Automatic differentiable Monte Carlo: Theory and application

Shi-Xin Zhang,[1,*] Zhou-Quan Wan,[1,*] and Hong Yao[1,2,†]
[1]*Institute for Advanced Study, Tsinghua University, Beijing 100084, China*
[2]*Department of Physics, Stanford University, Stanford, California 94305, USA*

(Received 31 December 2019; accepted 9 July 2023; published 21 July 2023)

Differentiable programming has emerged as a key programming paradigm empowering rapid developments of deep learning while its applications to important computational methods such as Monte Carlo remain largely unexplored. Here we present the general theory enabling infinite-order automatic differentiation on expectations computed by Monte Carlo with *unnormalized* probability distributions, which we call automatic differentiable Monte Carlo (ADMC). By implementing ADMC algorithms on computational graphs, one can also leverage state-of-the-art machine learning frameworks and techniques in traditional Monte Carlo applications in statistics and physics. We illustrate the versatility of ADMC by showing some applications: fast search of phase transitions and accurately finding ground states of interacting many-body models in two dimensions. ADMC paves a promising way to innovate Monte Carlo in various aspects to achieve higher accuracy and efficiency.

DOI: 10.1103/PhysRevResearch.5.033041

I. INTRODUCTION

Differentiation is a broadly important concept and a widely useful method in subjects such as mathematics and physics. Automatic differentiation (AD) evaluates derivatives of any function specified by computer programs [1,2] by propagating derivatives of primitive operations via chain rules. It is different from conventional symbolic differentiations by totally avoiding complicated analytic expressions of derivatives and is advantageous to numerical differentiations by totally eliminating discretization errors. Besides, AD is particularly successful in calculating higher-order derivatives and computing gradients with respect to large numbers of variables as in the case for gradient-based optimization algorithms. Emerging as a new programming paradigm, AD is now extensively utilized in machine learning. Being one of the most important infrastructures for machine learning, it enables massive exploration on neural network structures.

The great application potential of AD in fields beyond machine learning started to emerge. Specifically, AD has been applied to certain areas of computational physics; for instance, its interplay with tensor networks was investigated very recently [3–6]. One may naturally ask whether AD can be leveraged in Monte Carlo (MC) methods, another big family of computational algorithms. Initial investigations on encoding MC methods into an AD framework [7] all assumed normalized probability distributions for Monte Carlo sampling. However, for nearly all interesting problems the normalization factor is not known *a priori* and the probability distribution is usually unnormalized. Fortunately, knowing the ratio of probabilities between different configurations is sufficient to perform MC simulations in the Metropolis-Hasting algorithm [8]; Monte Carlo with unnormalized probability distribution is now a widely employed numerical approach in statistics and physics. Consequently, it is highly desired to integrate AD into generic Monte Carlo to achieve high accuracy and efficiency in various MC applications such as solving interacting many-body models in physics.

In this paper, we fill in the gap by proposing a general theory enabling infinite-order automatic differentiation on expectations computed by Monte Carlo with *unnormalized* probability distributions, which we call automatic differentiable Monte Carlo (ADMC). Specifically, ADMC employs the method of AD to compute gradients of MC expectations, which is a key quantity used in statistics and machine learning [7], without *a priori* knowledge of normalization factor or partition function, which is the case for nearly all application scenarios in Markov chain Monte Carlo (MCMC). As the MC gradient problem lies at the core of probabilistic programming [9] and plays a central role in various fields including optimization, variational inference, reinforcement learning, and variational Monte Carlo (VMC), ADMC can be employed in a wide range of MC applications to achieve high accuracy and efficiency.

ADMC not only works with gradient of expectations in MC with unnormalized distributions but also holds true for higher-order derivatives of MC expectations. In contrast, MC estimation of higher-order derivatives was rarely considered [10]. In addition, ADMC can be embedded in general stochastic computational graph [11] frameworks seamlessly and plays a critical role in the interplay between differentiable programming and probabilistic programming. By introducing ADMC, we can build Monte Carlo applications in the

*These authors contributed equally to this work.
†yaohong@tsinghua.edu.cn

Published by the American Physical Society under the terms of the Creative Commons Attribution 4.0 International license. Further distribution of this work must maintain attribution to the author(s) and the published article's title, journal citation, and DOI.

state-of-the-art machine learning infrastructure to achieve high accuracy and efficiency in addressing questions such as fast search of phase transitions and ground states of interacting quantum models. For models we studied by ADMC, comparable or higher accuracy has been obtained by comparing with previous methods such as restricted Boltzmann machine (RBM) and tensor networks. Moreover, ADMC paves a promising way to innovate Monte Carlo in various aspects, e.g., easing or solving the sign problem [12–21] of quantum Monte Carlo (QMC) [22–26].

The organization of this paper is as follows. In Sec. II, we review important background knowledge required to understand the general theory of ADMC and its applications, including automatic differentiation, estimation on Monte Carlo gradients, and variational Monte Carlo methods. In Sec. III, we elaborate our theory towards ADMC, including the detach function, ADMC estimator for normalized and unnormalized probability distributions as well as general theory on the Fisher information matrix (FIM) with unnormalized probabilities, and the general theory for the AD-aware version of VMC. In Sec. IV, we present two explicit ADMC applications in physics: fast searching of phase transitions and critical temperature in the two-dimensional (2D) Ising model, and end-to-end general-purpose ADVMC algorithms and accurately finding the ground state of the 2D quantum spin-1/2 Heisenberg model. We demonstrate how to leverage the power of state-of-the-art machine learning for ADMC algorithms in particular. In Sec. V, we further discuss other possible applications of ADMC as well as some outlooks on ADMC.

II. BACKGROUND

In this section, we would like to provide some background knowledge for the sake of being self-contained. Specifically, we will introduce some basic knowledge of AD, Monte Carlo gradient estimations, and variational Monte Carlo, which are related to the general theory and applications of ADMC.

A. Automatic differentiation

Conventional methods of computing gradients of a given function include symbolic and numerical approaches. It is challenging to symbolically compute gradients of complicated functions as deriving the analytical expression of the gradient is often nearly impossible. A numerical differentiation approach computes the gradient by finite discretization and thus normally suffers discretization errors. In addition, these two conventional methods encounter more challenges or errors in computing higher-order derivatives, especially when the number of input parameters is large.

AD, on the contrary, by tracing the derivative propagation of primitive operations via chain rules, can render numerically exact derivatives (including higher-order derivatives) for any programs [1,2,27]. The program is specified by a computational graph composed of function primitives. Such a directed acyclic graph shows the data shape and data flow of the corresponding program.

There are two ways to compute the derivative on the graph with respect to the graph's inputs: the forward AD and backward AD. The forward AD iteratively computes the recursive expression as shown in Fig. 1(a):

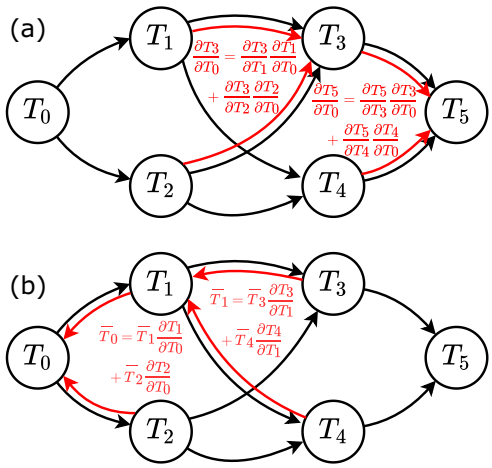

FIG. 1. (a) Forward mode and (b) reverse mode automatic differentiation on computational graphs. Black arrows label the forward pass from inputs to outputs. Red arrows represent forward chain rules in (a) and backpropagation for adjoints in (b).

$$\frac{\partial T_i}{\partial T_0} = \sum_{T_{i-1}\in \text{parent}\{T_i\}} \frac{\partial T_i}{\partial T_{i-1}} \frac{\partial T_{i-1}}{\partial T_0}, \tag{1}$$

where T_i stands for nodes on the computational graph; T_0 is the input and T_n the final output. The gradient we aim to obtain is $\frac{\partial T_n}{\partial T_0}$. Here $\frac{\partial T_i}{\partial T_{i-1}}$ corresponds to the derivatives of operator primitives $T_i = f(T_{i-1})$, and these derivatives are implemented as AD infrastructure built-in or user customizations. One drawback of the forward mode AD is that one needs to keep track of every derivative $\frac{\partial T_i}{\partial T_0[i]}$ in the middle of the graph when the input has many parameters, which is normally expensive and inefficient.

Reverse mode AD can avoid the inefficiency encountered by forward mode AD when input parameters are far more than output ones, which is the case for many applications including machine learning and variational Monte Carlo. By defining the adjoint as $\overline{T}_i = \frac{\partial T_n}{\partial T_i}$, as shown in Fig. 1(b), reverse mode AD iteratively computes the recursive expression:

$$\overline{T}_i = \sum_{T_{i+1}\in \text{child}\{T_i\}} \overline{T}_{i+1} \frac{\partial T_{i+1}}{\partial T_i}. \tag{2}$$

The final aim is to compute $\overline{T}_0$. In this approach of AD, one first computes the output and saves all intermediate node values T_i in the forward pass, and then backpropagates the gradients in the reverse pass. This workflow, denoted as backpropagation in machine learning language [28], is opposite to the forward mode AD where all computations happen in the same forward pass. Reverse mode AD is particularly suitable for scenarios with multiple input parameters and one output value, which is the case of most deep learning setups [29] and many MC approaches such as variational Monte Carlo.

B. Gradients of Monte Carlo expectations

As explained in the Introduction, it is of great importance in various fields to compute gradients of Monte Carlo

expectation values: $\nabla_{\boldsymbol{\theta}}\langle O(\boldsymbol{x},\boldsymbol{\theta})\rangle_{p(\boldsymbol{x},\boldsymbol{\theta})}$, where $\boldsymbol{\theta}$ represents the set of input parameters, $\boldsymbol{x}$ labels MC configurations, $p(\boldsymbol{x},\boldsymbol{\theta})$ is the (generally unnormalized) probability distribution, and $\langle O(\boldsymbol{x},\boldsymbol{\theta})\rangle_{p(\boldsymbol{x},\boldsymbol{\theta})}$ is the MC expectation value of O under the probability distribution $p(\boldsymbol{x},\boldsymbol{\theta})$. We often call O the loss function. Currently, there are two main methods for evaluating MC gradients: the score function estimator [30] (also denoted as REINFORCE [31]) and the pathwise estimator (also denoted as the reparametrization trick [32], stochastic backpropagation [33], or "push out" method [34]). Although the method of pathwise estimation in general gives a lower variance for MC gradient estimation, it can only be applied in quite limited settings due to the strict requirements on the differentiability of transformers and probability distributions. Therefore, it is nearly impossible to apply a pathwise estimator to evaluate MC gradients sampled from vastly complicated distributions encountered in most physics problems. We thus focus on the score function method in the present paper as it is more universal and general purpose.

