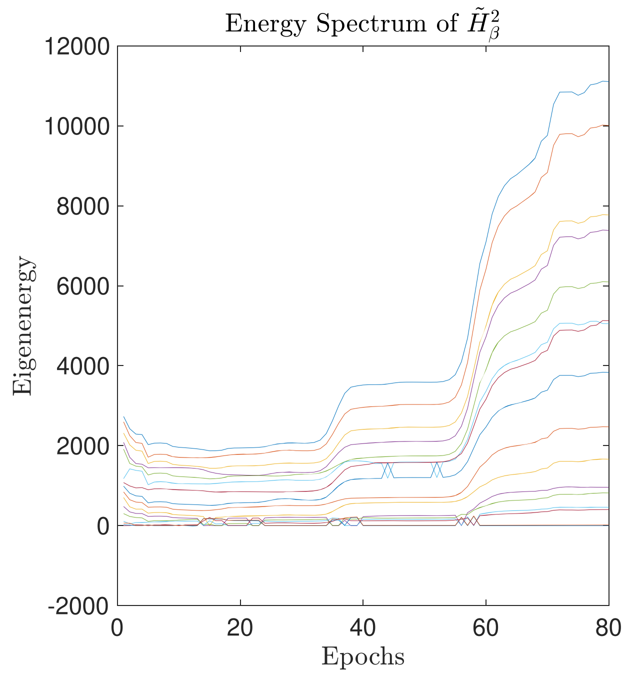
On the other hand, we can also examine the probability of the ground state of $\tilde{H}^2_\beta$ as the iteration goes. This probability can be expressed as

$$P(E = E_{\mathrm{g}}) = \frac{\mathrm{e}^{-E_{\mathrm{g}}}}{\mathrm{Tr}\ \mathrm{e}^{-\tilde{H}^2_\beta}}$$

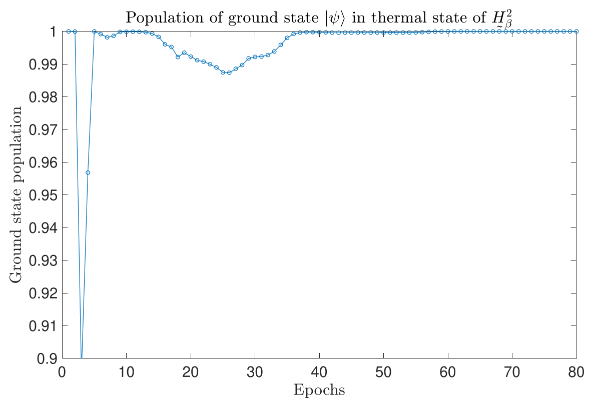
Since the ground state energy of $\tilde{H}^2_\beta$ is always have zero, the probability is

$$P(E = E_{\mathrm{g}}) = \frac{1}{\mathrm{Tr}\ \mathrm{e}^{-\tilde{H}^2_\beta}}.$$

The change of the probability $P(E = E_{\mathrm{g}})$ with iterations is shown in figure 9(b), from which we can see that finally, the probability goes to 1, which means that the thermal state form is (almost) ground state when β is a relatively large positive constant.



(a)



(b)

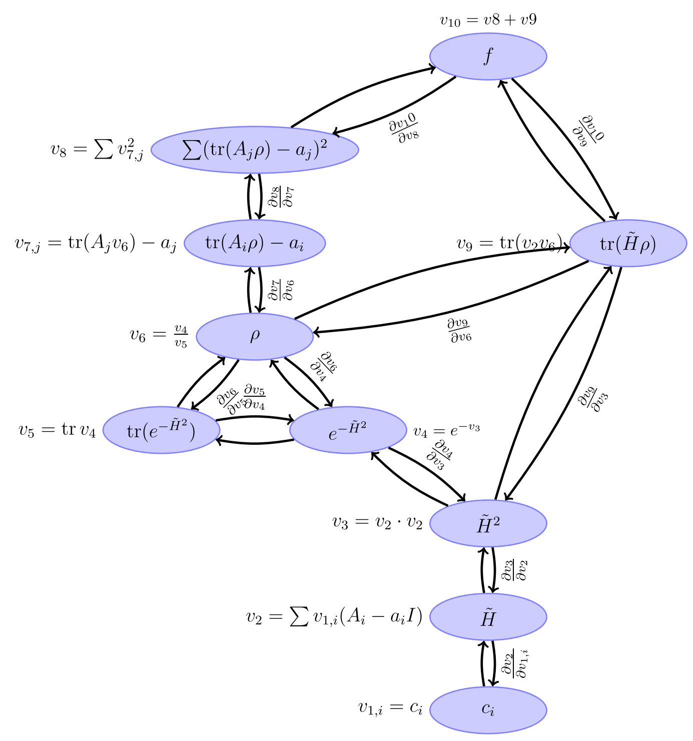
Figure 9. Behavior of $\tilde{H}_\beta^2$ in the optimization process. (a) Energy spectrum of $\tilde{H}_\beta^2$. (b) The ground state population during optimization.

