

Real time evolution with neural-network quantum states

Irene López Gutiérrez^{1,2} and Christian B. Mendl^{1,2}

¹Technische Universität München, Department of Informatics and Institute for Advanced Study, Boltzmannstraße 3, 85748 Garching, Germany

²Technische Universität Dresden, Institute of Scientific Computing, Zellescher Weg 12-14, 01069 Dresden, Germany

23rd December 2021

A promising application of neural-network quantum states is to describe the time dynamics of many-body quantum systems. To realize this idea, we employ neural-network quantum states to approximate the implicit midpoint rule method, which preserves the symplectic form of Hamiltonian dynamics. We ensure that our complex-valued neural networks are holomorphic functions, and exploit this property to efficiently compute gradients. Application to the transverse-field Ising model on a one- and two-dimensional lattice exhibits an accuracy comparable to the stochastic configuration method proposed in [Carleo and Troyer, *Science* 355, 602–606 (2017)], but does not require computing the (pseudo-)inverse of a matrix.

1 Introduction

The main difficulty in simulating strongly interacting many-body quantum systems on classical computers stems from the curse of dimensionality. However, a closer examination reveals that the manifold of physical quantum many-body states occupies an exponentially small volume in the Hilbert space [23]. The challenge, then, is to find an appropriate variational ansatz which has few degrees of freedom while faithfully representing physical states.

The recent successes of artificial neural network techniques have entailed a large interest in applying them to quantum many-body systems, in particular as ansatz for the wavefunction of (strongly correlated) quantum systems [5, 13, 21, 22]. Such neural-network quantum states have the principal capability to describe systems hosting chiral topological phases [8, 13, 17], or to handle large entanglement [11, 12, 19]. In view of real time evolution, this could turn out to be a considerable advantage compared to established tensor network methods [10, 26, 27, 33, 34], since the increase of entanglement with time demands an exponential increase of virtual bond dimensions, thus limiting the applicability of tensor network methods to relatively short time intervals [1, 4].

Irene López Gutiérrez: irene.lopez@tum.de

Christian B. Mendl: christian.mendl@tum.de

While the favorable capabilities of neural-network quantum states have been investigated theoretically [12, 19], demonstrations of their practical feasibility for quantum time evolution are still rather sparse (but see [5, 7, 24]). The canonical Dirac-Frenkel time-dependent variational principle can be regarded as projecting the time step vector onto the tangent space of the variational manifold [14]. Time-dependent variational Monte Carlo (tdVMC) [5, 6, 29] combines the Dirac-Frenkel principle with Monte Carlo sampling and exploits the locality of typical quantum Hamiltonians. This involves the application of the (pseudo-)inverse of a covariance matrix to evolve the variational parameters in time. However, we find that in practice tdVMC can be rather sensitive to the chosen cutoff tolerance for the pseudo-inverse, or demand a prohibitively small time step for “deep” neural-network quantum states. Here we propose and explore an alternative approach, namely directly approximating a time step of a conventional ordinary differential equation (ODE) method by “training” the neural-network quantum state at the next time step using (variations of) stochastic gradient descent.

2 Time evolution method

Our goal is to solve the time-dependent Schrödinger equation

$$i \frac{\partial \psi}{\partial t} = H \psi. \quad (1)$$

We denote the variational ansatz by $\psi[\theta]$, where $\theta \in \mathbb{C}^K$ is a complex vector containing all variational parameters, which are assumed to be time-dependent. From this perspective, it is possible to find the gradients of ψ with respect to θ and use the chain rule together with tangent space projections to derive an ODE for θ . In stochastic reconfiguration (SR) [5, 6], which is based on the Dirac-Frenkel variational principle, the final equation to be solved reads

$$S \dot{\theta} = -iF, \quad (2)$$

with the covariance matrix

$$S_{j,k} = \langle O_j^*; O_k \rangle \quad (3)$$

and force vector

$$F_j = \langle E_{\text{loc}}; O_j^* \rangle, \quad (4)$$

where $\langle A; B \rangle = \langle AB \rangle - \langle A \rangle \langle B \rangle$ is the connected correlation function. The brackets denote the Monte Carlo average over samples drawn from the probability distribution $|\psi[\theta]|^2$, since for large system sizes it is not possible to consider the full wavefunction. For each sample σ ,

$$O_j(\sigma) = \partial_{\theta_j} \log(\psi[\theta](\sigma)), \quad (5)$$

$$E_{\text{loc}}(\sigma) = \frac{(H\psi[\theta])(\sigma)}{\psi[\theta](\sigma)}. \quad (6)$$

The covariance matrix S is almost always effectively singular, which means it is not possible to exactly solve Eq. (2). Instead, the best update for the parameters should be found by minimising $\|S\hat{\theta} + iF\|$. One way of achieving this is by means of the Moore-Penrose pseudo-inverse. However, finding the appropriate pseudo-inverse requires choosing the right cut-off for small singular values, which can be rather challenging, specially for the real-time evolution. Krylov subspace methods, such as the conjugate gradient method or the MINRES algorithm, avoid this sensitivity problem by iteratively approximating the solution of the linear system, and have the added advantage of having low memory requirements. Their convergence to the optimal solution, however, is not guaranteed [16, 20]. Further note that Eq. (2) has dimension K (number of variational parameters), which limits the feasible Ansätze in practice. Recently, a regularization scheme has been developed [25] which avoids issues associated with the pseudo-inverse; nevertheless, this scheme still requires to diagonalize S , incurring an $\mathcal{O}(K^3)$ computational cost. In general, the sensitivity of Eq. (2) with respect to S might require a very large number of samples.