The score function estimator is a general-purpose MC gradient estimator with gradient given by

$$\nabla_{\boldsymbol{\theta}}\langle O(\boldsymbol{x},\boldsymbol{\theta})\rangle_p = \left\langle \nabla_{\boldsymbol{\theta}} O(\boldsymbol{x},\boldsymbol{\theta}) + O(\boldsymbol{x},\boldsymbol{\theta})\frac{\nabla_{\boldsymbol{\theta}} p(\boldsymbol{x},\boldsymbol{\theta})}{p(\boldsymbol{x},\boldsymbol{\theta})}\right\rangle_p, \tag{3}$$

where p is a shortcut for $p(\boldsymbol{x},\boldsymbol{\theta})$ that is normalized. Note that Eq. (3) is quite general by taking into account the dependence of the loss function O on the parameters $\boldsymbol{\theta}$. To leverage AD in the gradient estimation, it is desired to construct an AD-aware version of the MC expectation which can correctly obtain the MC gradient itself and its derivatives to all orders (including the gradients and all higher derivatives). For normalized probability distribution p, the following AD-aware version of the MC expectation,

$$\left\langle \frac{O(\boldsymbol{x},\boldsymbol{\theta})p(\boldsymbol{x},\boldsymbol{\theta})}{\perp(p(\boldsymbol{x},\boldsymbol{\theta}))}\right\rangle_p, \tag{4}$$

was proposed [10], where $\perp(\cdot)$ represents that $\cdot$ does not propagate gradients in the AD process (detailed explanations are given in Sec. III). However, for nearly all interesting physics problems, the normalization factor is not known *a priori* and the probability distribution is usually unnormalized. It is of central importance to sample from such unnormalized probability distributions for applications such as computing physical quantities without knowing the partition function *a priori* or approximating the posterior distributions of latent variables only with knowledge of likelihood and prior. Nevertheless, MC gradient estimation from such unnormalized probability distribution has not been constructed by any previous method. In Sec. III of the present paper, we develop a general framework and construct the AD-aware objective MC expectation which can correctly obtain both the expectation value and all its higher-order derivatives for unnormalized probability distributions.

C. Variational Monte Carlo in physics

VMC is a powerful numerical algorithm searching the ground state of a given quantum Hamiltonian based on the variational principle since a physical Hamiltonian has energy bounded from below [35,36]. By sampling the amplitude of variational wave function $|\psi_{\boldsymbol{\theta}}\rangle$, where $\boldsymbol{\theta}$ represents the set of variational parameters, one can compute the energy expectation $E_{\boldsymbol{\theta}} = \langle\psi_{\boldsymbol{\theta}}|H|\psi_{\boldsymbol{\theta}}\rangle/\langle\psi_{\boldsymbol{\theta}}|\psi_{\boldsymbol{\theta}}\rangle$, where the ansatz wave function $|\psi_{\boldsymbol{\theta}}\rangle$ is in general not normalized. The energy expectation can be evaluated through MC:

$$E_{\boldsymbol{\theta}} = \frac{\sum_\sigma p(\sigma,\boldsymbol{\theta})\frac{\langle\sigma|H|\psi_{\boldsymbol{\theta}}\rangle}{\langle\sigma|\psi_{\boldsymbol{\theta}}\rangle}}{\sum_\sigma p(\sigma,\boldsymbol{\theta})} = \langle E_{\mathrm{loc}}(\sigma,\boldsymbol{\theta})\rangle_{p(\sigma,\boldsymbol{\theta})}, \tag{5}$$

where $E_{\mathrm{loc}}(\sigma,\boldsymbol{\theta}) = \frac{\langle\sigma|H|\psi_{\boldsymbol{\theta}}\rangle}{\langle\sigma|\psi_{\boldsymbol{\theta}}\rangle}$, σ is complete basis of the quantum system's Hilbert space, and $p(\sigma,\boldsymbol{\theta}) = |\langle\sigma|\psi_{\boldsymbol{\theta}}\rangle|^2$ is the probability distribution. Note that the probability distribution $p(\sigma,\boldsymbol{\theta})$ is in general unnormalized since the ansatz wave function $\psi_{\boldsymbol{\theta}}(\sigma) = \langle\sigma|\psi_{\boldsymbol{\theta}}\rangle$ is in general unnormalized (as it is usually challenging to normalize the ansatz wave function due to complicated wave function structure). Since $E_{\boldsymbol{\theta}}$ depends on variational parameters $\boldsymbol{\theta}$, one thus can in principle optimize $E_{\boldsymbol{\theta}}$ obtained by Eq. (5) against $\boldsymbol{\theta}$, giving rise to the optimal ground state energy and wave function within the ansatz.

Stochastic gradient descent (SGD) is *de facto* for optimizations in machine learning [37–39] and can also be employed in computational physics such as optimization in VMC [40]. There are various generalizations beyond vanilla SGD optimizers that consider momentum and adaptive behaviors, among which Adam [41] is one common optimizer in training neural networks. Natural gradient descent, a concept emerged from information geometry, is one of the optimization techniques where the local curvature in distribution space defined by neural networks has been considered [42–45]. Efficient approximations on natural gradients have also been investigated such as FANG [46] and K-FAC [47–51]. For optimization problem such as VMC, gradient descent and natural gradient descent methods can be applied where various machine learning techniques can be utilized to boost VMC. Natural gradient optimization is exactly equivalent to the stochastic reconfiguration (SR) method [52,53] in VMC [54–57].

Recently, there were various studies focusing on using RBM or related neural networks as the ansatz wave function for quantum systems composed of spins [55,58–69], bosons [54,70,71], and fermions [56,72]. In previous studies, to incorporate such a wave function ansatz into the framework of VMC, one either computes all derivatives $\nabla_{\boldsymbol{\theta}}\psi_{\boldsymbol{\theta}}(\sigma)$ analytically when the neural network ansatz is simple enough [58] or applies AD on the wave function to compute $\nabla_{\boldsymbol{\theta}}\psi_{\boldsymbol{\theta}}(\sigma)$ [68,73], and then estimates the gradient $\nabla_{\boldsymbol{\theta}}E_{\boldsymbol{\theta}}$ by MC sampling given by

$$\nabla_{\boldsymbol{\theta}}E_{\boldsymbol{\theta}} = 2\mathrm{Re}\left[\left\langle\frac{\nabla\psi_\sigma{}^*}{\psi_\sigma^*}E_{\mathrm{loc}}\right\rangle - \langle E_{\mathrm{loc}}\rangle\left\langle\frac{\nabla\psi_\sigma{}^*}{\psi_\sigma^*}\right\rangle\right], \tag{6}$$

where $\langle\cdot\rangle$ denotes MC sampling of configurations σ from probability distribution $|\psi(\sigma)|^2$. However, applying AD directly in the energy expectation $E_{\boldsymbol{\theta}} = \langle\psi_{\boldsymbol{\theta}}|H|\psi_{\boldsymbol{\theta}}\rangle$ to obtain the gradient $\nabla_{\boldsymbol{\theta}}E_{\boldsymbol{\theta}}$, the most intuitive way to optimize the ground state, is still lacking partly due to the lack of AD technique for MC expectations sampled from unnormalized probability distributions. With the introduction of ADMC in this work, we can implement AD-aware VMC, which is much more straightforward and easy to implement by directly optimizing the energy expectation without any analytical derivation on

derivatives for MC expectations or wave functions, which we call "end-to-end" ADVMC.

III. THEORY

In this section, we present the general theory of the ADMC which enables infinite-order automatic differentiation on MC expectations with unnormalized probability distributions. We show the detailed derivations on the general theory.

A. Detach function

We first introduce detach function $\perp(x)$ which features $\perp(x) = x$ and $\frac{\mathrm{d}\perp(x)}{\mathrm{d}x} = 0$. Here we list some basic formulas in terms of detach functions utilized later: $f(\perp(x)) = \perp(f(x))$, $\perp(\perp(x)) = \perp(x)$, $\perp(x+y) = \perp(x) + \perp(y)$, and $\perp(xy) = \perp(x)\perp(y)$. The detach function can be easily implemented and simulated in modern machine learning frameworks (it corresponds to `stop_gradient` in TENSORFLOW [74] and `detach` in PYTORCH [75]). We call this function primitive as the detach function in this work. This weird-looking function has a natural explanation in the context of machine learning, especially in terms of computational graphs. Such an operator corresponds to nodes in the graph which only pass forward values while stopping the backpropagation of gradients.

By utilizing the detach function, we can construct functions whose derivatives of each order are not related. For example, the function $O(x) = x - \perp(x)$ equals zero irrespective of x although its first-order derivative is 1. Using the language of the detach function, the meanings of "function" and "equal" can be enlarged.

Theorem 1 For any "weird" function, whose value and every order of derivatives are defined separately, it can always be expressed by "normal" functions with detach function $\perp$.

Proof. For a function $\mathcal{F}(x)$ for which each order of derivatives is defined as $\mathcal{F}^{(n)}(x) = h_n(x)$, the construction with $\perp$ is

$$\mathcal{F}(x) = \sum_{n=0}^{\infty} \frac{1}{n!} h_n(\perp(x))(x - \perp(x))^n. \tag{7}$$

When translated into TENSORFLOW language, Theorem 1 states that every function defined with `tf.custom_gradient` can be instead defined with `tf.stop_gradient`.

Corollary 1. For a function with multiple variate input $\mathcal{F}(x_1, \ldots, x_m)$ whose derivatives $\mathcal{F}^{(n_1,\ldots,n_m)}$ are defined separately, irrelevant to the original function, it can always be expressed by "normal" functions together with single variate detach function $\perp$.

The corollary above is obvious by considering similar Taylor expansion construction as in Eq. (7).

The introduction of imaginary number i enlarges the meaning of the equal sign by twice the equivalence relation: $x = y \Leftrightarrow \mathrm{Re}(x) = \mathrm{Re}(y)$ and $\mathrm{Im}(x) = \mathrm{Im}(y)$. Similarly, with the introduction of the detach function, the equal signs are enlarged as infinite independent equivalence relations: $f(x) = g(x) \Leftrightarrow \perp(f^{(n)}(x)) = \perp(g^{(n)}(x))$ $(n = 0, 1, 2, \ldots)$. The conventional "equal" is reexpressed as one relation $(n = 0)$ of the above series: $\perp(f(x)) = \perp(g(x))$.