Appendix B. Calculation of the gradient of $f(\vec{c})$

B.1. General method

The computational graph of this technique is shown in figure 10. Each node of the graph represents an intermediate function. In principle, the chain rule can be applied. It seems that we can compute the gradient by using the existing deep learning frameworks. However, those auto differentiation tools embedded in the frameworks such as PyTorch [57] and TensorFlow [58] cannot harness our problem due to the following two reasons: first, most of the tools only deal with real numbers while in quantum physics we often deal with complex numbers; second and the most important, some of the intermediate functions in the computational graph use matrices (or in physics, operators) as variables, while in neural networks, the variables are real numbers. One should be careful while deriving matrices since a matrix usually does not commute with its derivative. For instance, even for some simple conditions, the node $v_3 = \tilde{H}^2 = v_2^2$, the derivative $\frac{\partial v_3}{\partial v_{1,k}}$ cannot be simply considered as $2v_2\frac{\partial v_2}{\partial v_{1,k}}$ because $[\tilde{H}, \frac{\partial \tilde{H}}{c_k}] \neq 0$. Instead, $\frac{\partial v_3}{\partial v_{1,k}} = v_2\frac{\partial v_2}{\partial v_{1,k}} + \frac{\partial v_2}{\partial v_{1,k}}v_2$.

The derivatives in the computational graph shown in figure 10 are listed in table 2. Here, due to the reason mentioned above, we use forward propagation to calculate the gradients. In table 2, the derivative could be separated into the following categories: 1. The variable(s) of the function is a number. This corresponds to the simplest case, and the derivatives could be obtained simply using chain rules; 2. The



$v_{10} = v8 + v9$
f
$v_8 = \sum v_{7,j}^2$
$\sum(\mathrm{tr}(A_j\rho) - a_j)^2$
$\frac{\partial v10}{\partial v8}$
$\frac{\partial v10}{\partial v9}$
$\frac{\partial v8}{\partial v7}$
$v_{7,j} = \mathrm{tr}(A_j v_6) - a_j$
$\mathrm{tr}(A_i\rho) - a_i$
$v_9 = \mathrm{tr}(v_3 v_6)$
$\mathrm{tr}(\tilde{H}\rho)$
$\frac{\partial v7}{\partial v6}$
$v_6 = \frac{v_4}{v_5}$
ρ
$\frac{\partial v9}{\partial v6}$
$\frac{\partial v6}{\partial v4}$
$\frac{\partial v6}{\partial v5}$
$\frac{\partial v5}{\partial v4}$
$\frac{\partial v9}{\partial v3}$
$v_5 = \mathrm{tr}\, v_4$
$\mathrm{tr}(e^{-\tilde{H}^2})$
$e^{-\tilde{H}^2}$
$v_4 = e^{-v_3}$
$\frac{\partial v4}{\partial v3}$
$v_3 = v_2 \cdot v_2$
$\tilde{H}^2$
$\frac{\partial v3}{\partial v2}$
$v_2 = \sum v_{1,i}(A_i - a_i I)$
$\tilde{H}$
$\frac{\partial v2}{\partial v_{1,i}}$
$v_{1,i} = c_i$
c_i

Figure 10. **Computational graph.** Each node represents an intermediate function with the label on its left. The arrows from lower nodes to upper nodes represent the propagation of function while those for upper nodes to lower nodes represent the propagation of the derivatives.