Here we propose a different approach, which fits more directly to the paradigm of neural network training: for each time step Δt , we optimize the network parameters to minimize the error

$$\|\psi[\theta_{n+1}] - \Phi^{\Delta t}(\psi[\theta_n])\| \quad (7)$$

with respect to θ_{n+1} , where $\Phi^{\Delta t}$ is the discrete flow of a numerical ODE method applied to the Schrödinger equation. We will use the implicit midpoint method here. For an ODE $y'(t) = f(t, y(t))$ and a time-step Δt , this method is defined by

$$y_{n+1} = y_n + \Delta t f\left(t_n + \frac{\Delta t}{2}, \frac{1}{2}(y_n + y_{n+1})\right). \quad (8)$$

In the specific case of the Schrödinger equation, this leads to (cf. the Cayley transform)

$$\psi[\theta_{n+1}] \approx \psi[\theta_n] - i\Delta t H \left(\frac{\psi[\theta_{n+1}] + \psi[\theta_n]}{2} \right). \quad (9)$$

The implicit midpoint method has two important favorable properties: firstly, it preserves the symplectic form of Hamiltonian dynamics [14], and secondly, it

does not contain intermediate quantities that would complicate the network optimization.

In the case of larger systems, where sampling becomes necessary, we minimize the following cost function for a single midpoint rule time step:

$$C(\theta_{n+1}) = \sum_{j=1}^N \left| \left(\left(I + \frac{i\Delta t}{2} H \right) \psi[\theta_{n+1}] - \left(I - \frac{i\Delta t}{2} H \right) \psi[\theta_n] \right) (\sigma^{(j)}) \right|^2, \quad (10)$$

with the $\sigma^{(j)}$, $j = 1, \dots, N$ a batch of input configurations, and the network parameters at the current time point, θ_n , regarded as fixed. To be specific, we consider spin variables as input in the following, and denote the system size by L , i.e., the quantum Hilbert space dimension (number of possible spin configurations) is 2^L . The cost function can be compactly represented in least squares form as

$$C(\theta) = \|A\psi[\theta] - b\|^2, \quad (11)$$

with $A = \mathbb{C}^{N \times 2^L}$ the (sparse) submatrix of $I + \frac{i\Delta t}{2} H$ containing the rows corresponding to the spin configurations $\sigma^{(j)}$, and the vector $b \in \mathbb{C}^N$ with entries

$$b_j = \left(\left(I - \frac{i\Delta t}{2} H \right) \psi[\theta_n] \right) (\sigma^{(j)}). \quad (12)$$

Assuming that $\psi[\theta]$ is a holomorphic function of the parameters θ and following the derivation in appendix A leads to

$$\frac{\partial C(\theta)}{\partial \theta_\ell} = \left\langle A\psi[\theta] - b \left| A \frac{\partial \psi[\theta]}{\partial \theta_\ell} \right. \right\rangle. \quad (13)$$

Note that since $C(\theta)$ is not holomorphic, the partial derivative on the left side of this expression is a Wirtinger derivative.

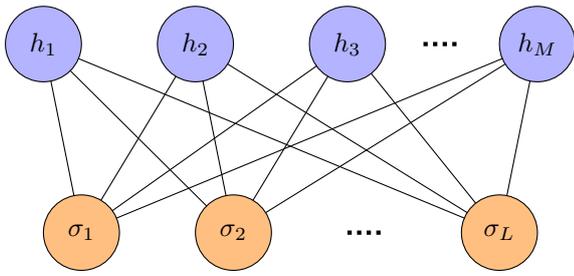
Appendix D provides a geometrical perspective on the SR and implicit midpoint methods.

3 Application to the Ising chain

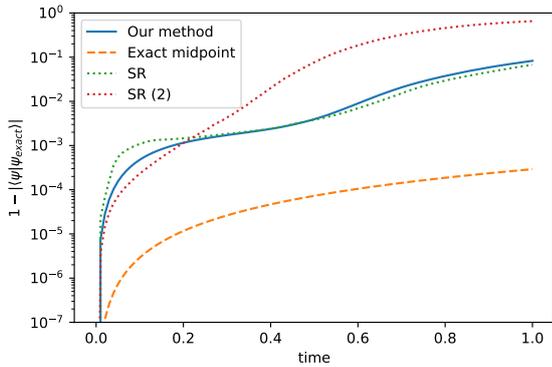
The neural-network architecture proposed as a wavefunction ansatz in [5] is the restricted Boltzmann machine (RBM). It has been shown to represent the ground state of various Hamiltonians with high accuracy [3]. As visualized in Fig. 1a, a RBM consists of two layers of neurons, referred to as the ‘visible’ and ‘hidden’ layers, which are connected with one another but have no intra-layer connections. The input of the visible layer is a specific spin configuration σ . From this architecture one obtains a variational ansatz for the wavefunction of the system

$$\psi(\sigma) = \sum_{\{h_i\}} e^{\sum_j a_j \sigma_j + \sum_i b_i h_i + \sum_{i,j} w_{ij} h_i \sigma_j}, \quad (14)$$

where a_j, b_i and w_{ij} are the network parameters, and h_i are the auxiliary spin variables which only take the



(a) Restricted Boltzmann machine architecture



(b) Time evolution error

Figure 1: (a) Schematic diagram of a restricted Boltzmann machine showing the visible (orange) and hidden (blue) layers and the connections between the neurons. (b) Error in the time evolution performed using a RBM with 80 hidden units (1700 complex parameters in total), time step 0.01 and 50,000 uniformly drawn samples per time step. For comparison, SR and SR (2) show the stochastic reconfiguration method proposed in [5] with two different pseudo-inverse thresholds.

values ± 1 . Due to the summation over h_i in Eq. (14), it is possible to trace out the hidden spin variables:

$$\psi(\sigma) = e^{\sum_j a_j \sigma_j} \prod_i 2 \cosh\left(b_i + \sum_j w_{ij} \sigma_j\right). \quad (15)$$

In this section, we apply the RBM ansatz to the one-dimensional transverse-field Ising model, which consists of a chain of spins that interact with their nearest neighbors and are subject to an external magnetic field, h . Its Hamiltonian for general lattice dimension is given by

$$H_{\text{TFI}} = -J \sum_{\langle i,j \rangle} \sigma_i^z \sigma_j^z - h \sum_i \sigma_i^x. \quad (16)$$

Such a system undergoes a phase transition from the ferromagnetic to the paramagnetic regime at $h = J$ in one dimension [30] and at $h/J = 3.04438(2)$ in two dimensions [2]. In the following, we set the coupling constant J to 1.