B. ADMC

We are ready to construct a general theory for MC expectation which can render AD to correctly obtain its directives at all orders (including the zero-order derivative, the expectation itself). We employ the extended score function method to enable AD on MC expectations for any complicated distribution, both normalized and unnormalized. Theorem 2 below is the central theoretical result of the present paper.

Theorem 2. The following MC estimator of $\langle O(\boldsymbol{x}, \boldsymbol{\theta})\rangle_p$,

$$\langle O(\boldsymbol{x}, \boldsymbol{\theta})\rangle_p = \frac{\left\langle \frac{p}{\perp(p)} O(\boldsymbol{x}, \boldsymbol{\theta})\right\rangle_{\perp(p)}}{\left\langle \frac{p}{\perp(p)}\right\rangle_{\perp(p)}}, \tag{8}$$

is automatic differentiable to all orders and works for both normalized and unnormalized probability distribution $p = p(\boldsymbol{x}, \boldsymbol{\theta})$.

To prove Theorem 2, we first introduce the following lemma.

Lemma 1. For both normalized and unnormalized probability distribution $p = p(\boldsymbol{x}, \boldsymbol{\theta})$,

$$\sum_{\boldsymbol{x}\in S(p)} \frac{p}{\perp(p)} \doteq \frac{Z}{\perp(Z)}. \tag{9}$$

Here Z is the shortcut for partition function $Z_{\boldsymbol{\theta}} = \sum_{\boldsymbol{x}\in \mathrm{all}} p(\boldsymbol{x}, \boldsymbol{\theta})$ with $\boldsymbol{x} \in$ all representing the summation over all configurations $\boldsymbol{x}$. $\sum_{\boldsymbol{x}\in S(p)}$ denotes the average obtained through MC sampling according to the probability distribution p. Note that, for brevity, we omit the $\frac{1}{N_s}$ factor before the MC sum $\sum_{\boldsymbol{x}\in S(p)}$ in Eq. (9) and hereafter; the sum should be understood as the average $\frac{1}{N_s}\sum_{\boldsymbol{x}\in S(p)}$, where N_s is the number of sample configurations. In Eq. (9) and hereafter, "$\doteq$" means that it is the same as the equal sign since MC estimation can be made exact in the limit of large N_s. The equal sign also makes sense in any order derivatives. Therefore, to prove the lemma we just need to demonstrate the following formula:

$$\perp\left(\sum_{\boldsymbol{x}\in S(p)} \nabla_{\boldsymbol{\theta}}^{(n)} \frac{p}{\perp(p)}\right) \doteq \perp\left(\nabla_{\boldsymbol{\theta}}^{(n)} \frac{Z}{\perp(Z)}\right), \tag{10}$$

where $\nabla_{\boldsymbol{\theta}}^{(n)}$ is a shortcut for $\nabla_{\theta_1,\ldots,\theta_m}^{(n_1,\ldots,n_m)}$, $n_j = 0, 1, 2, \ldots$.

For $n = 0$, the equation is simply true since both sides give 1. For arbitrary n, it is straightforward to show that

$$\begin{aligned}\perp\left(\sum_{\boldsymbol{x}\in S(p)} \frac{\perp(\nabla^{(n)} p)}{\perp(p)}\right) &\doteq \sum_{\boldsymbol{x}\in \mathrm{all}} \frac{\perp(p)}{\perp(Z)} \frac{\perp\nabla^{(n)} p}{\perp(p)} \\ &= \frac{\perp\nabla^{(n)} \sum_{\boldsymbol{x}\in \mathrm{all}} p}{\perp Z} \\ &= \frac{\perp(\nabla^{(n)} Z)}{\perp(Z)},\end{aligned} \tag{11}$$

which finishes the proof of the lemma.

The proof of the lemma above can be significantly simplified. With the enlarged meaning of the equal sign, each order of derivatives is automatically equal as long as expressions with the detach function are accordingly considered. In other words, to prove that some relation holds true for any order derivatives $\perp(f^{(n)}(x)) = \perp(g^{(n)}(x))$ $(n = 0, 1, 2, \ldots)$, we

only need to prove that $f(x) = g(x)$ is true. This simplification is the power of the detach function. The proof of the lemma can be simplified as

$$\sum_{x\in S(p)} \frac{p}{\perp(p)} \doteq \sum_{x\in \text{all}} \frac{\perp(p)}{\perp(Z)} \frac{p}{\perp(p)} = \frac{\sum_{x\in\text{all}} p}{\perp(Z)} = \frac{Z}{\perp(Z)}. \quad (12)$$

Note that the simplification from the involved proof in Eqs. (10) and (11) to the neat one in Eq. (12) reflects the brevity and power of the detach function and its algebra.

Now we are ready to prove Theorem 2. Note that, in the average $\langle\cdot\rangle_{\perp(p)}$, the probability distribution $\perp(p)$ is the background and is not involved in derivatives. Based on the spirit of detach function algebra, it is enough to show

$$\sum_{x\in S(p)} \frac{p}{\perp(p)} O \Big/ \sum_{x\in S(p)} \frac{p}{\perp(p)} \doteq \sum_{x\in\text{all}} p\frac{O}{Z}, \quad (13)$$

where O is the shortcut for $O(\boldsymbol{x}, \boldsymbol{\theta})$. By utilizing the lemma in Eq. (9) and observing the fact that $Z_{\boldsymbol{\theta}}$ is independent of $\boldsymbol{x}$, it is straightforward to prove Eq. (13) as follows:

$$\begin{aligned}\sum_{x\in S(p)} \frac{p}{\perp(p)} O \Big/ \sum_{x\in S(p)} \frac{p}{\perp(p)} &\doteq \sum_{x\in\text{all}} \frac{\perp(p)}{\perp(Z)} \frac{pO}{\perp(p)} \Big/ \left(\frac{Z}{\perp(Z)}\right) \\ &= \sum_{x\in\text{all}} p\frac{O}{Z}, \end{aligned} \quad (14)$$

where the first $\doteq$ comes from Eq. (9) and the properties of the detach function. Here we emphasize that both the equal signs, $\doteq$ and $=$, have enlarged meanings that expressions of both sides are equal to arbitrary order derivatives. Thus we get

$$\perp\left(\nabla_{\boldsymbol{\theta}}^{(n)} \frac{\left\langle \frac{p}{\perp(p)} O\right\rangle_{\perp(p)}}{\left\langle \frac{p}{\perp(p)}\right\rangle_{\perp(p)}}\right) \doteq \perp\left(\nabla_{\boldsymbol{\theta}}^{(n)} \sum_{x\in\text{all}} p\frac{O}{Z}\right), \quad (15)$$

where $n = 0, 1, 2, \ldots$. This finishes the proof of Theorem 2, which is the central result of the present paper. We believe Theorem 2 can provide endless opportunities to build applications combining AD infrastructure with MC algorithms.

We emphasize that Theorem 2 is general and applies for both normalized and unnormalized probability distribution p. For the case of normalized distribution $\sum_{x\in\text{all}} p = 1$, we obtain $\sum_{x\in S(p)} \frac{p}{\perp(p)} \doteq \sum_{x\in\text{all}} \perp(p)\frac{p}{\perp(p)} = \sum_{x\in\text{all}} p = 1$. Then, Eq. (8) in Theorem 2 can be simplified to $\langle O\rangle_p = \langle \frac{p}{\perp(p)} O\rangle_{\perp(p)}$, which is the MC estimator applicable only for the case of normalized probability distribution. For nearly all interesting applications with an unnormalized probability distribution p, Theorem 2 is the correct one to use, as we demonstrate in the applications below.

It is worth providing a heuristic explanation for Theorem 2. Through discretizing the parameters $\boldsymbol{\theta}$ in numerical differentiations, a rigorous MC gradient can be obtained in the limit of zero discretizing intervals. Specifically, to get gradients at $\boldsymbol{\theta_0}$, one can directly compute MC expectations of O by sampling separately from $p(\boldsymbol{\theta})$ and from $p(\boldsymbol{\theta}_0)$, with $\boldsymbol{\theta}$ very close to $\boldsymbol{\theta_0}$. However, it is highly inefficient to sample separately from $p(\boldsymbol{\theta})$ and $p(\boldsymbol{\theta}_0)$ distributions. As $\boldsymbol{\theta}$ is close to $\boldsymbol{\theta_0}$ (in the limit $\boldsymbol{\theta} \to \boldsymbol{\theta_0}$), one can actually reuse the samples from $p(\boldsymbol{\theta}_0)$ to evaluate the expectation at $\boldsymbol{\theta}$:

$$\begin{aligned}&\langle O(\boldsymbol{x}, \boldsymbol{\theta})\rangle_{p(\boldsymbol{\theta})} \\ &= \sum_{x\in\text{all}} p(\boldsymbol{\theta}) O(\boldsymbol{x}, \boldsymbol{\theta}) \Big/ \sum_{x\in\text{all}} p(\boldsymbol{\theta}) \\ &= \sum_{x\in\text{all}} p(\boldsymbol{x}, \boldsymbol{\theta}_0) \frac{p(\boldsymbol{x}, \boldsymbol{\theta})}{p(\boldsymbol{x}, \boldsymbol{\theta}_0)} O(\boldsymbol{x}, \boldsymbol{\theta}) \Big/ \sum_{x\in\text{all}} p(\boldsymbol{x}, \boldsymbol{\theta}_0) \frac{p(\boldsymbol{x}, \boldsymbol{\theta})}{p(\boldsymbol{x}, \boldsymbol{\theta}_0)} \\ &= \left\langle \frac{p(\boldsymbol{x}, \boldsymbol{\theta})}{p(\boldsymbol{x}, \boldsymbol{\theta}_0)} O(\boldsymbol{x}, \boldsymbol{\theta})\right\rangle_{p(\boldsymbol{\theta}_0)} \Big/ \left\langle \frac{p(\boldsymbol{x}, \boldsymbol{\theta})}{p(\boldsymbol{x}, \boldsymbol{\theta}_0)}\right\rangle_{p(\boldsymbol{\theta}_0)}. \end{aligned} \quad (16)$$

By comparing Eq. (16) with Eq. (8), one can observe the parallel relations between them and understand the physical rationale behind detach functions: when evaluating derivatives only $\boldsymbol{\theta}$ changes while $\boldsymbol{\theta}_0$ remains fixed; all terms related to $\boldsymbol{\theta}_0$ are wrapped with the detach function $\perp$ in the exact form of Eq. (8) in Theorem 2.

