Scalably learning quantum many-body Hamiltonians from dynamical data

F. Wilde,[1] A. Kshetrimayum,[2, 1] I. Roth,[3] D. Hangleiter,[4, 5] R. Sweke,[1, *] and J. Eisert[1, 2, 6]

[1]Dahlem Center for Complex Quantum Systems, Freie Universität Berlin, 14195 Berlin, Germany
[2]Helmholtz-Zentrum Berlin für Materialien und Energie, 14109 Berlin, Germany
[3]Quantum Research Centre, Technology Innovation Institute (TII), Abu Dhabi
[4]Joint Center for Quantum Information and Computer Science (QuICS), University of Maryland & NIST, College Park, MD 20742, USA
[5]Joint Quantum Institute (JQI), University of Maryland & NIST, College Park, MD 20742, USA
[6]Fraunhofer Heinrich Hertz Institute, 10587 Berlin, Germany

(Dated: September 30, 2022)

The physics of a closed quantum mechanical system is governed by its Hamiltonian. However, in most practical situations, this Hamiltonian is not precisely known, and ultimately all there is are data obtained from measurements on the system. In this work, we introduce a highly scalable, data-driven approach to learning families of interacting many-body Hamiltonians from dynamical data, by bringing together techniques from gradient-based optimization from machine learning with efficient quantum state representations in terms of tensor networks. Our approach is highly practical, experimentally friendly, and intrinsically scalable to allow for system sizes of above 100 spins. In particular, we demonstrate on synthetic data that the algorithm works even if one is restricted to one simple initial state, a small number of single-qubit observables, and time evolution up to relatively short times. For the concrete example of the one-dimensional Heisenberg model our algorithm exhibits an error constant in the system size and scaling as the inverse square root of the size of the data set.

Given the Hamiltonian of a quantum system we can, in principle, derive all physical properties of that system. Therefore, many studies of theoretical physics start by specifying the Hamiltonian. In practice, an effective Hamiltonian is typically hypothesised based on theoretical considerations. This hypothesis is then confirmed or rejected by comparing its predictions with experimental data. However, in many contexts, it is far from clear what the effective Hamiltonian describing a physical system actually is, at least to high levels of accuracy. In this situation, one might hope to infer the Hamiltonian governing the system directly from experimental data or, in other words, to *learn the Hamiltonian*.

In addition to their conceptual, foundational significance, there is a second more technologically minded reason why Hamiltonian learning is important: In the context of quantum information science, one aims at making predictions to high and often unprecedented accuracy. Analog quantum simulators in particular allow to assess the physics of interacting quantum systems presumably beyond the reach of classical computers [1–3]. To build trust in such devices it is neccessary to precisely determine the Hamiltonian at play. Moreover, in the context of near-term quantum computing, engineering and testing quantum processors requires precise knowledge of the underlying physics [4]. Hamiltonian learning can thus be seen as a primitive in the precise engineering of quantum devices. As a first step toward the ultimate goal of learning an effective Hamiltonian from scratch, one might consider the less demanding task of learning the precise parameters of a given Hamiltonian model.

In this work, we introduce a technique that allows to scalably learn the Hamiltonian parameters of interacting quantum many-body systems. To achieve applicability of our method to large systems, we bring together two computational methodologies. On the one hand, we make use of *tensor network techniques* that allow to efficiently compute the properties of a large class of quantum many-body systems in space and—at least for sufficiently small times—in time. On the other hand, we get inspiration from *machine-learning* approaches and make use of stochastic gradient descent optimization methods [5] and automatic differentiation [6]. On a high level, we solve a *maximum likelihood estimation* (MLE) problem on large sets of dynamical measurement data using (stochastic) gradient-based optimization. The data we use are simply given by the outcomes of Pauli measurements of time-evolved states starting from one simple reference state.

Previous work on many-body Hamiltonian learning [7] can be roughly divided into two categories. On the one hand, there is a body of theoretical work which rigorously shows that many-body Hamiltonians can be efficiently learned from either thermal states or eigenstates of the Hamiltonian [8–12], or short-time evolution [13–16], sometimes even in a way that is robust to *state preparation and measurement* (SPAM) errors [17]. On the other hand, there are practically motivated approaches to learn Hamiltonians that make use of the entire long-time dynamics of a system. Such methods face the challenge of predicting the time evolution. To overcome this challenge, one can utilize specific structure of certain classes of Hamiltonians [18–20], algebraic properties of the Hamiltonian [21, 22], or their conserved quantities [23, 24]. Alternatively, one can also make use of high-level machine-learning techniques such as neural networks [25–29] to predict the time series. Close in mindset to our approach are schemes such as the one of Ref. [30] which makes use of tensor-network techniques to directly learn a control model of an analogue quantum simulator.

However, few steps have been taken so far to devise practical methods for learning interacting quantum many-body Hamiltonians of large systems with a large number of unknown parameters. A large-scale demonstration on up to

* Currently at IBM Quantum, Almaden Research Center, San Jose, CA 95120, USA

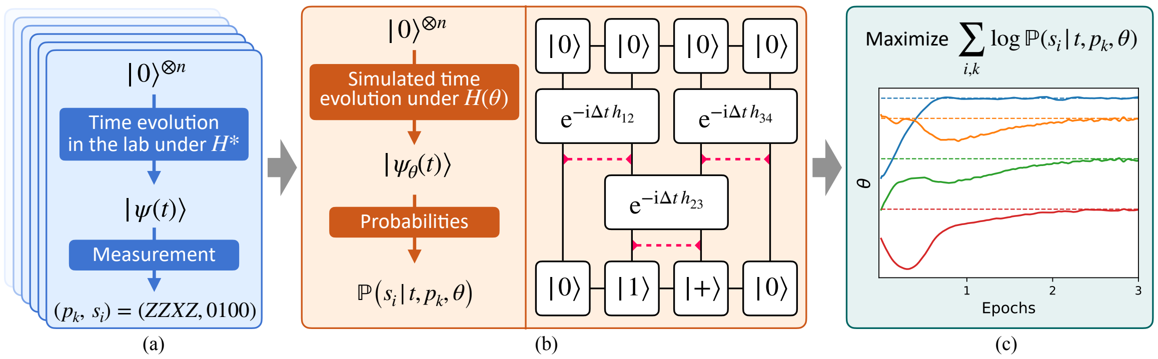
$|0\rangle^{\otimes n}$
Time evolution in the lab under H^*
$|\psi(t)\rangle$
Measurement
$(p_k, s_i) = (ZZXZ, 0100)$
(a)
$|0\rangle^{\otimes n}$
Simulated time evolution under $H(\theta)$
$|\psi_\theta(t)\rangle$
Probabilities
$\mathbb{P}(s_i\,|\,t, p_k, \theta)$
$|0\rangle$ $|0\rangle$ $|0\rangle$ $|0\rangle$
$e^{-i\Delta t\, h_{12}}$
$e^{-i\Delta t\, h_{34}}$
$e^{-i\Delta t\, h_{23}}$
$|0\rangle$ $|1\rangle$ $|+\rangle$ $|0\rangle$
(b)
Maximize $\sum_{i,k} \log \mathbb{P}(s_i\,|\,t, p_k, \theta)$
θ
1
2
3
Epochs
(c)

FIG. 1. The core idea of the method. **(a)** Data are accumulated from repeated measurements at different times. Each data point corresponds to a randomly chosen Pauli basis and a string of binary measurement outcomes. **(b)** The time evolution is simulated under a Hamiltonian $H(\theta) \in \mathcal{C}$ using the TEBD algorithm. After each contraction with a local time step $e^{-i\Delta t\, h_{\alpha,\beta}}$ the resulting rank-4 tensor is split and the bond dimensions are truncated to their original size, as indicated by the dashed red lines. **(c)** The best suitable parameters $\theta^{\mathcal{D}} \in \mathbb{R}^{\nu}$—according to the negative log-likelihood cost function—matching the observed data are learned using automatic differentiation and gradient-based optimization. In the case shown here $\nu = 4$ and the true parameter values (dashed lines) are recovered successfully.

100 sites with many parameters has only been achieved using steady states of the Hamiltonian [31], while methods that are in principle scalable and practical have not yet been demonstrated for large system sizes [23], or have prohibitively large (albeit polynomial) scaling [13, 17]. Moreover, such methods tend to require very specific state preparations and/or measurements, as well as the estimation of expectation values.

In contrast, our learning algorithm directly runs on the measurement outcomes of a small number of Pauli measurements, which can be chosen suitably depending on the measurement setup. This ensures that we do not discard any correlations contained in the measurement, and reduces the total number of experiments required. With sufficient computational memory, the run-time of our method scales linearly in the system size (see Fig. 4) and the reconstruction error scales as an inverse square root of the total number of measurements.

A scalable method for Hamiltonian learning. We consider the following setting, as illustrated in Fig. 1: A closed quantum many-body system consisting of n spins—or qubits—in the laboratory, governed by the Hamiltonian H^*, is initialized in a low-entanglement state. Specifically, here we consider the initial state to be the product state vector $|0\rangle^{\otimes n}$. Then the system evolves in time according to the Schrödinger equation up to some time t_j whereupon all spins are measured in some, potentially randomly chosen, Pauli basis $p_k \in \{X, Y, Z\}^n$. The measurement outcome is a binary string s of length n. In general this process is repeated M times for multiple time stamps $t_1, \dots, t_J$ and Pauli bases $p_1, \dots, p_K$. Hence, the resulting data set $\mathcal{D}$ contains $M \cdot J \cdot K$ bit-strings. Alternatively, one could prepare K initial states and only measure in one Pauli basis.

To simulate the laboratory system, we make use of tensor networks, which is a powerful tool for describing quantum many-body systems beyond the reach of exact diagonalization [32]. In particular, we use the *time-evolving block decimation* (TEBD) algorithm, illustrated in Fig. 1(b), which is a particularly simple and efficient tensor network algorithm that can be used for computing ground states, thermal states, and non-equilibrium dynamics of quantum many-body systems [33, 34].