We consider a chain of length $L = 20$, which is small enough to compute the exact time evolution as a reference. For the training we use the Adam optimizer

and choose the recommended hyperparameters from Ref. [18].

Fig. 1b shows a comparison between the error obtained with stochastic reconfiguration and the error of our method. The time evolution labeled SR was computed with a pseudo-inverse threshold of 10^{-10} , while for SR(2) a threshold of 10^{-9} was used. The importance and sensitivity of this threshold is reflected in the difference in the final error. We also plot the contribution to the error by the midpoint method. Even without a finely tuned optimization, our method yields comparable results to stochastic reconfiguration with an optimal pseudo-inverse cut-off. The additional error compared to the exact midpoint integration hints towards a lack of expressibility resulting from the RBM ansatz.

Since we are interested in physically realistic states, the initial state was found by performing a Hamiltonian quench with respect to the field strength h . We first optimize the network parameters to represent the ground state for $h = 1.5$ and then change h to 0.75 for the real time evolution. As a measure of the accuracy of the ground state, the deviation from the exact energy is $|(E[\theta] - E_0)/E_0| \approx 0.051$, where E_0 is the exact energy found by diagonalising the Hamiltonian and $E[\theta]$ is the energy of the quantum neural network state.

The training was performed on a single CPU and 64 GB memory, using multi-threading on 28 cores. We have implemented the neural network architectures and optimization part of the algorithm in a custom C code, with a Python interface to schedule and evaluate the runs. In this setup we find that a single optimization step of the RBM network with 20 lattice sites and 50,000 samples takes approximately 3.6s.

4 Application to the Ising model on a square lattice

To demonstrate the flexibility of our method we change the neural network architecture and consider the time evolution governed by the two-dimensional

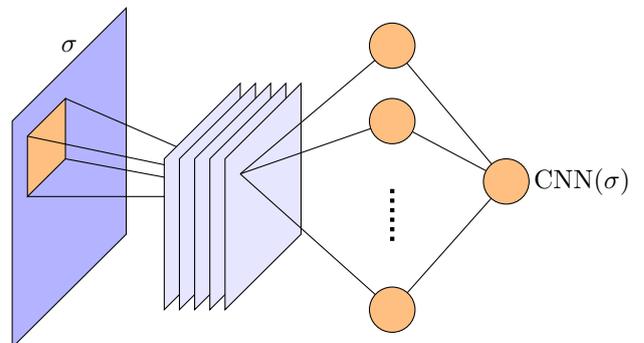


Figure 2: Schematic diagram of a convolutional neural network.

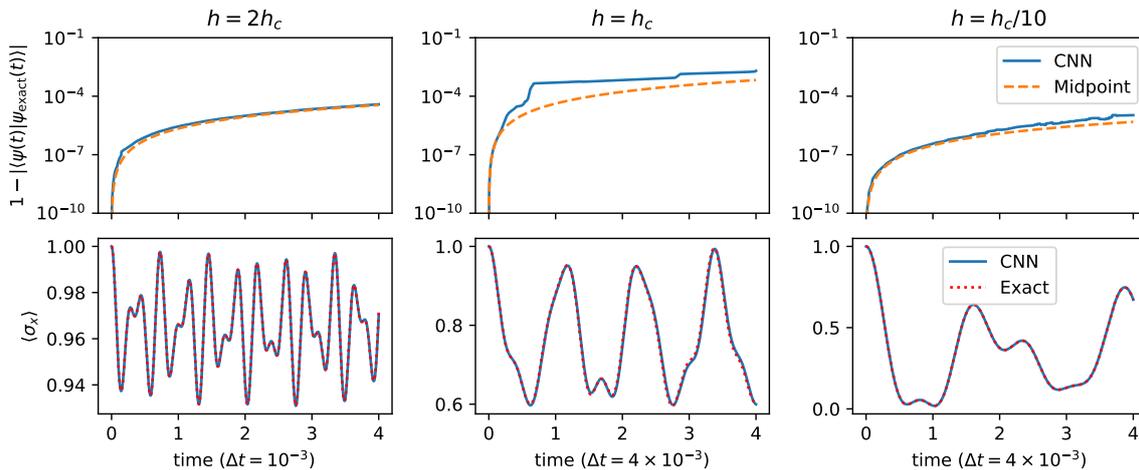


Figure 3: Overlap error (top row) and transverse magnetization (bottom row) of the real time evolution governed by the Ising Hamiltonian (16) on a 3×3 lattice after a quench of h . Each column corresponds to a different value of h after the quench, starting from “infinite” h . We used 500 uniformly drawn samples for each individual optimization step of our method (CNN).