Table 2. The types of the variables and values of the intermediate functions.

Node	Function	Derivative	Type of variable(s)	Type of function value
v_2	$v_2 = \sum c_i(A_i - a_i I)$	$A_i - a_i I$	Number	Matrix
v_3	$v_3 = v_2 \cdot v_2$	$\frac{\partial v_2}{v_{1,k}}\tilde{H} + \tilde{H}\frac{\partial v_2}{v_{1,k}}$	Matrix	Matrix
v_4	$v_4 = \mathrm{e}^{-v_3}$	$\int_0^1 \mathrm{e}^{\alpha v_3}\frac{\partial v_3}{v_{1,k}}\mathrm{e}^{(1-\alpha)v_3}\mathrm{d}\alpha$	Matrix	Matrix
v_5	$v_5 = \mathrm{tr} v_4$	$\mathrm{tr}(\frac{\partial v_4}{v_{1,k}})$	Matrix	Number
v_6	$v_6 = \frac{v_4}{v_5}$	$\frac{\partial v_4}{v_{1,k}}\frac{1}{v_5} - \frac{v_4}{v_5^2}\frac{\partial v_5}{\partial v_{1,k}}$	Matrix & number	Matrix
v_7	$v_7, j = \mathrm{tr}(A_j v_6) - a_j$	$\mathrm{tr}(A_j\frac{\partial v_6}{\partial v_{1,k}})$	Matrix	Number
v_8	$v_8 = \sum v_{7,j}^2$	$2v_{7,j}\frac{\partial v_{7,j}}{\partial v_{1,i}}$	Number	Number
v_9	$v_9 = \mathrm{tr}(v_3 v_6)$	$\mathrm{tr}(\frac{\partial v3}{\partial v_{1,i}}v_6 + v_3\frac{\partial v6}{\partial v_{1,i}})$	Matrix	Number
v_{10}	$v_1 0 = v_8 + v_9$	$\frac{\partial v8}{\partial v_{1,i}} + \frac{\partial v9}{\partial v_{1,i}}$	Number	Number

variables are matrices, but the function is the trace function tr. The derivatives could be obtained by applying $\frac{\mathrm{d}\ \mathrm{tr}(A)}{\mathrm{d}x} = \mathrm{tr}\ \frac{\mathrm{d}A}{\mathrm{d}x}$; 3. Some elementary functions take matrices as variables, i.e., $v_3 = v_2^2$. In this case, one should be careful about the commutators when dealing with it; 4. $v_4 = \mathrm{e}^{\tilde{H}^2}$, of which the gradients are difficult to calculate, which will be discussed in detail.

B.2. Derivative of matrix exponentials

The derivative of function $f(X) = \exp(X(c_k))$ with respect to c_k can be written as

$$\frac{\partial \mathrm{e}^{X(c_k)}}{\partial c_k} = \int_0^1 \mathrm{e}^{\alpha X(c_k)}\frac{\partial X(c_k)}{\partial c_k}\mathrm{e}^{(1-\alpha)X(c_k)}\mathrm{d}\alpha, \tag{B1}$$

where, in our cases, $X(c_k) = -\tilde{H}(c_k)^2 = -v_3$. Generally, the commutator $[X, \frac{\partial X}{\partial c_k}] \neq 0$. Therefore, the integration equation (B1) cannot be simply calculated. In reference [25], the authors approximate the integration using the value of the upper and lower limits. This approach does not work for our case. Thus we introduce a new way to calculate it. Let

$$A(\beta) = \frac{\partial \mathrm{e}^{X(c_k)}}{\partial c_k} = \int_0^\beta \mathrm{e}^{\alpha X(c_k)}\frac{\partial X(c_k)}{\partial c_k}\mathrm{e}^{-\alpha X(c_k)}\mathrm{d}\alpha, \tag{B2}$$

$$B(\beta) = \mathrm{e}^{-\beta X(c_k)} A(\beta), \tag{B3}$$

and

$$C(\beta) = \mathrm{e}^{-\beta X(c_k)}. \tag{B4}$$

Note that $A(1) = B(1)\exp[X(c_k)]$. It can be derived that

$$\frac{\mathrm{d}}{\mathrm{d}\beta}\begin{pmatrix} B(\beta) \\ C(\beta) \end{pmatrix} = \begin{pmatrix} -X(c_k) & \dfrac{\partial X(c_k)}{\partial c_k} \\ 0 & -X(c_k) \end{pmatrix}\begin{pmatrix} B(\beta) \\ C(\beta) \end{pmatrix}. \tag{B5}$$