Finally, we make a note on implementation. For numerical stability, $\ln p$ instead of p is in general referenced and the AD version of the MC estimator for generic probability distribution p is then given by

$$\langle O\rangle_p = \frac{\langle \exp(\ln p - \perp(\ln p)) O\rangle_{\perp p}}{\langle \exp(\ln p - \perp(\ln p))\rangle_{\perp p}}. \quad (17)$$

From the computational graph implementation perspective, p is never explicitly calculated since the numerical value of $\exp(\ln p - \perp(\ln p))$ is exactly 1. Therefore, the ADMC approach using $\ln p$ is automatically free from the numerical instability encountered in approaches directly using p.

C. Fisher information matrix and Kullback-Leibler divergence in ADMC

For the optimization method of natural gradient descent, the parameters $\boldsymbol{\theta}$ are updated in the following way: $\Delta\boldsymbol{\theta} = -\lambda F^{-1}\nabla_{\boldsymbol{\theta}} O_{\boldsymbol{\theta}}$, where F is the Fisher information matrix, λ is the learning rate, and $O_{\boldsymbol{\theta}} = \langle O(\boldsymbol{x}, \boldsymbol{\theta})\rangle_p$. The FIM is of great importance in numerical optimization and is defined as

$$F_{ij} = \left\langle \nabla_i \ln \frac{p}{Z} \nabla_j \ln \frac{p}{Z}\right\rangle_p, \quad (18)$$

where i, j represent θ_i, θ_j. The FIM is also the Hessian (with respect to $\boldsymbol{\theta}'$) of Kullback-Leibler (KL) divergence between $p(\boldsymbol{x}, \boldsymbol{\theta})$ and $p(\boldsymbol{x}, \boldsymbol{\theta}')$ with $\boldsymbol{\theta}'$ approaching $\boldsymbol{\theta}$. Hence, it defines the local curvature in distribution space.

In the following, we derive useful formulas related to FIM with unnormalized probability distribution p in the context of ADMC. For unnormalized p, the expectation of the score function is not zero and it is given by

$$\langle \nabla_{\boldsymbol{\theta}} \ln p\rangle_p = \frac{1}{Z} \sum_{x\in\text{all}} p\frac{\nabla_{\boldsymbol{\theta}} p}{p} = \frac{\nabla_{\boldsymbol{\theta}} Z}{Z} = \nabla_{\boldsymbol{\theta}} \ln Z. \quad (19)$$

Then, the FIM for unnormalized p can be defined as

$$F_{ij} = \langle \nabla_i \ln p \nabla_j \ln p\rangle_p - \langle \nabla_i \ln p\rangle_p \langle \nabla_j \ln p\rangle_p. \quad (20)$$

To apply the AD approach, we can obtain the FIM through the KL divergence whose Hessian is the FIM. The AD-aware

KL divergence is given by

$$\mathrm{KL}\left(\perp\left(\frac{p}{Z}\right)|\frac{p}{Z}\right) = \ln\frac{Z}{\perp(Z)} - \left\langle \ln\frac{p}{\perp(p)}\right\rangle_{\perp(p)} = \ln\left\langle\frac{p}{\perp(p)}\right\rangle_{\perp(p)} - \left\langle \ln\frac{p}{\perp(p)}\right\rangle_{\perp(p)}, \quad (21)$$

where the second equation is due to Eq. (9) in Lemma 1. Therefore, for any unnormalized p, we can construct an object function as Eq. (21) and compute the Hessian of it by ADMC. This approach is preferable to direct estimation from Eq. (20) in some scenarios. A further discussion is given in Appendix A.

Following the path of Eq. (21), we could further derive the AD-aware formula for general KL divergence with unnormalized probability p, q parametrized by $\boldsymbol{\theta}$ as

$$\mathrm{KL}(p_{\boldsymbol{\theta}}|q_{\boldsymbol{\theta}}) = \ln\frac{\left\langle\frac{p}{\perp p}\frac{q}{p}\right\rangle_{\perp p}}{\left\langle\frac{p}{\perp p}\right\rangle_{\perp p}} - \frac{\left\langle\frac{p}{\perp p}\ln\frac{q}{p}\right\rangle_{\perp p}}{\left\langle\frac{p}{\perp p}\right\rangle_{\perp p}}. \quad (22)$$

D. End-to-end ADVMC

As discussed in Sec. II, VMC is an important approach attempting to find the ground state wave function of a Hamiltonian by optimizing parametrized wave functions. Here we describe how to implement end-to-end VMC with ADMC, which we call ADVMC. We focus on the case where ansatz wave functions are positively valued. For the general case of complex-valued ansatz wave functions, ADVMC can also be implemented (see Appendix B for details).

As in Eq. (5), the energy expectation value $E_{\boldsymbol{\theta}} = \langle\psi_{\boldsymbol{\theta}}|H|\psi_{\boldsymbol{\theta}}\rangle$ of Hamiltonian H associated with the wave function $\psi_{\boldsymbol{\theta}}(\sigma) = \langle\sigma|\psi_{\boldsymbol{\theta}}\rangle$ can be evaluated through Monte Carlo sampling:

$$E_{\boldsymbol{\theta}} = \langle E_{\mathrm{loc}}(\sigma,\boldsymbol{\theta})\rangle_{p(\sigma,\boldsymbol{\theta})}, \quad (23)$$

where $p(\sigma,\boldsymbol{\theta}) = |\psi_{\boldsymbol{\theta}}(\sigma)|^2$ is usually an unnormalized probability distribution. To optimize (minimize) $E_{\boldsymbol{\theta}}$ using a gradient-based approach, we need to evaluate the gradients with respect to variational parameters $\boldsymbol{\theta}$:

$$\nabla_{\boldsymbol{\theta}}\langle E_{\mathrm{loc}}(\sigma,\boldsymbol{\theta})\rangle_{p(\sigma,\boldsymbol{\theta})}. \quad (24)$$

It is clear that $E_{\mathrm{loc}}(\sigma,\boldsymbol{\theta})$ in VMC plays a similar role as $O(\boldsymbol{x},\boldsymbol{\theta})$ in MC discussed earlier. It is natural to integrate AD into VMC so that an end-to-end ADVMC can be constructed. The ADVMC version of the energy estimator can be constructed as follows:

$$\langle E_{\mathrm{loc}}(\sigma,\boldsymbol{\theta})\rangle_p = \mathrm{Re}\frac{\left\langle\frac{p}{\perp p}E_{\mathrm{loc}}(\sigma,\boldsymbol{\theta})\right\rangle_{\perp p}}{\left\langle\frac{p}{\perp p}\right\rangle_{\perp p}}, \quad (25)$$

where Re guarantees that the energy estimator is real.

Taking account of the variance reduction trick, the ADVMC energy estimator for the real wave function can also be constructed as [76]

$$\langle E_{\mathrm{loc}}(\sigma,\boldsymbol{\theta})\rangle_p = \mathrm{Re}\frac{\left\langle\frac{\psi^2}{\perp(\psi^2)}\perp(E_{\mathrm{loc}}(\sigma,\boldsymbol{\theta}))\right\rangle_{\perp(p)}}{\left\langle\frac{\psi^2}{\perp(\psi^2)}\right\rangle_{\perp(p)}}. \quad (26)$$

Actually, the objective in Eq. (26) has better performance compared with the original estimator in Eq. (25) since $E(\sigma,\boldsymbol{\theta})$ is detached in Eq. (26) and no further backpropagations behind this node are needed. Note that Eq. (26) as the estimator of $E_{\boldsymbol{\theta}}$ can only reproduce a first-order derivative in the framework of AD, while the original estimator in Eq. (25) is correct for all-order derivatives (see Appendix B 3 for details).

The end-to-end ADVMC framework is universal and easy to implement. Instead of computing derivatives of wave functions and plugging the results into the formula of energy gradients by hand as conventional VMC approaches do in Eq. (6), the end-to-end ADVMC optimizes the energy expectation directly and leaves all remaining work to the machine learning (ML) infrastructure. Analytic and implementation works can be done automatically with AD infrastructure, the vectorization or broadcast mechanism, built-in optimizers, and GPU acceleration provided by the standard ML framework. For different quantum models, the only difference is different $E_{\mathrm{loc}}(\sigma,\boldsymbol{\theta})$. After implementing E_{loc}, we can bring it into Eq. (26) as an AD-aware energy estimator. Then, we can use AD to compute the gradients and gradient-based optimizer to optimize the energy.

Besides SGD-based optimizers, natural gradient optimizers (SR methods) can also be incorporated into the AD framework. In the context of VMC, the optimization method of natural gradient descent updates the variational parameters as follows: $\Delta\boldsymbol{\theta} = -\lambda F^{-1}\nabla_{\boldsymbol{\theta}}E_{\boldsymbol{\theta}}$, where F can be obtained by Monte Carlo,

$$F_{ij} = \mathrm{Re}\left[\left\langle\frac{\partial_i\psi^*}{\psi^*}\frac{\partial_j\psi}{\psi}\right\rangle_p - \left\langle\frac{\partial_i\psi^*}{\psi^*}\right\rangle_p\left\langle\frac{\partial_j\psi}{\psi}\right\rangle_p\right], \quad (27)$$

where ψ is a shortcut for $\psi_{\boldsymbol{\theta}}(\sigma)$ and dependence on parameters $\boldsymbol{\theta}$ is implicit. Note that Eq. (27) is connected to Eq. (20) when the distribution $p = |\psi|^2$ and ψ is real. The relation between the FIM and SR method with complex wave functions can also be analyzed by generalizing KL divergence in the complex distribution case (see Appendix B 4 for details).

IV. APPLICATIONS

The general theory of ADMC we presented above has broad applications, including achieving high accuracy and efficiency in studying interesting many-body interacting models in physics. As we mentioned earlier, by introducing ADMC, we can leverage not only AD but also other powerful features of machine learning frameworks in traditional Monte Carlo. AD together with vectorization, GPU acceleration, and state-of-the-art optimizers can build faster and more capable Monte Carlo applications to study challenging issues in statistics and physics. Here we present two explicit ADMC applications in studying interacting many-body systems [76] where comparable or higher accuracy can be achieved comparing with previous studies using RBM-based VMC, and tensor network methods.

A. Fast search of phase transitions by ADMC

For many-body systems in physics, it is of central interest to find distinct phases and phase transitions between them. We show that ADMC can provide a general and efficient way

to find phase transitions in many-body interacting models. At a given phase transition, certain quantities such as specific heat and ordering susceptibility reach a maximal value. This naturally enables ADMC to locate the phase transition in a fast and efficient way by searching for the maximum. A phase transition can occur when certain parameters such as temperature, pressure, and magnetic field are tuned across a critical value. ADMC can efficiently find the critical value of tuning parameter, such as the transition temperature.

