Neural Projected Quantum Dynamics: a systematic study

Luca Gravina[1,2], Vincenzo Savona[1,2], and Filippo Vicentini[3,4]

[1]Institute of Physics, Ecole Polytechnique Fédérale de Lausanne (EPFL), CH-1015 Lausanne, Switzerland
[2]Center for Quantum Science and Engineering, Ecole Polytechnique Fédérale de Lausanne (EPFL), CH-1015 Lausanne, Switzerland
[3]CPHT, CNRS, Ecole Polytechnique, Institut Polytechnique de Paris, 91120 Palaiseau, France.
[4]Collège de France, Université PSL, 11 place Marcelin Berthelot, 75005 Paris, France

arXiv:2410.10720v3 [quant-ph] 15 Jul 2025

We investigate the challenge of classical simulation of unitary quantum dynamics with variational Monte Carlo approaches, addressing the instabilities and high computational demands of existing methods. By systematically analyzing the convergence of stochastic infidelity optimizations, examining the variance properties of key stochastic estimators, and evaluating the error scaling of multiple dynamical discretization schemes, we provide a thorough formalization and significant improvements to the projected time-dependent Variational Monte Carlo (p-tVMC) method. We benchmark our approach on a two-dimensional Ising quench, achieving state-of-the-art performance. This work establishes p-tVMC as a powerful framework for simulating the dynamics of large-scale two-dimensional quantum systems, surpassing alternative VMC strategies on the investigated benchmark problems.

Contents

1 Introduction

Simulating the dynamics of a quantum system is essential for addressing various problems in material science, quantum chemistry, quantum optimal control, and for answering fundamental questions in quantum information [1, 2, 3]. However, the exponential growth of the Hilbert space makes this one of the most significant challenges in computational quantum physics, with only few tools available to accurately simulate the dynamics of large, complex systems.

To manage the exponential growth of the Hilbert space, quantum states can be encoded using efficient compression schemes [4]. While tensor network methods [5, 6, 7, 8, 9, 10], particularly Matrix Product States [11, 12, 13, 14], excel in simulating large one-dimensional models with short-range interactions, extending them to higher dimensions is problematic. Such extensions, either rely on uncontrolled approximations [15, 16], or incur in an exponential costs when encoding area-law entangled states [17], making them poorly suited for investigating strongly correlated, higher-dimensional systems or unstructured lattices, such as those encountered in chemistry or quantum algorithms [18, 19, 20, 21, 22].

Recently, Neural Quantum States (NQS) have garnered increasing attention as a non-linear variational encoding of the wave-function capable, in principle, of describing arbitrarily entangled states, both pure [23, 24, 25, 26, 27] and mixed [28, 29, 30, 31, 32, 33]. This approach compresses the exponentially large wave-function into a polynomial set of parameters, with no restrictions on the geometry of the underlying system. The added flexibility, however, comes at a cost: unlike matrix product states whose bond dimension can be adaptively tuned via deterministic algorithms, neural network optimizations are inherently stochastic, making it hard to establish precise error bounds.

Despite the limitations of neural networks not being fully understood [34, 35], recent studies have demonstrated that NQS can be reliably optimized to represent the ground state of computationally challenging, non-stoquastic, fermionic, or frustrated Hamiltonians, arising across various domains of quantum physics [36, 37, 38, 39, 40, 41, 42, 43, 44, 45]. However, for the more complex task of simulating quantum dynamics, NQS have yet to show significant advantages over existing methods.

Neural Quantum Dynamics. There are two families of variational algorithms for approximating the direct integration of the Schrödinger equation using variational ansatze: time-dependent Variational Monte Carlo (tVMC) [46] and projected tVMC (p-tVMC), formalized in Ref. [47]. The former, tVMC, linearizes both the unitary evolution and the variational ansatz, casting the Schrödinger equation into an explicit algebraic-differential equation for the variational parameters [46, 48]. The latter, p-tVMC, relies on an implicit optimization problem to compute the parameters of the wave function at each timestep, using low-order truncations of the unitary evolution such as Taylor or Trotter expansions.

Of the two methods, tVMC is regarded as the least computationally expensive, as it avoids the need to solve a nonlinear optimization problem at every step. It has been successfully applied to simulate sudden quenches in large spin [49, 50, 23, 51, 52] and Rydberg [53] lattices, quantum dots [54], as well as finite temperature [55, 56] and out of equilibrium [32, 57, 58] systems. However, while stable for (log-)linear variational ansatze such as Jastrow [53] or Tensor Networks, the stiffness of the tVMC equation [59] appears to increase with the nonlinearity of the ansatz, making integration particularly hard for deep networks. Contributing to this stiffness is the presence of a systematic statistical bias in the evaluation of the dynamical equation itself, which would be exponentially costly to correct [47]. Although the effect of this noise can be partially regularised away [49], this regularization procedure introduces additional bias that is difficult to quantify. As of today, the numerous numerical issues inherent to tVMC make its practical application to non-trivial problems difficult, with the estimation of the actual error committed by the method being unreliable at best.

Projected Neural Quantum Dynamics and open challenges. The projected time-dependent Variational Monte Carlo method offers a viable, albeit more computationally intensive, alternative by decoupling the discretization of the physical dynamics from the nonlinear optimization of the variational ansatz, thereby simplifying the analysis of each component. So far, the discretization problem has been tackled using established schemes such as Runge-Kutta [60] or Trotter [47, 61, 62, 63]. These methods do not fully leverage the specific properties of VMC approaches and, as a result, the existing body of work [60, 47, 64] has been limited to second-order accuracy in time, struggling to provide general, scalable solutions. Similarly, the nonlinear optimization problem has mainly been addressed using first-order gradient descent techniques, neglecting the benefits offered by second-order optimization strategies.

In this manuscript, we investigate both aspects of p-tVMC — discretization and optimization — independently, addressing the shortcomings detailed above with the goal of enhancing accuracy, reducing computational costs, and improving stability and usability.

Specifically, in Section 2 we introduce a new family of discretization schemes tailored for p-tVMC, achieving higher accuracy for equivalent computational costs. In Section 3 we conduct an in-depth analysis of the nonlinear optimization problem of infidelity minimization, identifying the most effective stochastic estimator and introducing a new adaptive optimization scheme that performs as well as manually tuned hyperparameters, eliminating the need for manual adjustment. Finally, in Section 4 we benchmark several of our methods against a challenging computational problem: a quench across the critical point of the two-dimensional transverse field Ising model.

2 Integration schemes

Consider the generic evolution equation

$$|\psi_{t+\mathrm{d}t}\rangle = e^{\hat{\Lambda}\mathrm{d}t}\,|\psi_t\rangle\,, \tag{1}$$

where $\hat{\Lambda} = -i\hat{H}$ for some K-local time-independent Hamiltonian $\hat{H}$ constant in the the system size N. The fundamental challenge for the numerical integration of Eq. (1) lies in the dimensionality of the Hilbert space $\mathcal{H}$ scaling exponentially with system size, that is, $\dim(\mathcal{H}) \sim \exp(N)$. This makes it impossible to merely store in memory the state-vector $|\psi\rangle$, let alone numerically evaluate or apply the propagator.

Variational methods address the first problem by encoding an approximate representation of the state at time t into the time-dependent parameter vector $\theta_t \in \mathbb{R}^{N_p}$ of a variational ansatz, while relying on Monte Carlo integration for computing expectation values [23, 65, 66]. Within this framework, the McLachlan variational principle is used to recast Eq. (1) as the optimization problem [4]

$$\theta_{t+\mathrm{d}t} = \underset{\theta}{\operatorname{argmin}}\ \mathcal{L}\Big(|\psi_\theta\rangle\,, e^{\hat{\Lambda}\mathrm{d}t}\,|\psi_{\theta_t}\rangle\Big), \tag{2}$$

where $\mathcal{L}$ is a suitable loss function quantifying the discrepancy between two quantum states. Various choices for $\mathcal{L}$ are possible, the one adopted throughout this work is presented and discussed in Section 3.

TVMC and p-tVMC confront Eq. (2) differently. The former, tVMC, linearizes both the unitary evolution operator and the ansatz, reducing Eq. (2) to an explicit first-order non-linear differential equation in the parameters [23, 48]. In contrast, p-tVMC, relies on higher-order discretizations of the evolution operator to efficiently solve the optimization problem in Eq. (2) at each timestep.

In Section 2.1, we present a general formulation of p-tVMC and identify a set of fundamental requirements that discretization schemes for p-tVMC should satisfy. From this perspective, we revisit the well-established Trotter and Runge-Kutta methods in Sections 2.2 and 2.3. In Sections 2.4 to 2.6, we introduce a new family of discretization schemes tailored to the specific structure of p-tVMC, achieving higher-order accuracy in $\mathrm{d}t$ with reduced computational complexity.

2.1 Generic formulation of p-tVMC schemes

The minimization problem in Eq. (2) aims at finding the parameters transformation that best approximates the infinitesimal evolution under $\hat{\Lambda}$, i.e.

$$\big|\psi_{\theta_{t+\mathrm{d}t}}\big\rangle = e^{\hat{\Lambda}\mathrm{d}t}\,|\psi_{\theta_t}\rangle\,. \tag{3}$$

The most generic problem of this form is that of finding the evolved parameters $\theta_{t+\mathrm{d}t}$ such that

$$\hat{V}\,\big|\psi_{\theta_{t+\mathrm{d}t}}\big\rangle = \hat{U}\,|\psi_{\theta_t}\rangle\,, \tag{4}$$

with arbitrary $\hat{U}$ and $\hat{V}$, not necessarily unitary. Equation (4) reduces to Eq. (3) for $\hat{V} = \mathbb{1}$ and $\hat{U} = \exp(\hat{\Lambda}\mathrm{d}t)$. Note that while the parameters $\theta_{t+\mathrm{d}t}$ satisfying Eq. (4) also satisfy

$$\big|\psi_{\theta_{t+\mathrm{d}t}}\big\rangle = \hat{V}^{-1}\hat{U}\,|\psi_{\theta_t}\rangle\,, \tag{5}$$

the associated optimization problems

$$\begin{aligned}\theta_{t+\mathrm{d}t} &= \underset{\theta}{\operatorname{argmin}}\ \mathcal{L}\Big(\hat{V}\,|\psi_\theta\rangle\,, \hat{U}\,|\psi_{\theta_t}\rangle\Big)\\ &= \underset{\theta}{\operatorname{argmin}}\ \mathcal{L}\Big(|\psi_\theta\rangle\,, \hat{V}^{-1}\hat{U}\,|\psi_{\theta_t}\rangle\Big)\end{aligned} \tag{6}$$

share the same minimum but have different optimization landscapes. It is natural, therefore, to consider expansions of the infinitesimal time-independent propagator in the form of a product series

$$e^{\hat{\Lambda}\mathrm{d}t} = \prod_{k=1}^{s} \hat{V}_k^{-1}\hat{U}_k + \mathcal{O}\Big(\mathrm{d}t^{o(s)+1}\Big), \tag{7}$$

where the number of elements s in the series is related to the order of the expansion $o = o(s)$. This decomposition is convenient because:

- There are no summations. Therefore, the terms $\hat{V}_k^{-1}\hat{U}_k$ in the series can be applied sequentially to a state, without the need to store intermediate states and recombine them.
- The single step of p-tVMC can efficiently embed an operator inverse at every sub-step.

By utilizing this discretization, the parameters after a single timestep $\mathrm{d}t$ are found by solving a sequence of s subsequent optimization problems, with the output of each substep serving as the input for the next. Specifically, setting $\theta_t \equiv \theta^{(0)}$, and $\theta_{t+\mathrm{d}t} \equiv \theta^{(s)}$, we can decompose Eq. (2) as

$$\theta^{(k)} = \underset{\theta}{\operatorname{argmin}}\ \mathcal{L}\Big(\hat{V}_k\,|\psi_\theta\rangle\,, \hat{U}_k\,|\psi_{\theta^{(k-1)}}\rangle\Big), \tag{8}$$

with $0 < k < s$. This optimization does not directly compress the variational state $|\psi_\theta\rangle$ onto the target state $|\phi\rangle$. Instead, it matches two versions of these states transformed by the linear operators $\hat{V}_k$ and $\hat{U}_k$. A careful tuning of $\hat{V}_k$ and $\hat{U}_k$ can be seen as a form of preconditioning, initializing the optimization closer to the solution and potentially accelerating convergence.

Equation (8) can be solved efficiently with variational Monte Carlo methods provided all operators $\{\hat{V}_k, \hat{U}_k\}$ are log-sparse (or K-local). In what follows we explore proficient choices for the set $\{\hat{V}_k, \hat{U}_k\}$. Two conditions guide our search for an optimal expansion scheme:

(i) Equation (7) should match Eq. (2) to a specified order in $\mathrm{d}t$, denoted as o, ensuring accurate time evolution up to this order.

Name	Sec.	substeps	N_c	unitary
Trotter	2.2	$5^{\frac{o}{2}-1}\mathcal{O}(2N)$	$\mathcal{O}(2)$	✓
Taylor	2.3	1	$\mathcal{O}(N^o)$	✗
LPE-o	2.4	o	$\mathcal{O}(N)$	✗
PPE-o	2.5	$\frac{o}{2}$	$\mathcal{O}(2N)$	✓
S-LPE-o	2.6	†	$\mathcal{O}(N)$	✗
S-PPE-o	2.6	†	$\mathcal{O}(2N)$	$o = 2$

Table 1: Discretization schemes compatible with p-tVMC. We denote by s the number of substeps (optimizations), o the order of the integration scheme, and N_c the connected elements (complexity) of the operators entering the optimization problem. We remark that the PPE-o scheme has only even orders o and is the only scheme that is exactly unitary for any choice of the truncation order o. The S-PPE scheme is exactly unitary only for $o = 2$. †: We were not able to derive an analytic expression connecting the order of the diagonally-exact split schemes. Semi-analytically we could determine that for S-LPE-o the first few substeps and orders are $(s, o) = (1, 1), (2, 2), (4, 3)$ and for S-PPE-o they are $(s, o) = (1, 2), (2, 3), (3, 4)$.

(ii) The computational complexity of solving Eq. (8), which is proportional to sN_c with N_c the number of connected elements of $\{\hat{V}_k, \hat{U}_k\}$, must scale at most polynomially in the number of particles N and in the order o of the method.

Table 1 summarizes our analysis, including both established discretization schemes, and the four new ones introduced in this manuscript.

2.2 Trotter decomposition

A prototypical product series decomposition of a unitary operator is the Suzuki–Trotter decomposition [67]. In this approach, $\hat{\Lambda}$ is expressed as a sum of local terms, and the exponential of the sum is approximated as a product of the exponentials of the individual terms. The decomposition of $\hat{\Lambda}$ is not unique and can be tailored to the specifics of the problem to maximize computational efficiency [68].

While Suzuki–Trotter decompositions can be extended to arbitrary order, in practice, their use in NQS is typically limited to second order in $\mathrm{d}t$, as seen in Refs. [47] and [61]. The key advantage of this approach is that it approximates the operator's action in a manner where state changes are highly localized, which simplifies the individual optimization problems in Eq. (8) and tends to improve their convergence.

For all their benefits, Suzuki–Trotter decompositions face two main limitations: the truncation to second order in $\mathrm{d}t$ and the scaling of the number of optimizations with the system size, both of which hinder computational efficiency in large-scale applications.

2.3 Taylor decomposition

Another relevant decomposition to consider is the order-o Taylor approximation of the propagator. Its expression

$$e^{\hat{\Lambda}\mathrm{d}t} = \sum_{k=0}^{o} \frac{(\hat{\Lambda}\mathrm{d}t)^k}{k!} + \mathcal{O}\big(\mathrm{d}t^{o+1}\big), \tag{9}$$

can be viewed within the framework of Eq. (7) as a single optimization problem ($s = 1$) of the form in Eq. (8) with $\hat{V}_1 = \mathbb{1}$ and $\hat{U}_1 = \sum_{k=0}^{o}(\hat{\Lambda}\mathrm{d}t)^k/k!$. This approach satisfies condition (i) by matching the desired order of accuracy in $\mathrm{d}t$, but it fails to meet condition (ii) related to computational efficiency. Indeed, computing $\hat{U}_1 |\psi_\theta\rangle$ requires summing over all N_c connected elements of $\hat{U}_1$. As the order o increases, higher powers of $\hat{\Lambda}$ introduce a growing number of such connected elements, with $N_c \sim \mathcal{O}(N^o)$. This approach incurs a computational cost that scales exponentially in o, making it unviable at higher orders. Furthermore, this approach cannot reasonably be used in continuous space simulations, where the square of the Laplacian cannot be efficiently estimated.

2.4 Linear Product Expansion (LPE)

We now introduce a new scheme circumventing the limitations of the previous approaches. We consider the linear operator $\hat{T}_a \equiv \mathbb{1} + a\hat{\Lambda}\mathrm{d}t$ with $a \in \mathbb{C}$ and expand the evolution operator as a series of products of such terms,

$$e^{\hat{\Lambda}\mathrm{d}t} = \prod_{i=1}^{s} \hat{T}_{a_i} + \mathcal{O}(\mathrm{d}t^{s+1}). \tag{10}$$

The expansion is accurate to order $o(s) = s$. The complex-valued coefficients a_i are determined semi-analitically by matching both sides of Eq. (10) order by order up to $o(s)$. For the second-order scheme ($s = 2$), for example, we find $a_1 = (1 - i)/2$ and $a_2 = (1 + i)/2$. Further details on the scheme and on the computation of the coefficients are provided in Appendix A. Tabulated values for a_i can be found in Table 4. We call this method LPE-o for *Linear product expansion*, where o is the order of the method, related to the number of sub-steps s by $o(s) = s$. Each substep corresponds to an optimization problem with $\hat{V}_k = \mathbb{1}$, and $\hat{U}_k = \hat{T}_{a_k}$. The advantage of LPE over Taylor schemes (Section 2.3) is that the former defines an s-substep scheme of order s with a step-complexity that is linear in N. This greatly outperforms Runge-Kutta style expansions for this particular application, enabling scaling to arbitrary order in $\mathrm{d}t$, simultaneously satisfying conditions (i) and (ii).

It was noted in Ref. [54] that the coefficients a_i of this expansion are the complex roots of the order s Taylor polynomial. While this is a handy trick to compute them numerically, this approach is not

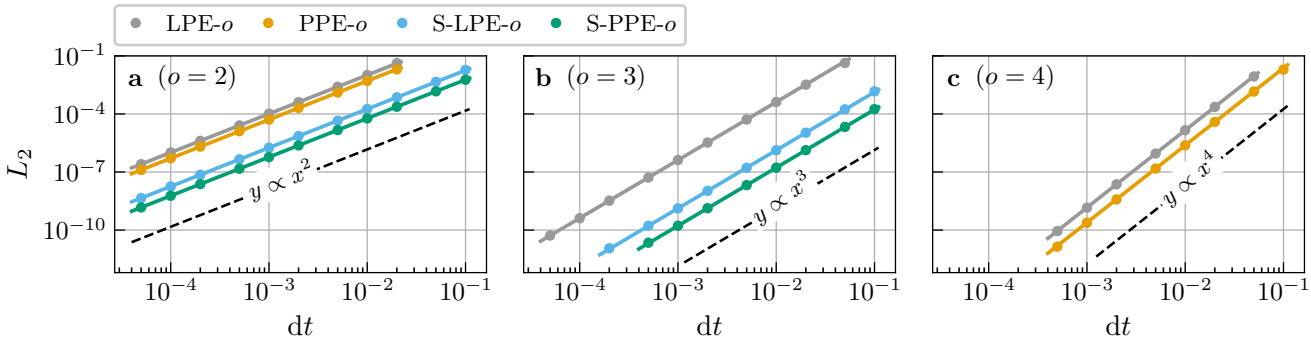

Figure 1: Global truncation error accumulated at time $t = 1$ for LPE and PPE integrators, and their split counterparts, of orders $o = 2$ (a), $o = 3$ (b), and $o = 4$ (c). The evolution is carried out under the Hamiltonian in Eq. (38) simulating on a 4×4 lattice the same quench dynamics investigated in Section 4.2. The accumulated error for an integrator of order o scales as $\mathrm{d}t^o$, with the local error scaling as $\mathrm{d}t^{o+1}$. Both split and non-split integrators are shown, demonstrating the power of the splitting in reducing the prefactor of the error.

general enough to represent the multi-operator expansions that we will analyze below in Sections 2.5 and 2.6.

2.5 Padé Product Expansion (PPE)

We now present schemes reaching order $2s$ with only s sub-steps of marginally-increased complexity. We consider the operator $\hat{P}_{b,a} \equiv \hat{T}_b^{-1}\hat{T}_a$ and expand the evolution operator as a series of products of such terms,

$$e^{\hat{\Lambda}\mathrm{d}t} = \prod_{i=1}^{s} \hat{P}_{b_i,a_i} + \mathcal{O}(\mathrm{d}t^{2s+1}). \tag{11}$$

The expansion is accurate to order $o(s) = 2s$. We call this method PPE-s for *Padé product expansion*, because the single term $\hat{P}_{b,a}$ corresponds to a (1,1) Padé approximation [69]. The scheme is explicitly constructed to take advantage of the structure of the optimization problems in Eq. (8), exploiting the presence of a matrix inverse in the expansion (7). While atypical for standard ODE integrators, as it would introduce an unjustified overhead, in our case this simply translates into optimizations where $\hat{V}_i = \hat{T}_{b_i}$, and $\hat{U}_i = \hat{T}_{a_i}$. The coefficients a_i and b_i are again obtained by matching both sides of Eq. (11) up to order o (see Appendix A). A remarkable property of PPE schemes is that they preserve the unitarity of the evolution map making them ideal candidates for the simulation of quantum circuits (proof in Appendix A.2.1). A comparison of PPE and LPE schemes of different orders is provided in Fig. 1 where we show the L_2-distance between the exact solution and the solution obtained from state-vector simulations of Eq. (7).