Ising model on a $L \times L$ lattice with periodic boundary conditions, setting $L = 3$ in the following. As an ansatz, we directly use the output of a convolutional neural network,

$$\psi(\sigma) = \text{CNN}(\sigma) \quad (17)$$

as shown in Fig. 2. Specifically, the network architecture consists of a convolutional layer with five complex-valued 2×2 filters and periodic boundary conditions, an intermediate dense layer with 10 neurons, and a single output. This correspond to 496 complex parameters in total. For the activation functions, we found the polynomial functions described in Ref. [25] to achieve errors several orders of magnitude smaller compared to other more commonly used activation functions, such as ReLUs or sigmoid functions. Specifically, we use

$$\sigma(z) = \frac{1}{2}z^2 - \frac{1}{12}z^4 + \frac{1}{45}z^6 \quad (18)$$

for the convolutional layer, and

$$\sigma(z) = z - \frac{1}{3}z^3 + \frac{2}{15}z^5 \quad (19)$$

for subsequent layers.

We perform three different Hamiltonian quenches, from “infinite” h to $h = 2h_c$, h_c and $h_c/10$, where h_c is the critical point at which the system undergoes a phase transition. Here “infinite” h is equivalent to retaining only the second term in the Hamiltonian (16), such that the corresponding ground state is the paramagnetic state $\psi_0 = \prod_{j=1}^L |+\rangle_j$ with $|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$. We are able to precisely represent ψ_0 using our network ansatz, such that the relative energy error is on the order of 10^{-11} .

The top row in Fig. 3 shows the overlap error with respect to the exact wavefunction as a function of

time, as well as the error resulting from a plain midpoint integration. Each time step was optimized using the same learning rate and number of iterations, but a more careful optimization could further lower the error. The bottom row of Fig. 3 shows the evolution of the transverse magnetization, based on our CNN ansatz and the numerically exact curve as reference. In agreement with Ref. [9], we find quenches to the critical point to be the hardest to optimize.

5 Conclusion

We have demonstrated that established methods for neural network optimization can be employed to describe the real time evolution of quantum wavefunctions. One additional advantage of this method, different from SR, is that it allows for the neural network architecture to be changed on the fly, which could be useful for capturing the growing complexity of the system as time progresses.

Taking full advantage of advanced machine learning techniques could further improve the results presented here, e.g., using deeper network architectures with batch normalization and residual blocks [28]. In this work, the network parameters were optimised without any restrictions, but imposing symmetries or a certain structure, especially to the CNN filters, could accelerate and improve the optimization.

Acknowledgments We thank Jan Budich and Lorenzo Pastori for useful discussions, and the Munich Center for Quantum Science and Technology for support.

Note: During the preparation of this manuscript we became aware of related work by Markus Schmitt and Markus Heyl [25], which appeared simultaneously on arXiv.

References

- [1] V. Alba and P. Calabrese. Entanglement and thermodynamics after a quantum quench in integrable systems. *PNAS*, 114:7947–7951, 2017. DOI: 10.1073/pnas.1703516114.
- [2] H. W. J. Blöte and Y. Deng. Cluster Monte Carlo simulation of the transverse Ising model. *Phys. Rev. E*, 66:066110, 2002. DOI: 10.1103/PhysRevE.66.066110.
- [3] A. Borin and D. A. Abanin. Approximating power of machine-learning ansatz for quantum many-body states. *Phys. Rev. B*, 101, 2020. DOI: 10.1103/PhysRevB.101.195141.
- [4] P. Calabrese and J. Cardy. Evolution of entanglement entropy in one-dimensional systems. *J. Stat. Mech.: Theory Exp.*, 2005:P04010, 2005. DOI: 10.1088/1742-5468/2005/04/p04010.
- [5] G. Carleo and M. Troyer. Solving the quantum many-body problem with artificial neural networks. *Science*, 355:602–606, 2017. DOI: 10.1126/science.aag2302.
- [6] G. Carleo, F. Becca, M. Schiró, and M. Fabrizio. Localization and glassy dynamics of many-body quantum systems. *Sci. Rep.*, 2:243, 2012. DOI: 10.1038/srep00243.
- [7] G. Carleo, F. Becca, L. Sanchez-Palencia, S. Sorella, and M. Fabrizio. Light-cone effect and supersonic correlations in one- and two-dimensional bosonic superfluids. *Phys. Rev. A*, 89:031602, 2014. DOI: 10.1103/PhysRevA.89.031602.
- [8] S. R. Clark. Unifying neural-network quantum states and correlator product states via tensor networks. *J. Phys. A Math. Theor.*, 51:135301, 2018. DOI: 10.1088/1751-8121/aaaaf2.
- [9] S. Czischek, M. Gärttner, and T. Gasenzer. Quenches near Ising quantum criticality as a challenge for artificial neural networks. *Phys. Rev. B*, 98:024311, 2018. DOI: 10.1103/PhysRevB.98.024311.
- [10] A. J. Daley, C. Kollath, U. Schollwöck, and G. Vidal. Time-dependent density-matrix renormalization-group using adaptive effective Hilbert spaces. *J. Stat. Mech. Theory Exp.*, 2004:P04005, 2004. DOI: 10.1088/1742-5468/2004/04/p04005.
- [11] D. Deng, X. Li, and S. Das Sarma. Quantum entanglement in neural network states. *Phys. Rev. X*, 7:021021, 2017. DOI: 10.1103/PhysRevX.7.021021.
- [12] X. Gao and L.-M. Duan. Efficient representation of quantum many-body states with deep neural networks. *Nat. Commun.*, 8:662, 2017. DOI: 10.1038/s41467-017-00705-2.
- [13] I. Glasser, N. Pancotti, M. August, I. D. Rodriguez, and J. I. Cirac. Neural-network quantum states, string-bond states, and chiral topological states. *Phys. Rev. X*, 8:011006, 2018. DOI: 10.1103/PhysRevX.8.011006.
- [14] E. Hairer, C. Lubich, and G. Wanner. *Geometric Numerical Integration. Structure-Preserving Algorithms for Ordinary Differential Equations*. Springer-Verlag Berlin Heidelberg, 2006. DOI: 10.1007/3-540-30666-8.
- [15] A. Hirose. *Complex-Valued Neural Networks*. Springer-Verlag Berlin Heidelberg, 2012. DOI: 10.1007/978-3-642-27632-3.
- [16] M. Hochbruck and C. Lubich. Error analysis of Krylov methods in a nutshell. *SIAM J. Sci. Comput.*, 19(2):695–701, 1998. DOI: 10.1137/S1064827595290450.
- [17] R. Kaubruegger, L. Pastori, and J. C. Budich. Chiral topological phases from artificial neural networks. *Phys. Rev. B*, 97:195136, 2018. DOI: 10.1103/PhysRevB.97.195136.
- [18] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *3rd International Conference for Learning Representations, San Diego*, 2015.
- [19] Y. Levine, O. Sharir, N. Cohen, and A. Shashua. Quantum entanglement in deep learning architectures. *Phys. Rev. Lett.*, 122:065301, 2019. DOI: 10.1103/PhysRevLett.122.065301.
- [20] J. Liesen and P. Tichý. Convergence analysis of Krylov subspace methods. *GAMM-Mitteilungen*, 27:153–173, 2004. DOI: 10.1002/gamm.201490008.
- [21] Y. Nomura, A. S. Darmawan, Y. Yamaji, and M. Imada. Restricted Boltzmann machine learning for solving strongly correlated quantum systems. *Phys. Rev. B*, 96:205152, 2017. DOI: 10.1103/PhysRevB.96.205152.
- [22] L. Pastori, R. Kaubruegger, and J. C. Budich. Generalized transfer matrix states from artificial neural networks. *Phys. Rev. B*, 99:165123, 2019. DOI: 10.1103/PhysRevB.99.165123.
- [23] D. Poulin, A. Qarry, R. Somma, and F. Verstraete. Quantum simulation of time-dependent Hamiltonians and the convenient illusion of Hilbert space. *Phys. Rev. Lett.*, 106:170501, 2011. DOI: 10.1103/PhysRevLett.106.170501.
- [24] M. Schmitt and M. Heyl. Quantum dynamics in transverse-field Ising models from classical networks. *SciPost Phys.*, 4:013, 2018. DOI: 10.21468/SciPostPhys.4.2.013.
- [25] M. Schmitt and M. Heyl. Quantum many-body dynamics in two dimensions with artificial neural networks. *Phys. Rev. Lett.*, 125:100503, 2020. DOI: 10.1103/PhysRevLett.125.100503.
- [26] U. Schollwöck. The density-matrix renormalization group. *Rev. Mod. Phys.*, 77:259–315, 2005. DOI: 10.1103/RevModPhys.77.259.
- [27] U. Schollwöck. The density-matrix renormalization group in the age of matrix product