In this work, we focus on the dynamics of one-dimensional systems, although an extension to higher dimensions is also possible [35–37]. By choosing an initial state with low entanglement, we can represent it using a *matrix-product state* (MPS) efficiently in space until intermediate times. To solve the Schrödinger equation, the time-evolution operator e^{-itH} is decomposed into a product of small Trotter steps $e^{-i\Delta t H} \cdots e^{-i\Delta t H}$. Additionally, the operator is decomposed into layers of mutually commmuting terms, as shown in the example in Fig. 1(b). These decompositions incur an approximation error, called *Trotter error*, which can be controlled by the Trotter-step size Δt. Each time, after contracting a layer of operators with the MPS, the tensors are split using a *singular value decomposition* (SVD) to retain the MPS structure with a given bond dimension. After each splitting the bond dimension between tensors gets multiplied by the physical dimension of the sites, which in our case is 2. To mitigate this exponential growth the bond dimension is truncated after splitting, which incurs the *truncation error*. Generally, the longer the time evolution is, the more entanglement is created in the system, which increases the truncation error. As such, TEBD is limited to intermediate times with a constant scaling in the system size, but importantly, not as short that expansions in powers of the Hamiltonian are suitable. Note that the computational complexity of TEBD scales linearly with the system size in one dimension. Hence, when the set of timestamps $\{t_j\}$ and the size d of the data set is constant, the computational cost of one gradient evaluation also scales linearly, due to the use of automatic differentiation, as shown in detail in Appendix A.

To recover H^*, we fix an ansatz class of Hamiltonians governed by certain Hamiltonian parameters. Formally, we define a *class of parametrized Hamiltonians* $\mathcal{C} = \{H(\theta) | \theta \in \Theta\}$ as the range of a twice continuously differentiable function $\theta \mapsto H(\theta)$ (see Appendix B). The parametrization is chosen according to physical intuition. For instance, the parameters could be the amplitudes of interactions one expects to occur in the lab experiment. Then, we classically simulate the time evolution under the parametrized Hamiltonian $H(\theta_{\mathrm{ini}}) \in \mathcal{C}$ using TEBD, where θ_{ini} is a, typically random, initialization of the parameters. Based on the data $\mathcal{D}$ we formulate a MLE problem by defining the *negative log-likelihood*

$$\mathcal{L}^{\mathcal{D}}(\theta) = -\frac{1}{d}\sum_{j=1}^{J}\sum_{k=1}^{K}\sum_{i=1}^{M} \log \mathbb{P}(s_{i,j,k}|t_j, p_k, \theta), \qquad (1)$$

with $d = |\mathcal{D}|$, which plays the role of a *loss function*. The minimizer $\theta^{\mathcal{D}} = \operatorname{argmin} \mathcal{L}^{\mathcal{D}}(\theta)$ of the loss corresponds to the Hamiltonian in $\mathcal{C}$ which best describes the given data. Note that due to this approach there is no need to post-process the data, for instance, by estimating single-spin expectation values. Therefore, all correlations contained in the data set are accessible to the learning algorithm. To evaluate the loss function we compute the probabilities according to the Born rule $\mathbb{P}(s_{i,j,k}|t_j, p_k, \theta) = |\langle\phi_{i,j,k}|\, \mathrm{e}^{-\mathrm{i}H(\theta)t_j}\,|0\rangle|^2$, where $|\phi_{i,j,k}\rangle$ is the i-th post-measurement state vector corresponding to the p_k measurement after evolution up to time t_j. This probability can be computed using TEBD as illustrated by the tensor-network diagram in Fig. 1(b).

When the number of parameters ν is large, i.e., the minimization problem is high dimensional, gradient-based optimization becomes advantageous over zeroth-order methods. In order to obtain the gradient efficiently and accurately, even for very large ν, we use automatic differentiation to back-propagate the derivative through the entire TEBD algorithm. This has been implemented using the *auto-differentiation framework* JAX [38], which supports complex numbers and matrix operations such as the matrix exponential and the singular value decomposition. To allow the differentiation of the SVD with complex inputs we derived and implemented the Jacobian-vector product rule and integrated it into JAX (see Appendix D). Additionally, we extensively exploit the potential of the Accelerated Linear Algebra (XLA) compiler which JAX is built on. Specifically, we use just-in-time compilation and vectorization by writing the TEBD using only XLA-compatible control flow syntax. Further implementation details can be found in Appendix A and the Github repository [39].

The algorithm we use to minimize $\mathcal{L}$ consists of two stages. In the first stage we use the popular optimizer ADAM and mini-batch stochastic gradient descent. In each step of the descent we select one bit-string per Pauli basis, while the sets of bit-strings get shuffled for each epoch. The resulting batch of data, containing $K \cdot J$ bit-strings, is then used to compute the stochastic gradient estimator. As this reduces the computation time per gradient step drastically, the first stage quickly moves to the vicinity of a local minimum. To find the minimum with high precision, in the second stage we use the entire data set to compute the gradient and employ the *pseudo Newton optimizer BFGS*. The gradient steps in the second stage require more computation time, but the BFGS algorithm takes only a few tens of steps to find the minimum with high precision. For a motivation of this particular optimization algorithm, see Appendix A.

Analysis. In practice, when the Hamiltonian parameters are unknown, the only accessible indicator of success is the loss function. Once the loss function has converged to a low value, the target Hamiltonian H^* might have been recovered. We find that one can distinguish successfully converged instances from instances which converged to a local minimum a posteriori only by the value of their loss function, as we elaborate in Appendix C. However, in this work, the data are generated synthetically and thus we exactly know the target Hamiltonian. This allows us to precisely gauge the performance of our method by defining the relative error

$$\epsilon(\theta) := \frac{\|\theta - \theta^*\|_2}{\|\theta^*\|_2}. \qquad (2)$$

The data set $\mathcal{D}$ consists of d randomly sampled bit-strings, representing the outcomes of quantum measurements. Thus it is natural to define the minimizer of the loss function as a random variable $\hat{\theta}^{(d)}$ itself, where the data set size d is an independent parameter and the sets of measurement bases $\{p_k\}$ and times $\{t_j\}$ are fixed. Consequently, the relative error is also a random variable $\hat{\epsilon}^{(d)} = \epsilon(\hat{\theta}^{(d)})$. By examining the MLE problem at hand we find that the relative error $\hat{\epsilon}^{(d)}$ scales as $d^{-1/2}$ in the large-sample limit.

Theorem 1 (Asymptotic error (informal))**.** *Suppose the class of parameterized Hamiltonians $\mathcal{C}$ is well conditioned, such that the estimator $\hat{\theta}^{(d)}$ is consistent and the Hessian of the loss is non-singular. Then for any $\delta \in (0,1]$ there exists a function $f(d) = \mathcal{O}(d^{-1/2})$ such that $\mathbb{P}\big[\hat{\epsilon}^{(d)} > f(d)\big] < \delta$ when d is sufficiently large.*

A rigorous statement of the theorem as well as a proof is provided in Appendix B. Note that this scaling can be seen in Fig. 2(a) even for moderate values of d, despite the fact that we cannot prove that our choice of $\mathcal{C}$ is well conditioned. Lastly, note that the theorem makes a statement about the global minimum $\theta^{\mathcal{D}}$ of the loss function. However, it does not guarantee the convergence to $\theta^{\mathcal{D}}$.

Results. To numerically analyze our method we generated data in two different ways. For systems consisting of up to 26 spins, the Schrödinger equation has been solved via exact state-vector simulation [40]. To go beyond the reach of exact diagonalization, the Schrödinger equation has been solved using TEBD for 100 spins with a sufficiently large bond dimension. Afterwards, the measurement data have been generated by sampling from the quantum state vector or the MPS, respectively. In one spatial dimension this can be done efficiently [41, 42]. The model we use for our analysis is a one-dimensional Heisenberg chain consisting of n spins (and hence sites)

$$H = \sum_i \left(J^x X_i X_{i+1} + J^y Y_i Y_{i+1} + J^z Z_i Z_{i+1} + h_i X_i\right). \qquad (3)$$

We set the three interaction parameters to the arbitrary values $J^x, J^y, J^z = -1, -0.5, -0.4$, while we draw the local field parameters h_i uniformly at random from the interval $[-1, 1]$. As such, there are $\nu = 3 + n$ unknown parameters. For each

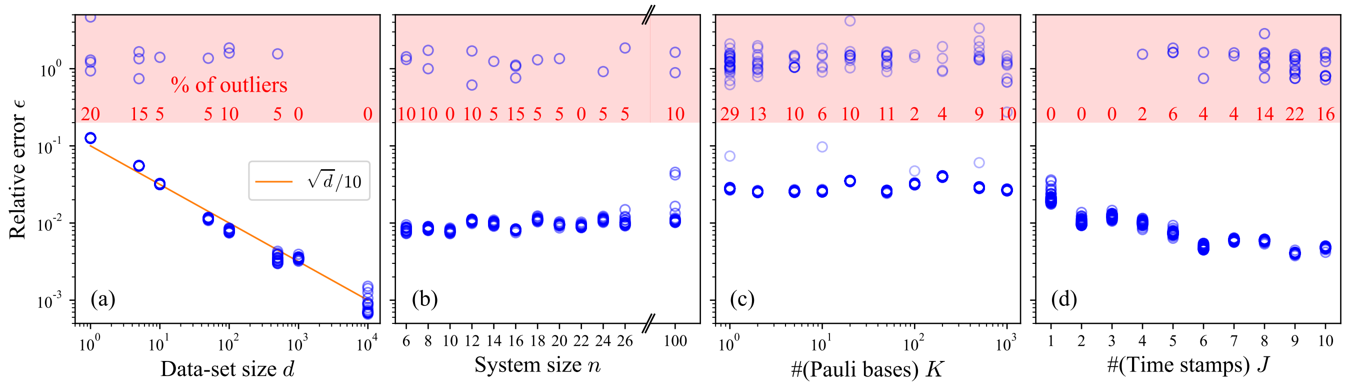

FIG. 2. The scaling of the relative recovery error ϵ for various experimental parameters. Each data point (blue circle) corresponds to a unique parameter initialization θ_{ini}, drawn at random. Vertically aligned data points correspond to the same data set $\mathcal{D}$. All points above the threshold $\epsilon = 0.2$ (red dashed line) are counted as outliers (red numbers) and deemed as *not successfully converged*. In all plots, except (c), the number of Pauli bases is 100. In all plots, except (d), the number of time stamps is 5. **(a)** The error decreases with the size of the data set. **(b)** The error does not appear to significantly depend on the system size and thereby also not on the number of parameters. **(c)** Even with only one Pauli basis, we can recover the Hamiltonian, however with a larger amount of outliers. The total number of samples has been kept fixed at $5 \cdot 10^3$. **(d)** The error decreases as one increases the number of time stamps J. One time stamp means measurements were only taken at time τ, two time stamps implies measurements were taken at times τ and 2τ, etc. The total number of measurements has been kept fixed to $\approx 6 \cdot 10^4$ in each column. The number of vertically aligned points, i.e., the number of randomly drawn parameter initializations, is 20 for (a) and (b), 100 for (c), and 50 for (d).

system size n, one parametrization is randomly drawn and used to simulate the time evolution. At time stamps in intervals of $\tau = 0.2$ (i.e., at $t = \tau, 2\tau, 3\tau, \ldots$) the state is then "measured" by sampling from the time-evolved state, typically in 100 uniformly randomly drawn Pauli bases. Fig. 2 displays the performance of the method as a function of various experimental parameters.