2.6 Diagonally-exact split schemes

Learning the parameter change connecting two states via state compression is challenging. Restricting the problem to scenarios where state changes are highly localized has proven effective in mitigating this issue, easing optimization and generally improving convergence [47]. This simplification, however, usually comes at the cost of an unfavourable scaling of the number of optimizations, typically scaling with N (c.f. Section 2.2).

We propose to reduce the complexity of the nonlinear optimizations by splitting $\hat{\Lambda}$ as $\hat{\Lambda} = \hat{X} + \hat{Z}$, where $\hat{Z}$ acts diagonally in the computational basis[1] while $\hat{X}$ is an off-diagonal matrix. The rationale will be to extract the diagonal operators which can be applied exactly to any variational parametrization.

We consider the decomposition

$$e^{\hat{\Lambda}\mathrm{d}t} = \prod_{i=1}^{s} S^{(T)}_{\alpha_i,a_i} + \mathcal{O}\Big(\mathrm{d}t^{o(s)+1}\Big), \tag{12}$$

where

$$S^{(T)}_{\alpha,a} = \Big(\mathbb{1} + a\hat{X}\mathrm{d}t\Big)\, e^{\alpha\hat{Z}\mathrm{d}t} \quad \text{with} \quad \alpha, a \in \mathbb{C}. \tag{13}$$

The expansion is accurate to order $o(s)$ but the analytical dependence on s is not straightforward to derive. For the lowest orders we find $o(1) = 1$, $o(2) = 2$, and $o(4) = 3$. Each term in the product consists in principle of two optimizations: the first compressing the off-diagonal transformation, the second the diagonal one. The advantage of this decomposition is that the latter optimization can be performed exactly with negligible computational effort (see Appendix G).

The same approach can be extended to Padé-like schemes by substituting in Eq. (12) the term

$$S^{(P)}_{\alpha,b,a} = \Big(\mathbb{1} + b\hat{X}\mathrm{d}t\Big)^{-1}\Big(\mathbb{1} + a\hat{X}\mathrm{d}t\Big)e^{\alpha\hat{Z}\mathrm{d}t}, \tag{14}$$

with $b, \alpha, a \in \mathbb{C}$. All coefficients are again obtained semi-analytically (see Appendix A). Though we do

[1] Acting diagonally in the computational basis means that $\langle a|\,\hat{Z}\,|b\rangle \propto \delta_{ab}$.

not have an explicit expression for the order $o(s)$ resulting from an s-substep expansion of this form, we find for the shallower schemes $o(1) = 2$, $o(2) = 3$, and $o(3) = 4$.

We will refer to these schemes as *split LPE* (S-LPE) and *split PPE* (S-PPE), respectively. They have the two advantages. First, they reduce the complexity of the optimizations in Eq. (8), and second, in many cases they reduce the prefactor of the error of their reciprocal non-split counterpart of the same order, as evidenced in Fig. 1.

3 State compression Optimizations

In Section 2, we introduced various schemes for efficiently decomposing unitary dynamics into a sequence of minimization problems, intentionally omitting the specific form of the loss function. The sole requirement was that the loss function quantify the dissimilarity between quantum states, reaching its minimum when the states perfectly match.

This section is organized as follows: In Section 3.1, we discuss a specific choice for the loss function — the infidelity — and explore some of its general properties. In Section 3.2, we examine the properties of different stochastic estimators for the fidelity and its gradient, identifying those that are most stable and perform best in practice. Finally, in Section 3.3 we review results on natural gradient optimization and introduce, in Section 3.3.1, an automatic regularization strategy that simplifies hyperparameter tuning for these simulations.

3.1 The generic fidelity optimization problem

A common measure of similarity between two pure quantum states is the fidelity, defined as [47, 70, 62]

$$\mathcal{F}\left(\hat{V}\left|\psi\right\rangle, \hat{U}\left|\phi\right\rangle\right) = \frac{\left\langle\psi\right|\hat{V}^\dagger\hat{U}\left|\phi\right\rangle\left\langle\phi\right|\hat{U}^\dagger\hat{V}\left|\psi\right\rangle}{\left\langle\psi\right|\hat{V}^\dagger\hat{V}\left|\psi\right\rangle\left\langle\phi\right|\hat{U}^\dagger\hat{U}\left|\phi\right\rangle}, \quad (15)$$

where the operators $\hat{U}$ and $\hat{V}$ are included as in Eq. (8). In this work, we adopt the infidelity $\mathcal{L} \equiv \mathcal{I} = 1 - \mathcal{F}$ as the loss function for each substep of Eq. (8), though alternative, less physically motivated metrics are also possible [64, 71].

The choice of operators $\hat{U}$ and $\hat{V}$ in Eq. (8) significantly influences the numerical complexity of the optimization. In Trotter-like decompositions, $\hat{U}$ and $\hat{V}$ act locally on a few particles, inducing minor changes to the wavefunctions. These localized transformations yield smoother optimization landscapes, which are effectively navigated using standard stochastic gradient methods, such as Adam [72]. This likely explains the findings of Sinibaldi et al. [47], where natural gradient optimization provided no significant improvements in performance.

In contrast, Taylor, LPE, and PPE schemes encode global transformations in $\hat{U}$ and $\hat{V}$, causing the target and variational wavefunctions to diverge substantially. These global transformations result in more complex optimization landscapes, where we find standard stochastic gradient descent methods inadequate (not shown). This issue is further exacerbated when training deep neural network architectures. To address these challenges, we employ parameterization-invariant optimization strategies, specifically natural gradient descent (NGD) which we discuss in Section 3.3. NGD adjusts the optimization path based on the geometry of the output space, enabling more efficient convergence in complex, high-dimensional problems. We find that NGD plays a critical role in improving both convergence and the overall efficiency of our proposed schemes (c.f. Appendix F).

3.2 Stochastic estimators

Estimators of the fidelity [Eq. (15)] take the general form

$$\mathcal{F}(\left|\psi\right\rangle, \left|\phi\right\rangle) = \mathbb{E}_{(x,y)\sim\chi}[f(x,y)], \quad (16)$$

for a suitable choice of random variable σ, sampling distribution $\chi(\sigma)$, and local estimator $f(\sigma)$. While this expression is exact, its direct evaluation incurs exponential scaling with system size N, rendering it computationally infeasible. Equation (16) is thus generally approximated by its sample mean

$$\bar{f}_{N_s} = \frac{1}{N_s}\sum_{i=1}^{N_s} f(x_i, y_i) \quad \text{with} \quad (x_i, y_i) \sim \chi, \quad (17)$$

evaluated over a number of samples N_s scaling polynomially in system size.

Importantly, different stochastic estimators, while sharing identical expectation values in the limit of infinite samples, can exhibit significantly different behavior over finite sample sizes. Though some attention has been given to characterizing the properties of different fidelity estimators [47], a systematic study of the variance properties of the estimators of the gradient is still lacking. As an accurate estimation of the gradient is crucial for driving fidelity optimization to convergence, this represents a central issue for quantum many-body dynamics.

In Section 3.2.1 we extend the analysis of Sinibaldi et al. [47], further improving on the variance properties of the fidelity estimator. In Section 3.2.2 we examine the properties of various gradient estimators, decoupling their analysis from that of the fidelity itself. Our findings are summarized in Tables 2 and 3, respectively.

3.2.1 Fidelity

As explained in Ref. [70], the natural choice for the fidelity estimator in Eq. (16) is the *simple Monte Carlo*

Name	Ref.	Eq.	+CV	+RW
smc	[47], 3.2.1	(18)	(20)	(85)
cmc	[61, 62, 63], 3.2.1	(21)	(22)	(87)

Table 2: List of stochastic fidelity estimators, their definition, and their expression with control variables (CV) and reweighting (RW). +CV links to the expressions of the estimators with control variables. +RW links to the expressions allowing sampling from the Born distribution of $|\phi\rangle$ rather than $\hat{U}|\phi\rangle$ when linear transformations are applied to the target or variational state. Figure 2 shows that both estimators perform similarly, and need CV to give accurate values.

Name	Ref.	Eq.	+RW	NTK	Stability
∇cmc	[61, 62, 63], 3.2.2	(25)	(87)	✓	High
∇smc	3.2.2	(24)	(85)	✓	Medium
∇cv-smc	[47, 54]	(26)	(85)	✗	Low

Table 3: List of estimators for the gradient of the infidelity. The NTK column lists the equation to implement the natural gradient estimator in the limit of large number of parameters. Not all estimators can be expressed that way. +RW links to the expressions allowing sampling from the Born distribution of $|\psi\rangle$ rather than $\hat{V}|\psi\rangle$ when linear transformations are applied to the target or variational state. The stability score is determined by the empirical results discussed in Fig. 3.

(smc) estimator defined by

$$\begin{aligned}\chi(x,y) &= \pi_\psi(x)\pi_\phi(y) \equiv \pi(x,y),\\ f(x,y) &= \frac{\phi(x)}{\psi(x)}\frac{\psi(y)}{\phi(y)} \equiv A(x,y),\end{aligned} \tag{18}$$

where the samples $\sigma = (x, y)$ are drawn from the joint Born distribution $\pi(x, y)$ of the two states. The estimator of the infidelity is $1 - f(x, y)$. Since $\pi(x, y)$ is separable, sampling can be carried out independently over the Born distributions $\pi_\psi(x) = |\psi(x)|^2/\langle\psi|\psi\rangle$ and $\pi_\phi(y) = |\phi(y)|^2/\langle\phi|\phi\rangle$. While Ref. [70] demonstrated that the variance of this estimator vanishes as $\mathcal{I} \to 0$, this does not guarantee we will be able to resolve arbitrarily small values of $\mathcal{I}$ from MC noise. Our ability to do so is measured by the signal to noise ratio (SNR) of the infidelity estimator, defined as

$$\frac{\mathrm{SNR}}{\sqrt{N_s}} = \frac{|1-\bar{f}_{N_s}|}{\sqrt{\frac{1}{N_s-1}\sum_{i=1}^{N_s}\left|f(\sigma_i)-\bar{f}_{N_s}\right|^2}}. \tag{19}$$

Reference [47] shows that the SNR of the smc estimator vanishes for $\mathcal{I} \to 0$ making it an unreliable indicator of progress in state compression problems [47]. In Fig. 2(a), we provide evidence of this by displaying the infidelity estimator's SNR as a function of the exact infidelity recorded along the best optimization amongst those presented in Fig. 3(a).

To address this issue, Sinibaldi et al. [47] leveraged the identity $\mathbb{E}_\pi[|A(x,y)|^2] = 1$ to construct the *control-variate-enhanced smc* (cv-smc) estimator

$$\begin{aligned}\chi(x,y) &= \pi(x,y)\\ f(x,y) &= \mathrm{Re}\{A(x,y)\} + c\Big(|A(x,y)|^2 - 1\Big) \equiv F(x,y).\end{aligned} \tag{20}$$

As shown in Fig. 2(a), the analytical control variable $|A(x, y)|^2$ allows for drastic variance reduction upon appropriate tuning of the control parameter c[2].

[2] As $|\psi\rangle \to |\phi\rangle$, the optimal choice for c has been shown to converge to $c = -1/2$.

An alternative variance reduction technique, readily applicable to Eq. (18), is Rao-Blackwellization. In its simplest form, Rao-Blackwellization reduces the variance of an estimator by replacing it with the conditional expectation with respect to a subset of its variables [73, 74]. For the particular case of a factorizable distribution such as $\chi(x, y) = \pi_\psi(x)\pi_\phi(y)$, Rao-Blackwellization amounts to marginalizing over the target state's distribution. In the following, we will refer to the Rao-Blackwellized smc estimator as the *conditional Monte Carlo* (cmc) estimator, expressed as (details in Appendix D)

$$\begin{aligned}&\chi(x,y) = \chi(x) = \pi_\psi(x),\\ &f(x) = \mathbb{E}[A(x,y)|x] = A_x(x)\,\mathbb{E}_{y\sim\pi_\phi}[A_y(y)] \equiv H_{\mathrm{loc}}(x),\end{aligned} \tag{21}$$

where $A(x, y) = A_x(x)A_y(y)$, $A_x(x) = \phi(x)/\psi(x)$, and $A_y(y) = \psi(y)/\phi(y)$.

We remark that the computational cost of Eqs. (18) and (21) is equivalent. Indeed, they use the same set of samples $\{(x_i, y_i)\}_{i=1}^{N_s}$ and evaluate ψ and ϕ over all of them. The difference is that while Eq. (18) sums over the diagonal pairs (x_i, y_i) alone, the sample average in Eq. (21) includes all cross terms (x_i, y_j).

Though the connection to conditional Monte Carlo was not drawn, the cmc estimator was used in Refs. [61, 62, 63]. We demonstrate in Fig. 2(a) that just like the smc estimator, the cmc estimator also suffers from a vanishing SNR in the limit of $\mathcal{I} \to 0$.

In the following, we combine the two variance reduction techniques detailed above, control variates and Rao-Blackwellization, to produce the *control-variate-enhanced cmc* (cv-cmc) estimator (derivation in Appendix B)

$$\begin{aligned}\chi(x) &= \pi_\psi(x)\\ f(x) &= \mathbb{E}[F(x,y)|x]\\ &= \mathrm{Re}\{H_{\mathrm{loc}}(x)\} + c\Big(|A_x(x)|^2\,\mathbb{E}_{y\sim\pi_\phi}\left[|A_y(y)|^2\right] - 1\Big).\end{aligned} \tag{22}$$

As a direct consequence of the Rao-Blackwell theorem, the cv-cmc estimator displays an SNR perfor-

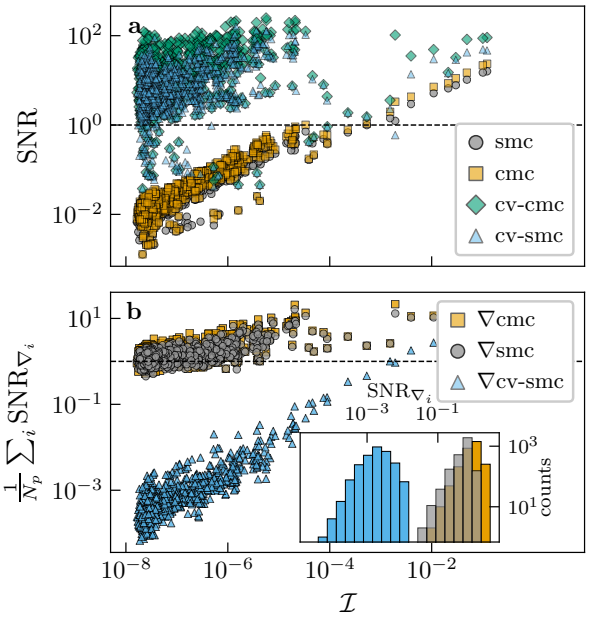

Figure 2: (a) Signal-to-noise ratio [Eq. (19)] of the infidelity as a function of its exact value for the estimators given in Eqs. (18) (smc), (21) (cmc), (20) (cv-smc), and (22) (cv-cmc). For all control-variate-enhanced estimators, the control coefficient is set to the asymptotically optimal value $c = -1/2$. (b) Component-averaged signal-to-noise ratio of the infidelity gradient estimators in Eqs. (25) (∇cmc), (24) (∇smc), and (26) (∇cv-smc). The inset shows the distribution of the component-wise signal-to-noise ratio at the 300-th iterate. All estimators are evaluated on the same states, specifically those visited along the curve with lowest final infidelity among those reported in Fig. 3(a). Parameters: $N = 16$, $J = 1$, $h = h_c/10$, $N_s = 2^{12}$, $n_{\text{iter}} = 10^3$, $\alpha = 0.27$, $\lambda = 3.17 \times 10^{-7}$, and $\boldsymbol{\Theta}_{\text{CNN}} = (10, 10, 10, 10; 3)$.

mance superior to all others, as illustrated in Fig. 2(a). We adopt this estimator in all simulations presented below.

We finally remark that p-tVMC might require evaluating $\mathcal{F}(\hat{V}|\psi\rangle, \hat{U}|\phi\rangle)$ thereby implicating the necessity of sampling from the Born distributions of the transformed states $\hat{V}|\psi\rangle$ and $\hat{U}|\phi\rangle$. This introduces an additional overhead which can however be sidestepped by means of importance sampling. We show this in Appendix E.

3.2.2 Gradient

While the preceding section provides compelling evidence that control variates are essential for accurate fidelity estimation, it does not address how this impacts the estimation of the gradient. In gradient-based optimization problems, such as Eq. (8), managing the variance of gradient estimates is crucial for achieving stable and efficient convergence. Historically, however, infidelity-optimization strategies have overlooked this aspect, simply adopting the gradient estimator obtained from automatic differentiation of the chosen fidelity estimator according to:

$$\begin{aligned} \boldsymbol{\nabla}\mathcal{F} &= \boldsymbol{\nabla}\mathbb{E}_\chi[f(\sigma)] = \mathbb{E}_\chi[g(\sigma)], \\ g(\sigma) &= 2\,\mathrm{Re}\{\Delta J(\sigma)\}f(\sigma) + \boldsymbol{\nabla} f(\sigma). \end{aligned} \tag{23}$$

While for an infinite number of samples, this expression is of course valid, when approximating the average with the sample mean $\bar{g}_{N_s}$, there is no guarantee that the selected gradient estimator will perform well. Indeed, as we show below, good variance properties of the fidelity estimator do not necessarily translate, under Eq. (23), into good variance properties of the gradient estimator. As a result, stable low-variance estimators for both fidelity and gradient have, until now, never been simultaneously employed. Specifically, while Refs. [47, 54] employ a control-variate-enhanced estimator for the fidelity, their choice of gradient estimator suffers from a vanishing SNR, rendering it ineffective for optimization. Conversely, the gradient estimator adopted in Refs. [62, 63, 61] has good variance properties, while the fidelity estimator exhibits a vanishing SNR. In the latter case, we observe this often leads to an underestimation of the quality of the results.

In support to this point, we explicitly evaluate the gradient estimators obtained from the smc, cmc, and cv-smc fidelity estimators, and compare their performance. We refer to such estimators as ∇smc, ∇cmc, and ∇cv-smc, respectively. For the smc estimator [Eq. (18)], automatic differentiation of Eq. (16) yields:

$$\begin{aligned} \chi(x,y) &= \pi(x,y), \\ g(x,y) &= 2\,\mathrm{Re}\{\Delta J(x) A(x,y)^*\}, \end{aligned} \tag{24}$$

with $\Delta J(x)$ defined in Eq. (28). Applying the same procedure to the cmc estimator [Eq. (21)] results in:

$$\begin{aligned} \chi(x,y) &= \chi(x) = \pi_\psi(x), \\ g(x,y) &= 2\,\mathrm{Re}\{\Delta J(x) H_{\text{loc}}(x)^*\}. \end{aligned} \tag{25}$$

This choice underlies the investigations in Refs. [61, 62, 63]. Notably, the expression in Eq. (25) can be derived directly via Rao-Blackwellization of Eq. (24), guaranteeing a reduction in variance. Finally, the gradient obtained by differentiation of the cv-smc estimator [Eq. (20)] reads:

$$\begin{aligned} \chi(x,y) &= \pi(x,y), \\ g(x,y) &= \mathrm{Re}\Big\{2\Delta F(x,y) J(x) + \\ &\quad + \Big(A(x,y) + 2c|A(x,y)|^2\Big)[J(y) - J(x)]\Big\}, \end{aligned} \tag{26}$$

with $F(x,y)$ as defined in Eq. (20). This expression underlies the studies in Refs. [47, 54].

To evaluate the performance of these gradient estimators, we first compare their SNRs. Since $\boldsymbol{\nabla}\mathcal{F} \in \mathbb{R}^{N_p}$, we compute the SNR of each component of the gradient vector. In Fig. 2(b), we present the component-averaged SNR as a function of the true

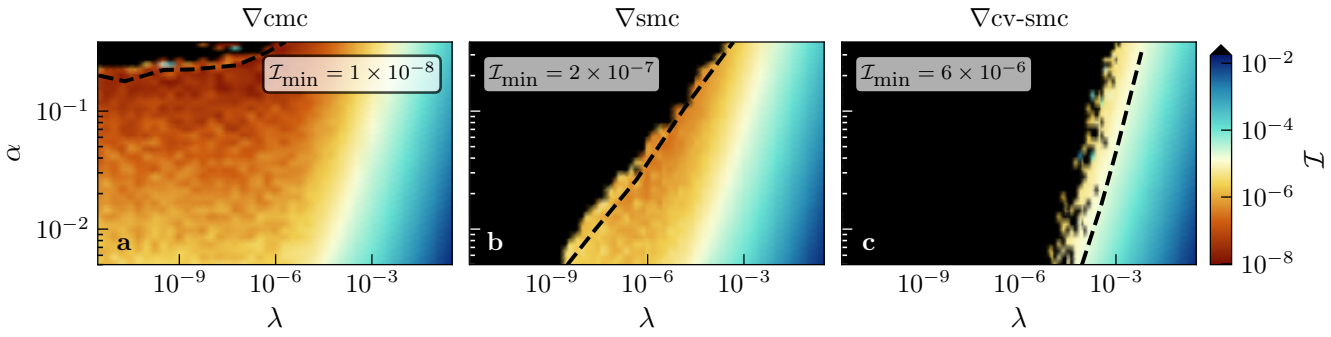

Figure 3: Stability diagram for the (a) ∇cmc [Eq. (25)], (b) ∇smc [Eq. (24)], and (c) ∇cv-smc [Eq. (26)] gradient estimators as a function of the regularization coefficient λ and learning rate α. Stability is quantified by the lowest infidelity (color scale) achieved across an ensemble of 10 optimizations with identical hyperparameters, identical initial parameters, but different random seeds for the Monte Carlo sampling. Comparisons are made for a fixed computational cost as all optimizations consist of $n_{\text{iter}} = 10^3$ iterations. Analogous conclusions can be drawn from the stability diagrams where $n_{\text{iter}}/\alpha = \text{const.}$ (not shown). Regions in black correspond to (α, λ) combinations where all simulations diverged, indicating instability. Dashed black lines outline areas where more than 50% of simulations diverged, providing a second visual stability threshold supporting the stability summary in Table 3. The inset in each panel shows the minimal infidelity $\mathcal{I}_{\min}$ achieved within the stable region (fewer than 50% of simulations diverged). For reliability and consistency, regardless of the chosen gradient estimator, we display the exact fidelity evaluated in full-summation. While the reference state wave function is exactly encoded into the parameters of a log-state-vector ansatz, the variational state is approximated with a complex CNN architecture. Parameters: $N = 16$, $J = 1$, $h = h_c/10$, $N_s = 2^{12}$, $n_{\text{iter}} = 10^3$, and $\mathbf{\Theta}_{\text{CNN}} = (10, 10, 10, 10; 3)$.

infidelity recorded during the best optimization run of Fig. 3(a). The inset shows a distribution of per-component SNR values. Interestingly, while the cv-smc estimator achieves superior SNR for fidelity estimation, this advantage is reversed when estimating the gradient. Specifically, the SNR of the cv-smc gradient estimator declines rapidly as the simulation progresses toward convergence. In contrast, gradient estimators derived from the bare smc and cmc fidelity estimators maintain an SNR of $\mathcal{O}(1)$ as convergence is approached. We attribute this disparity to the covariance structure present in Eqs. (24) and (25), but absent in Eq. (26). This covariance structure introduces sample-specific control variates that dynamically adapt to the non-stationary nature of the variables, significantly reducing variance and enhancing estimator stability (c.f. Appendix D).