- states. *Ann. Phys.*, 326:96–192, 2011. DOI: 10.1016/j.aop.2010.09.012.
- [28] A. Shrestha and A. Mahmood. Review of deep learning algorithms and architectures. *IEEE Access*, 7:53040–53065, 2019. DOI: 10.1109/ACCESS.2019.2912200.
- [29] S. Sorella. Generalized Lanczos algorithm for variational quantum Monte Carlo. *Phys. Rev. B*, 64:024512, 2001. DOI: 10.1103/PhysRevB.64.024512.
- [30] S. Suzuki, J. Inoue, and B. K. Chakrabarti. *Quantum Ising Phases and Transitions in Transverse Ising Models*. Springer, Berlin, Heidelberg, 2013. ISBN 978-3-642-33039-1. DOI: 10.1007/978-3-642-33039-1.
- [31] C. Trabelsi, O. Bilaniuk, Y. Zhang, D. Serdyuk, S. Subramanian, J. F. Santos, S. Mehri, N. Ros-tamzadeh, Y. Bengio, and C. J. Pal. Deep complex networks. In *International Conference on Learning Representations*, 2018. URL <https://openreview.net/forum?id=H1T2hmZAb>.
- [32] V. N. Vapnik. An overview of statistical learning theory. *IEEE Trans. Neural Netw.*, 10:988–999, 1999. DOI: 10.1109/72.788640.
- [33] G. Vidal. Efficient simulation of one-dimensional quantum many-body systems. *Phys. Rev. Lett.*, 93:040502, 2004. DOI: 10.1103/PhysRevLett.93.040502.
- [34] S. R. White and A. E. Feiguin. Real-time evolution using the density matrix renormalization group. *Phys. Rev. Lett.*, 93:076401, 2004. DOI: 10.1103/PhysRevLett.93.076401.

A Wirtinger formalism

We employ the Wirtinger formalism to compute gradients of the cost function with respect to complex-valued parameters. This formalism is not widely employed in the context of artificial neural networks, but see e.g. [15, 31] and references therein. The *Wirtinger* or *Dolbeault* operators are defined as

$$\frac{\partial}{\partial z} := \frac{1}{2} \left(\frac{\partial}{\partial x} - i \frac{\partial}{\partial y} \right), \quad \frac{\partial}{\partial z^*} := \frac{1}{2} \left(\frac{\partial}{\partial x} + i \frac{\partial}{\partial y} \right) \quad (20)$$

with $z = x + iy$, $x, y \in \mathbb{R}$. The operators act on (real-) differentiable (identifying $\mathbb{C} \simeq \mathbb{R}^2$) functions $f : U \rightarrow \mathbb{C}$ (with $U \subset \mathbb{C}$ some open subset of \mathbb{C}), which need not be holomorphic. However, in case f is indeed holomorphic, the Cauchy-Riemann equations imply that the Wirtinger derivative $\partial/\partial z$ is equal to the complex derivative of f , whereas the conjugated Wirtinger derivative vanishes:

$$\frac{\partial f(z)}{\partial z} = f'(z), \quad \frac{\partial f(z)}{\partial z^*} = 0 \quad \forall z \in U, \quad f \text{ holomorphic} \quad (21)$$

Complex conjugating the second identity leads to $\partial f^*(z)/\partial z = 0$ in this case.