Fig. 2(a) shows a $d^{-1/2}$ error scaling which is in line with Theorem 1. Note that the data points significantly fluctuate around the orange line. The reason for this is that for each system size only *one* Hamiltonian $H(\theta^*)$ was investigated. Fig. 2(b) indicates that at least for the Hamiltonian at hand, the error does not depend on the system size or the number of parameters ν. Fig. 2(c) clearly shows a larger amount of outliers (optimization processes that did not converge to the target θ^*) when only one Pauli basis is used to measure the system. Meanwhile the accuracy is not affected by the number of Pauli bases. In contrast, as shown in Fig. 2(d) a greater number of time stamps leads to greater accuracy, but also to a higher number of outliers. As the number of time stamps grows, the loss landscape seems to become more rugged, while the minimum of the loss function moves closer to the true parameter values. That is, the accuracy of the estimator given by the global minimum improves, but that minimum becomes more difficult to find. See Appendix C for additional discussions on the loss landscape. Given these results and the fact that one cannot guarantee efficient optimization of the non-convex loss function, we conclude that the number of initial points has to be considered as a hyper parameter, which has to be adjusted according to the number of outliers—as witnessed by the loss function—one encounters during learning.

Conclusion. In this work, we have introduced a scalable and experimentally friendly method of Hamiltonian learning based on the *native time evolution* only. Bringing together ideas and methods from machine learning and tensor networks, we have arrived at a highly practical scheme. In a way, the methods developed are reminiscent of methods for learning classical dynamical laws by making use of tensor networks [43, 44]. In contrast to other scalable methods based on dynamics for very small times [13, 17], our method achieves an error scaling of $d^{-1/2}$ as opposed to $d^{-1/3}$ or even $d^{-1/4}$ in the size of the data set. Furthermore, our results suggest that our method does not require the preparation of multiple distinct initial states and even the number of Pauli bases in the measurement can be kept small. Therefore, also in experiments where one has limited control over the system or extra manipulations (such as the preparation of specific states) are expensive and introduce additional noise, our method seems to be highly suitable. It does not require further and possibly infeasible state manipulation such as Pauli twirls as other methods do, but uses the native time evolution.

We believe that our method is placed at a "sweet spot" concerning the duration of time evolution: For *very short times*, expansions in powers of the Hamiltonian are feasible, so that the time evolution and its derivative can be estimated, but the signal-to-noise ratio is low, leading to a high sample complexity. For very *long times*, only certain families of quantum systems can be efficiently simulated [e.g., 20], while in general the simulation of time evolution is computationally hard [45]. In contrast, our method is based on intermediate times, leading to large signal strength that can at the same time be efficiently simulated using TEBD.

The framework introduced here immediately invites a number of exciting extensions. It would be interesting to extend the current framework to *time-dependent Hamiltonians*. Even more pressing seems the generalization to methods that are able to capture *dissipative systems* where the target is to learn the *Lindbladian noise* or the rate of dissipation [8], derived from tensor network methods simulating such dynamics [34, 46, 47]. Further work building on the present approach will aim at incorporating SPAM robustness, akin to the methods of Refs. [17, 20]. It will be interesting to explore methods of model selection to systematically identify meaningful hypothesis classes to give advice when choosing the parametrization. However, since the gradient-based optimization is efficient in the number of parameters, we can choose highly expressive parametrizations.

The picture that emerges from this work is that one can efficiently in space and up to intermediate times learn Hamiltonians of one-dimensional quantum many-body systems. This approach is therefore well suited to analogue simulation platforms such as trapped ions [48] or ultra-cold atoms [49], in which either the state preparation and/or the measurement is flexible. Based on this data, one can recover the Hamiltonian, but cannot efficiently make predictions for long times [45]: For this, one *has* to perform the experiment to have predictive power, but—using our method—for an accurately calibrated Hamiltonian.

Acknowledgements. This work has been supported by the DFG (EI 519 20-1 on Hamiltonian learning, MATH+ Cluster of Excellence on notions of quantum machine learning EF1-11, EI 519/15-1 on tensor networks and EXC-2046/1 – project ID: 390685689, GRK 2433 'Daedalus' on dynamical laws, as well as CRC 183, project B01, on notions of machine learning applied to quantum many-body physics), the BMBF (DAQC on digital-analog quantum devices, MUNIQC-ATOMS on cold atomic platforms), the BMWK (PlanQK on applied quantum machine learning), and the Einstein Foundation (Einstein Research Unit on Quantum Devices). This research is also part of the Munich Quantum Valley (K8), which is supported by the Bavarian state government with funds from the Hightech Agenda Bayern Plus. It has also received funding from the EU's Horizon 2020 research and innovation program under grant agreement No. 817482 (PASQuanS) on quantum simulations. D. H. acknowledges financial support from the U.S. Department of Defense through a QuICS Hartree fellowship. The authors would like to thank the HPC Service of ZEDAT, Freie Universität Berlin, for computing time [50].

[1] J. I. Cirac and P. Zoller, *Goals and opportunities in quantum simulation*, Nature Phys. **8**, 264 (2012).

[2] S. Trotzky, Y.-A. Chen, A. Flesch, I. P. McCulloch, U. Schollwöck, J. Eisert, and I. Bloch, *Probing the relaxation towards equilibrium in an isolated strongly correlated one-dimensional Bose gas*, Nature Phys. **8**, 325 (2012).

[3] J.-Y. Choi, S. Hild, J. Zeiher, P. Schauß, A. Rubio-Abadal, T. Yefsah, V. Khemani, D. Huse, A., I. Bloch, and C. Gross, *Exploring the many-body localization transition in two dimensions*, Science **352**, 1547 (2016).

[4] J. Carrasco, A. Elben, C. Kokail, B. Kraus, and P. Zoller, *Theoretical and experimental perspectives of quantum verification*, PRX Quantum **2**, 010102 (2021).

[5] S. Krastanov, S. Zhou, S. T. Flammia, and L. Jiang, *Stochastic estimation of dynamical variables*, Quant Sc. Tech. **4**, 035003 (2019).

[6] H.-J. Liao, J.-G. Liu, L. Wang, and T. Xiang, *Differentiable programming tensor networks*, Phys. Rev. X **9**, 031041 (2019).

[7] See Ref. [4] for a recent perspective article.

[8] E. Bairey, I. Arad, and N. H. Lindner, *Learning a local Hamiltonian from local measurements*, Phys. Rev. Lett. **122**, 020504 (2019).

[9] E. Bairey, C. Guo, D. Poletti, N. H. Lindner, and I. Arad, *Learning the dynamics of open quantum systems from their steady states*, New J. Phys. **22**, 032001 (2020).

[10] A. Anshu, S. Arunachalam, T. Kuwahara, and M. Soleimanifar, *Sample-efficient learning of interacting quantum systems*, Nat. Phys. , 1 (2021).

[11] J. Haah, R. Kothari, and E. Tang, *Optimal learning of quantum Hamiltonians from high-temperature Gibbs states*, (2021), arXiv:2108.04842 [quant-ph].

[12] X.-L. Qi and D. Ranard, *Determining a Local Hamiltonian from a Single Eigenstate*, Quantum **3**, 159 (2019).

[13] A. Zubida, E. Yitzhaki, N. H. Lindner, and E. Bairey, *Optimized Hamiltonian learning from short-time measurements*, (2021), arXiv:2108.08824.

[14] D. S. França, L. A. Markovich, V. V. Dobrovitski, A. H. Werner, and J. Borregaard, *Efficient and robust estimation of many-qubit Hamiltonians*, (2022), arXiv:2205.09567.

[15] A. Gu, L. Cincio, and P. J. Coles, *Practical black box Hamiltonian learning*, (2022), arXiv:2206.15464.

[16] R. Harper, W. Yu, and S. T. Flammia, *Fast estimation of sparse quantum noise*, PRX Quantum **2**, 010322 (2021).

[17] W. Yu, J. Sun, Z. Han, and X. Yuan, *Practical and efficient Hamiltonian learning*, (2022), arXiv:2201.00190.

[18] D. K. L. Oi and S. G. Schirmer, *Quantum system characterization with limited resources*, Phil. Trans. R. Soc. A **370**, 5386 (2012).

[19] D. Burgarth, K. Maruyama, and F. Nori, *Indirect quantum tomography of quadratic Hamiltonians*, New J. Phys. **13**, 013019 (2011).

[20] D. Hangleiter, I. Roth, J. Eisert, and P. Roushan, *Precise Hamiltonian identification of a superconducting quantum processor*, (2021), arXiv:2108.08319.