Let

$$G = \begin{pmatrix} -X(c_k) & \dfrac{\partial X(c_k)}{\partial c_k} \\ 0 & -X(c_k) \end{pmatrix}, \tag{B6}$$

The derivative equation (B5) can be solved as

$$\begin{pmatrix} B(1) \\ C(1) \end{pmatrix} = \mathrm{e}^{G}\begin{pmatrix} B(0) \\ C(0) \end{pmatrix}, \tag{B7}$$

where $B(0)$ is a matrix with all entries 0 and $C(0)$ is the identity matrix. With $B(1)$, we can obtain $A(1)$ as well as the integration equation (B1). This completes the gradient calculation for our algorithm.

Appendix C. Software implementations

According to the computational graph figure 10 and the table 2, we can define a function which accepts $\vec{c}$, the operators A, and returns the value of $f(\vec{c})$ and the gradient $\nabla f(\vec{c})$:

```
function [f,g]=fun(c, A)
% f is the value of the objective
    function
% g is the gradient
```

The function FMINUNC in MATLAB/Octave can accept the function and the gradient and doing the optimization:

```
f_handle=@(c)fun(c,A)
```

The default algorithm is BFGS. To make the algorithm using gradient we provide:

```
options=optimoptions('fminunc','
    SpecifyObjectiveGradient','true');
```

Then, we can start the optimization

```
[c1, fval]=fminunc(f_handle,c0)
```

Here, c1 is the final points, fval is the final value of the objective function and c0 is the initial value.

ORCID iDs

Shi-Yao Hou https://orcid.org/0000-0001-9739-2263

References

[1] Tarasov V O, Takhtadzhyan L A and Faddeev L D 2016 *Fifty Years of Mathematical Physics: Selected Works of Ludwig Faddeev* (Singapore: World Scientific) pp 339–53