For concreteness, we use ADMC to find the transition temperature of the 2D Ising model on a square lattice as an example, although the approach we present is general and can be applied to both classical and quantum models. For quantum models, we call the corresponding AD approach AD quantum Monte Carlo (ADQMC). The 2D Ising model is given by $H = -\sum_{\langle ij\rangle} J\sigma_i\sigma_j$, where $\sigma_i = \pm 1$ is the Ising spin on site i of the square lattice and we take $J = 1$ as the energy unit. It is well known that there is a phase transition at critical temperature T_c below which the system orders spontaneously [77]. The Ising model can be MC sampled with unnormalized probability distribution $p(\sigma, T) = \exp(-H(\sigma)/T)$. As specific heat reaches a maximal value at the phase transition, conventional MC methods usually compute specific heat for many temperature points and then locate the peak of the specific heat curve as the phase transition. In these conventional approaches, it requires analytical derivation of the formula for specific heat since MC sampling usually cannot compute the specific heat directly. It is relatively simple for specific heat due to the fluctuation-dissipation theorem, i.e., $C_v(T) = (\langle H^2\rangle_p - \langle H\rangle_p^2)/T$. However, it is generally challenging to analytically derive quantities such as gradient or higher-order derivatives of physical quantities.

ADMC provides a general way to search for phase transition by directly using the specific heat $C_v(T)$ or other physical quantities as the objective function, which avoids the drawback of the conventional MC approaches mentioned above. With the help of ADMC, we can find the peak of the specific heat curve much faster and more efficiently. Without the knowledge of the fluctuation-dissipation theorem, we can find the location of the peak very accurately with the total computation time which is orders of magnitude faster. In ADMC, we first directly compute energy using the AD-aware version of the MC energy estimator as Eq. (8) and then, following the spirit of SGD, we update the temperature (starting from any T_0) based on the second-order derivative of MC expectation energy in every few MC updates:

$$\Delta T \propto \frac{\partial C_v}{\partial T} = \frac{\partial^2 \left\langle \frac{p}{\perp(p)} H \right\rangle \Big/ \left\langle \frac{p}{\perp p} \right\rangle}{\partial T^2}. \tag{28}$$

Although the number of MC updates in each round of temperature update is small, rendering noisy estimation of specific heat, such a noisy gradient estimator can still converge to T_c very quickly. This is the essence of SGD: noisy gradient estimation might lead to better and faster convergence to the minimum or maximum. This is also why a mini-batch gradient estimate is used in general neural network training; for instance, one MC sample for each pass in the training of a variational autoencoder [32] and CD-1 algorithm in RBM training [78] work quite well. Following the same philosophy, we can combine SGD into the ADMC framework applied here. Specifically, to maximize some MC expectation values against variational parameters $\boldsymbol{\theta} = \mathrm{argmax}_{\boldsymbol{\theta}} \langle O(\boldsymbol{x}, \boldsymbol{\theta})\rangle_{p(\boldsymbol{x},\boldsymbol{\theta})}$, we may obtain noisy estimation of the gradients by doing few MC update steps. Such noisy estimation of the gradients can render stable and faster optimizations if the learning rate is small enough.

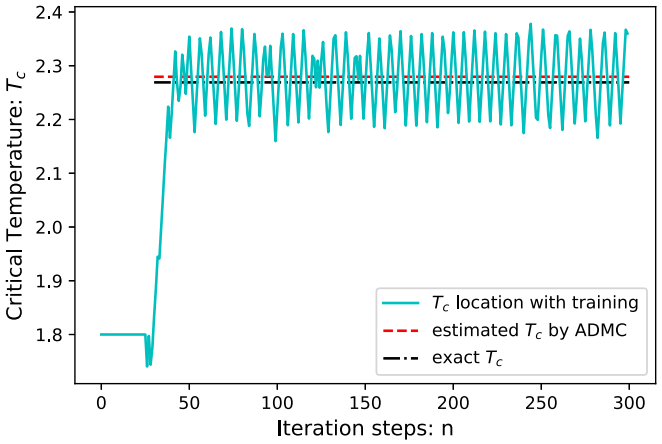

FIG. 2. Fast search for critical temperature T_c by the ADMC approach for the Ising model of lattice size 50×50. The obtained expectation value of T_c from training is 2.279, about 0.4% off from the exact value 2.269 [77]. Considering the short training time and finite size effect, this is a very good estimation of T_c and more accurate results can be obtained by larger system size and smaller learning rate.

Moreover, one can also utilize the third-order derivative of expectation energy and apply the Newton method to update the temperature, which convincingly shows the value of *infinitely* automatic differentiable MC estimators.

In terms of implementation, we also combine vectorization into the ADMC workflow above, which takes the Markov chain as one of the extra dimensions for spin configuration tensors, enabling MC simulation on tens of thousands of Markov chains simultaneously. Such a vectorization scheme is highly efficient compared to conventional parallel schemes, such as one Markov chain per CPU core. Besides, GPU supports such vectorization very well, providing further speed-up. The combination of SGD and vectorized Wolff algorithm leads to relatively accurate estimation of T_c in just a few seconds. The result is summarized in Fig. 2.

We emphasize that the approach we present here is general and can be straightforwardly generalized to other classical or quantum models, where fast estimation on critical values is desired. The knowledge of critical values is helpful to reduce unnecessary calculations on data points deeply in phases and renders a fast search of phase transitions in interacting many-body systems.

B. Accurate search of ground states by ADVMC

The integration of AD with VMC provides a powerful tool to accurately study ground states of many-body quantum models in one and higher dimensions. In particular, ADVMC can be used to study generic quantum models (including those

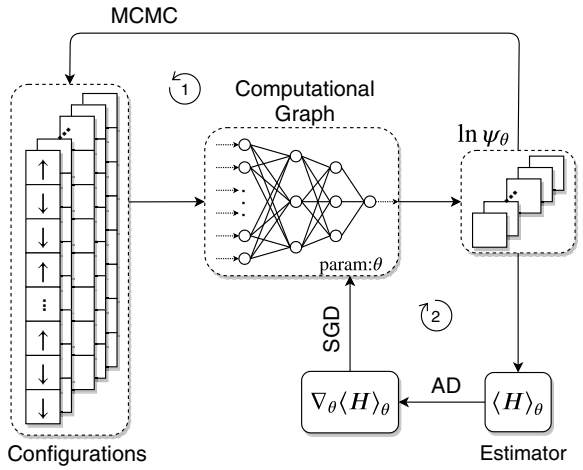

FIG. 3. Schematic illustration of the end-to-end ADVMC. Configurations (spins) are vectorized in an extra dimension as different Markov chains. A computational graph is constructed to give the logarithm of wave function $\ln \psi_{\boldsymbol{\theta}}$ which is also vectorized. Loop 1 is the conventional MCMC approach for updating the configurations according to the Metropolis-Hasting algorithm. Loop 2 is the ADMC approach to evaluate the AD-aware energy estimator and update the parameters $\boldsymbol{\theta}$ by optimizers. One iteration of our algorithm includes many (often the size of the system) configuration updates (loop 1) to decrease the autocorrelation and one step of parameter update (loop 2).

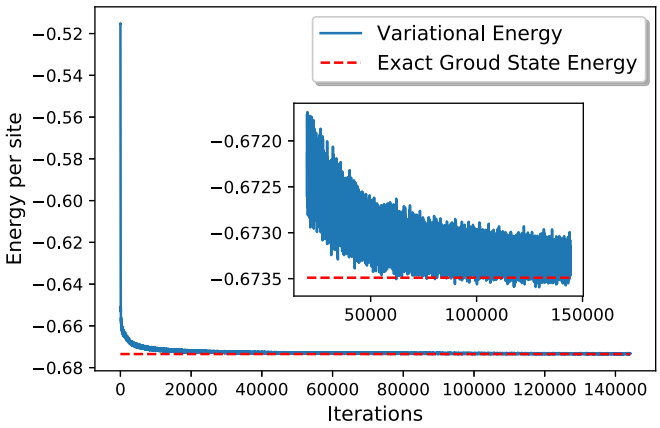

FIG. 4. Numerical results of end-to-end ADVMC approach for the spin-1/2 quantum Heisenberg model on the square lattice with system size 8×8 and periodic boundary condition. The variational wave function was chosen to be a fully connected neural network with seven layers. The number of nodes on each layer is $(16l^2, 8l^2, 4l^2, 8l^2, 4l^2, l^2, 1)$, with $l = 8$. The activations were set to be RELU for all these layers except the last one. The Adam optimizer was used to update the parameters. The dashed line is the benchmark ground state energy given by the stochastic series expansion method. The inset shows the convergence of energy near the exact value. The ADVMC result is energetically competitive with results obtained by various state-of-the-art methods including entangled plaquette statets, projected entangled pair states, and RBM-based VMC.

models with frustration) in two and higher dimensions using general neural-network states as ansatz wave functions.

The workflow of the end-to-end ADVMC is sketched in Fig. 3. In the ADVMC algorithm, we can take advantage of the vectorization technique to watch and update thousands of independent Markov chains in a parallel fashion. As shown in Fig. 3, the (spin) configurations of different Markov chains are vectorized in a new dimension of size n_{mc} (the number of Markov chains). The configurations are sent to an arbitrary computational graph with variational parameters $\boldsymbol{\theta}$ where the logarithm of wave function $\ln \boldsymbol{\psi}_{\boldsymbol{\theta}}(\sigma)$ is evaluated as the output. The computational graph can be constructed by mean-field wave functions with Jastrow factors, matrix product state [79,80], deep neural networks, or any other programs with variational parameters and one scalar output. $\ln \boldsymbol{\psi}_{\boldsymbol{\theta}}$ also has an extra dimension with the same size as n_{mc}. In evaluating the computational graph, the extra dimension behaves as the batch dimension in ML language which can be easily taken care of using the broadcast technique supported by ML. With the knowledge of wave function amplitudes, we can update the configurations using the MCMC method to make them satisfy the distribution given by the computational graph wave function ansatz.