[21] J. Zhang and M. Sarovar, *Quantum Hamiltonian identification from measurement time traces*, Phys. Rev. Lett. **113**, 080401 (2014).

[22] X. Chen, Y. Li, Z. Wu, R. Liu, Z. Li, and H. Zhou, *Experimental realization of Hamiltonian tomography by quantum quenches*, Phys. Rev. A **103**, 042429 (2021).

[23] Z. Li, L. Zou, and T. H. Hsieh, *Hamiltonian tomography via quantum quench*, Phys. Rev. Lett. **124**, 160502 (2020).

[24] L. Pastori, T. Olsacher, C. Kokail, and P. Zoller, *Characterization and verification of Trotterized digital quantum simulation via Hamiltonian and Liouvillian learning*, PRX Quantum **3**, 030324 (2022).

[25] P. Bienias, A. Seif, and M. Hafezi, *Meta Hamiltonian learning*, (2021), arXiv:2104.04453.

[26] T. Schuster, M. Niu, J. Cotler, T. O'Brien, J. R. McClean, and M. Mohseni, *Learning quantum systems via out-of-time-order correlators*, (2022), arXiv:2208.02254.

[27] N. Mohseni, T. Fösel, L. Guo, C. Navarrete-Benlloch, and F. Marquardt, *Deep learning of quantum many-body dynamics*

via random driving, Quantum **6**, 714 (2022).

[28] C.-D. Han, B. Glaz, M. Haile, and Y.-C. Lai, *Tomography of time-dependent quantum Hamiltonians with machine learning*, Phys. Rev. A **104**, 062404 (2021).

[29] A. Valenti, G. Jin, J. Léonard, S. D. Huber, and E. Greplova, *Scalable Hamiltonian learning for large-scale out-of-equilibrium quantum dynamics*, Phys. Rev. A **105**, 023302 (2022).

[30] Y.-J. Xie, H.-N. Dai, Z.-S. Yuan, Y. Deng, X. Li, Y.-A. Chen, and J.-W. Pan, *Bayesian learning for optimal control of quantum many-body states in optical lattices*, Phys. Rev. A **106**, 013316 (2022).

[31] T. J. Evans, R. Harper, and S. T. Flammia, *Scalable Bayesian Hamiltonian learning*, (2019), arXiv:1912.07636.

[32] U. Schollwöck, *The density-matrix renormalization group in the age of matrix product states*, Ann. Phys. January 2011 Special Issue, **326**, 96 (2011).

[33] G. Vidal, *Efficient simulation of one-dimensional quantum many-body systems*, Phys. Rev. Lett. **93**, 040502 (2004).

[34] M. Zwolak and G. Vidal, *Mixed-state dynamics in one-dimensional quantum lattice systems: A time-dependent superoperator renormalization algorithm*, Phys. Rev. Lett. **93**, 207205 (2004).

[35] A. Kshetrimayum, M. Goihl, and J. Eisert, *Time evolution of many-body localized systems in two spatial dimensions*, Phys. Rev. B **102**, 235132 (2020).

[36] C. Hubig and J. I. Cirac, *Time-dependent study of disordered models with infinite projected entangled pair states*, SciPost Phys. **6**, 031 (2019).

[37] A. Kshetrimayum, M. Goihl, D. M. Kennes, and J. Eisert, *Quantum time crystals with programmable disorder in higher dimensions*, Phys. Rev. B **103**, 224205 (2021).

[38] J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, and Q. Zhang, *JAX: composable transformations of Python+NumPy programs*, (2018).

[39] *github.com/frederikwilde/scalable-dynamical-hamiltonian-learning*, .

[40] We use the Runge-Kutta ODE solver implemented in `scipy.integrate.solve_ivp` to integrate the Schrödinger equation in its exact form on the Hilbert space of dimension 2^n.

[41] Z.-Y. Han, J. Wang, H. Fan, L. Wang, and P. Zhang, *Unsupervised generative modeling using matrix product states*, Phys. Rev. X **8**, 031012 (2018).

[42] A. J. Ferris and G. Vidal, *Perfect sampling with unitary tensor networks*, Phys. Rev. B **85**, 165146 (2012).

[43] P. Gelß, S. Klus, J. Eisert, and C. Schütte, *Multidimensional approximation of nonlinear dynamical systems*, J. Comput. Nonlinear Dynam. **14**, 061006 (2019).

[44] A. Goeßmann, M. Götte, I. Roth, R. Sweke, G. Kutyniok, and J. Eisert, *Tensor network approaches for learning non-linear dynamical laws*, (2020), arXiv:2002.12388.

[45] K. G. H. Vollbrecht and J. I. Cirac, *Quantum simulators, continuous-time automata, and translationally invariant systems*, Phys. Rev. Lett. **100**, 010501 (2008).

[46] H. Weimer, A. Kshetrimayum, and R. Orús, *Simulation methods for open quantum many-body systems*, Rev. Mod. Phys. **93**, 015008 (2021).

[47] A. Kshetrimayum, H. Weimer, and R. Orús, *A simple tensor network algorithm for two-dimensional steady states*, Nature Comm. **8**, 1291 (2017).

[48] R. Blatt and C. F. Roos, *Quantum simulations with trapped ions*, Nature Phys. **8**, 277 (2012).

[49] I. Bloch, J. Dalibard, and S. Nascimbene, *Quantum simulations with ultracold quantum gases*, Nature Phys. **8**, 267 (2012).

[50] L. Bennett, B. Melchers, and B. Proppe, *Curta: A General-purpose high-performance computer at ZEDAT, Freie Universität Berlin*, (2020).

[51] R. Bellman, *Introduction to matrix analysis* (Society for Industrial & Applied Mathematics, 1987).

[52] W. K. Newey and D. McFadden, *Large sample estimation and hypothesis*, Handbook Econom. **36**, 135 (2005).

[53] Z.-Q. Wan and S.-X. Zhang, *Automatic differentiation for complex valued SVD*, (2019), arXiv:1909.02659.

[54] J.-G. Liu, *Linear algebra autodiff (complex valued)*, (2019).

[55] M. Giles, *An extended collection of matrix derivative results for forward and reverse mode algorithmic differentiation*, (2007), tech. Rep. NA07.

[56] M. Bartholomew-Biggs, S. Brown, B. Christianson, and L. Dixon, *Automatic differentiation of algorithms*, J. Comp. Appl. Math. **124**, 171 (2000).

[57] We can imagine A being dependent on some parameter p such that we can identify $\mathrm{d}A$, $\mathrm{d}U$, $\mathrm{d}S$, and $\mathrm{d}V^\dagger$ with $\partial A/\partial p$, $\partial U/\partial p$, $\partial S/\partial p$, and $\partial V^\dagger/\partial p$, respectively.

[58] J. Townsend, *Differentiating the singular value decomposition*, (2016).

Appendix A: Details of the numerical computations

The key component of the loss function is the computation of probabilities of bit-strings. This is achieved by creating an initial MPS in the $|0\rangle^{\otimes n}$ state vector and applying Trotter steps to it in an iterative fashion until the state has evolved up to the desired time. We found a suitable value of the *bond dimension*, used for all results presented here, to be $\chi = 30$. For longer evolution times or a Hamiltonian that generates more entanglement, the bond dimension will have to be increased. We implemented the *second-order Trotter* formula, as this reduces the Trotter error with little computational overhead. The time-evolved MPS can then be contracted with the post-measurement state, as depicted in Fig. 1(b), to compute the probability of the corresponding bit-string.

To enable the efficient use of *automatic differentiation* of the probabilities, it is important to keep the dimensions of the arrays fixed, during the execution of the TEBD algorithm. Therefore the MPS is a fixed array of dimensions $(n, \chi, 2, \chi)$, where the physical dimension is 2 since we are considering a spin-1/2 system. Furthermore, the algorithm provides several opportunities for *vectorization*: we vectorize the probability computation over Pauli bases and bit-strings, the basis transformation of the MPS, and the application of individual Trotter gates. Lastly, we *just-in-time compile* the entire TEBD function. This takes several seconds to complete in the first step (see Fig. 4), but reduces the overall computation time significantly. To limit excessive memory usage during back-propagation we *checkpoint* each Trotter step. In the case of mini-batch gradient descent checkpointing is not necessary, which reduces the time for evaluating the gradient. However, for gradients on the full data set, without checkpointing the method quickly exceeds the memory constraints, as can be seen in Fig. 4. Even with checkpointing we had to iterate over chunks of the data set to compute the gradient, for larger data

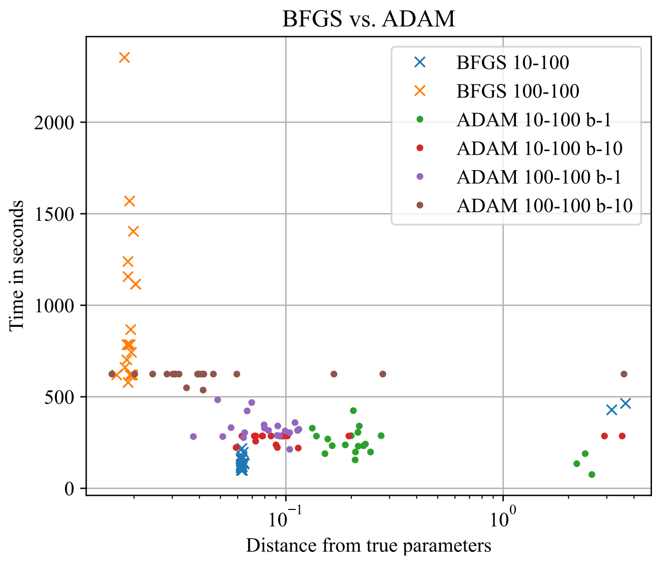

FIG. 3. Optimization times for different optimizers. The numbers K-M denote the number of Pauli bases and bit-strings, respectively. b-B denotes batch size B, which refers to the number of bit-strings used per Pauli basis in a given gradient computation.

set sizes d. For instance, for a system of $n = 10$ spins consisting of 10^6 samples per time step, we have divided the data set into 10 chunks which required about 22GiB of memory. Therefore, we predict that for even larger data sets, system sizes, evolution times, and bond dimensions, due to memory requirements, it will become necessary to only optimize with stochastic gradients, perhaps with a tailored learning rate schedule.