To further characterize the performance of the gradient estimators, we examine their ability to drive infidelity optimizations to convergence. As a benchmark, we consider a state-matching problem in which we optimize the variational state to match a given reference state. As a physical reference we take $|\psi(t)\rangle = \exp(-i\hat{H}t)\,|{\rightarrow}\ldots{\rightarrow}\rangle$ with $\hat{H}$ the transverse field Ising Hamiltonian in Eq. (38). In general, we observe that states at shorter times are easier to learn, with smaller differences in the results produced by the various gradient estimators. Conversely, states at longer times are significantly more challenging to accurately match. It is important to emphasize that this state-matching setup is inherently more demanding than the dynamics itself. In the simulation of dynamics, the state at each timestep is initialized from the preceding timestep, offering a more informed starting point closer to the target state. By contrast, the state-matching problem begins in general from a random state, making the optimization considerably more difficult.

In Fig. 3 we present the accuracies with which we were able to recover the state at $Jt = 0.5$, a time point that we consider *relatively challenging* and representative of the general trends observed. Convergence is quantified by the final optimization infidelity (colormap) which we display as a function of the NGD regularization parameter λ (see Section 3.3) and learning rate α for the different gradient estimators detailed above. For each pair (α, λ) we consider 10 different replicas with different sampling histories. Black regions in the figure represent instability zones, where all replicas diverged, indicating poor stability of the method for the selected hyperparameter combination.

Overall, our results indicate that the ∇cmc gradient estimator consistently delivers the best convergence and exhibits the highest stability among all the gradient estimators examined in this study and reported in the available literature. We rely on this estimator for all optimizations presented in Section 4.

3.3 Natural Gradient Descent

Throughout this work we use NGD to solve the optimizations in Eq. (8). In its simplest implementation, given the current iterate $\theta_k \in \mathbb{R}^{N_p}$, NGD proposes a new iterate $\theta_{k+1} = \theta_k - \alpha_k\,\delta_0$, where α_k is a schedule of learning rates, and δ_0 the natural gradient at the current iterate. δ_0 is determined by minimizing a local quadratic model $M(\delta)$ of the objective, formed using gradient and curvature information at θ_k [75, 76, 77].

Formally,

$$M(\delta) = \mathcal{L}(\theta_k) + \boldsymbol{\nabla}\mathcal{L}(\theta_k)^\top \delta + \frac{1}{2}\delta^\top \boldsymbol{B}\,\delta,$$
$$\delta_0 = \operatorname*{argmin}_\delta M(\delta) = \boldsymbol{B}^{-1}\boldsymbol{\nabla}\mathcal{L}, \tag{27}$$

where $\boldsymbol{B}$ is a symmetric positive definite *curvature matrix*. This matrix is taken to be the Fisher information matrix when modeling probability distributions, or the Fubini-Study (FS) metric tensor for quantum states [48, 78, 79]. The latter is a Gram matrix estimated as[3]

$$\boldsymbol{S} = \mathrm{Re}\left\{\mathbb{E}_{\pi_\psi}\left[\Delta J^*(x)\Delta J(x)^\top\right]\right\} = \boldsymbol{X}^\top\boldsymbol{X} \in \mathbb{R}^{N_p\times N_p}, \tag{28}$$

where $J(x) = \boldsymbol{\nabla}\log\psi(x) \in \mathbb{C}^{N_p}$ is the quantum score function, $\Delta J(x) = J(x) - \mathbb{E}_{\pi_\psi}[J(x)]$, and

$$\boldsymbol{X} = \begin{pmatrix}\mathrm{Re}\{\boldsymbol{O}\}\\ \mathrm{Im}\{\boldsymbol{O}\}\end{pmatrix} \in \mathbb{R}^{2N_s\times N_p}, \tag{29}$$

with

$$\boldsymbol{O} = \frac{1}{\sqrt{N_s}}\begin{pmatrix}\Delta J(x_1)^\top\\ \vdots\\ \Delta J(x_{N_s})^\top\end{pmatrix} \in \mathbb{C}^{N_s\times N_p}. \tag{30}$$

Here, the samples $\{x_1,\dots,x_{N_s}\}$ are taken from π_ψ. In essence, NGD is a way of implementing steepest descent in state space rather than parameter space. In practice, however, NGD still operates in parameter space, computing directions in the space of distributions and translating them back to parameter space before implementing the step [75, 76]. As Eq. (27) stems from a Taylor approximation of the loss, it is important that the step in parameter space be small, for the expansion (and the update direction) to be reliable. As the FS tensor is often ill-conditioned or rank-deficient, this requirement is enforced by hand by adding to the objective a damping term with a regularization coefficient λ that penalizes large moves in parameter space. This yields the update

$$\delta_\lambda = \operatorname*{argmin}_\delta M(\delta) + \frac{\lambda}{2}\|\delta\|^2 = (\boldsymbol{X}^\top\boldsymbol{X} + \lambda\mathbb{1}_{N_p})^{-1}\boldsymbol{X}^\top\boldsymbol{\epsilon}, \tag{31}$$

where we used that $\boldsymbol{B} = \boldsymbol{S} = \boldsymbol{X}^\top\boldsymbol{X} \in \mathbb{C}^{N_p\times N_p}$ and $\boldsymbol{\nabla}\mathcal{L} = \boldsymbol{X}^\top\boldsymbol{\epsilon}$ with $\boldsymbol{\epsilon} \in \mathbb{R}^{2N_s}$[4]. Alternative ways of formulating this constraint exist, such as trust-region methods [77], proximal optimization, or Tikhonov damping [76], all eventually leading to Eq. (31).

It is important to note that evaluating the curvature matrix over a subset of configurations stochastically sampled from the full Hilbert space ensures at most a decrease of the loss over the sampled configurations. This does not guarantee a reduction in the overall loss or in the loss evaluated on a different set of configurations. A desirable property of any stochastic optimization method is its ability to approach optimal (or near-optimal) behavior in the limit of infinite samples. The expectation is that, with a sufficiently large sample size, a reduction in the loss on the sampled data will correspond to a global decrease in the loss.

The main challenge with NGD is the high computational cost of inverting $(\boldsymbol{X}^\top\boldsymbol{X}+\lambda\mathbb{1}_{N_p})$ in large-scale models with many parameters ($N_p \gg N_s$). Various approximate approaches have been proposed to address this, such as layer-wise block diagonal approximations [80, 81], Kronecker-factored approximate curvature [82, 83], and unit-wise approximations [84, 85]. At the moment, the only method enabling the use of NGD in deep architectures without approximating the curvature matrix is the tangent kernel method [86, 87], recently rediscovered in the NQS community as minSR [88]. This approach leverages a simple linear algebra identity [86, 89, 90] to rewrite Eq. (31) as

$$\delta_\lambda = \boldsymbol{X}^\top(\boldsymbol{X}\boldsymbol{X}^\top + \lambda\mathbb{1}_{2N_s})^{-1}\boldsymbol{\epsilon}, \tag{32}$$

where the matrix $\boldsymbol{T} = \boldsymbol{X}\boldsymbol{X}^\top \in \mathbb{R}^{2N_s\times 2N_s}$ is known as the *neural tangent kernel* (NTK) [91, 92]. In the limit where $N_p \gg N_s$, inverting the NTK becomes much more tractable than inverting the FS tensor, shifting the computational bottleneck on the number of parameters to the number of samples. Importantly, the NTK formulation relies on the gradient of the objective being of the form $\boldsymbol{\nabla}\mathcal{F} = \boldsymbol{X}^\top\boldsymbol{\epsilon}$. As shown in Appendix C, this is not a prerogative of all estimators investigated in Section 3.2.2. Specifically, while the ∇smc and ∇cmc estimators admit such decomposition with

$$\boldsymbol{\epsilon} = \frac{2}{\sqrt{N_s}}\left(\mathrm{Re}\{A(x_1,y_1)\},\dots,\mathrm{Im}\{A(x_1,y_1)\},\dots\right)^\top,$$
$$\boldsymbol{\epsilon} = \frac{2}{\sqrt{N_s}}\left(\mathrm{Re}\{H_{\mathrm{loc}}(x_1)\},\dots,\mathrm{Im}\{H_{\mathrm{loc}}(x_1)\},\dots\right)^\top, \tag{33}$$

respectively, the ∇cv-smc estimator does not. This restriction prevents these estimators from being efficiently computed in the NTK framework, making them a poor choice for scaling to the deep network limit.

In addition to not supporting an NTK formulation, the absence of a form like $\boldsymbol{\nabla}\mathcal{F} = \boldsymbol{X}^\top\boldsymbol{\epsilon}$ also precludes the use of L-curve and generalized cross-validation methods for adaptively selecting the regularization coefficient λ. While these automatic damping strategies

[3] The expression of the FS tensor in Eq. (28) applies to a generic variational state $|\psi\rangle$ and is used to precondition the gradient of the objective as $\boldsymbol{S}^{-1}\boldsymbol{\nabla}\mathcal{L}(|\psi\rangle, |\phi\rangle)$, where $|\phi\rangle$ is an arbitrary target state. This formulation remains valid if the state $|\psi\rangle$ is transformed by an operator $\hat{V}$, with the corresponding replacement $\psi \to \tilde{\psi}$.

[4] Note that the identity $\boldsymbol{\nabla}\mathcal{L} = \boldsymbol{X}^\top\boldsymbol{\epsilon}$ does not hold universally for all loss functions, although it is verified for many prototypical choices, such as the mean squared error or the variational energy. In Appendix C, we demonstrate that the fidelity can exhibit this structure, although this is not guaranteed for all gradient estimators.

have been used in the literature [93], they are less suited to the NGD problem compared to those discussed in Section 3.3.1.

3.3.1 Autonomous damping

Choosing an appropriate value for λ is crucial for successful optimization. If λ is too large, the update resembles standard gradient descent with a very small learning rate. Conversely, if λ is too small, updates can become excessively large, particularly in low-curvature directions, possibly leading to increasing the loss rather than decreasing it [75]. Identifying an optimal value for λ is nontrivial, and it is rare for a single value of λ to be suitable across all optimization iterations. This challenge is exacerbated in p-tVMC calculations, where numerous successive infidelity optimizations, with very different target and initial states, must be performed. Performing an hyperparameter search analogous to that in Fig. 3 for every state compression is of course impractical, and an automated approach is required. Hereunder, we introduce a custom adaptive control strategy inspired by heuristics commonly employed in the numerical optimization literature [75, 76, 77, 82, 94].

Let θ_k be the parameters at the k-th iterate. The parameters at the following iterate are $\theta_{k+1} = \theta_k + \delta\theta_k$ with $\delta\theta_k = -\alpha_k \delta_{\lambda_k}$. Here, δ_{λ_k} is the natural gradient update, and α_k is the learning rate. Under ideal conditions (accurate curvature matrix, infinite sample size, and proper regularization), the optimal learning rate $\alpha_k = 1$ maximizes the decrease of the quadratic model. However, these conditions are rarely satisfied in practice and mitigating the overfitting caused by finite-sample effects typically requires lowering α_k [75]. We find that $\alpha_k \gtrsim 5\times10^{-2}$ is a reasonable choice in most cases.

To dynamically adapt λ_k, we build on the Levenberg-Marquardt heuristic [94], which relies on the reduction ratio

$$\rho_k = \frac{\mathcal{L}(\theta_k + \delta\theta_k) - \mathcal{L}(\theta_k)}{M(\delta\theta_k) - \mathcal{L}(\theta_k)}. \tag{34}$$

This ratio measures the accuracy of the quadratic model $M(\delta\theta_k)$ [c.f. Eq. (27)] in predicting $\mathcal{L}(\theta_k+\delta\theta_k)$. Small ρ_k ($\rho_k \ll 1$) suggests poor agreement between the model and the actual loss, indicating the need to increase λ_k. Larger ρ_k ($\rho_k \gtrsim 1$) indicates good agreement, allowing λ_k to be reduced [5]. The standard Levenberg-Marquardt algorithm aims at keeping

[5] In a trust-region approach to second-order optimization problems, an increase in λ_k corresponds to tightening the trust region. Conversely, a reduction in λ_k corresponds to a relaxation of the trust region [77].

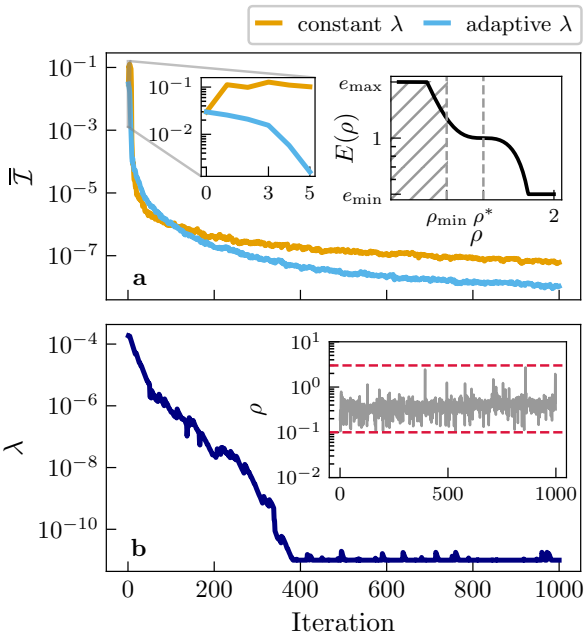

Figure 4: Performance of the autonomous damping algorithm described in Eq. (36). (a) Comparison of the infidelity across the optimization. The infidelity is computed exactly and averaged over 10 Monte Carlo runs, with parameter updates derived from noisy estimates of the ∇cmc gradient and curvature matrix. Results are compared to the best configuration ($\alpha = 0.27$, $\lambda = 3\times10^{-7}$) from Fig. 3(a). The first inset highlights an initial increase in the loss due to a suboptimal initial choice of λ. The second inset characterizes the behavior of the error function $E(\rho)$ in Eq. (37). Hatching is used to highlight the region $\rho < \rho_{\min}$ where updates are discarded and recomputed. (b) Evolution of the diagonal shift. The data corresponds to the replica achieving the lowest infidelity. The inset displays the value of reduction ratio ρ at each substep. Parameters: $q = 1$, $\beta_1 = 0.9$, $\beta_2 = 0.1$, $\alpha = 0.05$, $\rho_{\max} = 3$, $\rho_{\min} = 0.1$, $\rho^* = 0.35$, $\gamma_s = 0.9$, $e_{\max} = 2$, and $e_{\min} = 0.05$.

$1/4 \le \rho_k \le 3/4$ by taking $\lambda_{k+1} = m_k\lambda_k$ with

$$m_k = \begin{cases} 3/2 & \text{if} \quad \rho_k \le \frac{1}{4} \\ 2/3 & \text{if} \quad \rho_k > \frac{3}{4} \\ 1 & \text{if } \frac{1}{4} < \rho_k \le \frac{3}{4} \end{cases}. \tag{35}$$

Steps where $\rho_k < 0$ would actually lead to an increase of the loss are thus discarded and recomputed with the adjusted λ_k.

While effective in many cases, the standard heuristic can result in rapid and uncontrolled variations in ρ_k and λ_k, leading to suboptimal performance. Drawing inspiration from step-size control in differential equations, we enhance this heuristic by incorporating proportional-integral controllers, which enhance stability by combining proportional responses to current errors with integral corrections from accumulated past errors [95]. As a measure of the error we take the deviation of ρ_k from a desired target value ρ^*. The

multiplier m_k for adjusting λ_k is then taken to be

$$m_k = \min\Big[m_{\max}, \max\Big[m_{\min}, E^{\beta_1}(\rho_k)\, E^{-\beta_2}(\rho_{k-1})\Big]\Big], \tag{36}$$

where $e_{\min}$ and $e_{\max}$ constrain the scaling factor within a safe range, and β_1 and β_2 control the influence of current and past errors, respectively. Contrary to standard controllers where the magnitude of the (positive) error alone modulates m_k, here the sign of the error plays an important role as well. Specifically, the error functional $E(\rho)$ is chosen to ensure that $m_k > 1$ for $\rho_k < \rho^*$ and $m_k < 1$ for $\rho_k > \rho^*$. While various choices are possible, we choose

$$E(\rho) = \mathrm{sgn}(\rho^* - \rho)|\rho^* - \rho|^q + 1, \tag{37}$$

whose functional form we display as an inset to Fig. 4(a). To further enhance robustness, we reject steps where $\rho_k > \rho_{\max}$ or $\rho_k < \rho_{\min}$ and recompute the update with the newly adjusted λ_k.

We find the proposed approach to work well in practice and to be insensitive to minor changes in the selected thresholds. In general, we find $q = 1$, $\beta_1 = 0.9$, $\beta_2 = 0.1$, $\alpha = 0.05$, $\rho_{\max} = 3$, $\rho_{\min} = 0.1$, $\rho^* = 0.35$, $e_{\max} = 2$, and $e_{\min} = 0.05$ to be good choices.

4 Numerical Experiments

In the following sections, we benchmark our methods on the paradigmatic example of the two-dimensional transverse-field Ising model (TFIM), a widely used testbed for NQS dynamics [23, 47, 49, 50, 52, 60, 64]. The Hamiltonian is given by

$$\hat{H} = -J \sum_{\langle i,j \rangle} \hat{\sigma}_i^z \sigma_j^z - h \sum_i \hat{\sigma}_i^x, \tag{38}$$

where J is the nearest-neighbor coupling strength and h represents the transverse field strength. Throughout this work, we set $J = 1$, we adopt periodic boundary conditions, and we fix the z-axis as the quantization axis.

At zero temperature, this model undergoes a quantum phase transition at the critical point $h_c = 3.044$. This separates the ferromagnetic phase ($h < h_c$), from the paramagnetic phase ($h > h_c$). Deep in the ferromagnetic phase ($h \to 0$) the ground state is degenerate and lies in the subspace spanned by $|\uparrow \ldots \uparrow\rangle$ and $|\downarrow \ldots \downarrow\rangle$. In the opposite limit ($h \to \infty$), the ground state is $|\rightarrow \ldots \rightarrow\rangle$, with spins aligned along the transverse field direction.

We demonstrate that the far-from-equilibrium dynamics characteristic of the quantum quenches of this model can be efficiently captured using p-tVMC. Our results are consistent across architectures, integration schemes, and quench parameters.

4.1 Small-scale experiments

Before presenting results for large system sizes, we validate our method against exact diagonalization results on a 4×4 lattice: small enough to allow simulating the exact dynamics but large enough for MC sampling to be non-trivial. We consider the quench dynamics in which the system is initialized in the paramagnetic ground state at $h = \infty$, and evolved under a finite transverse field of strength $h = 2h_c$. For these simulations we adopt the Convolutional Neural Network (CNN) ansatz described in Appendix H.1.

We compare several integration schemes: S-LPE-3, S-PPE-3 (third-order in $\mathrm{d}t$), and S-LPE-2, S-PPE-2 (second-order in $\mathrm{d}t$). We intentionally choose a fixed step size of $h\mathrm{d}t = 3 \times 10^{-2}$, too big for product schemes of second order to accurately approximate the dynamics at hand, to underscore the advantages of our higher-order schemes. Optimizations are performed using NGD and the autonomous damping strategies detailed in Section 3.3.1.

The variational evolution closely follows the dynamics predicted by the integration scheme, achieving infidelities with respect to the exact solution below 10^{-5} for the best-performing S-PPE-3 scheme. The ideal behavior of each integrator is estimated by applying the product expansion directly to the full state vector [equivalent to solving exactly the optimization problem in Eq. (8) on a log-state vector ansatz]. This analysis reveals that the variational dynamics is influenced by two primary sources of error: optimization error and integration error. In the absence of optimization error, the variational dynamics would follow the approximate integrator's dynamics which, in the absence of integration error, would in turn match the exact evolution. For most schemes shown in Fig. 5, the dynamics is limited by integration error. An exception is the S-PPE-3 scheme, where the discretization is sufficiently accurate that optimization error becomes the dominant source at $ht \gtrsim 0.6$. This crossover between integration and optimization errors is more clearly illustrated in Fig. 6, where we analyze the error contributions for S-LPE-3 and S-PPE-2, the two best-performing schemes. While S-LPE-3 remains dominated by integration error throughout, S-PPE-3 exhibits a crossover point where optimization error overtakes, as indicated by the intersection of the dashed and dotted lines.