Here we demonstrate this new paradigm of VMC by AD-VMC study of the spin-1/2 quantum Heisenberg model on the square lattice. The model is given by $H = \sum_{\langle ij\rangle} J\vec{S}_i \cdot \vec{S}_j$, where $\vec{S}_j$ is a spin-1/2 operator on site j. Because of the Marshal-sign rule the ground state wave function amplitudes of the Heisenberg model can be rendered positive definite. Consequently, for simplicity we use positive ansatz wave functions in our ADVMC simulations. The computational graph we utilize in this problem is a fully connected neural network with seven layers and with rectified linear unit (RELU) activations [81]. The numbers of nodes on these layers are $16l^2, 8l^2, 4l^2, 8l^2, 4l^2, l^2, 1$, where $l^2 = 64$ is the size of the system. Such neural network design is general without considering any symmetry and geometric information. In total there are more than one million variational parameters and the number of Markov chains is 5000 in our ADVMC simulations. With such a large number of independent Markov chains and variational parameters, the ADVMC implementation is still very efficient on a GPU in terms of time due to the highly parallelized structure of our algorithm.

The approximation ground state energy optimized by Adam converges to -0.6733 (in the unit of J) per site averaged by the last 5000 energy data points, as shown in Fig. 4. This result has 3×10^{-4} relative error compared with the benchmark ground state energy obtained by SSE [82]. It is also energetically competitive with results obtained by various state-of-the-art methods including EPS [83], PEPS [84,85], and RBM-based VMC [58]. This convincingly demonstrates that end-to-end ADVMC can enable us to reach state-of-the-art numerical results with very moderate effort for quantum models.

V. DISCUSSIONS AND CONCLUSIONS

One central issue in Monte Carlo simulations of interacting many-body quantum models is the notorious sign problem. Although it is shown to be NP-hard to solve the sign problem generically [86], it is still possible to ease [87–89] or solve [16] the sign problem of a given specific quantum model in

QMC simulations by certain basis transformations. We propose to employ ADMC as a general way to find the optimal basis which can ease or solve the notorious sign problem in QMC simulations of interesting quantum models. One appropriate objective in ADMC would be the expectation value of the sign, which depends on the parameters characterizing the basis choice. ADMC can help to find an optimal basis in which the sign problem is alleviated. From the ADMC-optimized basis with eased sign problem, one may simulate strongly correlated models with lower temperature and larger system size than QMC with usual basis. In fact, we have successfully applied this idea in mitigating the sign problem in determinant QMC [90].

ADMC proposed in the present paper is based on score function estimators. For the specific models we have studied, the present ADMC obtains accurate results without suffering any high variance in MC estimations. It is possible to further improve ADMC by reducing variance in MC estimations of expectations. In other words, it would be desired to find baselines or general control variables that could systematically reduce the variance of MC estimations. It is one of the future routes to improve ADMC by introducing baselines suitable for any order derivatives as in the case of normalized probability distribution [91].

In conclusion, we have presented the general theory and framework of ADMC. We also showed how Monte Carlo expectations, KL divergence, and objectives from various settings can be expressed in an infinitely AD fashion. We further applied the ADMC approach to various Monte Carlo applications including classical Monte Carlo and end-to-end VMC. We believe that the ADMC approach can inspire more accurate and efficient Monte Carlo designs with machine learning toolboxes in the future. At the intersection of differentiable programming and probabilistic programming, the ADMC framework provides a promising route to advancing Monte Carlo applications in the fields of statistics, machine learning, and physics.

ACKNOWLEDGMENTS

We thank Shuai Chen, Zi-Xiang Li, Jin-Guo Liu, Rong-Yang Sun, and Lei Wang for helpful discussions. This work is supported in part by the NSFC under Grant No. 11825404 (S.-X.Z., Z.-Q.W., and H.Y.) and the Strategic Priority Research Program of Chinese Academy of Sciences under Grant No. XDB28000000 (H.Y.). H.Y. would also like to acknowledge support in part by the Gordon and Betty Moore Foundations EPiQS Initiative through Grant GBMF4302 at Stanford.

APPENDIX A: AUTOMATIC DIFFERENTIATION APPROACH FOR FISHER INFORMATION MATRIX

In this part, we further discuss the implementation details and advantages of AD approach toward FIM.

The test case for algorithm implementations of the FIM we utilized involves simple distributions such as multivariate Gaussian distribution $N(\boldsymbol{\mu}|\boldsymbol{\sigma})$, in which $\boldsymbol{\mu}, \boldsymbol{\sigma}$ depend on variational parameters $\boldsymbol{\theta}$. For the simplest case, $\boldsymbol{\sigma}$ is constant and $\boldsymbol{\mu}(\boldsymbol{\theta})$ is determined by parameters $\boldsymbol{\theta}$. We can obtain an analytical expression for the FIM in this case:

$$F_{ij} = \frac{\partial \boldsymbol{\mu}^T}{\partial \theta_i} \boldsymbol{\sigma}^{-1} \frac{\partial \boldsymbol{\mu}}{\partial \theta_j}. \tag{A1}$$

f we further assume $\boldsymbol{\sigma} = I$ and $\mu_i = \mu(\theta_i)$ is in the same function form, we can further simplify the FIM analytically as

$$F = (\partial \mu)^2 I. \tag{A2}$$

In our code example, we test with a three-dimensional Gaussian distribution and $\mu(\theta) = (\theta + 1)^2$ where $\theta = 0$. The expected FIM should be $4\, I_{3\times 3}$ in this case. Such test cases can also be used for testing the implementation of unnormalized probability cases if the normalization factor of the Gaussian distribution is deliberately dropped out.

The first advantage for the FIM with AD approach is that zero elements might be kept without MC fluctuations or error bars. Taking the test case above for an example, all off-diagonal elements of the FIM should be zero analytically. If one utilized the conventional way of computing the FIM by MC averaging first-order derivatives of $\ln p$, the resulting off-diagonal elements are not zero due to the error bar introduced by MC. However, with advanced graph optimization and smart compiler infrastructure provided by TENSORFLOW, unnecessary computations can be identified and removed from the runtime graph. With such a state-of-the-art executing engine of the computational graph, the off-diagonal terms can be pinned at zero with the AD approach. This is because the zero nature of these terms has already been identified at graph building time by TENSORFLOW engines. That is to say, the numerical result can even reach theoretical precision with the help of AD. It is worth noting that such gain is not guaranteed since the TENSORFLOW engine can fail to recognize complicated series of unnecessary operations. For example, AD with unnormalized probability objectives gives nonzero off-diagonal elements in the FIM using the same Gaussian distribution test case.

The second advantage of the AD approach is the better compatibility with the vectorization scheme. Suppose we vectorize Markov chains as the batch dimension as the case in our implementation of example applications. The conventional way to evaluate the FIM involves terms like $\sum_{x\sim p} \partial_i \ln p(x)\, \partial_j \ln p(x)$ where x are different configurations living on the extra vectorization dimension in our setup. It would be hard to evaluate such terms by treating the batch dimension as a whole where x are different for different chains. This restriction is mainly brought by the modern AD infrastructure of ML libraries in which derivatives for multiple outputs can only be obtained one by one and no tensorized fashion of AD is implemented. Instead, the KL divergence objective only concerns terms like $\sum_{x\sim p} \frac{p}{\perp p}$ which is very easy to parallelize by a simple reduce mean. The computation time of the conventional approach scales with the number of Markov chains or configuration samples, N, which is typically thousands to millions, while the computation time of the AD approach scales with the parameter number (one has to apply AD on each derivative to get the Hessian in ML libraries) which could be way less than the configuration numbers vectorized in the batch dimension. And our numerical experi-

ments indeed show that the AD approach is clearly faster than the conventional approach either in graph building time or in graph executing time.

APPENDIX B: END-TO-END ADVMC SETUP FOR GENERAL COMPLEX WAVE FUNCTIONS

1. Computational graph setup for general wave function

If the ground state wave function is not always real positive, the general form can be expressed as $\psi_\sigma = e^{r_\sigma} e^{i\theta_\sigma}$, where r characterizes the real norm part $\ln|\psi|$ and θ characterizes the complex angle for the wave function. Therefore, we need two separate computational graphs for computing r and θ, and train them together towards minimal energy. We discuss about the most reliable form of AD-aware energy estimators and the assistant estimator for natural gradients in the following.

2. Infinite-order AD estimator for VMC

The reason why VMC works is due to the following fact: the quantum expectation energy can be approximated by classical Monte Carlo–averaged $E_{\rm loc}$:

$$\begin{aligned}\langle H\rangle &= \frac{\sum_{\sigma\sigma'}\psi_\sigma^* H_{\sigma\sigma'}\psi_{\sigma'}}{\sum_\sigma \psi_\sigma^*\psi_\sigma} = \frac{\sum_{\sigma\sigma'}\psi_\sigma^*\psi_\sigma(H_{\sigma\sigma'}\psi_{\sigma'}/\psi_\sigma)}{\sum_\sigma \psi_\sigma^*\psi_\sigma}\\ &= \frac{\sum_\sigma \psi_\sigma^*\psi_\sigma \sum_{\sigma'}(H_{\sigma\sigma'}\psi_{\sigma'}/\psi_\sigma)}{\sum_\sigma \psi_\sigma^*\psi_\sigma}\\ &= \frac{\sum_\sigma \psi_\sigma^*\psi_\sigma E_{\rm loc}(\sigma)}{\sum_\sigma \psi_\sigma^*\psi_\sigma} = {\rm Re}\frac{\sum_\sigma \psi_\sigma^*\psi_\sigma E_{\rm loc}(\sigma)}{\sum_\sigma \psi_\sigma^*\psi_\sigma}\\ &= \frac{\sum_\sigma \psi_\sigma^*\psi_\sigma {\rm Re}E_{\rm loc}(\sigma)}{\sum_\sigma \psi_\sigma^*\psi_\sigma},\end{aligned} \tag{B1}$$

where $E_{\rm loc}(\sigma) = \sum_{\sigma'}(H_{\sigma\sigma'}\psi_{\sigma'}/\psi_\sigma)$, and the summation over σ' can be done efficiently because $H_{\sigma\sigma'}$ is a sparse matrix for the general local Hamiltonian. If we treat $\psi_\sigma^*\psi_\sigma$ as the classical probability $p(\sigma)$, then we have

$$\langle H\rangle = \sum_\sigma p(\sigma)({\rm Re}E_{\rm loc}(\sigma))/\sum_\sigma p(\sigma) = \langle {\rm Re}E_{\rm loc}(\sigma)\rangle_{\sigma\sim p(\sigma)}, \tag{B2}$$

which indicates $\langle H\rangle$ is just the expected value of ${\rm Re}E_{\rm loc}(\sigma)$ when σ is sampled from an unnormalized distribution $p(\sigma)$. For this problem, our ADMC approach gives an accurate infinite-order AD-aware estimator of it:

$$\mathbf{Es}\langle H\rangle = \frac{\sum_{\sigma\in S(p)} \frac{\psi_\sigma^*\psi_\sigma}{\perp(\psi_\sigma^*\psi_\sigma)} {\rm Re}E_{\rm loc}(\sigma)}{\sum_{\sigma\in S(p)} \frac{\psi_\sigma^*\psi_\sigma}{\perp(\psi_\sigma^*\psi_\sigma)}}. \tag{B3}$$

Here $\sum_{\sigma\in S(p)}$ means doing summation on the set of configurations σ sampled from distribution $p = \psi_\sigma^*\psi_\sigma$ using the MCMC method. This estimator is correct for arbitrary-order derivatives no matter whether the wave function ansatz is real or not. Nevertheless, we can design more efficient estimators in the VMC context with lower variance and better optimization results as we will show below.