Initially we investigated conjugate gradient descent, plain stochastic gradient descent, BFGS, and ADAM. However the latter two appeared to be more reliable and faster in our context. The results, shown in Fig. 3, indicate that ADAM with a small batch size can quickly and reliably reduce the error, while BFGS can reduce the error reliably to the minimal value, however at longer computation times and with the necessity to evaluate the gradient w.r.t. the full data set.

Since the negative log-likelihood is a non-convex function, it is difficult to make guarantees on the number of iterations needed to find the global minimum. However, we observe that the number of iterations shows only a weak dependence on the system size n, as shown in the left plot of Fig. 4. Therefore, the relevant figure of merit for determining the time complexity is the computation time needed for one iteration, i.e. one gradient evaluation. As shown in the center plot of Fig. 4, with sufficient memory this computation time scales linearly in system size. Lastly, it is noteworthy that the system considered in our analysis the system size is connected to the number of parameters $\nu = 3 + n$.

Appendix B: Asymptotic error

For this section, we use the abbreviated notation $\partial_l = \partial/\partial\theta_l$ and $\nabla = \nabla_\theta$. Let us first precisely define the data set $\mathcal{D}$, its associated loss function $\mathcal{L}^{\mathcal{D}}$, and the minimizer of the loss $\theta^{\mathcal{D}}$ function as random variables. Suppose we are given non-empty sets of time stamps $\{t_1, \dots, t_J\}$ and Pauli bases $\{p_1, \dots, p_K\} \subseteq \{X, Y, Z\}^n$, where $t_j > 0$ for all $j = 1, \dots, J$. Moreover, we denote the family of target Hamiltonians—under which the data set is being generated—by $H(\theta^*)$. Changing the target Hamiltonian, the Pauli bases, or the time stamps would obviously impact the measurement outcomes and therefore the obtained data set $\mathcal{D}$. However, for the sake of lucidity, we do not explicity write out this dependency and assume that those three are fixed.

Definition 1 (Data set as a random variable)**.** *Each Pauli basis p_k gives rise to a positive projector-valued measure (PVM) $\{\pi^k_{0,0,\dots,0}, \dots, \pi^k_{1,1,\dots,1}\}$. The PVM elements together with a time stamp t_j give rise to a probability distribution (over bit-strings of length n)*

$$\begin{aligned} \rho_{j,k} : \{0,1\}^n &\longrightarrow [0,1] \\ s &\longmapsto \operatorname{tr}(\pi^k_s \left|\psi_j\right\rangle \left\langle\psi_j\right|), \end{aligned}$$

where $\left|\psi_j\right\rangle = \mathrm{e}^{-\mathrm{i}H(\theta^*)t_j}\left|0\right\rangle$ is the quantum state vector at time t_j. Suppose further that we select a set of positive sample numbers $\{d_{j,k} | 1 \le j \le J,\ 1 \le k \le K\}$, specifying how often we measure each Pauli basis at each time. Denote by $d = \sum_{j,k} d_{j,k}$ the total size of the data set. Now we define a realization $\mathcal{D}$ of the data-set random variable $\hat{\mathcal{D}}$ to be specified by a collection of samples (bit-strings of length n)

$$\mathcal{D} = \{s_{i,j,k} \in \{0,1\}^n | 1 \le i \le d_{j,k}, 1 \le j \le J,\ 1 \le k \le K\}$$

and the probability of observing that realization is

$$\mathbb{P}(\mathcal{D}) = \prod_{j,k} \prod_{i=1}^{d_{j,k}} \rho_{j,k}(s_{i,j,k}).$$

The definition of the empirical loss $\hat{\mathcal{L}}^{(d)}$ and the minimizer $\hat{\theta}^{(d)}$ follow straightforwardly. A realization of $\hat{\mathcal{L}}^{(d)}$ and $\hat{\theta}^{(d)}$ is given by $\mathcal{L}^{\mathcal{D}}$ and its minimizer $\theta^{\mathcal{D}}$, respectively, where $\mathcal{D}$ is a data set realization of size d.

Definition 2 (Expected loss function)**.** *The expected negative log-likelihood in the limit $d \to \infty$ is formally given by*

$$\mathcal{L}(\theta) := -\frac{1}{JK} \sum_{j,k} \sum^{\star}_{s\in\{0,1\}^2} \rho_{j,k}(s) \log(\rho_{j,k}(s|\theta)), \tag{B1}$$

where $\star$ indicates that only terms satisfying $\rho_{j,k}(s) \neq 0$ appear in the sum. Here $\rho_{j,k}(s|\theta)$ denotes the probability of measuring s in the basis p_k under the parameter values θ at time t_j.

Now we show that $\mathcal{L}^{\mathcal{D}}$ is twice continuously differentiable for all data sets $\mathcal{D}$.

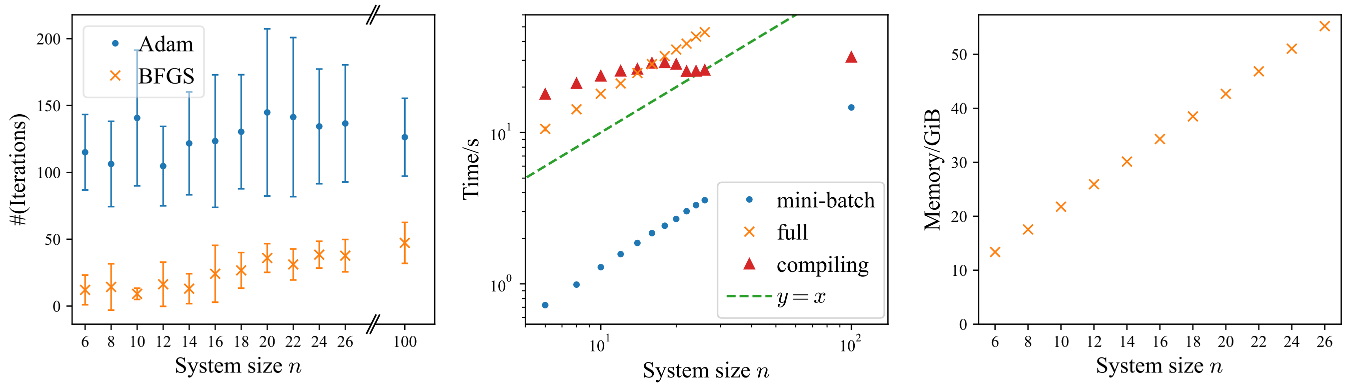

FIG. 4. Resource requirements for learning the Heisenberg Hamiltonian at varying system sizes n. **Left:** The number of iterations needed to converge. Note that our optimization algorithm consists of two stages: an initial phase of stochastic gradient descend using the ADAM optimizer followed by a fine-tuning phase to converge to the minimum using the BFGS optimizer. **Center:** Times for computing one gradient evaluation on a data set of size $d = 50000$ bit-strings ($J, K, M = 5, 100, 100$). For the mini-batch stochastic gradient descent with ADAM only one bit-string per basis is randomly chosen. In this case we have an effective data-set size $d_{\mathrm{eff}} = 500$. For $n = 100$ sites the memory requirement for the gradient w.r.t. the full data set is prohibitively large. In practice we computed the gradient w.r.t. chunks of the data set and summed up the results subsequently. The computation time was recorded on an Intel Core i9-10900 CPU@2.80GHz processor. However, the program utilizes at most 4 cores simultaneously. The time required for JIT compiling the computation is only required once for a given system size. The function $y = x$ (green dashed line) has the same slope as the run times, which illustrates the linear scaling. **Right:** As the system size grows, so does the memory requirement of the gradient computation. The data shown is the requirement for computing the gradient w.r.t. the full data set. For mini-batch stochastic gradients the memory requirement is reduced by a factor 100.

Lemma 1 (Derivatives of the loss function)**.** *The loss function $\mathcal{L}^{\mathcal{D}}$ is twice differentiable in every component.*

Proof. We start with the observation that the directional derivative of the matrix exponential can be expressed as

$$\frac{\mathrm{d}}{\mathrm{d}h}\mathrm{e}^{A+hV} = \int_0^1 \mathrm{e}^{(1-\tau)A} V \mathrm{e}^{\tau A}\mathrm{d}\tau. \tag{B2}$$

This can be verified as follows [51]. e^{A+hV} is the solution of the matrix-valued differential equation

$$\frac{\mathrm{d}X}{\mathrm{d}t} = (A + hV)X, \quad X(0) = \mathbb{1} \tag{B3}$$

at time $t = 1$. By subtracting AX and multiplying by e^{-At} from the left we can write the differential equation as

$$\frac{\mathrm{d}}{\mathrm{d}t}(\mathrm{e}^{-At}X) = \mathrm{e}^{-At}hVX, \tag{B4}$$

which has the formal solution

$$\mathrm{e}^{-At}X(t) = \mathbb{1} + \int_0^t \mathrm{e}^{-A\tau}hVX(\tau)\mathrm{d}\tau. \tag{B5}$$

Multiplying by e^{At} from the left and expanding the series we identify the first derivative of X with respect h

$$X(t) = \mathrm{e}^{At} + h\underbrace{\int_0^t \mathrm{e}^{A(t-\tau)}V\mathrm{e}^{A\tau}\mathrm{d}\tau}_{=\frac{\mathrm{d}X}{\mathrm{d}h}} + \mathcal{O}(h^2), \tag{B6}$$

which, evaluated at $t = 1$ proves the validity of Eq. (B2). Now we note that in the loss

$$\mathcal{L}^{\mathcal{D}}(\theta) = -\frac{1}{d}\sum_{i,j,k}\log\rho_{j,k}(s_i|\theta), \tag{B7}$$

the probabilities can be written as

$$\rho_{j,k}(s_i|\theta) := \langle\phi_{i,j,k}|\,\mathrm{e}^{-\mathrm{i}H(\theta)t}\,|0\rangle\,\langle 0|\,\mathrm{e}^{\mathrm{i}H(\theta)t}\,|\phi_{i,j,k}\rangle\,, \tag{B8}$$

where, in line with the main text, $|\phi_{i,j,k}\rangle$ is the post measurement state vector corresponding to $s_{i,j,k}$. In this form, it becomes apparent that if H is twice continuously differentiable, the same holds for the loss function. □