4.2 Large-scale experiments

We now demonstrate the effectiveness of our methods for simulating large-scale quantum dynamics. We again focus on the quench dynamics of the TFIM, this time on a 10×10 lattice and for the more challenging quench from $h = \infty$ to $h = h_c/10$. While no exact solution is available at this scale, this problem is widely used as a benchmark in quantum dynamics, allowing for comparison with established methods.

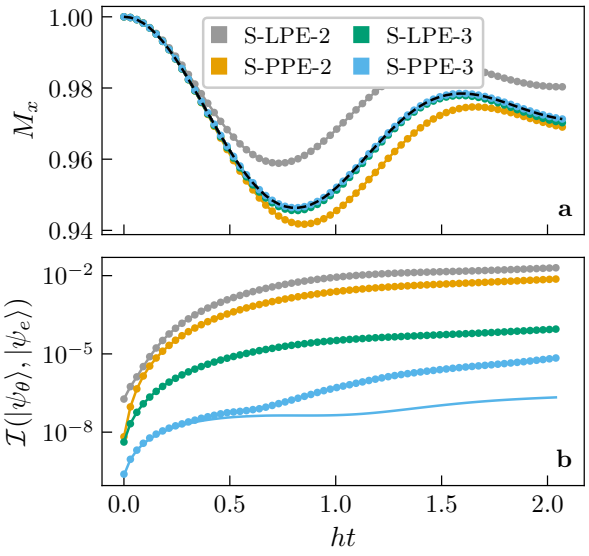

Figure 5: Quench dynamics ($h = \infty \to 2h_c$) of the transverse-field Ising model on a 4×4 lattice obtained using different integration schemes: S-LPE-2, S-LPE-3, S-PPE-2, S-PPE-3. We show the evolution of the average magnetization M_x (a), and of the infidelity between the exact solution $|\psi_e\rangle$ and its variational approximation $|\psi_\theta\rangle$ (b). Full dots are used to mark variational data, obtained by solving the optimization problems in Eq. (8). Solid lines detail the ideal behavior of each integrator scheme, estimated from a state-vector simulation of the dynamics resulting from the product expansion of the evolution operator. The dashed black line in (a) corresponds to the exact evolution. Parameters: $N = 16$, $J = 1$, $h = 2h_c$, $N_s = 2^{14}$, $\mathbf{\Theta}_{\text{CNN}} = (5, 4, 3; 3)$, and $\mathrm{d}t = 0.025$.

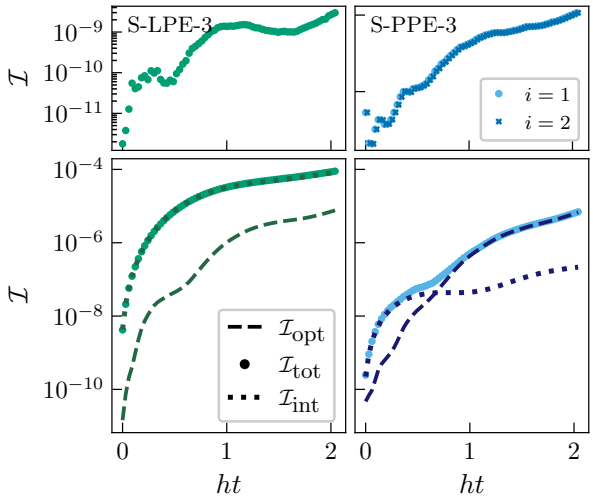

Figure 6: Evolution of the total infidelity ($\mathcal{I}_{\text{tot}}$), integration infidelity ($\mathcal{I}_{\text{int}}$), and optimization infidelity ($\mathcal{I}_{\text{opt}}$) as a function of time. The exact dynamics $|\psi_e(t)\rangle$, the dynamics obtained from state-vector simulations of product schemes $|\psi_a(t)\rangle$, and the variational dynamics $|\psi_\theta(t)\rangle$ are compared. The infidelities are defined as follows: $\mathcal{I}_{\text{tot}} = \mathcal{I}(|\psi_e(t)\rangle, |\psi_\theta(t)\rangle)$, $\mathcal{I}_{\text{int}} = \mathcal{I}(|\psi_e(t)\rangle, |\psi_a(t)\rangle)$, and $\mathcal{I}_{\text{opt}} = \mathcal{I}(|\psi_a(t)\rangle, |\psi_\theta(t)\rangle)$. The final step infidelities obtained at the end of the optimization problems in Eq. (8) are reported in the top panels. The different substeps of the S-PPE-3 scheme are indexed by i. Parameters: same as in Fig. 5.

In Fig. 7, we present results obtained using the S-LPE-3 integration scheme (c.f. Section 2.6) for times up to $Jt = 2$ where comparison with existing results is meaningful. To rigorously evaluate our approach, we utilize two architectures: a CNN with configuration $\mathbf{\Theta}_{\text{CNN}} = (5, 4, 3; 6)$ to explore the regime $N_s \gg N_p$, and a Vision Transformer (ViT) with $\mathbf{\Theta}_{\text{ViT}} = (2, 12, 6, 4; 6)$ for the opposite case where $N_s \ll N_p$. In the latter regime, direct inversion of the FS tensor becomes prohibitively expensive and the gradient estimator introduced in Eq. (25) proves essential for the evaluation of the natural gradient. As shown in Fig. 7(a), both models provide consistent predictions, underscoring the effectiveness of p-tVMC in yielding consistent results across different architectures, provided they are sufficiently expressive. These findings establish the ViT as a promising ansatz not only for ground-state simulations, as previously shown in Refs. [96, 39], but also for large-scale simulations of quantum dynamics.

While the p-tVMC results, the iPEPS data from Ref. [97], and the largest tVMC simulations from Ref. [49], generally show strong agreement, subtle differences emerge around the more challenging region $Jt \gtrsim 1.4$. Here, the tVMC curves deviate from the iPEPS predictions, while p-tVMC maintains closer alignment, suggesting superior stability at longer times. This is further quantified in Fig. 7(b), where we report the absolute deviation from the iPEPS baseline. Notably, p-tVMC captures long-time dynamics more accurately, highlighting its potential for enhanced long-term stability. We further observe, at the early stages of the simulation ($0.1 \lesssim Jt \lesssim 0.4$), the existence of a small region where tVMC data are marginally better aligned with the iPEPS baseline compared to p-tVMC. This region coincides with the high-activity region of the adaptive timestepping algorithm used in the tVMC simulations suggesting the origin of this discrepancy to be found in the larger, fixed, timestep employed in our calculations. While our current results are competitive with state-of-the-art methods, implementing adaptive timestepping in future iterations will further enhance the overall accuracy of the results. Finally, in Fig. 7(c) we display the final step-infidelities at each time step. We attribute the degradation of the quality of the fidelity optimizations to finite-sample effects that become more pronounced at later times. To validate this hypothesis we ran the same optimizations in a smaller 4×4 lattice evaluating expectation values exactly by summing over the entire Hilbert space. In the absence of MC noise, we observe convergence to numerical precision (see Appendix F). We remark that to stabilize our simulations, we employed the autonomous damp-

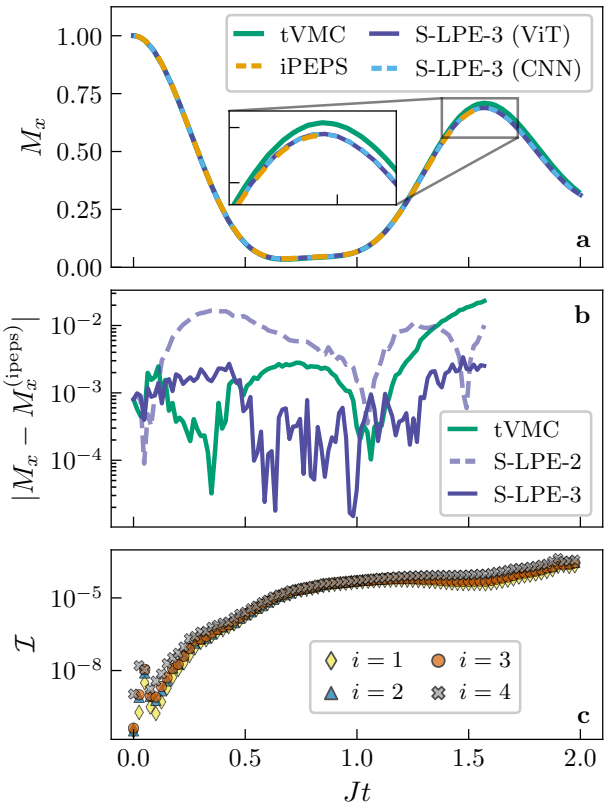

Figure 7: Quench dynamics across the critical point of the TFIM ($h = \infty \to h_c/10$) on a 10×10 lattice. (a) Average magnetization as a function of time. Results are compared with existing methods: tVMC [49] and iPEPS [97]. P-tVMC simulations are obtained using the S-LPE-3 scheme on two architectures: CNN and ViT. (b) Absolute difference in average magnetization between the iPEPS baseline and the variational simulations. P-tVMC results obtained using the SLPE2 and SLPE3 schemes are displayed. (c) Final optimization infidelity (step infidelity) achieved at each substep $i = 1, \ldots, 4$ of the S-LPE-3 scheme. For clarity, in panels (b) and (c) only the results for the ViT architecture are displayed. Similar curves were observed with the CNN (not shown). Parameters: $N = 100$, $J = 1$, $h = h_c/10$, $N_s = 2^{17}$ (CNN), $\mathbf{\Theta}_{\text{CNN}} = (5, 4, 3; 6)$, $N_s = 2^{15}$ (ViT), $\mathbf{\Theta}_{\text{ViT}} = (2, 12, 6, 4; 6)$, and $\mathrm{d}t = 0.025$.

ing strategy outlined in Section 3.3.1. All systems were initialized using a two-step process where we first use traditional VMC to approximate the ground state of $\hat{H}x = \sum_i \hat{\sigma}_i^x$, and then perform infidelity minimization to align the variational state with the constant-amplitude target state $\phi_{\text{gs}}(y) = \text{const}$ for all $y \in [-1, +1]^N$. .

5 Conclusions and outlooks

In this work, we provide a rigorous formalization of the p-tVMC method, decoupling the discretization of time evolution from the state compression task, performed by infidelity minimization.

In our analysis of the discretization scheme, we identify key criteria for constructing schemes that are simultaneously accurate and computationally efficient. Building on these principles, we address the limitations in prior approaches to p-tVMC through two novel families of integration schemes capable of achieving arbitrary-order accuracy in time while scaling favorably with the number of degrees of freedom in the system. This is made possible by making efficient use of the specific structure of the p-tVMC problem.

In the study of the fidelity optimization, we demonstrate the critical role of natural gradient descent in compressing non-local transformations of the wavefunction into parameter updates. Additionally, we introduce an automatic damping mechanism for NGD which provides robust performance without the need for extensive hyperparameter tuning at each timestep of the dynamics.

We further clarify which among the available stochastic estimators are most reliable for computing both the infidelity and its gradient, addressing open questions regarding the role of control variates in the estimation of infidelity and their necessity in gradient-based optimization.

By integrating these advances into the p-tVMC framework, we demonstrate the potential to achieve machine precision in small-scale state compression and time evolution tasks. Applying these methods to larger systems, we show that p-tVMC not only reproduces state-of-the-art tVMC results with higher accuracy and improved control, but also surpasses previous methods in terms of generality and stability.

While fully numerically exact simulations on large systems remain beyond reach — likely due to limitations in Monte Carlo sampling — this work establishes p-tVMC as a highly promising approach, capable of overcoming several intrinsic challenges faced by other methods, and bringing us closer to achieving precise, large-scale classical quantum simulations.

Software

Simulations were performed with NetKet [98, 99], and at times parallelized with mpi4JAX [100]. This software is built on top of JAX [101], equinox [102] and Flax [103]. We used QuTiP [104, 105] for exact benchmarks. The accompanying code and Mathematica scripts for the manuscript are available at github.com/NeuralQXLab/ptvmc-systematic-study.

Acknowledgments

We acknowledge the useful contributions of R. Chen towards the derivation of PPE and S-PPE schemes. We acknowledge insightful discussions with F. Becca, D. Poletti, G. Carleo, F. Ferrari and F. Minganti. We thank F. Caleca, M. Schmitt, and J. Dziarmaga for

sharing their data with us. We are grateful to L. L. Viteritti, C. Giuliani, A. Kahn, A. Shokry, L. Fioroni, and A. Sinibaldi for assisting in our fight against non-converging simulations, Jax bugs and complicated equations. F.V. acknowledges support by the French Agence Nationale de la Recherche through the NDQM project, grant ANR-23-CE30-0018. V.S. acknowledges support by the Swiss National Science Foundation through Projects No. 200020_185015, 200020_215172, and support from the EPFL Science Seed Fund 2021. This project was provided with computing HPC and storage resources by GENCI at IDRIS thanks to the grant 2023-AD010514908 on the supercomputer Jean Zay's V100/A100 partition.
During the redaction of this manuscript we were made aware of a late revision of Ref. [54] applying the LPE scheme to fermionic dynamics, referring to it as *Taylor Root Expansion*.

References

[1] I.M. Georgescu, S. Ashhab, and Franco Nori. "Quantum simulation". Reviews of Modern Physics **86**, 153–185 (2014).

[2] John Preskill. "Quantum Computing in the NISQ era and beyond". Quantum **2**, 79 (2018). url: https://doi.org/10.22331/q-2018-08-06-79.

[3] Olivier Ezratty. "Understanding quantum technologies 2023" (2021) arXiv:arxiv:2111.15352.

[4] Xiao Yuan, Suguru Endo, Qi Zhao, Ying Li, and Simon C. Benjamin. "Theory of variational quantum simulation". Quantum **3**, 191 (2019).

[5] Mari Carmen Bañuls. "Tensor network algorithms: A route map". Annual Review of Condensed Matter Physics **14**, 173–191 (2023).

[6] Román Orús. "Tensor networks for complex quantum systems". Nature Reviews Physics **1**, 538–550 (2019).

[7] U. Schollwöck. "The density-matrix renormalization group". Reviews of Modern Physics **77**, 259–315 (2005).

[8] Noa Feldman, Augustine Kshetrimayum, Jens Eisert, and Moshe Goldstein. "Entanglement estimation in tensor network states via sampling". PRX Quantum**3** (2022).

[9] J. Eisert. "Entanglement and tensor network states" (2013) arXiv:1308.3318.

[10] A. H. Werner, D. Jaschke, P. Silvi, M. Kliesch, T. Calarco, J. Eisert, and S. Montangero. "Positive tensor network approach for simulating open quantum many-body systems". Phys. Rev. Lett. **116**, 237201 (2016).

[11] J. Ignacio Cirac, David Pérez-García, Norbert Schuch, and Frank Verstraete. "Matrix product states and projected entangled pair states: Concepts, symmetries, theorems". Reviews of Modern Physics**93** (2021).

[12] Sebastian Paeckel, Thomas Köhler, Andreas Swoboda, Salvatore R. Manmana, Ulrich Schollwöck, and Claudius Hubig. "Time-evolution methods for matrix-product states". Annals of Physics **411**, 167998 (2019).

[13] Román Orús. "A practical introduction to tensor networks: Matrix product states and projected entangled pair states". Annals of Physics **349**, 117–158 (2014).

[14] Ulrich Schollwöck. "The density-matrix renormalization group in the age of matrix product states". Annals of Physics **326**, 96–192 (2011).

[15] Michael Lubasch, J Ignacio Cirac, and Mari-Carmen Bañuls. "Unifying projected entangled pair state contractions". New Journal of Physics **16**, 033014 (2014).

[16] F. Verstraete and J. I. Cirac. "Renormalization algorithms for quantum-many body systems in two and higher dimensions" (2004) arXiv:cond-mat/0407066.

[17] L. Tagliacozzo, G. Evenbly, and G. Vidal. "Simulation of two-dimensional quantum systems using a tree tensor network that exploits the entropic area law". Phys. Rev. B **80**, 235127 (2009).

[18] Timo Felser, Simone Notarnicola, and Simone Montangero. "Efficient tensor network ansatz for high-dimensional quantum many-body problems". Physical Review Letters**126** (2021).

[19] Pietro Silvi, Ferdinand Tschirsich, Matthias Gerster, Johannes Jünemann, Daniel Jaschke, Matteo Rizzi, and Simone Montangero. "The tensor networks anthology: Simulation techniques for many-body quantum lattice systems". SciPost Physics Lecture Notes (2019).

[20] Daniel Jaschke, Simone Montangero, and Lincoln D Carr. "One-dimensional many-body entangled open quantum systems with tensor network methods". Quantum Science and Technology **4**, 013001 (2018).

[21] Daniel Jaschke, Michael L. Wall, and Lincoln D. Carr. "Open source matrix product states: Opening ways to simulate entangled many-body quantum systems in one dimension". Computer Physics Communications **225**, 59–91 (2018).

[22] G. Evenbly and G. Vidal. "Tensor network states and geometry". Journal of Statistical Physics **145**, 891–918 (2011).

[23] Giuseppe Carleo and Matthias Troyer. "Solving the quantum many-body problem with artificial neural networks". Science **355**, 602–606 (2017).

[24] Or Sharir, Amnon Shashua, and Giuseppe Carleo. "Neural tensor contractions and the expressive power of deep neural quantum states". Phys. Rev. B **106**, 205136 (2022).

[25] Dong-Ling Deng, Xiaopeng Li, and S. Das Sarma. "Quantum entanglement in neural network states". Phys. Rev. X **7**, 021021 (2017).

[26] Julio Ureña, Antonio Sojo, Juani Bermejo-Vega, and Daniel Manzano. "Entanglement detection with classical deep neural networks". Scientific Reports**14** (2024).

[27] Giacomo Passetti and Dante M. Kennes. "Entanglement transition in deep neural quantum states" (2023) arXiv:2312.11941.

[28] Giacomo Torlai and Roger G. Melko. "Latent space purification via neural density operators". Physical Review Letters**120** (2018).

[29] Filippo Vicentini, Riccardo Rossi, and Giuseppe Carleo. "Positive-definite parametrization of mixed quantum states with deep neural networks" (2022).

[30] Di Luo, Zhuo Chen, Juan Carrasquilla, and Bryan K. Clark. "Autoregressive neural network for simulating open quantum systems via a probabilistic formulation". Physical Review Letters**128** (2022).

[31] Filippo Vicentini, Alberto Biella, Nicolas Regnault, and Cristiano Ciuti. "Variational neural-network ansatz for steady states in open quantum systems". Physical Review Letters**122** (2019).

[32] Debbie Eeltink, Filippo Vicentini, and Vincenzo Savona. "Variational dynamics of open quantum systems in phase space" (2023).

[33] Moritz Reh, Markus Schmitt, and Martin Gärttner. "Time-dependent variational principle for open quantum systems with artificial neural networks". Phys. Rev. Lett. **127**, 230501 (2021).

[34] Sidhartha Dash, Luca Gravina, Filippo Vicentini, Michel Ferrero, and Antoine Georges. "Efficiency of neural quantum states in light of the quantum geometric tensor". Communications Physics**8** (2025).

[35] Haimeng Zhao, Giuseppe Carleo, and Filippo Vicentini. "Empirical sample complexity of neural network mixed state reconstruction". Quantum **8**, 1358 (2024).

[36] Kenny Choo, Titus Neupert, and Giuseppe Carleo. "Two-dimensional frustrated J_1-J_2 model studied with neural network quantum states". Phys. Rev. B **100**, 125124 (2019).

[37] Or Sharir, Yoav Levine, Noam Wies, Giuseppe Carleo, and Amnon Shashua. "Deep autoregressive models for the efficient variational simulation of many-body quantum systems". Phys. Rev. Lett. **124**, 020503 (2020).

[38] Dian Wu, Riccardo Rossi, Filippo Vicentini, Nikita Astrakhantsev, Federico Becca, Xiaodong Cao, Juan Carrasquilla, Francesco Ferrari, Antoine Georges, Mohamed Hibat-Allah, Masatoshi Imada, Andreas M. Läuchli, Guglielmo Mazzola, Antonio Mezzacapo, Andrew Millis, Javier Robledo Moreno, Titus Neupert, Yusuke Nomura, Jannes Nys, Olivier Parcollet, Rico Pohle, Imelda Romero, Michael Schmid, J. Maxwell Silvester, Sandro Sorella, Luca F. Tocchio, Lei Wang, Steven R. White, Alexander Wietek, Qi Yang, Yiqi Yang, Shiwei Zhang, and Giuseppe Carleo. "Variational benchmarks for quantum many-body problems". Science **386**, 296–301 (2024).

[39] Luciano Loris Viteritti, Riccardo Rende, and Federico Becca. "Transformer variational wave functions for frustrated quantum spin systems". Phys. Rev. Lett. **130**, 236401 (2023).

[40] Xiao Liang, Wen-Yuan Liu, Pei-Ze Lin, Guang-Can Guo, Yong-Sheng Zhang, and Lixin He. "Solving frustrated quantum many-particle models with convolutional neural networks". Phys. Rev. B **98**, 104426 (2018).

[41] James Stokes, Javier Robledo Moreno, Eftychios A. Pnevmatikakis, and Giuseppe Carleo. "Phases of two-dimensional spinless lattice fermions with first-quantized deep neural-network quantum states". Phys. Rev. B **102**, 205122 (2020).

[42] Attila Szabó and Claudio Castelnovo. "Neural network wave functions and the sign problem". Phys. Rev. Res. **2**, 033075 (2020).

[43] Kenny Choo, Antonio Mezzacapo, and Giuseppe Carleo. "Fermionic neural-network states for ab-initio electronic structure". Nature Communications**11** (2020).