[2] Tarasov V O 1985 *Theor. Math. Phys.* **63** 440

[3] Zanardi P 2002 *Phys. Rev.* A **65** 042101

[4] Feynman R P 1982 *Int. J. Theor. Phys.* **21** 467

[5] Cirac J I and Zoller P 2012 *Nat. Phys.* **8** 264

[6] Lloyd S 1996 *Science* **273** 1073

[7] Buluta I and Nori F 2009 *Science* **326** 108
[8] Freedman M, Kitaev A, Larsen M and Wang Z 2003 *Bull. Am. Math. Soc.* **40** 31
[9] Nagaj D 2008 arXiv:0808.2117
[10] Aharonov D, Van Dam W, Kempe J, Landau Z, Lloyd S and Regev O 2008 *SIAM Rev.* **50** 755
[11] Jordan S P, Farhi E and Shor P W 2006 *Phys. Rev.* A **74** 052322
[12] Kempe J, Kitaev A and Regev O 2006 *SIAM J. Comput.* **35** 1070
[13] Kempe J and Regev O 2003 arXiv:quant-ph/0302079
[14] Bravyi S, Divincenzo D P, Oliveira R I and Terhal B M 2006 arXiv:quant-ph/0606140
[15] Xin T, Lu S, Cao N, Anikeeva G, Lu D, Li J, Long G and Zeng B 2018 arXiv:1807.07445
[16] Bairey E, Arad I and Lindner N H 2019 *Phys. Rev. Lett.* **122** 020504
[17] Deng D-L, Li X and Sarma S D 2017 *Phys. Rev.* B **96** 195145
[18] Lu S, Gao X and Duan L-M 2019 *Phys. Rev.* B **99** 155136
[19] Pudenz K L and Lidar D A 2013 *Quant. Inf. Process.* **12** 2027
[20] Liu J, Qi Y, Meng Z Y and Fu L 2017 *Phys. Rev.* B **95** 041101
[21] Garrison J R and Grover T 2018 *Phys. Rev.* X **8** 021026
[22] Qi X-L and Ranard D 2019 *Quantum* **3** 159
[23] Chen J, Ji Z, Wei Z and Zeng B 2012 *Phys. Rev.* A **85** 040303
[24] Zhou D 2008 *Phys. Rev. Lett.* **101** 180505
[25] Niekamp S, Galla T, Kleinmann M and Gühne O 2013 *J. Phys. A: Math. Theor.* **46** 125301
[26] Kalev A, Baldwin C H and Deutsch I H 2015 arXiv:1511.01433
[27] Linden N, Popescu S and Wootters W 2002 *Phys. Rev. Lett.* **89** 207901
[28] Linden N and Wootters W 2002 *Phys. Rev. Lett.* **89** 277906
[29] Diósi L 2004 *Phys. Rev.* A **70** 010302
[30] Chen J, Ji Z, Ruskai M B, Zeng B and Zhou D-L 2012 *J. Math. Phys.* **53** 072203
[31] Chen J, Ji Z, Zeng B and Zhou D 2012 *Phys. Rev.* A **86** 022339
[32] Chen J, Dawkins H, Ji Z, Johnston N, Kribs D, Shultz F and Zeng B 2013 *Phys. Rev.* A **88** 012109
[33] Zeng B, Chen X, Zhou D-L and Wen X-G 2015 arXiv:1508.02595
[34] Klyachko A A 2006 *J. Phys.: Conf. Ser.* **36** 72
[35] Liu Y-K 2006 *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques* (*Lecture Notes in Computer Science* vol 4110) ed J Diaz, K Jansen, J D Rolim and U Zwick (Berlin: Springer) pp 438–49
[36] Liu Y-K, Christandl M and Verstraete F 2007 *Phys. Rev. Lett.* **98** 110503
[37] Wei T-C, Mosca M and Nayak A 2010 *Phys. Rev. Lett.* **104** 040501
[38] Coleman A J 1963 *Rev. Mod. Phys.* **35** 668
[39] Erdahl R M 1972 *J. Math. Phys.* **13** 1608
[40] Klyachko A 2004 arXiv:quant-ph/0409113
[41] Altunbulak M and Klyachko A 2008 *Commun. Math. Phys.* **282** 287
[42] Schilling C, Gross D and Christandl M 2013 *Phys. Rev. Lett.* **110** 040404
[43] Walter M, Doran B, Gross D and Christandl M 2013 *Science* **340** 1205
[44] Sawicki A, Oszmaniec M and Kuś M 2014 *Rev. Mod. Phys.* **26** 1450004
[45] Deutsch J M 1991 *Phys. Rev.* A **43** 2046
[46] Srednicki M 1994 *Phys. Rev.* E **50** 888
[47] Srednicki M 1996 *J. Phys. A: Math. Gen.* **29** L75
[48] Srednicki M 1999 *J. Phys. A: Math. Gen.* **32** 1163
[49] D'Alessio L, Kafri Y, Polkovnikov A and Rigol M 2016 *Adv. Phys.* **65** 239
[50] Li J, Huang S, Luo Z, Li K, Lu D and Zeng B 2017 *Phys. Rev.* A **96** 032307
[51] Broyden C G 1970 *IMA J. Appl. Math.* **6** 76
[52] Fletcher R 1970 *Comput. J.* **13** 317
[53] Goldfarb D 1970 *Math. Comput.* **24** 23
[54] Shanno D F 1970 *Math. Comput.* **24** 647
[55] Fortunato E M, Pravia M A, Boulant N, Teklemariam G, Havel T F and Cory D G 2002 *J. Chem. Phys.* **116** 7599
[56] To generate a random Hermitian matrix, we first generate d^2 complex number $\{e_{ij}|0 \leqslant i, j \leqslant d\}$ as the entries of a matrix E, then we construct $A = E + E^\dagger$. It can be easily seen A is an Hermitian matrix.
[57] Paszke A *et al* 2017 Automatic differentiation in pytorch
[58] Abadi M *et al* 2015 TensorFlow: Large-scale machine learning on heterogeneous systems software available from tensorflow.org