Lemma 2 (Zero-mean property of the score)**.** *In the limit of large d, where each $d_{j,k} \to \infty$, the scaled derivative of the loss converges in distribution to the normal distribution*

$$\sqrt{d}\,\nabla_\theta\hat{\mathcal{L}}^{(d)}(\theta^*) \xrightarrow{\text{dist.}} \mathcal{N}(0,\Sigma). \tag{B9}$$

Proof. Let $\hat{A}_{j,k} = \nabla_\theta \log\left(\rho_{j,k}(\hat{s}|\theta)\right)\big|_{\theta=\theta^*}$ denote the so called score, so that we can write

$$\nabla_\theta\hat{\mathcal{L}}^{(d)}(\theta^*) = \frac{1}{JK}\sum_{j,k}\hat{A}_{j,k}^{(d_{j,k})}, \tag{B10}$$

where $\hat{A}_{j,k}^{(N)}$ is the N-sample mean estimator of $\hat{A}_{j,k}$. To apply the central limit theorem to the score, we show that it has zero mean and finite variance. The mean is readily calculated to be

$$\mathbb{E}(\hat{A}_{j,k}) = \sum_{s\in\{0,1\}^n} \rho_{j,k}(s)\nabla_\theta \log\big(\rho_{j,k}(s|\theta)\big)\big|_{\theta=\theta^*} \tag{B11}$$

$$= \nabla_\theta \underbrace{\sum_{s\in\{0,1\}^n} \rho_{j,k}(s)|\theta}_{=1}\Bigg|_{\theta=\theta^*} \tag{B12}$$

$$= 0.$$

The (l, m)-th element of the covariance matrix is given by

$$\mathrm{Cov}_{l,m}(\hat{A}_{j,k}) \tag{B13}$$

$$= \sum_{s\in\{0,1\}^n} \rho^+_{j,k}(s)\big[\partial_l \rho_{j,k}(s|\theta)\big]\big[\partial_m \rho_{j,k}(s|\theta)\big]\Bigg|_{\theta=\theta^*},$$

where x^+ denotes the pseudo inverse of x. Since $\rho_{j,k}$ is twice differentiable all summands are finite and thus the sum is finite. Now, by the multivariate central limit theorem $\hat{A}^{(d_{j,k})}_{j,k} \xrightarrow{\text{dist.}} \mathcal{N}\big(0, \mathrm{Cov}(\hat{A}_{j,k})\big)$ as $d_{j,k} \to \infty$. Therefore, the sum of all scores converges in distribution to $\mathcal{N}(0, \Sigma)$, where

$$\Sigma = \frac{1}{JK}\sum_{j,k} \mathrm{Cov}(\hat{A}_{j,k}). \tag{B14}$$

□

Theorem 2 (Normality of the maximum likelihood estimator [52]). *Suppose the following conditions hold:*

1. *The estimator $\hat{\theta}^{(d)}$ is consistent, i.e., $\hat{\theta}^{(d)} \xrightarrow{\text{p.}} \theta^*$.*
2. $\theta^* \in \mathrm{interior}(\Theta)$.
3. *The loss $\hat{\mathcal{L}}^{(d)}$ is twice continuously differentiable in a neighborhood N of θ^*.*
4. *The loss satisfies $\sqrt{d}\nabla_\theta \hat{\mathcal{L}}^{(d)}(\theta^*) \xrightarrow{\text{dist.}} \mathcal{N}(0, \Sigma)$.*
5. *There exists a function $H : \mathbb{R}^p \to \mathbb{R}^{p\times p}$ that is continuous at θ^* with the property*
$$\sup_{\theta\in\mathcal{N}} \|\nabla^2_\theta \hat{\mathcal{L}}^{(d)}(\theta) - H(\theta)\| \xrightarrow{\text{p.}} 0.$$
6. *$H(\theta^*)$ is non-singular.*

Then the estimator $\hat{\theta}^{(d)}$ is asymptotically normal, i.e.,

$$\sqrt{d}(\hat{\theta}^{(d)} - \theta^*) \xrightarrow{\text{dist.}} \mathcal{N}\big(0, \tilde{\Sigma}\big),$$

where $\tilde{\Sigma} = H^{-1}(\theta^*)\Sigma H^{-1}(\theta^*)$.

Here $\xrightarrow{\text{p.}}$ and $\xrightarrow{\text{dist.}}$ denote convergence in probability and distribution, respectively. A proof can be found in Ref. [52]. We are now equipped to provide a rigorous statement and proof of Theorem 1 in the main text.

Theorem 3 (Asymptotic error). *Let $\hat{\mathcal{L}}^{(d)}$ and $\hat{\theta}^{(d)}$ be defined as above. Let $\Theta \subset \mathbb{R}^\nu$ be compact and suppose the true parameters θ^* are contained in the interior of Θ. Suppose the class*

$$\mathcal{C} = \{H(\theta)|\theta \in \Theta\}$$

of parametrized Hamiltonians is well conditioned, such that the estimator $\hat{\theta}^{(d)}$ is consistent and the Hessian at the true parameter value is non-singular. Furthermore, let the parametrization $\theta \mapsto H(\theta)$ be twice continuously differentiable. Then for any $\delta \in (0, 1]$ there exists a function $f(d) = \mathcal{O}(d^{-1/2})$ and some value $D \in \mathbb{N}$ such that $\mathbb{P}\big[\hat{\epsilon}^{(d)} > f(d)\big] < \delta$ for all $d \geq D$.

Proof. First we verify the conditions in Theorem 2 are satisfied. By assumption condition *1*, *2* and *6* are satisfied. By Lemma 1 and 2 conditions *3* and *4* are satisfied, respectively. Furthermore, both $\mathcal{L}$ and $\hat{\mathcal{L}}^{(d)}$ are twice continuously differentiable and by the law of large numbers $\hat{\mathcal{L}}^{(d)}(\theta) \xrightarrow{\text{p.}} \mathcal{L}(\theta)$ for all $\theta \in \Theta$. Hence, we also have $\nabla^2_\theta \hat{\mathcal{L}}^{(d)}(\theta) \xrightarrow{\text{p.}} \nabla^2\mathcal{L}(\theta)$, element-wise in the interior of Θ, which suffices to show that condition *5* is satisfied, where $\nabla^2\mathcal{L}$ is the required function H.

Now denote a Euclidean ball of radius r centered at 0 in $\mathbb{R}^\nu$ by B_r and its complement by $\bar{B}_r = \mathbb{R}^\nu \setminus B_r$. Furthermore, let Z be a random variable with distribution $\mathcal{N}(0, \tilde{\Sigma})$ and denote its probability density function by p. Now define

$$r(\delta) := \min\{r|\mathbb{P}(Z \in \bar{B}_r) \leq \delta\}. \tag{B15}$$

Since p is normalized we have $r(\delta) < \infty$ for all $\delta \in (0, 1]$. Now note that

$$\mathbb{P}\big[\sqrt{d}\,Z \in \bar{B}_{r(\delta)}\big] = \mathbb{P}\big[Z \in \bar{B}_{r(\delta)/\sqrt{d}}\big], \tag{B16}$$

which can be verified by writing out the integral over p and scaling the argument of p by a factor of $1/\sqrt{d}$. Together with this identity and the fact that $\sqrt{d}(\hat{\theta}^{(d)} - \theta^*)$ converges in distribution to Z, we can conclude that

$$\lim_{d\to\infty} a_d = \delta, \tag{B17}$$

where $a_d(\delta) := \mathbb{P}\big[\hat{\theta}^{(d)} - \theta^* \in \bar{B}_{r(\delta)/\sqrt{d}}\big]$. Now consider the sequence $a_d(\delta/2)$. Since it converges to $\delta/2$ there exists a value $D \in \mathbb{N}$ such that $a_d(\delta/2) < \delta$ for all $d \geq D$. Lastly, the condition $\hat{\theta}^{(d)} - \theta^* \in \bar{B}_{r(\delta/2)/\sqrt{d}}$ translates to

$$\hat{\epsilon}^{(d)} < \frac{r(\delta/2)}{\sqrt{d}\,\|\theta^*\|_2}, \tag{B18}$$

which gives us the function f described in the theorem. □

Note that showing that the class of parametrized Hamiltonians $\mathcal{C}$ is well conditioned, in the sense of the theorem, is not trivial. Additional restrictions are necessary to exclude pathological cases in which the times $\{t_j\}$ are commensurate with the oscillations in the state amplitudes during time evolution. Also one must ensure that the initial state has non-zero

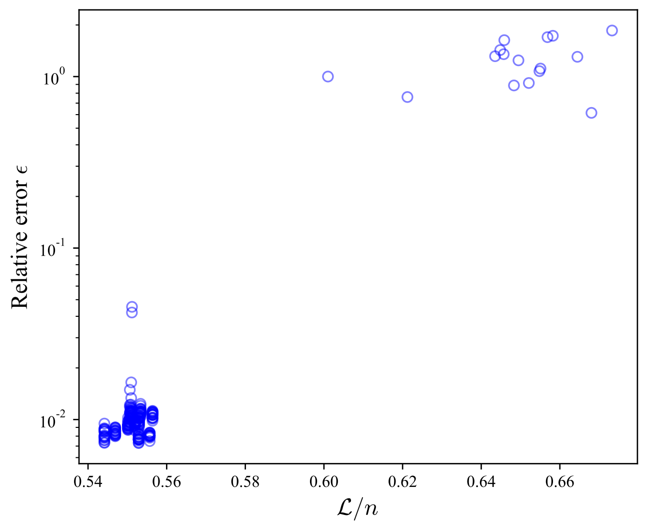

FIG. 5. The same data points as in Fig. 6(b) are shown. Here the relative error ϵ (for various system sizes) is plotted against the loss function value $\mathcal{L}$, normalized by the system size n. One can clearly divide outliers from successfully converged parameter values purely based on the loss function value.