[44] Gino Cassella, Halvard Sutterud, Sam Azadi, N. D. Drummond, David Pfau, James S. Spencer, and W. M. C. Foulkes. "Discovering quantum phase transitions with fermionic neural networks". Phys. Rev. Lett. **130**, 036401 (2023).

[45] Javier Robledo Moreno, Giuseppe Carleo, Antoine Georges, and James Stokes. "Fermionic wave functions from neural-network constrained hidden states". Proceedings of the National Academy of Sciences**119** (2022).

[46] Giuseppe Carleo, Lorenzo Cevolani, Laurent Sanchez-Palencia, and Markus Holzmann. "Unitary dynamics of strongly interacting bose gases with the time-dependent variational monte carlo method in continuous space". Physical Review X**7** (2017).

[47] Alessandro Sinibaldi, Clemens Giuliani, Giuseppe Carleo, and Filippo Vicentini. "Unbiasing time-dependent variational monte carlo by projected quantum evolution". Quantum **7**, 1131 (2023).

[48] James Stokes, Brian Chen, and Shravan Veerapaneni. "Numerical and geometrical aspects of flow-based variational quantum monte carlo". Machine Learning: Science and Technology **4**, 021001 (2023).

[49] Markus Schmitt and Markus Heyl. "Quantum many-body dynamics in two dimensions with artificial neural networks". Physical Review Letters**125** (2020).

[50] Tiago Mendes-Santos, Markus Schmitt, and Markus Heyl. "Highly resolved spectral functions of two-dimensional systems with neural quantum states". Phys. Rev. Lett. **131**, 046501 (2023).

[51] Ashish Joshi, Robert Peters, and Thore Posske. "Quantum skyrmion dynamics studied by neural network quantum states". Physical Review B**110** (2024).

[52] Stefanie Czischek, Martin Gärttner, and Thomas Gasenzer. "Quenches near ising quantum criticality as a challenge for artificial neural networks". Phys. Rev. B **98**, 024311 (2018).

[53] Linda Mauron, Zakari Denis, Jannes Nys, and Giuseppe Carleo. "Predicting topological entanglement entropy in a rydberg analog simulator" (2024) arXiv:2406.19872.

[54] Jannes Nys, Gabriel Pescia, Alessandro Sinibaldi, and Giuseppe Carleo. "Ab-initio variational wave functions for the time-dependent many-electron schrödinger equation". Nature Communications**15** (2024).

[55] Dennis Wagner, Andreas Klümper, and Jesko Sirker. "Thermodynamics based on neural networks". Physical Review B**109** (2024).

[56] Jannes Nys, Zakari Denis, and Giuseppe Carleo. "Real-time quantum dynamics of thermal states with neural thermofields". Phys. Rev. B **109**, 235120 (2024).

[57] Filippo Vicentini, Riccardo Rossi, and Giuseppe Carleo. "Positive-definite parametrization of mixed quantum states with deep neural networks" (2022) arXiv:2206.13488.

[58] Joshua Lin, Di Luo, Xiaojun Yao, and Phiala E. Shanahan. "Real-time dynamics of the schwinger model as an open quantum system with neural density operators". Journal of High Energy Physics**2024** (2024).

[59] L. F. Shampine and C. W. Gear. "A user's view of solving stiff ordinary differential equations". SIAM Review **21**, 1–17 (1979).

[60] Kaelan Donatella, Zakari Denis, Alexandre Le Boité, and Cristiano Ciuti. "Dynamics with autoregressive neural quantum states: Application to critical quench dynamics". Phys. Rev. A **108**, 022210 (2023).

[61] Wenxuan Zhang, Bo Xing, Xiansong Xu, and Dario Poletti. "Paths towards time evolution with larger neural-network quantum states" (2024) arXiv:2406.03381.

[62] Matija Medvidović and Giuseppe Carleo. "Classical variational simulation of the quantum approximate optimization algorithm". npj Quantum Information**7** (2021).

[63] Bjarni Jónsson, Bela Bauer, and Giuseppe Carleo. "Neural-network states for the classical simulation of quantum computing" (2018).

[64] Irene López Gutiérrez and Christian B. Mendl. "Real time evolution with neural-network quantum states". Quantum **6**, 627 (2022).

[65] Matija Medvidović and Javier Robledo Moreno. "Neural-network quantum states for many-body physics". The European Physical Journal Plus**139** (2024).

[66] Hannah Lange, Anka Van de Walle, Atiye Abedinnia, and Annabelle Bohrdt. "From architectures to applications: A review of neural quantum states" (2024) arXiv:2402.09402.

[67] Naomichi Hatano and Masuo Suzuki. "Finding exponential product formulas of higher orders". Page 37–68. Springer Berlin Heidelberg. (2005).

[68] Alexander Müller-Hermes, J Ignacio Cirac, and Mari Carmen Bañuls. "Tensor network techniques for the computation of dynamical observables in one-dimensional quantum spin systems". New Journal of Physics **14**, 075003 (2012).

[69] Cleve Moler and Charles Van Loan. "Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later". SIAM Review **45**, 3–49 (2003).

[70] Vojtech Havlicek. "Amplitude ratios and neural network quantum states". Quantum **7**, 938 (2023).

[71] Eimantas Ledinauskas and Egidijus Anisimovas. "Scalable imaginary time evolution with neural network quantum states". SciPost Physics**15** (2023).

[72] Diederik P. Kingma and Jimmy Ba. "Adam: A method for stochastic optimization" (2014) arXiv:1412.6980.

[73] Reuven Y. Rubinstein and Dirk P. Kroese. "Simulation and the monte carlo method". Chapter 5. Wiley. (2016).

[74] Rajesh Ranganath, Sean Gerrish, and David Blei. "Black box variational inference". In Artificial intelligence and statistics. Pages 814–822. PMLR (2014).

[75] James Martens and Ilya Sutskever. "Training deep and recurrent networks with hessian-free optimization". Page 479–535. Springer Berlin Heidelberg. (2012).

[76] James Martens. "New insights and perspectives on the natural gradient method". Journal of Machine Learning Research **21**, 1–76 (2020). url: http://jmlr.org/papers/v21/17-678.html.

[77] Jorge Nocedal and Stephen J. Wright. "Numerical optimization". Volume 2 of Springer Series in Operations Research and Financial Engineering. Springer New York. (2006).

[78] Ran Cheng. "Quantum geometric tensor (fubini-study metric) in simple quantum system: A pedagogical introduction" (2010) arXiv:1012.1337.

[79] James Stokes, Josh Izaac, Nathan Killoran, and Giuseppe Carleo. "Quantum natural gradient". Quantum **4**, 269 (2020).

[80] Tom Heskes. "On "natural" learning and pruning in multilayered perceptrons". Neural Computation **12**, 881–901 (2000).

[81] Roger Grosse and Ruslan Salakhudinov. "Scaling up natural gradient by sparsely factorizing the inverse fisher matrix". In Francis Bach and David Blei, editors, Proceedings of the 32nd International Conference on Machine Learning. Volume 37 of Proceedings of Machine Learning Research, pages 2304–2313. Lille, France (2015). PMLR. url: https://proceedings.mlr.press/v37/grosse15.html.

[82] James Martens and Roger Grosse. "Optimizing neural networks with kronecker-factored approximate curvature". In Francis Bach and David Blei, editors, Proceedings of the 32nd International Conference on Machine Learning. Volume 37 of Proceedings of Machine Learning Research, pages 2408–2417. Lille, France (2015). PMLR. url: https://proceedings.mlr.press/v37/martens15.html.

[83] Roger Grosse and James Martens. "A kronecker-factored approximate fisher matrix for convolution layers". In Maria Florina Balcan and Kilian Q. Weinberger, editors, Proceedings of The 33rd International Conference on Machine Learning. Volume 48 of Proceedings of Machine Learning Research, pages 573–582. New York, New York, USA (2016). PMLR. url: https://proceedings.mlr.press/v48/grosse16.html.

[84] Y. Ollivier. "Riemannian metrics for neural networks i: feedforward networks". Information and Inference **4**, 108–153 (2015).

[85] Shun-ichi Amari, Ryo Karakida, and Masafumi Oizumi. "Fisher information and natural gradient learning in random deep networks". In Kamalika Chaudhuri and Masashi Sugiyama, editors, Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics. Volume 89 of Proceedings of Machine Learning Research, pages 694–702. PMLR (2019). url: https://proceedings.mlr.press/v89/amari19a.html.

[86] Ryo Karakida and Kazuki Osawa. "Understanding approximate fisher information for fast convergence of natural gradient descent in wide neural networks*". Journal of Statistical Mechanics: Theory and Experiment **2021**, 124010 (2021).

[87] Qinxun Bai, Steven Rosenberg, and Wei Xu. "A geometric understanding of natural gradient" (2022) arXiv:2202.06232.

[88] Ao Chen and Markus Heyl. "Empowering deep neural quantum states through efficient optimization". Nature Physics **20**, 1476–1481 (2024).

[89] K. B. Petersen and M. S. Pedersen. "The matrix cookbook" (2012).

[90] Riccardo Rende, Luciano Loris Viteritti, Lorenzo Bardone, Federico Becca, and Sebastian Goldt. "A simple linear algebra identity to optimize large-scale neural network quantum states". Communications Physics**7** (2024).

[91] Arthur Jacot, Franck Gabriel, and Clément Hongler. "Neural tangent kernel: convergence and generalization in neural networks". In Proceedings of the 32nd International Conference on Neural Information Processing Systems. Page 8580–8589. NIPS'18Red Hook, NY, USA (2018). Curran Associates Inc.

[92] Roman Novak, Jascha Sohl-Dickstein, and Samuel S Schoenholz. "Fast finite width neural tangent kernel". In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, Proceedings of the 39th International Conference on Machine Learning. Volume 162 of Proceedings of Machine Learning Research, pages 17018–17044.

PMLR (2022). url: https://proceedings.mlr.press/v162/novak22a.html.

[93] Julien Gacon, Jannes Nys, Riccardo Rossi, Stefan Woerner, and Giuseppe Carleo. "Variational quantum time evolution without the quantum geometric tensor". Phys. Rev. Res. **6**, 013143 (2024).

[94] James Martens. "Deep learning via hessian-free optimization". In Proceedings of the 27th International Conference on International Conference on Machine Learning. Page 735–742. ICML'10Madison, WI, USA (2010). Omnipress. url: https://dl.acm.org/doi/10.5555/3104322.3104416.

[95] Gerhard Wanner and Ernst Hairer. "Solving ordinary differential equations ii". Volume 375. Springer Berlin Heidelberg New York. (1996).

[96] Luciano Loris Viteritti, Riccardo Rende, Alberto Parola, Sebastian Goldt, and Federico Becca. "Transformer wave function for the shastry-sutherland model: emergence of a spin-liquid phase" (2023) arXiv:2311.16889.

[97] Piotr Czarnik, Jacek Dziarmaga, and Philippe Corboz. "Time evolution of an infinite projected entangled pair state: An efficient algorithm". Phys. Rev. B **99**, 035115 (2019).

[98] Filippo Vicentini, Damian Hofmann, Attila Szabó, Dian Wu, Christopher Roth, Clemens Giuliani, Gabriel Pescia, Jannes Nys, Vladimir Vargas-Calderón, Nikita Astrakhantsev, and Giuseppe Carleo. "NetKet 3: Machine Learning Toolbox for Many-Body Quantum Systems". SciPost Phys. CodebasesPage 7 (2022).

[99] Giuseppe Carleo, Kenny Choo, Damian Hofmann, James E. T. Smith, Tom Westerhout, Fabien Alet, Emily J. Davis, Stavros Efthymiou, Ivan Glasser, Sheng-Hsuan Lin, Marta Mauri, Guglielmo Mazzola, Christian B. Mendl, Evert van Nieuwenburg, Ossian O'Reilly, Hugo Théveniaut, Giacomo Torlai, Filippo Vicentini, and Alexander Wietek. "Netket: A machine learning toolkit for many-body quantum systems". SoftwareXPage 100311 (2019).

[100] Dion Häfner and Filippo Vicentini. "mpi4jax: Zero-copy mpi communication of jax arrays". Journal of Open Source Software **6**, 3419 (2021).

[101] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. "JAX: composable transformations of Python+NumPy programs". https://github.com/google/jax (2018). Version 0.3.13.

[102] Patrick Kidger and Cristian Garcia. "Equinox: neural networks in jax via callable pytrees and filtered transformations" (2021) arXiv:2111.00254.

[103] Jonathan Heek, Anselm Levskaya, Avital Oliver, Marvin Ritter, Bertrand Rondepierre, Andreas Steiner, and Marc van Zee. "Flax: A neural network library and ecosystem for JAX". https://github.com/google/flax (2024). Version 0.10.6.

[104] J.R. Johansson, P.D. Nation, and F. Nori. "QuTiP: An open-source Python framework for the dynamics of open quantum systems". Computer Physics Communications **183**, 1760–1772 (2012).

[105] J.R. Johansson, P.D. Nation, and F. Nori. "QuTiP 2: A Python framework for the dynamics of open quantum systems". Computer Physics Communications **184**, 1234–1240 (2013).

[106] Richard P. Stanley. "Enumerative combinatorics, volume 2". Volume 2 of Cambridge Studies in Advanced Mathematics. Cambridge University Press. (1999).

[107] Uri M. Ascher, Steven J. Ruuth, and Brian T. R. Wetton. "Implicit-explicit methods for time-dependent partial differential equations". SIAM Journal on Numerical Analysis **32**, 797–823 (1995).

[108] Uri M. Ascher, Steven J. Ruuth, and Raymond J. Spiteri. "Implicit-explicit runge-kutta methods for time-dependent partial differential equations". Applied Numerical Mathematics **25**, 151–167 (1997).

[109] Ruichen Li, Haotian Ye, Du Jiang, Xuelan Wen, Chuwei Wang, Zhe Li, Xiang Li, Di He, Ji Chen, Weiluo Ren, and Liwei Wang. "A computational framework for neural network-based variational monte carlo with forward laplacian". Nature Machine Intelligence (2024).

[110] Fabrizio Minganti, Adam Miranowicz, Ravindra W. Chhajlany, and Franco Nori. "Quantum exceptional points of non-hermitian hamiltonians and liouvillians: The effects of quantum jumps". Physical Review A**100** (2019).

A Details on LPE and PPE schemes

The two tables below report the coefficients for the first few orders of the LPE and PPE schemes. The details on the derivation can be found in the subsections following the tables.

o	1	2	3	4
a_1	1	$(1-i)/2$	0.6265	$0.0426-0.3946i$
a_2		$(1+i)/2$	$0.1867-0.4808i$	$0.0426+0.3946i$
a_3			$0.1867+0.4808i$	$0.4573-0.2351i$
a_4				$0.4573+0.2351i$

Table 4: Coefficients for the LPE schemes of lowest order. The sets of coefficients presented for each order are not unique: all $s!$ permutations are also solutions. Irrational numbers are reported with a precision of four decimal points. We remark that the first order scheme with a single timestep is equivalent to a standard Euler scheme.

o	2	4	6
a_1	$1/2$	$(3-\sqrt{3}i)/12$	0.2153
a_2		$(3+\sqrt{3}i)/12$	$0.1423-0.1358i$
a_3			$0.1423+0.1358i$
b_1	$-1/2$	$(-3-\sqrt{3}i)/12$	-0.2153
b_2		$(-3+\sqrt{3}i)/12$	$-0.1423-0.1358i$
b_3			$-0.1423+0.1358i$

Table 5: Coefficients for the PPE schemes of lowest order. The sets of coefficients presented for each order are not unique. All the permutations of the a_js and of b_js are also solutions leading to a total of $[(s/2)!]^2$ combinations. We remark that the second order scheme with a single substep corresponds to a simple midpoint scheme.

The two tables below report the coefficients for the first few orders of the S-LPE and S-PPE schemes.

o	1	2	3
a_1	1	$(1-i)/2$	$0.1057-0.3943i$
a_2		$(1+i)/2$	$0.3943+0.1057i$
a_3			$0.3943-0.1057i$
a_4			$0.1057+0.3943i$
α_1	1	$(1-i)/2$	$0.1057-0.3943i$
α_2		$(1+i)/2$	$0.3943+0.1057i$
α_3			$0.3943-0.1057i$
α_4			$0.1057+0.3943i$

Table 6: Coefficients for the S-LPE schemes of lowest order. We remark that $a_i = \alpha_i$. The sets of coefficients presented for each order are unique up to conjugation. The first order scheme with a single timestep is equivalent to the first order Baker–Campbell–Hausdorff expansion. For S-LPE schemes the coefficient β in Eq. (12) is always vanishing.

o	2	3	4
a_1	$1/2$	$(3+\sqrt{3}i)/12$	$(3-\sqrt{15}i)/24$
a_2		$(3-\sqrt{3}i)/12$	$1/4$
a_3			$(3+\sqrt{15}i)/24$
b_1	$-1/2$	$(-3-\sqrt{3}i)/12$	$(-3+i\sqrt{15})/24$
b_2		$(-3+\sqrt{3}i)/12$	$-1/4$
b_3			$(-3-i\sqrt{15})/24$
α_1	$1/2$	$(3+\sqrt{3}i)/12$	$(3-\sqrt{15}i)/24$
α_2	$1/2$	$1/2$	$(9-\sqrt{15}i)/24$
α_3		$(3-\sqrt{3}i)/12$	$(9+\sqrt{15}i)/24$
α_4			$(3+\sqrt{15}i)/24$

Table 7: Coefficients for the S-PPE schemes of lowest order. The sets of coefficients presented for each order are unique up to conjugation. We note that $\alpha_s = 0$ is always followed by $\alpha_{s+1} \neq 0$. This corresponds to a final diagonal operation after the sequence of optimizations.

A.1 Linear Product Expansion (LPE)

Consider the ordinary differential equation of the form in Eq. (1), where the solution $|\psi(t)\rangle$ is discretized over a set of times $\{t_n\}_{n=1,2,\dots}$, such that $|\psi_n\rangle = |\psi(t_n)\rangle$ and $t_{n+1} - t_n = \mathrm{d}t$. The LPE scheme introduced in Section 2.4 provides the following prescription for approximately updating $|\psi_n\rangle$:

$$|\psi_{n+1}\rangle = \left(\prod_{j=1}^{s} \hat{T}_{a_j}\right)|\psi_n\rangle = |\psi_n\rangle + \mathrm{d}t \sum_{j=1}^{s} a_j |\kappa_j\rangle \tag{39}$$

where

$$|\kappa_j\rangle = \hat{\Lambda}\left(|\psi_n\rangle + \mathrm{d}t \sum_{\ell=1}^{j-1} a_\ell |\kappa_\ell\rangle\right). \tag{40}$$

This expression is in direct correspondence with the evolution equations of an explicit Runge–Kutta method with update function $f(|\psi\rangle) = \hat{\Lambda}|\psi\rangle$. Although the LPE scheme can be cast as an explicit Runge-Kutta

approximation, its scalability relies on avoiding this direct interpretation. Instead, Eq. (39) is treated as a recursive process defined by

$$|\phi_k\rangle \quad \text{s.t.} \quad |\phi_k\rangle = \hat{T}_{a_k} |\phi_{k-1}\rangle \quad (k = 1, \ldots, s) \tag{41}$$

where $|\psi_{n+1}\rangle \equiv |\phi_s\rangle$, and $|\psi_n\rangle \equiv |\phi_0\rangle$. This is analogous to the formulation given in Eq. (8) in terms of transformations of the variational parameters of the wave function. As explained in the main text, each subproblem in Eq. (41) involves at most linear powers of $\hat{\Lambda}$ making its application to NQS far more practical than a direct implementation of the Taylor expansion. The numerical values of the coefficients $(a_1, \ldots, a_s)$ are obtained by Taylor expanding both sides of Eq. (10) and matching the terms order by order. This leads to to the following linear system of equations for the coefficients:

$$e_k(a_1, \ldots, a_s) = \frac{1}{k!} \tag{42}$$

for $1 \leq k \leq s$. Here, $e_k(a_1, \ldots, a_s)$ is the elementary symmetric polynomial of degree k in s variables, defined for $k \leq s$ as the sum of the products of all possible combinations of k distinct elements chosen from the set $\{a_j\}_{j=1}^s$ [106]. We solve these equations in Mathematica for $s \leq 4$ and report the solutions in Table 4. Interestingly, all coefficients a_j for which $\mathrm{Im}\{a_j\} \neq 0$ appear in complex conjugate pairs and $\mathrm{Re}\{a_j\} > 0 \; \forall j$.

A.2 Padé Product Expansion (PPE)

The PPE scheme introduced in Section 2.5 provides the following prescription for approximately updating $|\psi_n\rangle$:

$$|\psi_{n+1}\rangle = \Big(\prod_{j=1}^{s} \hat{P}_{b_j,a_j}\Big) |\psi_n\rangle = \Big(\prod_{j=1}^{s} \hat{T}_{b_j}^{-1} T_{a_j}\Big) |\psi_n\rangle \,. \tag{43}$$

While a correspondence with explicit Runge–Kutta methods could be established via the Neumann series expansion of each $\hat{T}_{b_j}^{-1}$ term, the scalability of the method relies on avoiding this expansion. Instead, Eq. (43) is treated as a recursive problem defined by

$$|\phi_k\rangle \quad \text{s.t.} \quad \hat{T}_{b_k} |\phi_k\rangle = \hat{T}_{a_k} |\phi_{k-1}\rangle \quad (k = 1, \ldots, s), \tag{44}$$

and where $|\psi_{n+1}\rangle \equiv |\phi_s\rangle$, and $|\psi_n\rangle \equiv |\phi_0\rangle$. This recursion relation can be alternatively stated as

$$|\phi_k\rangle = |\psi_n\rangle + \mathrm{d}t \sum_{j=1}^{k} b_j(-\hat{\Lambda}) |\phi_j\rangle + \mathrm{d}t \sum_{j=1}^{k} a_j \hat{\Lambda} |\phi_{j-1}\rangle \quad (k = 1, \ldots, s), \tag{45}$$

which puts the method in direct correspondence with the evolution equations of an implicit-explicit (IMEX) Runge–Kutta method with implicit and explicit update function $g(|\psi\rangle) = -\hat{\Lambda} |\psi\rangle$ and $f(|\psi\rangle) = \hat{\Lambda} |\psi\rangle$ respectively [107, 108].