3. First-order efficient AD estimator for VMC

In most of the cases, the knowledge about the gradients (first-order derivatives) of $\langle H\rangle$ is enough, while our estimator in Eq. (B3) is correct for any order of derivatives. There is the possibility that we can further increase our precision if we focus on the first-order derivative.

By analytically deriving the gradients of $\langle H\rangle$ from a quantum expectation perspective, we have

$$\begin{aligned}\nabla\langle H\rangle &= \nabla\frac{\sum_{\sigma\sigma'}\psi_\sigma^* H_{\sigma\sigma'}\psi_{\sigma'}}{\sum_\sigma \psi_\sigma^*\psi_\sigma} = \frac{\sum_{\sigma\sigma'}(\nabla\psi_\sigma^*)H_{\sigma\sigma'}\psi_{\sigma'} + \psi_\sigma^* H_{\sigma\sigma'}(\nabla\psi_{\sigma'})}{\sum_\sigma \psi_\sigma^*\psi_\sigma} - \frac{\sum_{\sigma\sigma'}\psi_\sigma^* H_{\sigma\sigma'}\psi_{\sigma'}}{\sum_\sigma \psi_\sigma^*\psi_\sigma}\frac{\nabla\sum_\sigma \psi_\sigma^*\psi_\sigma}{\sum_\sigma \psi_\sigma^*\psi_\sigma}\\ &\overset{H_{\sigma\sigma'}=H_{\sigma'\sigma}^*}{\Rightarrow} \frac{\sum_\sigma(\psi_\sigma^*\psi_\sigma)\left(\frac{\nabla\psi_\sigma^*}{\psi_\sigma^*}E_{\rm loc}(\sigma) + \frac{\nabla\psi_\sigma}{\psi_\sigma}E_{\rm loc}^*(\sigma)\right)}{\sum_\sigma \psi_\sigma^*\psi_\sigma} - \frac{\sum_{\sigma\sigma'}\psi_\sigma^* H_{\sigma\sigma'}\psi_{\sigma'}}{\sum_\sigma \psi_\sigma^*\psi_\sigma}\frac{\sum_\sigma(\psi_\sigma^*\psi_\sigma)\left(\frac{\nabla\psi_\sigma^*}{\psi_\sigma^*} + \frac{\nabla\psi_\sigma}{\psi_\sigma}\right)}{\sum_\sigma \psi_\sigma^*\psi_\sigma}\\ &= 2{\rm Re}\left(\frac{\sum_\sigma \psi_\sigma^*\psi_\sigma\frac{\nabla\psi_\sigma^*}{\psi_\sigma^*}E_{\rm loc}(\sigma)}{\sum_\sigma \psi_\sigma^*\psi_\sigma} - \frac{\sum_\sigma \psi_\sigma^*\psi_\sigma E_{\rm loc}(\sigma)}{\sum_\sigma \psi_\sigma^*\psi_\sigma}\frac{\sum_\sigma \psi_\sigma^*\psi_\sigma\frac{\nabla\psi_\sigma^*}{\psi_\sigma^*}}{\sum_\sigma \psi_\sigma^*\psi_\sigma}\right).\end{aligned} \tag{B4}$$

Note all the terms are in the form of $\frac{\sum_\sigma \psi_\sigma^*\psi_\sigma O(\sigma)}{\sum_\sigma \psi_\sigma^*\psi_\sigma}$, as this kind of term can be estimated by $\sum_{\sigma\in S(p)} O(\sigma)/N_{mc}$, where N_{mc} is the number of Monte Carlo sampling. Thus we have

$$\nabla\langle H\rangle \doteq 2{\rm Re}\left(\frac{\sum\frac{\nabla\psi_\sigma^*}{\psi_\sigma^*}E_{\rm loc}(\sigma)}{N_{mc}} - \frac{\sum E_{\rm loc}(\sigma)}{N_{mc}}\frac{\sum\frac{\nabla\psi_\sigma^*}{\psi_\sigma^*}}{N_{mc}}\right). \tag{B5}$$

From the MC expectation perspective, we calculate the gradients of the general estimator in Eq. (B3); the result is not the same as Eq. (B5). The difference terms are

$$\begin{aligned}\mathbf{diff} &= \frac{1}{N_{mc}}{\rm Re}\sum_{\sigma\in S(p)}\left(\frac{\nabla\psi_\sigma^*}{\psi_\sigma^*}E_{\rm loc} - \frac{\nabla\psi_\sigma}{\psi_\sigma}E_{\rm loc} - \nabla E_{\rm loc}\right)\\ &= \frac{1}{N_{mc}}{\rm Re}\sum_{\sigma\in S(p)}\left(\frac{\nabla\psi_\sigma^*}{\psi_\sigma^*}E_{\rm loc} - \frac{\nabla\psi_\sigma}{\psi_\sigma}E_{\rm loc} - \nabla\sum_{\sigma'}H_{\sigma\sigma'}\frac{\psi_{\sigma'}}{\psi_\sigma}\right) = \frac{1}{N_{mc}}{\rm Re}\sum_{\sigma\in S(p)}\left(\frac{\nabla\psi_\sigma^*}{\psi_\sigma^*}E_{\rm loc} - \sum_{\sigma'}H_{\sigma\sigma'}\frac{\nabla\psi_{\sigma'}}{\psi_\sigma}\right).\end{aligned} \tag{B6}$$

diff normally is not zero numerically, but it goes to zero when N_{mc} goes to infinity as it should be. This is because H is Hermitian:

$$\begin{aligned}
&\frac{1}{N_{mc}}\mathrm{Re}\sum_{\sigma\in S(p)}\sum_{\sigma'}H_{\sigma\sigma'}\frac{\nabla\psi_{\sigma'}}{\psi_\sigma}\overset{N_{mc}\to\infty}{=}\mathrm{Re}\sum_{\sigma\sigma'}\psi^*_\sigma\psi_\sigma H_{\sigma\sigma'}\frac{\nabla\psi_{\sigma'}}{\psi_\sigma}\Big/\sum_\sigma\psi^*_\sigma\psi_\sigma\\
&=\mathrm{Re}\sum_{\sigma\sigma'}\psi^*_{\sigma'}\psi_{\sigma'}\frac{\nabla\psi_{\sigma'}}{\psi_{\sigma'}}H_{\sigma\sigma'}\frac{\psi^*_\sigma}{\psi^*_{\sigma'}}\Big/\sum_{\sigma'}\psi^*_{\sigma'}\psi_{\sigma'}\overset{H_{\sigma\sigma'}=H^*_{\sigma'\sigma}}{=}\mathrm{Re}\sum_{\sigma'}\psi^*_{\sigma'}\psi_{\sigma'}\frac{\nabla\psi_{\sigma'}}{\psi_{\sigma'}}\sum_\sigma\left(H_{\sigma'\sigma}\frac{\psi_\sigma}{\psi_{\sigma'}}\right)^*\Big/\sum_{\sigma'}\psi^*_{\sigma'}\psi_{\sigma'}\\
&=\mathrm{Re}\sum_{\sigma'}\psi^*_{\sigma'}\psi_{\sigma'}\frac{\nabla\psi_{\sigma'}}{\psi_{\sigma'}}E^*_{\mathrm{loc}}(\sigma')\Big/\sum_{\sigma'}\psi^*_{\sigma'}\psi_{\sigma'}=\frac{1}{N_{mc}}\mathrm{Re}\sum_{\sigma\in S(p)}\frac{\nabla\psi^*_\sigma}{\psi^*_\sigma}E_{\mathrm{loc}}.
\end{aligned}\tag{B7}$$

Thus we proved $\lim_{N_{mc}\to\infty}\mathbf{diff}=0$. In other words, If we directly use Eq. (B3) as the estimator, AD will give the gradients with another term whose expectation is theoretically zero. Since this term is in general nonzero in MC calculations, it adds more variance to the estimation of energy. We can safely drop the **diff** term from the philosophy of the baseline method. In other words, it would be better to find the estimator whose first-order derivative is directly Eq. (B5) without the **diff** term.

Such an estimator is easy to construct:

$$\mathbf{Es_1}\langle H\rangle=2\mathrm{Re}\frac{\sum_{\sigma\in S(p)}\frac{\psi^*_\sigma}{\perp\psi^*_\sigma}\perp E_{\mathrm{loc}}(\sigma)}{\sum_{\sigma\in S(p)}\frac{\psi^*_\sigma}{\perp\psi^*_\sigma}}.\tag{B8}$$

We can prove it by directly calculating the gradient of this estimator:

$$\begin{aligned}
\nabla\mathbf{Es_1}\langle\mathbf{H}\rangle=&\,2\mathrm{Re}\frac{\sum_{\sigma\in S(p)}\frac{\nabla\psi^*_\sigma}{\perp\psi^*_\sigma}\perp E_{\mathrm{loc}}(\sigma)\sum_{\sigma\in S(p)}\frac{\psi^*_\sigma}{\perp\psi^*_\sigma}}{\left(\sum_{\sigma\in S(p)}\frac{\psi^*_\sigma}{\perp\psi^*_\sigma}\right)^2}\\
&-2\mathrm{Re}\frac{\sum_{\sigma\in S(p)}\frac{\psi^*_\sigma}{\perp\psi^*_\sigma}\perp E_{\mathrm{loc}}(\sigma)\sum_{\sigma\in S(p)}\frac{\nabla\psi^*_\sigma}{\perp\psi^*_\sigma}}{\left(\sum_{\sigma\in S(p)}\frac{\psi^*_\sigma}{\perp\psi^*_\sigma}\right)^2}.
\end{aligned}\tag{B9}$$

In the sense of its numerical value

$$\begin{aligned}
\perp\nabla\mathbf{Es_1}\langle\mathbf{H}\rangle=&\perp 2\mathrm{Re}\frac{\sum_{\sigma\in S(p)}\frac{\nabla\psi^*_\sigma}{\psi^*_\sigma}E_{\mathrm{loc}}(\sigma)}{N_{mc}}\\
&-\perp 2\mathrm{Re}\frac{\sum_{\sigma\in S(p)}E_{\mathrm{loc}}(\sigma)}{N_{mc}}\frac{\sum_{\sigma\in S(p)}\frac{\nabla\psi^*_\sigma}{\psi^*_\sigma}}{N_{mc}},
\end{aligned}\tag{B10}$$

which is just the same as Eq. (B5). Thus the AD-aware estimator in Eq. (B8) gives the right approximation of the gradients of $\langle H\rangle$ with lower variance than the general estimator, Eq. (B3). Though it is only valid for the first order derivatives.