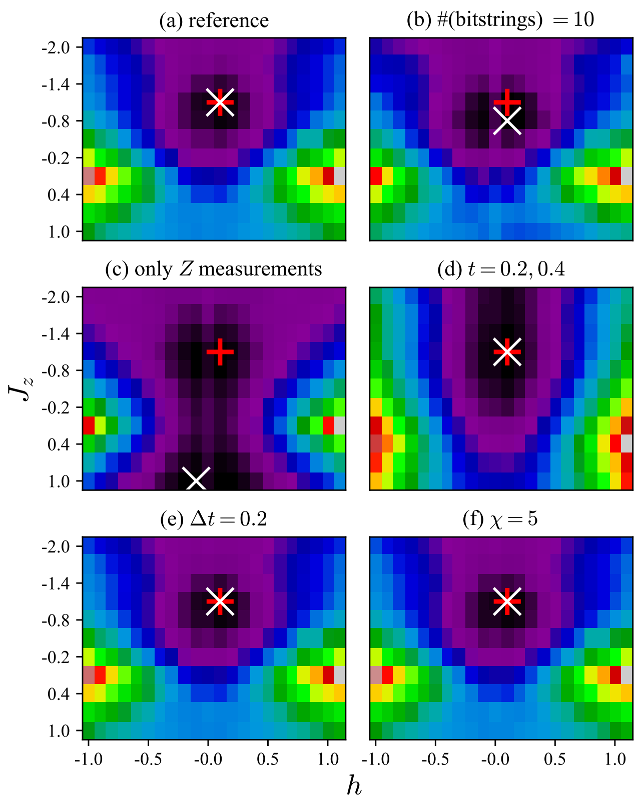

FIG. 6. The loss function in the parameters h (uniform over all spins) and J_z in arbitrary units (black is low). To compute the loss 1000 bit-strings have been sampled for each basis choice, all-X, all-Y, all-Z, and for each time stamp $t = 0.2, 0.4, 0.6, 0.8, 1.0$. The Trotter step size is $\Delta t = 0.05$ and the bond dimension is $\chi = 10$. The red cross shows the true parameters, while the white cross shows the minimum of the loss function. For reference the loss landscape is shown in **(a)** with these exact meta parameters.

overlap with sufficiently many eigenstates of the Hamiltonian to be able to observe meaningful dynamics in the system. In practice one is unlikely to encounter these issues, which is also exemplified by the fact that one can, in principle, recover a generic geometrically local and traceless Hamiltonian from only one time stamp using a generic initial state [23].

Appendix C: Loss function

The analysis of our method is based on the relative error ϵ. However, in a practical setting, where one doesn't know the Hamiltonian, the error is not accessible. Instead one needs to gauge convergence and success of the optimization by the loss function $\mathcal{L}$. To demonstrate that one can indeed distinguish outliers, as shown in Fig. 6 from successfully converged parameters the error is plotted against the loss function value in Fig. 5.

To investigate the impact various experimental and meta parameters have on the loss function landscape and the ability to recover the true Hamiltonian, we plotted the landscape in a model with $\nu = 2$ parameters. In this example instead of sampling Pauli bases at random, only the bases $X^{\otimes n}$, $Y^{\otimes n}$, and $Z^{\otimes n}$ have been considered. The results are shown in Fig. 6. In **(b)** the number of bit-strings has been reduced. In **(c)**, only bit-strings from all-Z measurements have been used. In **(d)**, only samples from the first two time stamps have been made use of. In **(e)**, the Trotter step size has been increased leading to higher errors. In **(f)**, the bond dimension has been reduced, again increasing the errors. It becomes apparent, that only measuring in the Z basis has negative effects on the loss landscape. Also one can clearly see that too short times are not sufficient for the loss function to form a well pronounced minimum.

Appendix D: Differentiation of the singular value decomposition

The *singular value decomposition* (SVD) is an integral part of many tensor network algorithms [32]. Apart from linear algebra applications such as computing the rank or the kernel of a matrix, the SVD is widely used in physics, for instance to access the Schmidt spectrum, the entanglement entropy, and reduced density matrices of bipartite quantum states.

The Jacobian-vector product can be used for forward-mode automatic differentiation, but allows reverse-mode automatic differentiation as well by using transposition rules, thereby avoiding the need to work with the dual transformation (vector-Jacobian product) explicitly. As a design choice, the popular open-source automatic differentiation framework JAX [38] contains only Jacobian-vector product rules for non-linear transformations (such as the SVD) along with transposition rules for linear transformations. The results derived here have been included into JAX (github.com/google/jax/pull/5225).

Previous work. We emphasize that the vector-Jacobian product of the complex SVD has been worked out previously [53, 54]. These results provide the basis for our derivation of the JVP rule. Our aim is to make the differentiation of the SVD accessible to a broad audience, not necessarily familiar with the field of automatic differentiation. Additionally, we provide the Jacobian-vector product and discuss its lack of uniqueness. The reader may also be referred to a larger collection of automatic differentiation formulas for linear algebra transformations [55].

Notation. Let $\mathbb{1}$ be the identity matrix and $\bar{\mathbb{1}}$ its complement, i.e., where the diagonal is zero and all off-diagonal entries are one. If necessary we will annotate the matrix dimensions, e.g., $\mathbb{1}_{n\times m}$. Let $\circ$ denote the element-wise matrix product, say, $(A\circ B)_{i,j} = A_{i,j}B_{i,j}$.

Automatic differentiation. Imagine we have a function $L:\mathbb{R}\to\mathbb{R}$ which is composed of several functions $L(p) = f(\mathsf{S}(g(p)))$, where S may denote the SVD. The derivative of L is then given by the chain rule $\mathrm{D}L = \mathrm{D}f\cdot\mathrm{D}\mathsf{S}\cdot\mathrm{D}g$. To compute the derivative, we can multiply these three Jacobians beginning from the left or the right, which is called backward-mode or forward-mode differentiation, respectively. In the former case one needs the vector-Jacobian product, in the latter the Jacobian-vector product. This technique can be used on more complex functions L as well. Instead of elaborating on more general cases we refer the reader to Ref. [56].

Derivation. We begin by stating the relation between the input $A\in\mathbb{C}^{n\times m}$ of the SVD and its output $U\in\mathbb{C}^{n\times k}, S=\mathrm{diag}(s_1,\dots,s_k)\in\mathbb{C}^{k\times k}, V^\dagger\in\mathbb{C}^{k\times m}$, where $s_i\in\mathbb{R}$ for all i and $k=\min(n,m)$. We have

$$A = USV^\dagger. \tag{D1}$$

The total derivative of this relation [57]

$$\mathrm{d}A = \mathrm{d}USV^\dagger + U\mathrm{d}SV^\dagger + US\mathrm{d}V^\dagger \tag{D2}$$

allows us to derive the differentiation rules. As we will see, this derivative is not well defined as it stands, since there is a gauge freedom in the matrices U and $V^\dagger$. Our aim is to find expressions for $\mathrm{d}U$, $\mathrm{d}S$, and $\mathrm{d}V^\dagger$ in terms of $\mathrm{d}A$.

Square matrices. We assume $m=n$ for the sake of lucidity and explain additional steps necessary for non-square matrices later. First we transform $\mathrm{d}A$ by multiplying $U^\dagger$ and V from the right and left, respectively. We obtain

$$\mathrm{d}\tilde{A} = \mathrm{d}\tilde{U}S + \mathrm{d}S + S\mathrm{d}\tilde{V}^\dagger, \tag{D3}$$

where $\mathrm{d}\tilde{U} = U^\dagger\mathrm{d}U$ and $\mathrm{d}\tilde{V} = V^\dagger\mathrm{d}V$. Note that

$$\mathrm{d}\tilde{U} = -\mathrm{d}\tilde{U}^\dagger \quad \mathrm{d}\tilde{V} = -\mathrm{d}\tilde{V}^\dagger \tag{D4}$$

due to U and V being unitary, i.e., $\mathrm{d}(U^\dagger U) = \mathrm{d}\mathbb{1} = 0$. This implies that the diagonal of $\mathrm{d}\tilde{U}$ and $\mathrm{d}\tilde{V}$ are purely imaginary and we can eliminate them by adding $\mathrm{d}\tilde{A}$ to its Hermitian conjugate. With this we can already state the derivative of the singular values

$$\mathrm{d}S = \frac{1}{2}(\mathrm{d}\tilde{A} + \mathrm{d}\tilde{A}^\dagger). \tag{D5}$$

To compute $\mathrm{d}\tilde{U}$ and $\mathrm{d}\tilde{V}$ we first focus on the off-diagonal portion. This is closely related to the derivation for SVD of real matrices [58]. In contrast, the diagonals need extra care, which has no counterpart in the real case. Let $i\neq j$ such that we can disregard $\mathrm{d}S$ and consider the matrix element

$$\begin{aligned}(\mathrm{d}\tilde{A}S + S\mathrm{d}\tilde{A}^\dagger)_{i,j} = {} & \mathrm{d}\tilde{U}_{i,j}S_{j,j}^2 + S_{i,i}\mathrm{d}\tilde{V}_{i,j}^\dagger S_{j,j} \\ & + S_{i,i}^2\mathrm{d}\tilde{U}_{i,j}^\dagger + S_{i,i}\mathrm{d}\tilde{V}_{i,j}S_{j,j},\end{aligned} \tag{D6}$$

using the Einstein summation convention. Using (D4), we see that the two terms involving $\mathrm{d}\tilde{V}$ vanish while the other two can be combined. We can solve the equation element-wise for $\mathrm{d}\tilde{U}$ using the matrix with entries $F_{i,j} = 1/(s_j^2 - s_i^2)$ for $i\neq j$ and zero otherwise. We can use the same matrix to solve for $\mathrm{d}\tilde{V}^\dagger$ and obtain

$$\bar{\mathbb{1}}\circ\mathrm{d}\tilde{U} = F\circ\frac{1}{2}(\mathrm{d}\tilde{A}S + S\mathrm{d}\tilde{A}^\dagger), \tag{D7}$$

$$\bar{\mathbb{1}}\circ\mathrm{d}\tilde{V} = F\circ\frac{1}{2}(S\mathrm{d}\tilde{A} + \mathrm{d}\tilde{A}^\dagger S). \tag{D8}$$