As for the LPE scheme, the superiority in scalability of the method relies on avoiding this expansion and casting instead the expression onto the nested series of optimizations in Eq. (8). The numerical values of the coefficients $(a_1, \ldots, a_s, b_1, \ldots, b_s)$ are obtained by Taylor expanding both sides of Eq. (11) and matching the terms order by order. This leads to to the following linear system of equations for the coefficients:

$$\sum_{j=0}^{k} (-1)^{k-j} \, e_j(a_1, \ldots, a_s) h_{k-j}(b_1, \ldots, b_s) = 1/k! \tag{46}$$

for $1 \leq k \leq 2s$. Here, $h_{k-j}(b_1, \ldots, b_s)$ is the complete homogeneous symmetric polynomial of degree k in s variables, defined as the sum of all monomials of degree k that can be formed from the set $\{b_j\}_{j=1}^s$ allowing repetition of variables [106]. We adopt the convention that $e_0(a_1, \ldots, a_s) = h_0(b_1, \ldots, b_s) = 1$, and $e_{j>s}(a_1, \ldots, a_s) = 0$. The values of $\{a_j, b_j\}_{j=1}^s$ for $s \leq 3$ satisfying Eq. (46) are provided in Table 5. Interestingly, we note that $a_j = -b_j^*$ and that $\mathrm{Re}\{a_j\} > 0$, $\mathrm{Re}\{b_j\} < 0 \; \forall j$. As before, if a_j or b_j have nonvanishing imaginary part they appear in conjugate pairs.

A.2.1 Unitarity of Padè Product Expansions

We here show that PPE schemes are exactly (to all orders in $\mathrm{d}t$) unitary. The fundamental building blocks of PPEs are of the form

$$U_i = P_{b_i,a_i} = \hat{T}_{b_i}^{-1} \hat{T}_{a_i} = \Big(\mathbb{1} + b_i \hat{\Lambda}\Big)^{-1} \Big(\mathbb{1} + a_i \hat{\Lambda}\Big), \tag{47}$$

where the timestep $\mathrm{d}t$ has been absorbed into $\hat{\Lambda} = -i\hat{H}\mathrm{d}t$. Since $\hat{H}$ is Hermitian, $\hat{\Lambda} = -\hat{\Lambda}^\dagger$ is anti-Hermtian. It follows that

$$\begin{aligned} U_i^\dagger U_i &= \left(\mathbb{1} - a_i^*\hat{\Lambda}\right)\left(\mathbb{1} - b_i^*\hat{\Lambda}\right)^{-1}\left(\mathbb{1} + b_i\hat{\Lambda}\right)^{-1}\left(\mathbb{1} + a_i\hat{\Lambda}\right) \\ &= \left(\mathbb{1} - a_i^*\hat{\Lambda}\right)\left(\mathbb{1} + b_i\hat{\Lambda}\right)^{-1}\left(\mathbb{1} - b_i^*\hat{\Lambda}\right)^{-1}\left(\mathbb{1} + a_i\hat{\Lambda}\right), \end{aligned} \tag{48}$$

where in the last equality we used that $[(\mathbb{1}+a\hat{A})^{-1}, \mathbb{1}+\hat{A}] = 0$: a trivial consequence of the fact that $[\hat{A}, f(\hat{A})] = 0$ for any $\hat{A} \in \mathcal{H}$ and $f : \mathcal{H} \to \mathcal{H}$. From Table 5 we observe that the expansion coefficients are related by $a_i = -b_i^*$. Substituting this in Eq. (48) leads to

$$U_i^\dagger U_i = \left(\mathbb{1} + b_i\hat{\Lambda}\right)\left(\mathbb{1} + b_i\hat{\Lambda}\right)^{-1}\left(\mathbb{1} + a_i\hat{\Lambda}\right)^{-1}\left(\mathbb{1} + a_i\hat{\Lambda}\right) = \mathbb{1}. \tag{49}$$

PPE expansions of arbitrary order $o = 2s$ obtained as $\hat{U} = \prod_{i=1}^{s} U_i$ are therefore unitary, that is $\hat{U}^\dagger\hat{U} = \mathbb{1}$. To be precise, the relation $a_i = -b_i^*$ holds only for a subset of all possible solutions to Eq. (46). In general, however, the relation $a_i = -\operatorname{Re}\{b_i\} \pm \operatorname{Im}\{b_i\}$ holds. Moreover, the b_is always present in conjugate pairs so that the proof can be carried out analogously.

Analogous calculations can be applied to the S-PPE-2 scheme. Here, since all coefficients are real, the diagonal terms in $\hat{U}^\dagger\hat{U}$ cancel out, making the algorithm exactly unitary. This does not hold for S-PPE schemes of higher order for which the α_is are in general complex.

B Control variates for the conditional Monte-Carlo estimator [Eq. (21)]

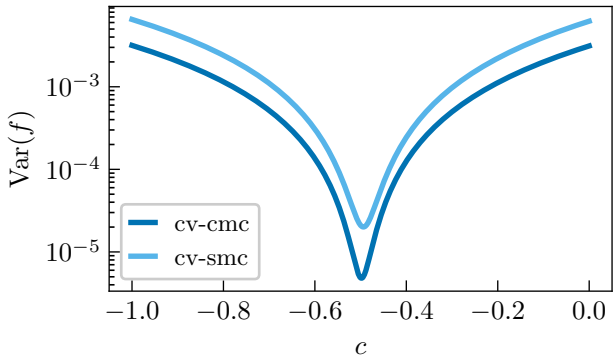

Figure 8: Variance comparison of controlled estimators Eq. (20) (cv-smc) and Eq. (22) (cv-cmc) as a function of the control coefficient c.

Control variates are particularly effective when both the control variable and the ideal control coefficient can be derived analytically. For the simple Monte Carlo estimator of the fidelity in Eq. (18), this is done in Ref. [47], where the identity $\mathbb{E}_{(x,y)\sim\pi}[|A(x,y)|^2] = 1$ is employed to reduce the variance of the estimator leading to Eq. (20). To identify an analogous control variable for the cmc estimator we can proceed in one of two ways. The first approach recognizes that the cmc estimator arises from conditioning the smc estimator, and therefore the control-variate-enhanced cmc estimator naturally follows from conditioning the cv-smc estimator. Specifically,

$$\mathcal{F}(|\psi\rangle, |\phi\rangle) = \mathbb{E}_{(x,y)\sim\pi}[F(x,y)] = \mathbb{E}_{x\sim\pi_\psi}[\mathbb{E}_{y\sim\pi_\phi}[F(x,y)|x]], \tag{50}$$

corresponds exactly to Eq. (22). The second approach observes that $|A(z)|^2$ is correlated with $H_{\mathrm{loc}}(x)$ and thus remains a valid control variable for the cmc estimator. By leveraging the separability of both $\pi(x,y) = \pi_\psi(x)\pi_\phi(y)$, and $A(x,y) = A_x(x)A_y(y)$, we can use $|A(x,y)|^2$ as control variate for the cmc estimator as well:

$$\begin{aligned} \mathcal{F}(|\psi\rangle, |\phi\rangle) &= \mathbb{E}_{x\sim\pi_\psi}\Big[\operatorname{Re}\{H_{\mathrm{loc}}\} + c\Big(\mathbb{E}_{(x,y)\sim\pi}\left[|A(x,y)|^2\right] - 1\Big)\Big], \\ &= \mathbb{E}_{x\sim\pi_\psi}\Big[\operatorname{Re}\{H_{\mathrm{loc}}\} + c\Big(|A_x(x)|^2\mathbb{E}_{y\sim\pi_\phi}\left[|A_y(y)|^2\right] - 1\Big)\Big], \end{aligned} \tag{51}$$

which again corresponds to Eq. (22). Here, $A_x(x) = \phi(x)/\psi(x)$, and $A_y(y) = \psi(y)/\phi(y)$. With a derivation similar to that in Ref. [47], we find that the optimal control coefficient c, which minimizes the variance of the cv-cmc estimator, converges to $c = -1/2$ as $|\psi\rangle \to |\phi\rangle$.

In Fig. 8 we illustrate the variance reduction achieved by applying control variates to the smc and cmc estimators. The data clearly highlight the efficiency of control variates in reducing the variance for both estimators. These calculations were performed far from ideal convergence, resulting in slight deviations from the expected optimal value of $c = -1/2$. In accordance with the Rao-Blackwell theorem, the variance of the cv-cmc estimator is consistently lower than that of the cv-smc estimator for any choice of c.

Note that in both control-variate-enhanced estimators, we retain only the real part of the original estimator, as the fidelity is known to be real. This ensures that $\mathbb{E}[\mathrm{Im}\{f\}] = 0$ for $f = A, H_{\mathrm{loc}}$. Including the imaginary part would amount to introducing an additional control variate with zero mean. However, any variability explained by the correlation between $\mathrm{Im}\{f\}$ and $\mathrm{Re}\{f\}$ is already captured through the variate $|A(x,y)|^2$. As a result, this additional control variate does not contribute to further variance reduction and is therefore omitted.

C Derivation of the gradient estimators

In this section we derive and discuss the properties of different possible Monte Carlo estimators for the gradient of the fidelity. To do in an efficient manner we introduce the notation

$$z = (x,y), \quad \pi_\psi(x) = \frac{|\psi(x)|^2}{\langle\psi|\psi\rangle}, \quad \pi(z) = \pi_\psi(x)\pi_\phi(y), \quad A(z) = \frac{\phi(x)}{\phi(y)}\frac{\psi(y)}{\psi(x)}. \tag{52}$$

In the following we consider a complex ansatz $[\psi_\theta(x) \in \mathbb{C}]$ and real parameters $(\theta \in \mathbb{R}^{N_P})$. As we have done throughout the paper, we identify the variational state as $\psi = \psi_\theta$, making implicit the dependence of the variational state $|\psi\rangle$ on the variational parameters.

C.1 Derivation of the ∇cmc gradient estimator [Eq. (25)]

We derive the ∇cmc estimator of the gradient in Eq. (25) by differentiating the cmc fidelity estimator $\mathcal{F} = \mathbb{E}_{x\sim\pi_\psi}[H_{\mathrm{loc}}(x)]$ in Eq. (21). We show that this gradient estimator can be written in the form $\boldsymbol{\nabla}\mathcal{F} = \boldsymbol{X}^\top\boldsymbol{\varepsilon}$.

Applying the chain rule to $\mathcal{F}$ yields two contributions to the gradient

$$\boldsymbol{\nabla}\mathcal{F} = \boldsymbol{\nabla}\sum_x \pi_\psi(x)H_{\mathrm{loc}}(x) = \underbrace{\sum_x H_{\mathrm{loc}}(x)\boldsymbol{\nabla}\pi_\psi(x)}_{①} + \underbrace{\sum_x \pi_\psi(x)\boldsymbol{\nabla}H_{\mathrm{loc}}(x)}_{②}. \tag{53}$$

First off, we have that

$$\boldsymbol{\nabla}\pi_\psi(x) = \boldsymbol{\nabla}\frac{\langle\psi|x\rangle\langle x|\psi\rangle}{\langle\psi|\psi\rangle} = \frac{\langle\boldsymbol{\nabla}\psi|x\rangle\langle x|\psi\rangle + \langle\psi|x\rangle\langle x|\boldsymbol{\nabla}\psi\rangle}{\langle\psi|\psi\rangle} - \frac{\langle\psi|x\rangle\langle x|\psi\rangle}{\langle\psi|\psi\rangle}\frac{\langle\boldsymbol{\nabla}\psi|\psi\rangle + \langle\psi|\boldsymbol{\nabla}\psi\rangle}{\langle\psi|\psi\rangle} = 2\pi_\psi(x)\Delta J^{\mathrm{re}}(x), \tag{54}$$

where we denote $A^{\mathrm{re}} \equiv \mathrm{Re}\{A\}$ and $A^{\mathrm{im}} \equiv \mathrm{Im}\{A\}$. It follows that

$$① = \sum_x H_{\mathrm{loc}}(x)\boldsymbol{\nabla}\pi_\psi(x) = \mathbb{E}_{x\sim\pi_\psi}[2\Delta J^{\mathrm{re}}(x)H_{\mathrm{loc}}(x)]. \tag{55}$$

We next compute

$$\boldsymbol{\nabla}H_{\mathrm{loc}}(x) = \boldsymbol{\nabla}\frac{\langle x|\hat{H}|\psi\rangle}{\langle x|\psi\rangle} = \frac{\langle x|\hat{H}|\boldsymbol{\nabla}\psi\rangle}{\langle x|\psi\rangle} - H_{\mathrm{loc}}(x)\frac{\langle x|\boldsymbol{\nabla}\psi\rangle}{\langle x|\psi\rangle} = \frac{\langle x|\hat{H}|\boldsymbol{\nabla}\psi\rangle}{\langle x|\psi\rangle} - H_{\mathrm{loc}}(x)J(x), \tag{56}$$

and note that

$$\begin{aligned}
\mathbb{E}_{\pi_\psi}\left[\frac{\langle x|\hat{H}|\boldsymbol{\nabla}\psi\rangle}{\langle x|\psi\rangle}\right] &= \sum_x \pi_\psi(x)\frac{\langle x|\hat{H}|\boldsymbol{\nabla}\psi\rangle}{\langle x|\psi\rangle} = \sum_x \pi_\psi(x)\frac{\langle x|\hat{H}\left(\sum_y |y\rangle\langle y|\right)|\boldsymbol{\nabla}\psi\rangle}{\langle x|\psi\rangle} = \sum_{x,y}\frac{\langle\psi|x\rangle\langle x|\hat{H}|y\rangle}{\langle\psi|\psi\rangle}\langle y|\boldsymbol{\nabla}\psi\rangle \\
&= \sum_y \frac{\langle\psi|\left(\sum_x |x\rangle\langle x|\right)\hat{H}|y\rangle}{\langle\psi|\psi\rangle}\langle y|\boldsymbol{\nabla}\psi\rangle = \sum_x \pi_\psi(x)\frac{\langle\psi|\hat{H}|x\rangle}{\langle\psi|x\rangle}J(x) = \sum_x \pi_\psi(x)H^*_{\mathrm{loc}}(x)J(x) \\
&= \mathbb{E}_{\pi_\psi}[H^*_{\mathrm{loc}}(x)J(x)].
\end{aligned} \tag{57}$$

so that,

$$\begin{aligned} \textcircled{2} &= \mathbb{E}_{\pi_\psi}[\boldsymbol{\nabla} H_{\mathrm{loc}}(x)] = \mathbb{E}_{\pi_\psi}\left[\frac{\langle x|\hat{H}|\boldsymbol{\nabla}\psi\rangle}{\langle x|\psi\rangle}\right] - \mathbb{E}_{\pi_\psi}[H_{\mathrm{loc}}(x)J(x)] = \mathbb{E}_{\pi_\psi}[J(x)(H^*_{\mathrm{loc}}(x) - H_{\mathrm{loc}}(x))] \\ &= -2i\,\mathbb{E}_{\pi_\psi}[J(x)H^{\mathrm{im}}_{\mathrm{loc}}(x)]. \end{aligned} \tag{58}$$

Since $\hat{H}$ is Hermitian, we know that $\langle\psi|\hat{H}|\psi\rangle = \mathbb{E}_{\pi_\psi}[H_{\mathrm{loc}}(x)] \in \mathbb{R}$ and therefore that $\mathbb{E}_{\pi_\psi}[H^{\mathrm{im}}_{\mathrm{loc}}(x)] = 0$. We can thus use that $H^{\mathrm{im}}_{\mathrm{loc}} = \Delta H^{\mathrm{im}}_{\mathrm{loc}}$ to obtain

$$\textcircled{2} = -2i\mathbb{E}_{\pi_\psi}[J(x)\Delta H^{\mathrm{im}}_{\mathrm{loc}}(x)] = -2i\mathbb{E}_{\pi_\psi}[\Delta J(x)H^{\mathrm{im}}_{\mathrm{loc}}(x)]. \tag{59}$$

Finally,

$$\begin{aligned} \boldsymbol{\nabla}\mathcal{F} &= \textcircled{1} + \textcircled{2} = \mathbb{E}_{\pi_\psi}[2\Delta J^{\mathrm{re}}(x)H_{\mathrm{loc}}(x) - 2i\Delta J(x)H^{\mathrm{im}}_{\mathrm{loc}}(x)] \\ &= \mathbb{E}_{\pi_\psi}[2\,\mathrm{Re}\{\Delta J(x)H_{\mathrm{loc}}(x)^*\}] = 2\mathbb{E}_{\pi_\psi}[\Delta J^{\mathrm{re}}_x H^{\mathrm{re}}_{\mathrm{loc}}(x) + \Delta J^{\mathrm{im}}_x H^{\mathrm{im}}_{\mathrm{loc}}(x)] \\ &= \mathbb{E}_{\pi_\psi}\left[\begin{pmatrix} \Delta J^{\mathrm{re}}(x) \\ \Delta J^{\mathrm{im}}(x) \end{pmatrix} \cdot \begin{pmatrix} 2\,\mathrm{Re}\{H_{\mathrm{loc}}(x)\} \\ 2\,\mathrm{Im}\{H_{\mathrm{loc}}(x)\} \end{pmatrix}\right]. \end{aligned} \tag{60}$$

We can now explicit the Monte-Carlo sampling of the expectation value which is in practice evaluated as

$$\boldsymbol{\nabla}\mathcal{F} \approx \frac{2}{N_s}\sum_{i=1}^{N_s} \Delta J^{\mathrm{re}}(x_i)H^{\mathrm{re}}_{\mathrm{loc}}(x_i) + \Delta J^{\mathrm{im}}(x_i)H^{\mathrm{im}}_{\mathrm{loc}}(x_i) = \frac{2}{N_s^2}\sum_{i,j=1}^{N_s} \Delta J^{\mathrm{re}}(x_i)A^{\mathrm{re}}(x_i,y_j) + \Delta J^{\mathrm{im}}(x_i)A^{\mathrm{im}}(x_i,y_j), \tag{61}$$

with N_s the number of samples. Note that the second expression follows from the fact that the local estimator itself is evaluated with MC sampling as

$$H_{\mathrm{loc}}(x) = \frac{\phi(x)}{\psi(x)}\frac{1}{N_s}\sum_{j=1}^{N_s}\frac{\psi(y_j)}{\phi(y_j)}. \tag{62}$$

We now want to express the above in a form compatible with NTK and automatic damping strategies. To do so, we define

$$\boldsymbol{X} = \begin{pmatrix} \mathrm{Re}\{\boldsymbol{O}\} \\ \mathrm{Im}\{\boldsymbol{O}\} \end{pmatrix} \in \mathbb{R}^{2N_s\times N_p}, \qquad \boldsymbol{O} = \frac{1}{\sqrt{N_s}}\begin{pmatrix} \Delta J(x_1)^\top \\ \vdots \\ \Delta J(x_{N_s})^\top \end{pmatrix} \in \mathbb{C}^{N_s\times N_p}. \tag{63}$$

In this way, the Monte Carlo estimate of the Fubini-Study metric tensor reads $\boldsymbol{S} = \boldsymbol{X}^\top\boldsymbol{X}$, and the neural tangent kernel $\boldsymbol{T} = \boldsymbol{X}\boldsymbol{X}^\top$. We then define the complex-valued local energy vector as

$$\boldsymbol{f} = \frac{2}{\sqrt{N_s}}\Big(H_{\mathrm{loc}}(x_1),\dots,H_{\mathrm{loc}}(x_{N_s})\Big) \in \mathbb{C}^{N_s}, \tag{64}$$

and

$$\boldsymbol{\varepsilon} = \begin{pmatrix} \mathrm{Re}\{\boldsymbol{f}\} \\ \mathrm{Im}\{\boldsymbol{f}\} \end{pmatrix} \in \mathbb{R}^{2N_s}. \tag{65}$$

It is easy to see that we can express Eq. (61) as $\boldsymbol{\nabla}\mathcal{F} = \boldsymbol{X}^\top\boldsymbol{\varepsilon}$.

C.2 Derivation of the ∇cv-smc gradient estimator [Eq. (26)]

We derive the ∇cv-smc estimator of the gradient in Eq. (26) by differentiating the cv-smc fidelity estimator $\mathcal{F} = \mathbb{E}_{z\sim\pi}[F(z)] = \mathbb{E}_{z\sim\pi}[\mathrm{Re}\,A(z) + c(|A(z)|^2 - 1)]$ in Eq. (20). We show that this gradient estimator does not admit the form $\boldsymbol{\nabla}\mathcal{F} = \boldsymbol{X}^\top\boldsymbol{\varepsilon}$ and is therefore unsuited for NTK calculations.