Note that for real-valued functions $f : U \rightarrow \mathbb{R}$ (like cost functions considered below), the partial derivatives with respect to x and y can be obtained from the Wirtinger derivative via

$$\frac{\partial f}{\partial x} = 2 \operatorname{Re} \left(\frac{\partial f}{\partial z} \right), \quad \frac{\partial f}{\partial y} = -2 \operatorname{Im} \left(\frac{\partial f}{\partial z} \right). \quad (22)$$

The following chain rule can be verified by a straightforward calculation:

$$\frac{\partial}{\partial z} (g \circ f) = \left(\frac{\partial g}{\partial w} \circ f \right) \cdot \frac{\partial f}{\partial z} + \left(\frac{\partial g}{\partial w^*} \circ f \right) \cdot \frac{\partial f^*}{\partial z}. \quad (23)$$

The formalism generalizes naturally to functions of several variables; for $z \in \mathbb{C}^n$, we write

$$\nabla_z^W = \left(\frac{\partial}{\partial z_1}, \dots, \frac{\partial}{\partial z_n} \right)^T \quad (24)$$

for the Wirtinger nabla operator.

Let $f_\theta : U \subset \mathbb{C}^n \rightarrow \mathbb{C}^m$ denote the map defined by an artificial neural network with complex-valued parameters $\theta \in \mathbb{C}^p$, input dimension n and output dimension m . In abstract terms, our goal is to minimize a cost function with respect to the network parameters via some version of gradient descent (or more precisely, minimizing the expected prediction error for data not used during training [32]):

$$\min_{\theta} C(\theta), \quad C(\theta) = \sum_{j=1}^N c(f_\theta(x^{(j)}), y^{(j)}) \quad (25)$$

with $c : \mathbb{C}^m \times \mathbb{C}^k \rightarrow \mathbb{R}$ depending on the network output and training labels $y^{(j)} \in \mathbb{C}^k$. Here $(x^{(j)}, y^{(j)})_{j=1, \dots, N}$ a given sequence of training samples. Since C is real-valued, it cannot be holomorphic (except for the trivial case of a constant function), which motivates the use of Wirtinger derivatives in the first place.

Nevertheless, we assume that f_θ is holomorphic as function of the parameters θ . Applying the chain rule (23) and using that $\nabla_{\theta}^W f_{\theta}^* = 0$ leads to

$$\nabla_{\theta}^W C(\theta) = \sum_{j=1}^N \sum_{k=1}^m \frac{\partial c(f_\theta(x^{(j)}), y^{(j)})}{\partial f_{\theta,k}(x^{(j)})} \nabla_{\theta}^W f_{\theta,k}(x^{(j)}) \quad (26)$$

with the subscript k denoting the k -th output component of f_θ . From Eq. (26), one obtains the gradient with respect to the real and imaginary parts of θ via Eq. (22).

As basic example, the Wirtinger derivative of the quadratic cost (for $a, y \in \mathbb{C}^m$)

$$c(a, y) = \|a - y\|^2 = \sum_{j=1}^m |a_j - y_j|^2 \quad (27)$$

reads

$$\frac{\partial c(a, y)}{\partial a_j} = (a_j - y_j)^*. \quad (28)$$

Combined with the chain rule, the gradients of the cost function in Eq. (11) thus read

$$\frac{\partial C}{\partial \theta_\ell} = \frac{\partial C}{\partial (A\psi)} \frac{\partial (A\psi)}{\partial \psi} \frac{\partial \psi}{\partial \theta_\ell} = \left\langle A\psi - b \left| A \frac{\partial \psi}{\partial \theta_\ell} \right. \right\rangle \quad (29)$$

B Error analysis

We distinguish between three different wave functions: the exact one, $\psi(t)$; the one obtained by the exact midpoint time-evolution, $\psi_\Delta(t)$; and the one represented by the network, $\psi_N(t)$. We want to find the error of the network with respect to the exact state:

$$\varepsilon(t) = \psi(t) - \psi_N(t) \quad (30)$$

This can be split into the error due to the the midpoint method and the error due to the network optimization:

$$\varepsilon(t) = \varepsilon_\Delta(t) + \varepsilon_N(t) \quad (31)$$

with

$$\varepsilon_\Delta(t) = \psi(t) - \psi_\Delta(t) \quad (32)$$

and

$$\varepsilon_N(t) = \psi_\Delta(t) - \psi_N(t). \quad (33)$$

Using the triangle inequality we can set an upper bound to the absolute error:

$$|\varepsilon(t)| \leq |\varepsilon_\Delta(t)| + |\varepsilon_N(t)| \quad (34)$$

As implied by Eq. (9), in order to obtain the state at the next time step using the midpoint rule one must solve the following linear matrix equation:

$$A\psi_\Delta(t_{n+1}) - B\psi_\Delta(t_n) = 0 \quad (35)$$

where

$$A = I + \frac{i\Delta t}{2}H \quad (36)$$

and

$$B = I - \frac{i\Delta t}{2}H. \quad (37)$$

As a second-order Runge-Kutta method, its global error is of the order of $O(\Delta t^2)$. However, instead of solving this exactly, our method optimizes $\psi_N(t_{n+1})$ to minimize the above. That means we are actually solving the equation

$$A\psi_N(t_{n+1}) - B\psi_N(t_n) = r^{(n+1)}, \quad (38)$$

where $r^{(n+1)}$ is the residual obtained at the end of the optimization in time-step $n + 1$. We are interested in finding an expression for $\varepsilon_N(t_n)$. Using its definition into Eq. (38) we obtain

$$A(\psi_\Delta(t_{n+1}) - \varepsilon_N(t_{n+1})) - B(\psi_\Delta(t_n) - \varepsilon_N(t_n)) = r^{(n+1)}, \quad (39)$$

and since the exact midpoint rule must be 0 (see Eq. (35)) this simplifies to

$$\varepsilon_N(t_{n+1}) = A^{-1} \left(B\varepsilon_N(t_n) - r^{(n+1)} \right). \quad (40)$$