What is left is to determine the diagonals. We do this by considering the diagonal of $\mathrm{d}\tilde{A} - \mathrm{d}\tilde{A}^\dagger$. Since this eliminates the (purely real) components of $\mathrm{d}S$ we are left with

$$S^{-1}\circ\frac{1}{2}(\mathrm{d}\tilde{A} - \mathrm{d}\tilde{A}^\dagger) = \mathbb{1}\circ(\mathrm{d}\tilde{U} + \mathrm{d}\tilde{V}^\dagger). \tag{D9}$$

This tells us the sum of $\mathrm{d}\tilde{U}$'s and $\mathrm{d}\tilde{V}^\dagger$'s diagonals, but not the individual summands. We will now show that we can distribute this sum in any arbitrary way (for instance half-half) between the two matrices. To see this consider an arbitrary matrix element of $\mathrm{d}A$

$$\mathrm{d}A_{i,l} = U_{i,j}\mathrm{d}\tilde{U}_{j,k}S_{k,k}V_{l,k}^* \tag{D10}$$
$$+ U_{i,j}S_{j,j}\mathrm{d}\tilde{V}_{k,j}^* V_{l,k}^* \tag{D11}$$
$$+ U_{i,j}\mathrm{d}S_{j,j}V_{l,j}^*.$$

We are interested in all summands in (D10) and (D11) that contain diagonal elements of $\mathrm{d}\tilde{U}$ and $\mathrm{d}\tilde{V}^\dagger$, i.e., summands where $j=k$. This reduced sum turns out to be $U_{i,j}S_{j,j}V_{l,j}^*(\mathrm{d}\tilde{U}_{j,j} + \mathrm{d}\tilde{V}_{j,j}^*)$ which implies the above claim, that any solution of equation (D9) suffices to recover $\mathrm{d}A$. Hence one valid Jacobian-vector product rule is

$$\mathrm{d}U = \frac{U}{2}\left[F\circ(\mathrm{d}\tilde{A}S + S\mathrm{d}\tilde{A}^\dagger) + S^{-1}\circ(\mathrm{d}\tilde{A} - \mathrm{d}\tilde{A}^\dagger)\right], \tag{D12}$$

$$\mathrm{d}V = \frac{V}{2}\left[F\circ(S\mathrm{d}\tilde{A} + \mathrm{d}\tilde{A}^\dagger S)\right]. \tag{D13}$$

Gauge freedom. In addition to the freedom of choice of the diagonal of $\mathrm{d}\tilde{U}$ and $\mathrm{d}\tilde{V}$ there is a freedom in the choice of U and V in the SVD. Notice that the SVD is invariant under the transformation $U\mapsto U\Lambda$ and $V^\dagger\mapsto\Lambda^\dagger V^\dagger$, with $\Lambda = \mathrm{diag}(\mathrm{e}^{\mathrm{i}\phi_1},\dots,\mathrm{e}^{\mathrm{i}\phi_k})$ for arbitrary ϕ_i's as $\Lambda^\dagger S\Lambda = S$. Consequently the derivatives transform as $\mathrm{d}U\mapsto\mathrm{d}U\Lambda$ and

$\mathrm{d}V^\dagger \mapsto \Lambda^\dagger \mathrm{d}V^\dagger$. Therefore, we can only consider functions L (discussed in the beginning) which, at some point during the calculation, eliminate the gauge freedom, perhaps by multiplying U with $f(S)$ and then $V^\dagger$ for some differentiable function f.

Non-square matrices. To solve the case where $n \neq m$ we assume that $n > m$, such that $U \in \mathbb{C}^{n\times m}$, $S \in \mathbb{R}^{m\times m}$, and $V \in \mathbb{C}^{m\times m}$. It will be straightforward to repeat the following arguments for $n < m$. Following the reasoning presented in Ref. [58] we first notice that U is no longer an element of the unitary group. In fact, it is an element of the *Stiefel manifold* $V_m(\mathbb{C}^n)$. While it is still true that $U^\dagger U = \mathbb{1}_{m\times m}$, we now have $UU^\dagger \neq \mathbb{1}_{n\times n}$. It follows that $\mathrm{d}\tilde{U} \in \mathbb{C}^{m\times m}$ is still well defined and anti-Hermitian $\mathrm{d}\tilde{U} = -\mathrm{d}\tilde{U}^\dagger$. Since these are the properties we have used to derive the expressions for $\mathrm{d}S$, $\mathrm{d}\tilde{U}$ and $\mathrm{d}\tilde{V}$ (Eqs. (D5), (D7), and (D8)) they remain valid in the non-square case.

However, extra care is required when recovering $\mathrm{d}U$ since we cannot simply multiply $\mathrm{d}\tilde{U}$ by U from the left. Instead we need to extend $U \in \mathbb{C}^{n\times m}$ by $U_\perp \in \mathbb{C}^{n\times(n-m)}$ such that the stacked matrix $[U, U_\perp] \in \mathbb{C}^{n\times n}$ is unitary. Here the columns of $U_\perp$ consist of orthonormal vectors in the complement of the image of U. This allows us to recover the $n \times n$ identity

$$[U, U_\perp] \cdot \begin{bmatrix} U^\dagger \\ U_\perp^\dagger \end{bmatrix} = UU^\dagger + U_\perp U_\perp^\dagger = \mathbb{1}_{n\times n}. \quad \text{(D14)}$$

Analogously, we extend $\mathrm{d}\tilde{U}$ by an element $\mathrm{d}\tilde{U}_\perp \in \mathbb{C}^{(n-m)\times m}$ which acts in the image of $U_\perp$ such that

$$\mathrm{d}U = U\mathrm{d}\tilde{U} + U_\perp \mathrm{d}\tilde{U}_\perp. \quad \text{(D15)}$$

This allows us to compute $\mathrm{d}\tilde{U}_\perp$ by multiplying $\mathrm{d}A$ by $U_\perp^\dagger$ which yields $\mathrm{d}\tilde{U}_\perp = U_\perp^\dagger \mathrm{d}AVS^{-1}$. Plugging this into (D15) we have the recovery in terms of known matrices

$$\mathrm{d}U = U\mathrm{d}\tilde{U} + (\mathbb{1} - UU^\dagger)\mathrm{d}AVS^{-1}, \quad \text{(D16)}$$

where we have used (D14) to substitute $U_\perp U_\perp^\dagger$ and as before we may choose

$$\mathrm{d}\tilde{U} = \frac{1}{2}\left[F \circ (\mathrm{d}\tilde{A}S + S\mathrm{d}\tilde{A}^\dagger) + S^{-1} \circ (\mathrm{d}\tilde{A} - \mathrm{d}\tilde{A}^\dagger)\right]. \quad \text{(D17)}$$

Analogously we can derive the Jacobian-vector product rule for $n < m$ and obtain

$$\mathrm{d}V^\dagger = \mathrm{d}\tilde{V}^\dagger V^\dagger + S^{-1}U^\dagger \mathrm{d}A(\mathbb{1} - VV^\dagger). \quad \text{(D18)}$$

Choice of the diagonal component. As shown in (D9) and subsequent arguments one has the freedom to split the diagonal given in (D9) and add its parts to $\mathrm{d}\tilde{U}$ and $\mathrm{d}\tilde{V}^\dagger$ given in (D7) and (D8). In order to demonstrate the effect of this splitting on the precision of the Jacobian-vector product rule we differentiated the function

$$f(p) = \mathrm{Re}(U(p) \cdot S(p) \cdot V^\dagger(p)), \quad \text{(D19)}$$
$$U(p), S(p), V^\dagger(p) = \mathsf{SVD}(\exp(A + pB))$$

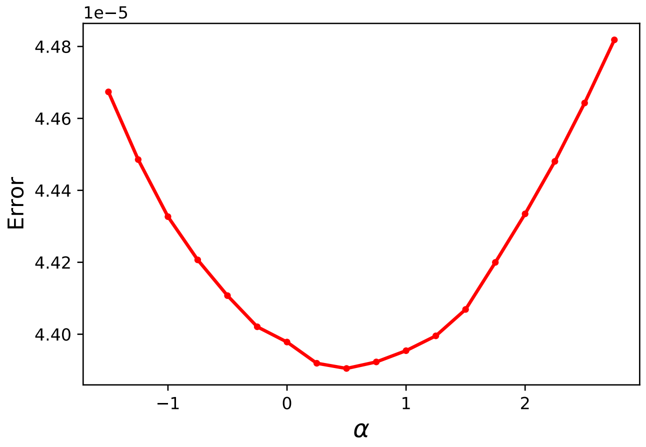

FIG. 7. The error for 100 pairs of random matrices A and B for various values of α. The error is defined by the Hilbert-Schmidt distance between the analytical derivative and the automatic-differentiation result $\|\mathrm{d}f/\mathrm{d}p - \mathsf{autodiff}(f)\|_{\mathrm{HS}}$.

for 100 pairs of random 100×100 matrices A and B, where the exponential function is applied *element-wise*. A splitting parameter α is multiplied to the diagonal (Eq. D9) which is then added to $\mathrm{d}\tilde{U}$ (Eq. D7) while a $(1-\alpha)$ multiple of the diagonal is added to $\mathrm{d}\tilde{V}^\dagger$ (Eq. D8). The differential of the function (D19) is evaluated at $p = 20$ and compared to the analytical derivative $\mathrm{d}f/\mathrm{d}p = \exp(A + pB) \circ B$. The results shown in Fig. 7 suggest that an even splitting, i.e., $\alpha = 1/2$ yields the highest precision of the Jacobian-vector product rule. However, it might be more efficient to choose $\alpha = 0$ or $\alpha = 1$ to reduce computational operations.