Once again

$$\boldsymbol{\nabla}\mathcal{F} = \boldsymbol{\nabla}\sum_z \pi(z)F(z) = \underbrace{\sum_z F(z)\boldsymbol{\nabla}\pi(z)}_{\textcircled{1}} + \underbrace{\sum_z \pi(z)\boldsymbol{\nabla}F(z)}_{\textcircled{2}}. \tag{66}$$

Since $\boldsymbol{\nabla}\pi = \pi_\phi \boldsymbol{\nabla}\pi_\psi$, Eq. (54) gets us ① $= \mathbb{E}_{z\sim\pi}[2\Delta F(z) J^{\mathrm{re}}(x)]$. Differentiating $A(z)$ yields $\boldsymbol{\nabla}A(z) = A(z)[J(y) - J(x)]$, so that

$$\boldsymbol{\nabla}F(z) = \frac{\boldsymbol{\nabla}A(z) + \boldsymbol{\nabla}A^*(z)}{2} + c\Big(A(z)\boldsymbol{\nabla}A^*(z) + A^*(z)\boldsymbol{\nabla}A(z)\Big) = \mathrm{Re}\Big\{\Big(A(z) + 2c|A(z)|^2\Big)\Big(J(y) - J(x)\Big)\Big\}. \quad (67)$$

Inserting this into the expression for ② leads to

$$② = \mathbb{E}_{z\sim\pi}\Big[\Big(A^{\mathrm{re}}(z) + 2c|A(z)|^2\Big)\Big(J^{\mathrm{re}}(y) - J^{\mathrm{re}}(x)\Big) + A^{\mathrm{im}}(z)\Big(J^{\mathrm{im}}(x) - J^{\mathrm{im}}(y)\Big)\Big]. \quad (68)$$

The gradient is then found to be

$$\boldsymbol{\nabla}_\theta \mathcal{F} = ① + ② = \mathbb{E}_{z\sim\pi}\left[\begin{pmatrix} J^{\mathrm{re}}(x) \\ J^{\mathrm{im}}(x) \\ J^{\mathrm{re}}(y) \\ J^{\mathrm{im}}(y) \end{pmatrix} \cdot \begin{pmatrix} 2\Delta F(z) - A^{\mathrm{re}}(z) - 2c|A(z)|^2 \\ A^{\mathrm{im}}(z) \\ A^{\mathrm{re}}(z) + 2c|A(z)|^2 \\ -A^{\mathrm{im}}(z) \end{pmatrix}\right]. \quad (69)$$

We remark that this estimator evaluates the Jacobian of ψ not only on the samples of ψ as we would normally expect, but on those of ϕ as well. Equation (69) makes it manifest that this estimator cannot be expressed in the form $\boldsymbol{\nabla}\mathcal{F} = \boldsymbol{X}^\top \boldsymbol{\varepsilon}$.

C.3 Derivation of the ∇smc gradient estimator [Eq. (24)]

We derive the ∇smc estimator of the gradient $\mathcal{F} = \mathbb{E}_{(x,y)\sim\pi}[A(x,y)]$ in Eq. (24) and show that it admits the form $\boldsymbol{\nabla}\mathcal{F} = \boldsymbol{X}^\top\boldsymbol{\varepsilon}$.

One straightforward approach is to reverse the marginalization yielding the ∇cmc gradient estimator [Eq. (25)]

$$\begin{aligned}
\boldsymbol{\nabla}_\theta \mathcal{F} &= \mathbb{E}_{\pi_\psi}[2\,\mathrm{Re}\{\Delta J(x)^* H_{\mathrm{loc}}(x)\}] = 2\,\mathrm{Re}\Big\{\sum_x \pi_\psi(x)\Delta J^*(x) H_{\mathrm{loc}}(x)\Big\} \\
&= 2\,\mathrm{Re}\Big\{\sum_x \pi_\psi(x)\Delta J(x)^* \frac{\phi(x)}{\psi(x)} \sum_y \pi_\phi(y)\frac{\psi(y)}{\phi(x)}\Big\} = \mathbb{E}_{z\sim\pi}[2\,\mathrm{Re}\{\Delta J(x) A(z)^*\}] \\
&= \mathbb{E}_{z\sim\pi}\left[\begin{pmatrix} \Delta J^{\mathrm{re}}(x) \\ \Delta J^{\mathrm{im}}(x) \end{pmatrix} \cdot \begin{pmatrix} 2\,\mathrm{Re}\{A(z)\} \\ 2\,\mathrm{Im}\{A(z)\} \end{pmatrix}\right].
\end{aligned} \quad (70)$$

In practice, the expectation value above is evaluated using MC sampling as

$$\boldsymbol{\nabla}\mathcal{F} \approx \frac{1}{N_s}\sum_{i=1}^{N_s} \Delta J^{\mathrm{re}}(x_i)\,\mathrm{Re}\{A(x_i,y_i)\} + \Delta J^{\mathrm{im}}(x_i)\,\mathrm{Im}\{A(x_i,y_i)\}, \quad (71)$$

which makes manifest the possibility of expressing the gradient as $\boldsymbol{\nabla}\mathcal{F} = \boldsymbol{X}\varepsilon$ with

$$\boldsymbol{f} = \frac{2}{\sqrt{N_s}}\Big(A(x_1,y_1),\ldots,A(x_{N_s},y_{N_s})\Big) \in \mathbb{C}^{N_s}, \quad (72)$$

and

$$\boldsymbol{\varepsilon} = \begin{pmatrix} \mathrm{Re}\{\boldsymbol{f}\} \\ \mathrm{Im}\{\boldsymbol{f}\} \end{pmatrix} \in \mathbb{R}^{2N_s}. \quad (73)$$

Although this derivation would suffice, it is still insightful to explore how the same expression can be obtained by manipulating the ∇cv-smc estimator [Eq. (26), or equivalently Eq. (69)]. The value in this alternative derivation stems from the properties of $A(z)$ that this approach reveals. We believe these properties could be useful in the future to derive control variables tailored to the gradient. For starters, we find that

$$\mathbb{E}_{z\sim\pi}\Big[|A(z)|^2 f(x)\Big] = \mathbb{E}_{y\sim\pi_\phi}[f(y)]. \quad (74)$$

Using this identity, one can easily show that

$$\mathbb{E}_{z\sim\pi}[A^*(z)f(x)] = \mathbb{E}_{z\sim\pi}[A(z)f(y)], \quad (75)$$
$$\mathbb{E}_{z\sim\pi}[A^*(z)f(y)] = \mathbb{E}_{z\sim\pi}[A(z)f(x)], \quad (76)$$
$$\mathbb{E}_{z\sim\pi}[A(z)f(x)] = \mathbb{E}_{z\sim\pi}[A^*(z)f(y)], \quad (77)$$
$$\mathbb{E}_{z\sim\pi}[A(z)f(y)] = \mathbb{E}_{z\sim\pi}[A^*(z)f(x)]. \quad (78)$$

Again, these equations can be combined to show that

$$\mathbb{E}_{z\sim\pi}[f(x)\,\mathrm{Re}\{A(z)\}] = \mathbb{E}_{z\sim\pi}[f(y)\,\mathrm{Re}\{A(z)\}], \tag{79}$$
$$\mathbb{E}_{z\sim\pi}[f(x)\,\mathrm{Im}\{A(z)\}] = -\mathbb{E}_{z\sim\pi}[f(y)\,\mathrm{Im}\{A(z)\}]. \tag{80}$$

Substitution into Eq. (69) yields Eq. (70).

D Covariance structure, Rao-Blackwellization, and variance reduction

We use this appendix to expand on the differences in variance exhibited by the gradient estimators presented in the main text. First off, we note that while both the ∇cmc and ∇smc estimators take the form of a covariance estimator [c.f. Eqs. (60) and (70)], the ∇cv-smc estimator does not. Generally, estimators structured as covariances tend to exhibit lower variance largely due to the inherent variance reduction from centering each component. Specifically, the sample covariance used to estimate the covariance between two random variables X and Y is given by

$$\mathrm{Cov}(X,Y) = \mathbb{E}[(X-\mathbb{E}[X])(Y-\mathbb{E}[Y])] \approx \frac{\gamma}{N_s}\sum_{i=1}^{N_s}\left(X_i - \frac{1}{N_s}\sum_{j=1}^{N_s}X_j\right)\left(Y_i - \frac{1}{N_s}\sum_{j=1}^{N_s}Y_j\right), \tag{81}$$

where $\gamma = N_s/(N_s-1)$ is the Bessel correction factor used to remove the bias introduced by centering. The centering process effectively provides sample-specific control variates that automatically adjust to the non-stationarity of the variables, enhancing estimator stability, and justifying the results presented in the main text and further corroborated by Fig. 9.

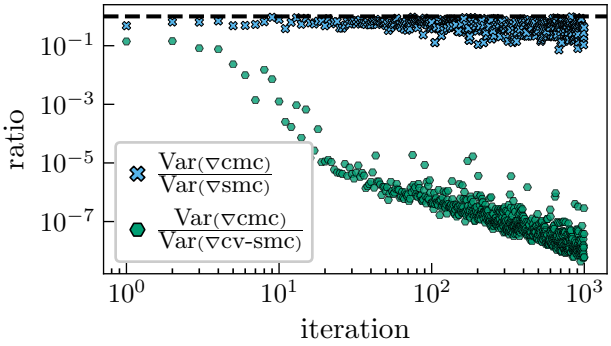

Figure 9: Comparison of the variance of the different estimators of the fidelity gradient. As the gradient is a vector in $\mathbb{R}^{N_p}$, for each estimator we consider the a single direction in parameter space. The results are independent of the chosen directions. Statistics are computed on the same data of Fig. 3.

While the poor performance of the ∇cv-smc estimator is attributed to its lack of a covariance structure, the smaller performance difference between the ∇cmc and ∇smc estimators can be understood through the Rao-Blackwell theorem. Identical arguments apply for the cmc and smc estimators of the fidelity itself. In the simplest setting, Rao-Blackwellization replaces a function of two random variables with its conditional expectation resulting in a new estimator whose variance is no greater than that of the original function (Rao-Blackwell theorem). Consider two random variables, X and Y, and a function $J(X,Y)$. Our goal is to compute its expectation $\mathbb{E}[J(X,Y)]$ with respect to the joint distribution of X and Y. To do so, we define [74]

$$\mathbb{E}[J(X,Y)\mid X] = \sum_y J(x,y)\cdot P(Y=y\mid X=x) \equiv \hat{J}(X) \tag{82}$$

and note that $\mathbb{E}[\hat{J}(X)] = \mathbb{E}[J(X,Y)]$. This implies that $\hat{J}(X)$ can be used in place of $J(X,Y)$ in a Monte Carlo approximation of $\mathbb{E}[J(X,Y)]$. The variance of $\hat{J}(X)$ can be easily computed to be

$$\mathrm{Var}(\hat{J}(X)) = \mathrm{Var}(J(X,Y)) - \mathbb{E}[(J(X,Y)-\hat{J}(X))^2], \tag{83}$$

from which it follows that $\mathrm{Var}(\hat{J}(X)) \le \mathrm{Var}(J(X,Y))$. To compute Rao-Blackwellized estimators, we generally need to calculate conditional expectations. In many cases, this cannot be done exactly; however, under a

mean-field assumption, the conditional expectation simplifies due to factorization:

$$\hat{J}(X) = \mathbb{E}[J(X,Y) \mid X] = \sum_y J(x,y) \cdot P(Y=y \mid X=x) = \sum_y J(x,y) \cdot P(Y=y) = \mathbb{E}_Y[J(X,Y)], \tag{84}$$

where we used that $P(X,Y) = P(X)P(Y)$ and thus $P(Y \mid X) = P(Y)$. Therefore, to construct a lower variance estimator when the joint distribution factorizes, all we need to do is integrate out some variables and use this marginal as the estimator.

E Lowering sampling cost by reweighting

Direct evaluation of the fidelity (and of its gradient) between two transformed states $|\tilde{\psi}\rangle = \hat{V}|\psi\rangle$ and $|\tilde{\phi}\rangle = \hat{U}|\phi\rangle$ requires, in principle, sampling from their Born distributions $\pi_{\tilde{\psi}}(x)$ and $\pi_{\tilde{\phi}}(y)$, respectively. This process introduces a computational overhead that scales with N_c, the number of connected elements in the transformations. For local spin Hamiltonians, this introduces a sampling overhead proportional to the system size N, which often becomes the dominant cost (in the simulations presented in Section 4, for instance, around 90% of the computational time is spent sampling). The computational burden grows substantially for other Hamiltonians, such as for those arising in natural-orbital chemistry or for the kinetic term in first-quantisation formulations [109].

To address this overhead, one can resort to self-normalized importance sampling [73], adapting the estimators discussed in Section 3 to sample directly from the bare distributions. In line with the procedure outlined in Ref. [47], we avoid direct sampling from the target state. While previous works were concerned with unitary transformations applied to the target state alone, here we extend the approach to arbitrary transformations that act on both the target and variational states.

As all estimators discussed in Section 3 are of this form, it is convenient to write down the reweighting procedure just once for the generic estimator $\mathbb{E}_{\sigma\sim\chi}[f(\sigma)]$. In this case, we have [73]

$$\mathbb{E}_{\sigma\sim\chi}[f(\sigma)] = \frac{\mathbb{E}_{\sigma\sim\eta}[w(\sigma)f(\sigma)]}{\mathbb{E}_{\sigma\sim\eta}[w(\sigma)]}, \tag{85}$$

where $w = \eta/\chi$. Specializing to the case where we sample from a joint distribution, e.g. in Eqs. (18), (20), (24) and (26), yields

$$\mathbb{E}_{(x,y)\sim\tilde{\pi}}[f(x,y)] = \frac{\sum_{x,y} \pi(x,y)\frac{\tilde{\pi}(x,y)}{\pi(x,y)} f(x,y)}{\sum_{x,y} \pi(x,y)\frac{\tilde{\pi}(x,y)}{\pi(x,y)}}, \tag{86}$$

where $\tilde{\pi}(x,y) = \pi_{\tilde{\psi}}(x)\pi_{\tilde{\phi}}(y)$. Applying Rao-Blackwellization to this estimator is still possible and results in

$$\mathbb{E}_{(x,y)\sim\tilde{\pi}}[f(x,y)] = \mathbb{E}_{x\sim\tilde{\pi}_\psi}[f_x(x)]\mathbb{E}_{y\sim\tilde{\pi}_\phi}[f_y(y)] = \frac{\sum_x \pi_\psi(x)\frac{\tilde{\pi}_\psi(x)}{\pi_\psi(x)} f_x(x)}{\sum_x \pi_\psi(x)\frac{\tilde{\pi}_\psi(x)}{\pi_\psi(x)}} \frac{\sum_y \pi_\phi(y)\frac{\tilde{\pi}_\phi(y)}{\pi_\phi(y)} f_y(y)}{\sum_y \pi_\phi(y)\frac{\tilde{\pi}_\phi(y)}{\pi_\phi(y)}}, \tag{87}$$

where we made use of the separability of the Born distributions, and of the integrator $f(x,y) = f_x(x)f_y(y)$ which we have observed in Section 3.

F Machine precision on small systems, the limitation of Monte Carlo sampling, and the importance of curvature

We now demonstrate the theoretical possibility of solving infidelity-based optimizations to machine precision. Specifically, we revisit the quench dynamics studied in Section 4.2, involving a quench from $h = \infty$ to $h = h_c/10$, but now on a smaller 4×4 lattice—small enough to allow exact summation over the entire Hilbert space.

In Fig. 10(a-b), we show the variational p-tVMC dynamics computed by evaluating all statistical averages relevant to the optimization exactly. These results exhibit perfect agreement with the exact solution, with optimizations converging to the target state within machine precision.

By contrast, performing the same simulations using Monte Carlo sampling—i.e., evaluating statistical averages over a finite number of samples—yields results similar to those shown in Section 4.1. While convergence remains qualitatively good, it no longer reaches machine precision. As discussed in Section 3.3.1, this discrepancy is likely due to poor estimation of the gradient and curvature matrix when using limited sample sizes, which leads

to unreliable updates. Indeed, increasing the number of samples steadily improves the results, bringing them closer to the exact, noiseless solution. Achieving results comparable to full-summation, however, appears to require a number of samples on the order of the Hilbert space dimension itself. This is illustrated in Fig. 10(c), where we display optimization trajectories for the following infidelity minimization problem:

$$\theta^* = \underset{\theta}{\operatorname{argmin}}\ \mathcal{I}(|\psi_t\rangle, |\psi_\theta\rangle). \tag{88}$$

The initial parameter configuration corresponds to the state at time $Jt = 1.25$, obtained from the noiseless p-tVMC simulation shown in Fig. 10(a-b). The target state $|\psi_t\rangle$ at time $Jt = 1.3$, by contrast, is taken from the exact evolution of the TFIM dynamics and encoded exactly, using one complex-valued parameter per entry of the state vector.

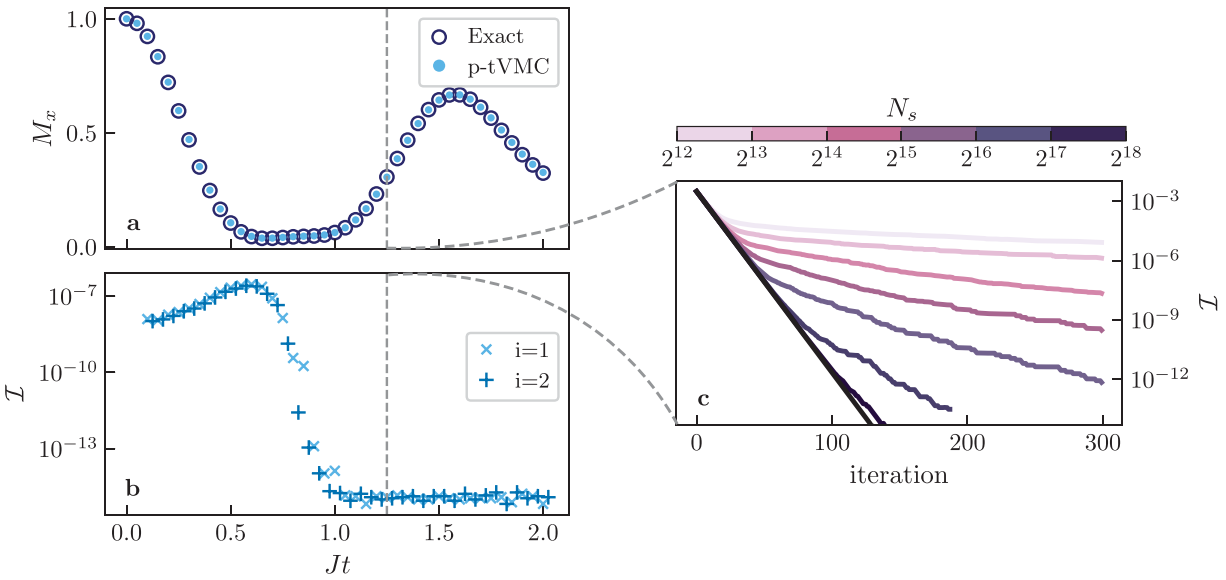

Figure 10: Quenched dynamics ($h = \infty \to h_c/10$) on a 4×4 lattice. (a,b) Average magnetization and optimization infidelity as a function of time. The variational results (full dots) are obtained in full summation using the S-LPE-2 scheme with $\mathrm{d}t = 0.05$. We compare this to the exact calculation (open circles). (c) Optimization profiles for the update $Jt = 1.25 \to 1.30$ for different sample sizes N_s. The convergence is negatively impacted by the lack a number of samples sufficient to properly reconstruct the curvature matrix and resolve the ideal descent direction. Stochastic optimization achieves performance compatible with full summation (black line) only for very large sample sizes, comparable or superior to the size of the Hilbert space. The regularization coefficient λ is fixed to $\lambda = 10^{-8}$ for $N_s = \infty, 2^{18}, 2^{17}, 2^{16}$. Smaller sample sizes require stronger regularization. We use $\lambda = 10^{-7}$ for $N_s = 2^{15}, 2^{14}$ and $\lambda = 10^{-6}$ for $N_s = 2^{13}, 2^{12}$. Parameters: $\alpha = 0.05$, $\mathbf{\Theta}_{\mathrm{CNN}} = (10, 10, 10, 10; 3)$.

We now use this simple toy problem to provide a clear illustration of the importance of incorporating curvature information into the optimization strategy to ensure reliable convergence. To isolate the effect of the optimization method from other sources of error—such as Monte Carlo noise and model expressivity—we consider an idealized setting in which all statistical averages required to compute the objective's gradient and curvature matrix are evaluated exactly by summing over the entire Hilbert space at each optimization step. Moreover, as shown in Fig. 10, the chosen CNN model with $\mathbf{\Theta}_{\mathrm{CNN}} = (10, 10, 10, 10; 3)$ is able to represent the target state at time $Jt = 1.3$ with machine precision accuracy. In Fig. 11, we compare the performance of NGD and Adam on the same state-compression task studied in Fig. 10(c). The results show that Adam converges significantly more slowly and often becomes trapped in local minima, failing to accurately compress the target state. By contrast, NGD achieves rapid and reliable convergence. To further explore the role of curvature information, we modulate the damping parameter λ, which interpolates between NGD and standard gradient descent. For large values of λ, the curvature information is heavily suppressed by regularization, and the behavior approaches that of vanilla gradient descent or Adam. As λ is decreased, curvature plays a more prominent role in shaping the optimization path, culminating in exponential convergence for $\lambda = 10^{-4}$. Overall, this toy problem highlights the critical role of NGD in learning large, global transformations of the quantum state, especially in regimes where precise convergence is required.

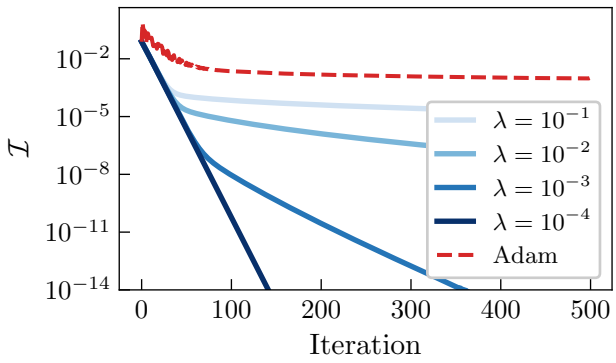

Figure 11: Performance of NGD with varying damping parameters λ, and of Adam on the same state-compression problem investigate in Fig. 10(c). All gradients and curvature matrices are computed exactly. Parameters: $\alpha_{\text{ngd}} = 0.05$, $\boldsymbol{\Theta}_{\text{CNN}} = (10, 10, 10, 10; 3)$, $\alpha_{\text{adam}} = 10^{-4}$. Note that larger values of the learning rate for Adam are unstable and result in divergences in the objective. Different warm up strategies progressively raising and then lowering the learning rate yield no visible improvement over the presented results.