Similarly as what we did for the midpoint error, we now have a recursive relation for the network error for the next time step. Then, by proof of induction one can show that

$$\varepsilon_N(t_n) = -B^{-1} \sum_{m=1}^n A^{-m} B^m r^{(n+1-m)}, \quad (41)$$

for all $n > 0$. Note that the application of $A^{-1}B$ corresponds to one time step of the midpoint time evolution. This means that the residual from each time step is added to the error and time-evolved unitarily, so the residual will not dramatically increase the error as an artifact of the chosen integration method.

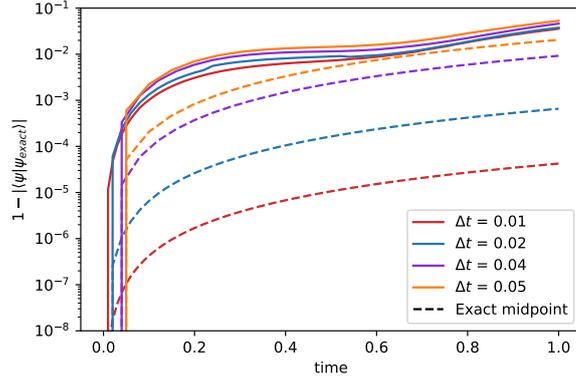


Figure 4: Overlap error achieved with different time steps for the same system under the same optimization hyperparameters (RBM with 60 hidden units, same learning rate and same number of optimization steps).

C Effect of time step size, and comparison with the forward Euler method

Fig. 1b in the main text directly compares SR with the midpoint optimization using the same number of time steps. However, the implicit midpoint integration method is a second-order method, which suggests the possibility of achieving a similar final error using a larger time step size for the real time evolution. Here we consider the effects of varying the time step size. Furthermore, we compare the results with an analogous neural network optimization based on the forward Euler method.

We apply the midpoint optimization to simulate a quench from $h = 1.5$ to $h = 0.75$ for a one-dimensional system with 15 lattice sites. As shown in Fig. 4 the overlap error resulting from the optimization shows very little variation as one decreases Δt , different from an exact midpoint integration. These results further strengthen our intuition that most of the error originates from the expressiveness of the architecture, and not the integration method. The variation that is seen can be explained by the fact that the same learning rate and number of optimization parameters are being used in each simulation. For a larger Δt one can expect a larger update in the network parameters, and a longer optimization may be necessary. In practice, the small improvement may not be worth the computational expense of decreasing Δt nor increasing the number of optimization steps.

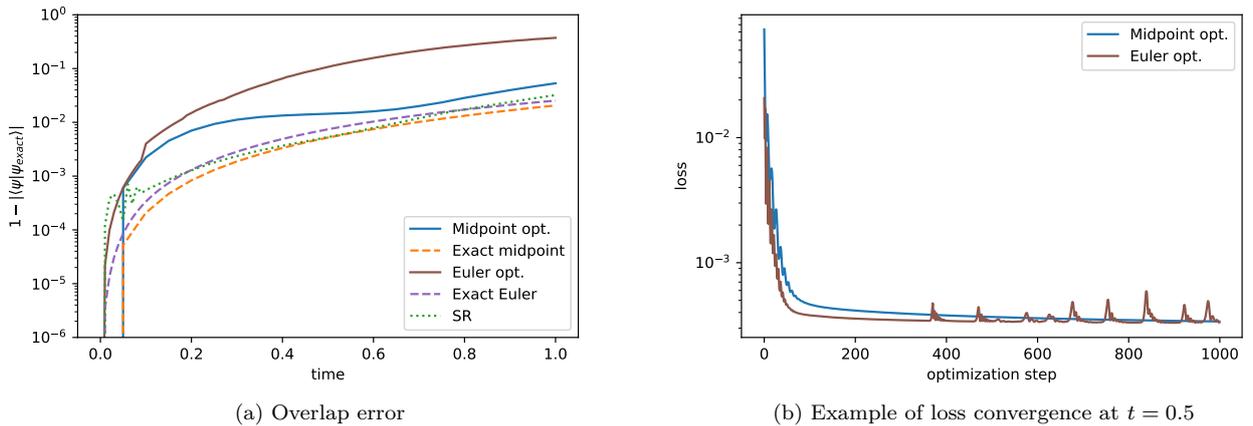


Figure 5: Comparison of different integration methods and time steps: SR and the Euler method with $\Delta t = 0.01$ and the midpoint method with $\Delta t = 0.05$. In (b) the loss is normalized with respect to the number of samples and time steps.

Next we compare these results with those obtained using an Euler step for the optimization. In a similar fashion to Eq. (10) we choose as cost function

$$C(\theta_{n+1}) = \sum_{j=1}^N \left| (\psi[\theta_{n+1}] - i\Delta t H \psi[\theta_n]) (\sigma^{(j)}) \right|^2. \quad (42)$$

Fig. 5a shows that the results obtained are worse than those obtained with the implicit midpoint rule, despite the similar errors of the exact Euler and midpoint integrations for the chosen Δt sizes. Taking a closer look at the optimization landscape shows that the loss exhibits a smoother convergence with the midpoint optimization, as shown in Fig. 5b. Implementing an early stop mechanism may be beneficial for the Euler optimization.