G Exact application of diagonal operators

Let $|\psi_\theta\rangle = \sum_x \psi_\theta(x)\,|x\rangle$ be a variational state and $\hat{A} = \sum_{x,y} A_{xy}\,|x\rangle\langle y|$ a generic operator acting on it. We want to find a way to reduce the application of $\hat{A}$ on $|\psi_\theta\rangle$ to a change of parameters $\theta \to \theta'$. In other words, we want to find θ' such that

$$|\psi_{\theta'}\rangle = \hat{A}\,|\psi_\theta\rangle\,. \tag{89}$$

If successful, this procedure would allow us to apply $\hat{A}$ on $|\psi_\theta\rangle$ exactly and at no computational expense. Unfortunately, for a generic parametrization of the state, or a generic operator, this is not possible. The problem greatly simplifies if we restrict to diagonal operators of the form $\hat{A} = \sum_x A_x\,|x\rangle\langle x|$, whose application on a generic variational state reads $\hat{A}\,|\psi_\theta\rangle = \sum_x A_x \psi_\theta(x)\,|x\rangle$. Even in this simple case, the transformation $\theta \to \theta'$ satisfying $|\psi_{\theta'}\rangle = \hat{A}\,|\psi_\theta\rangle$ is not guaranteed to exist. Consider, however, the improved ansatz $|\psi_{\theta,\phi}\rangle = \sum_x \psi_{\theta,\phi}(x)\,|x\rangle$ where

$$\psi_{\theta,\phi}(x) = (A_x)^\phi \psi_\theta(x) \quad \text{with} \quad \phi \in \mathbb{C}. \tag{90}$$

The application of $\hat{A}$ on this state reads

$$\hat{A}\,|\psi_{\theta,\phi}\rangle = \sum_x A_x \psi_{\theta,\phi}(x)\,|x\rangle = \sum_x (A_x)^{\phi+1} \psi_\theta(x)\,|x\rangle = |\psi_{\theta,\phi+1}\rangle \tag{91}$$

The action of $\hat{A}$ on $|\psi_{\theta,\phi}\rangle$ can thus be *exactly* reduced to the parameter transformation $(\theta, \phi) \to (\theta, \phi+1)$ which can be computed at virtually no computational expense. Note that while the additional multiplicative layer has a structure determined by $\hat{A}$, the network $\psi_\theta(x)$ to which this layer is added is completely arbitrary.

This procedure can be easily generalized to multiple diagonal operations which, of course, commute with each other. Given $\hat{A} = \sum_x A_x\,|x\rangle\langle x|$ and $\hat{B} = \sum_x B_x\,|x\rangle\langle x|$, we define the improved ansatz as

$$\psi_{\theta,\phi_A,\phi_B}(x) = (B_x)^{\phi_B} (A_x)^{\phi_A} \psi_\theta(x). \tag{92}$$

The application of $\hat{A}$ without $\hat{B}$ is equivalent to $(\theta, \phi_A, \phi_B) \to (\theta, \phi_A + 1, \phi_B)$. The application of $\hat{B}$ without $\hat{A}$ is equivalent to $(\theta, \phi_A, \phi_B) \to (\theta, \phi_A, \phi_B + 1)$. The simultaneous application of $\hat{A}$ and $\hat{B}$ is equivalent to the transformation $(\theta, \phi_A, \phi_B) \to (\theta, \phi_A + 1, \phi_B + 1)$. Note that we never act on the network itself (θ is never changed).

G.1 ZZ-operations

Let $|\psi_\theta\rangle$ be an arbitrary ansatz for the state of a system of N spin-1/2 particles, and

$$\hat{A} = \exp\{\alpha \hat{\sigma}^z_\mu \hat{\sigma}^z_\nu\} = \sum_x e^{\alpha\, x_\mu x_\nu}\,|x\rangle\langle x| \equiv \sum_x A_x\,|x\rangle\langle x| \quad \text{with} \quad \mu, \nu \in [1, \ldots, N]. \tag{93}$$

the ZZ-operation acting on spins μ and ν. To encode the action of $\hat{A}$ as a change of parameters we define the improved ansatz

$$\psi_{\theta,\phi}(x) = (A_x)^\phi \psi_\theta(x) = e^{\alpha\phi\, x_\mu x_\nu} \psi_\theta(x) \equiv e^{\phi\, x_\mu x_\nu} \psi_\theta(x), \tag{94}$$

where, in the last equality, we absorb α in the parameter ϕ without loss of generality. As expected, $|\psi_{\theta,\phi}\rangle \to \hat{A}\,|\psi_{\theta,\phi}\rangle \equiv (\theta,\phi) \to (\theta,\phi+1)$. This ansatz accounts for the application of ZZ-operations on a fixed pair of spins, namely spin μ and ν. For this reason the additional parameter ϕ is a scalar. In general, however, we want to reserve the right to apply the operation between any pair of spins and/or on multiple pairs simultaneously.

Consider now the application of ZZ-rotations on two pairs of spins: (μ,ν) and (μ',ν'). Said differently, we want to find an ansatz to incorporate the action of $\hat{A} = \exp\{\alpha_{\mu\nu}\hat{\sigma}^z_\mu\hat{\sigma}^z_\nu\}$, of $\hat{B} = \exp\{\alpha_{\mu'\nu'}\hat{\sigma}^z_{\mu'}\hat{\sigma}^z_{\nu'}\}$, and of their product AB, as a simple change of parameters [6]. To do so we can incorporate the single-operator ansatz from Eq. (94) into the two-operator structure in Eq. (92) as

$$\psi_{\theta,\boldsymbol{\phi}}(x) = (B_x)^{\phi_{\mu'\nu'}}(A_x)^{\phi_{\mu\nu}}\psi_\theta(x) = \exp\{x_\mu\phi_{\mu\nu}x_\nu\}\exp\{x_{\mu'}\phi_{\mu'\nu'}x_{\nu'}\}\psi_\theta(x) = \exp\{x_\mu\phi_{\mu\nu}x_\nu + x_{\mu'}\phi_{\mu'\nu'}x_{\nu'}\}\psi_\theta(x). \tag{95}$$

Note that now $\boldsymbol{\phi} = (\phi_{\mu\nu},\phi_{\mu'\nu'})$ is a two-dimensional vector. This can be further generalized to account for ZZ-operations between any two spins via the ansatz

$$\psi_{\theta,\boldsymbol{\phi}}(x) = \exp\Big\{\sum_{ij} x_i\phi_{ij}x_j\Big\}\psi_\theta(x) = \exp\{\boldsymbol{x}\boldsymbol{\phi}\boldsymbol{x}^\top\}\psi_\theta(x). \tag{96}$$

Note that the multiplicative layer added to the network is exactly a two-body Jastrow ansatz where $\boldsymbol{\phi} = [\phi_{\mu\nu}]$ is an $N\times N$ matrix. Application of the operator $\hat{A}_{\mu\nu} = \exp\{\alpha_{\mu\nu}\hat{\sigma}^z_\mu\hat{\sigma}^z_\nu\}$ is equivalent to the parameter transformation $\phi_{\mu\nu} \to \phi_{\mu\nu} + \alpha_{\mu\nu}$. In general we parameterize the log-amplitude of the wave function $\log\psi_{\theta,\boldsymbol{\phi}}(x) = \boldsymbol{x}\boldsymbol{\phi}\boldsymbol{x}^\top + \log\psi_\theta(x)$, so that the multiplicative layer actually becomes an additive one.

H Neural network architectures

In this section we review the two architectures used in this work.

H.1 Convolutional neural networks

Convolutional Neural Networks (CNNs) are particularly well-suited for processing and analyzing grid-like data, such as images or quantum systems on a lattice. The architecture of a CNN consists of multiple layers indexed by $\ell \in [1, N_L]$, typically structured as alternating non-linear and affine transformations. For a system defined on a square lattice of linear length L and number of particles $N = L^2$, the input layer ($\ell = 1$) receives the configuration vector $x = (s_1,\dots,s_N)$, which is reshaped into an $L\times L$ matrix as $X = \mathrm{vec}^{-1}(x)$, where vec represents the vectorization operation [110].

The building block of CNNs is the convolutional layer, where filters (or kernels) are applied to local regions of the input, learning spatial hierarchies of features. This is analogous to performing convolution operations over a lattice, capturing local correlations across the system. Let the output of the ℓ-th layer be

$$X^{(\ell)} = [X^{(\ell)}]^\alpha_{i,j} \in \mathbb{C}^{C_\ell}\otimes\mathbb{C}^{H_\ell\times W_\ell} \quad \text{with} \quad \begin{cases} i \in [0, H_\ell - 1], \\ j \in [0, W_\ell - 1], \\ \alpha \in [0, C_\ell - 1] \end{cases} \tag{97}$$

where (H_ℓ, W_ℓ) are the height and width of the processed data at layer step ℓ, and C_ℓ is the number of channels. The convolution operation yielding the data structure of the $(\ell+1)$-th layer is

$$\Big(X^{(\ell+1)}\Big)^\beta_{m,n} = \sigma_\ell\Big([X^{(\ell)} \circledast F^{(\ell)}]^\beta_{m,n}\Big) = \sigma_\ell\left(\sum_\alpha\sum_{i,j}[F^{(\ell)}]^{\alpha\beta}_{i,j}\,[X^{(\ell)}]^\alpha_{m+i,\,n+j}\right) \quad \text{with} \quad \begin{cases} i \in [0, h_\ell - 1], \\ j \in [0, w_\ell - 1], \\ \beta \in [0, c_\ell - 1] \end{cases} \tag{98}$$

where σ_ℓ is the activation function, (h_ℓ, w_ℓ) is the size of the convolutional kernel $F^{(\ell)}$, and c_ℓ is its output dimension.

For the activation functions, we follow the approach in Ref. [49]. The activation function in the first layer conists of the first three non-vanishing terms of the series expansion of $\mathrm{logcosh}(z)$, ensuring the incorporation of the system's $\mathbb{Z}_2$ symmetry in the absence of bias in the first layer. It is defined as

$$\sigma_1(z) = \frac{z^2}{2} - \frac{z^4}{12} + \frac{z^6}{45}. \tag{99}$$

[6] Here $\alpha_{\mu\nu}\in\mathbb{C}$ is the phase of the gate operation acting on the pair (μ,ν).

In subsequent layers, its derivative is used

$$\sigma_{\ell>1}(z) = \frac{z}{2} - \frac{z^3}{3} + \frac{2}{15}z^5. \tag{100}$$

We use circular padding to respect periodic boundary conditions, ensuring moreover that the spatial dimensions remain constant across layers, $H_\ell = W_\ell = L$. Both dilation and stride are set to one across all layers, and a fixed kernel size $h_\ell = w_\ell = k$ is used. A final dense layer is used to pool the output of the CNN into a scalar output. After the convolutional layers, a fully connected layer reduces the output of the convolutional sequence to a scalar.

The CNN structure can is summarized by a tuple $\boldsymbol{\Theta}_{\text{CNN}} = (c_1, \ldots, c_{N_L}; k)$, where c_ℓ represents the number of channels in each layer, and k denotes the kernel size. A sketch of the architecture is shown in Fig. 12.

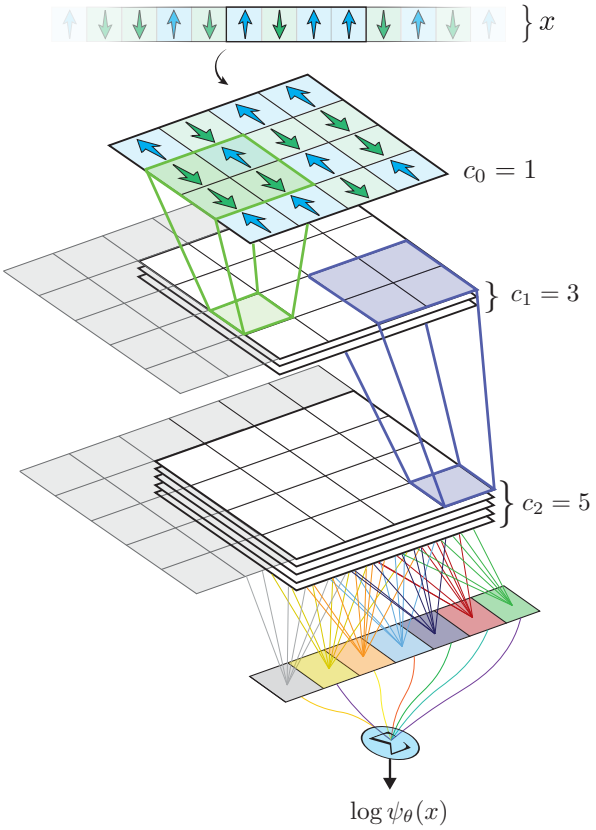

Figure 12: Illustrative representation of the CNN architecture described in Appendix H.1.

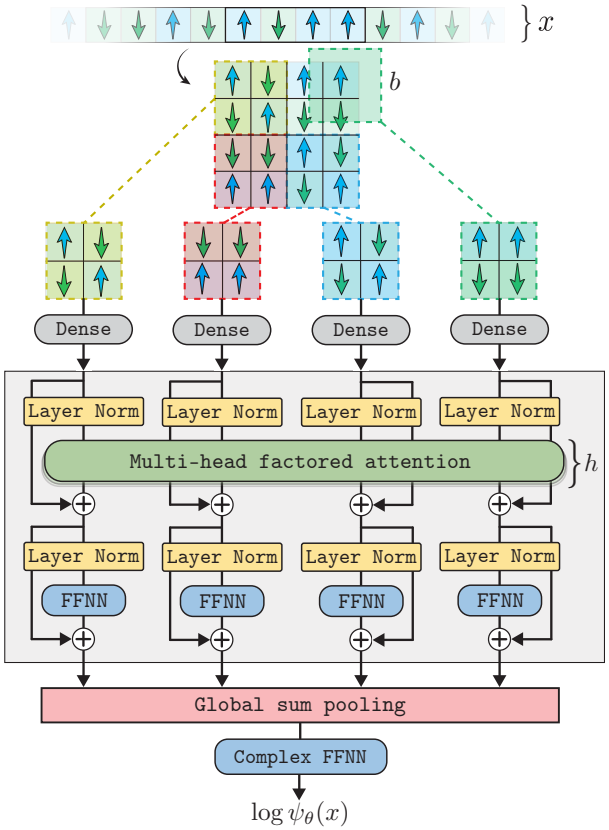

Figure 13: Illustrative representation of the ViT architecture described in Appendix H.2. We refer to Refs. [96, 39] for further details.

H.2 Vision Transformer

The Vision Transformer (ViT) is a state-of-the-art architecture in machine learning. While originally developed for image classification and segmentation, it has recently been adopted as a powerful ansatz for quantum many-body wavefunctions [39, 96]. Below, we describe the key components and parameters of this architecture as applied to NQS. For a spin-1/2 system defined on a square lattice of linear length L, the input consists of a configuration vector $\boldsymbol{x} = (s_1, \ldots, s_N)$ with $s_i \in \{\pm 1\}$. This vector is reshaped into an $L \times L$ matrix $\boldsymbol{X} = \text{vec}^{-1}(\boldsymbol{x})$, with vec representing the vectorization operation [110]. This matrix corresponds to a specific configuration of spins on the lattice.

Patch Extraction and Embedding. The matrix is divided into $n \in \mathbb{N}$ non-overlapping patches of size $b \times b$, namely $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_n$ with $\boldsymbol{X}_i \in \{\pm 1\}^{b \times b}$. Of course, for this to be possible, the total number of sites must be an integer multiple of b. Each patch is flattened, and linearly projected into a high-dimensional embedding space of dimension d. This transforms the spin values into a vector representation that is processed by the encoder blocks:

$$\left\{\boldsymbol{X}_i \in \{\pm 1\}^{b \times b}\right\}_{i=1}^{n} \;\to\; \left\{\text{vec}(\boldsymbol{X}_i) \in \{\pm 1\}^{b^2}\right\}_{i=1}^{n} \to \left\{\boldsymbol{x}_i \in \mathbb{R}^d\right\}_{i=1}^{n}. \tag{101}$$

Encoder Blocks. The core of the ViT architecture consists of N_L encoder blocks. The input to each block is a set of n vectors $\{\boldsymbol{x}_i^{(\ell)} \in \mathbb{R}^d\}_{i=1}^n$ with $\ell = 1, \ldots, N_L$ and $\boldsymbol{x}_i^{(1)} = \boldsymbol{x}_i$. Each encoder block correlates the input vectors thereby capturing long-range correlations and intricate interactions in the quantum system. Each block consists of the following components:

Multi-Head Factored Attention Mechanism. The generic *self-attention* mechanism is defined by three rectangular matrices of parameters $\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}$ known as Query, Key, and Value. For each of the n input vectors $\boldsymbol{x}_i$, three new vectors are computed: $\boldsymbol{q}_i = \boldsymbol{Q}\boldsymbol{x}_i$, $\boldsymbol{k}_i = \boldsymbol{K}\boldsymbol{x}_i$, $\boldsymbol{v}_i = \boldsymbol{V}\boldsymbol{x}_i$. These vectors are then combined with one another to create n attention vectors $\boldsymbol{A}_i = \sum_{j=1}^n \alpha(\boldsymbol{q}_i, \boldsymbol{k}_j)\, \boldsymbol{v}_j$ where the attention weights $\alpha(\boldsymbol{q}_i, \boldsymbol{k}_j)$ modulate the contribution of each value vector $\boldsymbol{v}_j$ to the globally mixed vector $\boldsymbol{A}_i$. To improve the performance of the model, a *multi-head attention mechanism* is often preferred to this self-attention. In this case, a set of $h \in \mathbb{N}$ matrices $\{\boldsymbol{Q}^\mu, \boldsymbol{K}^\mu, \boldsymbol{V}^\mu\}_{\mu=1}^h$ are used to compute in parallel a set of h sets of n self-attention vectors $\{\boldsymbol{A}_1^\mu, \ldots, \boldsymbol{A}_n^\mu\}_{\mu=1}^h$ with $\boldsymbol{A}_i^\mu \in \mathbb{R}^r$ and $r = d/h$. While in general we would have

$$\boldsymbol{A}_i^\mu = \sum_{j=1}^n \alpha(\boldsymbol{Q}\boldsymbol{x}_i^\mu, \boldsymbol{K}\boldsymbol{x}_j^\mu)\, \boldsymbol{V}\boldsymbol{x}_j, \tag{102}$$

the simplification proposed in [96, 39] takes

$$\boldsymbol{A}_i^\mu = \sum_{j=1}^n \alpha_{ij}^\mu\, \boldsymbol{V}^\mu \boldsymbol{x}_j, \tag{103}$$

where $\boldsymbol{V}^\mu$ is an $r \times d$ matrix and α_{ij}^μ is a tensor of parameters depending on the positions of groups of spins (patches i, j) and independent instead of the actual spin configuration at those positions (no dependence on $\boldsymbol{x}_i$ or $\boldsymbol{x}_j$). To summarize:

$$\{\boldsymbol{x}_i \in \mathbb{R}^d\}_{i=1,\ldots,n} \;\to\; \{\boldsymbol{A}_i^\mu \in \mathbb{R}^r\}_{\substack{i=1,\ldots,n \\ \mu=1,\ldots,h}}. \tag{104}$$

Feed-Forward Neural Network. The h sets of vectors from the attention layer $\boldsymbol{A}_i^\mu$ are first concatenated into a set of n d-dimensional vectors $\{\boldsymbol{y}_i\}_{i=1}^n$ with

$$\boldsymbol{y}_i = \mathrm{Concat}(\boldsymbol{A}_i^1, \ldots, \boldsymbol{A}_i^h) \in \mathbb{R}^d. \tag{105}$$

A feed-forward network (FFN) is finally used to process these vectors independently. The hidden layer in this FFN has a dimensionality of $n_{\mathrm{up}}\, d$ with n_{up} the upscaling factor. A GeLU (Gaussian Error Linear Unit) activation function is used between the layers of the FFN, introducing a non-linearity that helps the model capture more complex features. The output of the encoder block is a set of n vectors

$$\boldsymbol{x}_i^{(\ell+1)} = \mathrm{FFN}(\boldsymbol{y}_i) \in \mathbb{R}^{n_{\mathrm{up}}\, d}. \tag{106}$$

This output is then used as the input to the subsequent block.

Skip Connections and Layer Normalization. Skip connections are applied across the attention and FFN layers. These connections help alleviate the vanishing gradient problem, allowing the model to train deeper architectures. Layer normalization is applied before both the attention and FFN layers to stabilize the training process and improve convergence.

Output and Wave Function Representation. After the spin configurations pass through the encoder blocks, the output vectors corresponding to each patch are summed to create a final hidden representation vector

$$\boldsymbol{z} = \sum_{i=1}^n \boldsymbol{x}_i^{(N_L+1)}. \tag{107}$$

This vector represents the configuration in a high-dimensional space and is passed through a final complex-valued fully connected neural network yielding the log-amplitude of our variational wave function:

$$\log \psi_\theta(\boldsymbol{x}) = \sum_{\alpha=1}^K \mathrm{logcosh}(b_\alpha + \boldsymbol{w}_\alpha \cdot \boldsymbol{z}), \tag{108}$$

where $\boldsymbol{W} = \{b_\alpha, \boldsymbol{w}_\alpha\}_{\alpha=1}^K$ are complex-valued parameters so that $\{\mathrm{Re}(\boldsymbol{W}), \mathrm{Im}(\boldsymbol{W})\} \subset \theta$.

We summarize the ViT configuration with the tuple $\Theta_{\mathrm{ViT}} = (b, h, r, n_{\mathrm{up}}; N_L)$. A schematic representation is shown in Fig. 13.