D Geometrical perspective

From a geometrical perspective, the architecture of the neural network defines the manifold \mathcal{M} to which the time-evolution is restrained. As shown in Fig. 6, in SR the parameters are updated by projecting the true derivative at the current time onto \mathcal{M} . Similarly, the minimization of Eq. (7), implicitly projects the next timestep obtained from the integration flow Φ , as the point of minimum distance in the manifold is its projection.

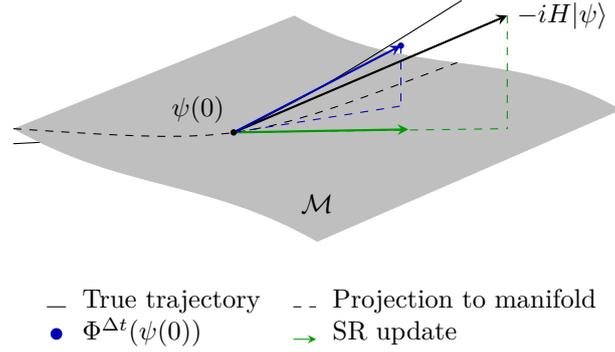


Figure 6: Geometric interpretation of the update achieved by SR and a gradient descent based method from a numerical integration flow $\Phi^{\Delta t}$

E Details of stochastic reconfiguration calculation

Fig. 1b in the main text shows a comparison between our method and SR. For our simulations, we determine the cut-off of the pseudo-inverse as follows: we perform the time evolution several times decreasing the cut-off by one order of magnitude at each run, from 0.1 to 10^{-11} , and chose the one that resulted in the smallest error at $t = 1$. It is worth noting, however, that the best cut-off at some time t_1 may not be the best at a later time t_2 . Fig. 7a illustrates this point for an Ising chain with 6 lattice sites.

We also tried several Krylov subspace methods for solving Eq. (2), but found their performance to be worse than using the pseudo-inverse with optimal threshold, as shown in Fig. 7b.

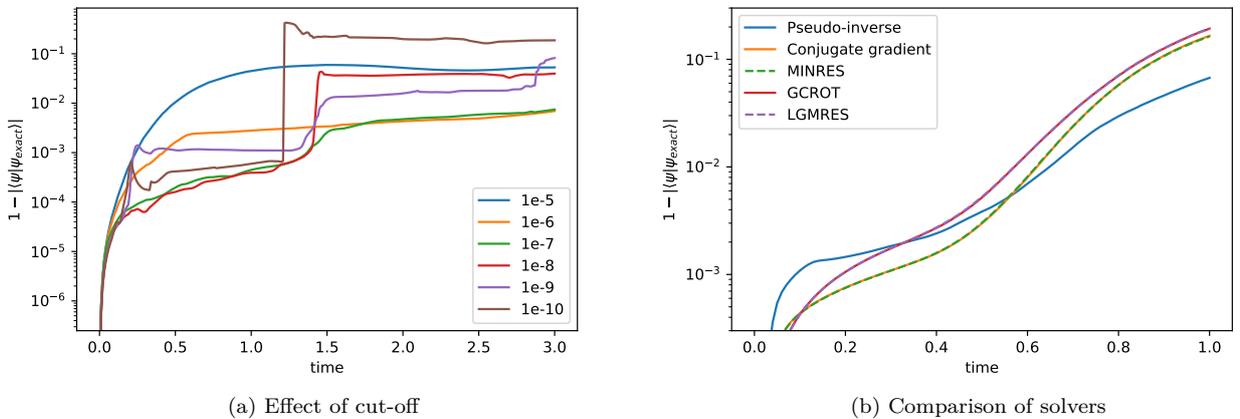
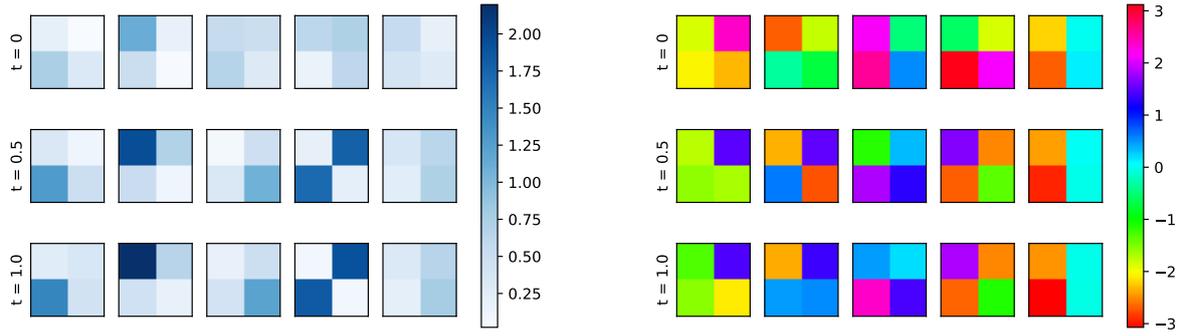


Figure 7: Comparison in error using SR with (a) different cut-offs for the pseudo-inverse and (b) different iterative solvers, as well as using a pseudo-inverse with cut-off at 10^{-10} .



(a) Amplitudes of filter weights

(b) Arguments of filter weights

Figure 8: (a) Amplitudes and (b) arguments of the complex weights for each filter of the CNN in Fig. 2, for the $h = 2h_c$ simulation shown in Fig. 3. The rows corresponds to different time points.

F Filter weights visualization

As illustration, Fig. 8 shows the five complex-valued convolution layer filter weights for the time evolution of the 2D system at three time points, for the quench to $h = 2h_c$. It appears that the amplitudes vary rather slowly as compared to the complex arguments (phases).