If the wave function is real, Eq. (B8) reduce to

$$\mathbf{Es}_{1r}=\frac{\sum_{\sigma\in S(p)}\frac{p_\sigma}{\perp p_\sigma}\mathrm{Re}\perp E_{\mathrm{loc}}(\sigma)}{\sum_{\sigma\in S(p)}\frac{p_\sigma}{\perp p_\sigma}}.\tag{B11}$$

One can again verify it by directly differentiating on Eq. (B11). The only change in Eq. (B11) is detached E_{loc} comparing with the general estimator, Eq. (B3). The new estimator also has better performance when running the computational graph since E_{loc} is detached and no backward propagation passes through it.

4. SR and natural gradients

The SR method (natural gradient descent) is reported to give faster convergence on VMC. In this part, we explore the relation between natural gradient descent and the SR method in general settings where the wave function could be complex.

For real wave function cases, KL divergence plays the role of the distance of distribution space whose Hessian FIM gives the same formalism as the SR method as we have shown in the main text. In terms of complex cases, traditional KL divergence defined with $p=\psi^*\psi$ loses the information of wave function's phases. Thus we need a better distance measure to describe the difference between different wave functions.

The natural choice is the Fubini-Study distance defined in the Hilbert space:

$$s(\psi,\phi)=\arccos\sqrt{\frac{\langle\psi|\phi\rangle\langle\phi|\psi\rangle}{\langle\psi|\psi\rangle\langle\phi|\phi\rangle}}.\tag{B12}$$

Infinitesimal distances are thus given by

$$\begin{aligned}
ds^2=s(\psi,\psi+\delta\psi)^2=&\frac{\langle\delta\psi|\delta\psi\rangle}{\langle\psi|\psi\rangle}-\frac{\langle\delta\psi|\psi\rangle}{\langle\psi|\psi\rangle}\frac{\langle\psi|\delta\psi\rangle}{\langle\psi|\psi\rangle}\\
=&\left\langle\frac{\delta\psi_\sigma}{\psi_\sigma}\frac{\delta\psi^*_\sigma}{\psi^*_\sigma}\right\rangle_p-\left\langle\frac{\delta\psi_\sigma}{\psi_\sigma}\right\rangle_p\left\langle\frac{\delta\psi^*_\sigma}{\psi^*_\sigma}\right\rangle_p,
\end{aligned}\tag{B13}$$

where $\delta\psi=\partial_i\psi d\theta_i$ and $\langle O\rangle_p=\frac{\sum_\sigma|\psi_\sigma|^2O(\sigma)}{\sum_\sigma|\psi_\sigma|^2}$.

Thus we have

$$\begin{aligned}
ds^2=&\sum_{\alpha,\beta}\left(\left\langle\frac{\partial_\alpha\psi_\sigma}{\psi_\sigma}\frac{\partial_\beta\psi^*_\sigma}{\psi^*_\sigma}\right\rangle_p-\left\langle\frac{\partial_\alpha\psi_\sigma}{\psi_\sigma}\right\rangle_p\left\langle\frac{\partial_\beta\psi^*_\sigma}{\psi^*_\sigma}\right\rangle_p\right)d\theta_\alpha d\theta_\beta\\
=&\sum_{\alpha,\beta}\mathrm{Re}\left(\left\langle\frac{\partial_\alpha\psi_\sigma}{\psi_\sigma}\frac{\partial_\beta\psi^*_\sigma}{\psi^*_\sigma}\right\rangle_p-\left\langle\frac{\partial_\alpha\psi_\sigma}{\psi_\sigma}\right\rangle_p\left\langle\frac{\partial_\beta\psi^*_\sigma}{\psi^*_\sigma}\right\rangle_p\right)d\theta_\alpha d\theta_\beta\\
=&\sum_{\alpha,\beta}S_{\alpha\beta}d\theta_\alpha d\theta_\beta,
\end{aligned}\tag{B14}$$

where

$$S_{\alpha\beta}=\mathrm{Re}\left(\left\langle\frac{\partial_\alpha\psi_\sigma}{\psi_\sigma}\frac{\partial_\beta\psi^*_\sigma}{\psi^*_\sigma}\right\rangle_p-\left\langle\frac{\partial_\alpha\psi_\sigma}{\psi_\sigma}\right\rangle_p\left\langle\frac{\partial_\beta\psi^*_\sigma}{\psi^*_\sigma}\right\rangle_p\right)\tag{B15}$$

is identical to the quantum version of the Fisher information matrix utilized in the SR method.

Equation (B15) can be estimated by the ADMC approach, that is,

$$\begin{aligned} S_{\alpha\beta} &\doteq \mathrm{Re}\left(\left\langle \frac{\partial_\alpha \psi_\sigma}{\psi_\sigma}\frac{\partial_\beta \psi_\sigma^*}{\psi_\sigma^*}\right\rangle_p - \left\langle \frac{\partial_\alpha \psi_\sigma}{\psi_\sigma}\right\rangle_p \left\langle \frac{\partial_\beta \psi_\sigma^*}{\psi_\sigma^*}\right\rangle_p\right) \\ &= \mathrm{Re}(\langle \partial_\alpha \ln \psi_\sigma \partial_\beta \ln \psi_\sigma^*\rangle_p - \langle \partial_\alpha \ln \psi_\sigma\rangle_p \langle \partial_\beta \ln \psi_\sigma^*\rangle_p) \\ &= \langle \partial_\alpha \ln |\psi_\sigma| \partial_\beta \ln |\psi_\sigma|\rangle_p - \langle \partial_\alpha \ln |\psi_\sigma|\rangle_p \langle \partial_\beta \ln |\psi_\sigma|\rangle_p \\ &\quad + \langle \partial_\alpha \theta_\sigma \partial_\beta \theta_\sigma\rangle_p - \langle \partial_\alpha \theta_\sigma\rangle_p \langle \partial_\beta \theta_\sigma\rangle_p. \end{aligned} \tag{B16}$$

Using the detach function, we also have the relationship already utilized in FIM formalisms with unnormalized distribution p:

$$\begin{aligned} &\perp \partial_{\alpha\beta}^2 \left(\ln \left\langle \frac{O_\sigma}{\perp O_\sigma}\right\rangle_{\sigma\in S(p)} - \left\langle \ln \frac{O_\sigma}{\perp O_\sigma}\right\rangle_{\sigma\in S(p)}\right) \\ &= \perp \partial_\alpha \left(\frac{\left\langle \frac{\partial_\beta O_\sigma}{\perp O_\sigma}\right\rangle_{\sigma\in S(p)}}{\left\langle \frac{O_\sigma}{\perp O_\sigma}\right\rangle_{\sigma\in S(p)}} - \left\langle \frac{\partial_\beta O_\sigma}{O_\sigma}\right\rangle_{\sigma\in S(p)}\right) \\ &= \perp(\langle \partial_\alpha O_\sigma \partial_\beta O_\sigma\rangle_{\sigma\in S(p)} - \langle \partial_\alpha O_\sigma\rangle_{\sigma\in S(p)} \langle \partial_\beta O_\sigma\rangle_{\sigma\in S(p)}). \end{aligned} \tag{B17}$$

Note the detach function at the beginning used to emphasize this relationship is only true in value for the second derivatives of the left-hand side.

Consider the general computational graph setup with two graphs r and θ which gives $\psi_\sigma = e^{r_\sigma} e^{i\theta_\sigma}$. Using this relationship, Eq. (B16) can be calculated as the Hessian of an AD-aware estimator. The estimator is

$$\begin{aligned} \mathbf{Es}_{ng} = &\ln \left\langle \frac{r_\sigma}{\perp r_\sigma}\right\rangle_{\sigma\in S(p)} - \left\langle \ln \frac{r_\sigma}{\perp r_\sigma}\right\rangle_{\sigma\in S(p)} \\ &+ \ln \left\langle \frac{\theta_\sigma}{\perp \theta_\sigma}\right\rangle_{\sigma\in S(p)} - \left\langle \ln \frac{\theta_\sigma}{\perp \theta_\sigma}\right\rangle_{\sigma\in S(p)}. \end{aligned} \tag{B18}$$

This estimator can be viewed as the generalized version of KL divergence in complex distribution space.

For the real positive wave function case, there will be no θ term; the Hessian of $\mathbf{Es}_{ng}$ is just $\mathbf{FIM}/4$, where $\mathbf{FIM}$ is the classical Fisher information matrix, i.e., the Hessian of conventional KL divergence.

To summarize, the natural distance measure in wave function Hilbert space is the Fubini-Study distance as in Eq. (B12). The Hessian of it gives the inverse matrix to be applied before the gradients utilized in the natural gradient method. From the implementation perspective, such distance can be substituted by the extended version of KL divergence as in Eq. (B18). In this context, the Hessian of extended KL estimator, quantum version of FIM, the Hessian of the Fubini-Study distance, and the matrix required in the SR method are literally the same thing. All these objects are connected with each other and it is interesting to see how the SR method can emerge in the context of natural gradient descent from information geometry without knowledge about imaginary time evolution by the Schrödinger equation.

It is worth noting that, in the SR method, we need to inverse the FIM for natural gradient descent. However, the FIM is often peculiar for very large condition number, rendering the inverse of the FIM numerically unstable. The singular spectrum of the FIM has been investigated very recently [45,57]. From the implementation perspective, the most simple workaround is adding εI on F before the inverse.

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

[2] A. G. Baydin, B. A. Pearlmutter, A. A. Radul, and J. M. Siskind, Automatic differentiation in machine learning: A survey, J. Mach. Learn. Res. **18**, 153 (2017).

[3] H.-J. Liao, J.-G. Liu, L. Wang, and T. Xiang, Differentiable Programming Tensor Networks, Phys. Rev. X **9**, 031041 (2019).

[4] G. Torlai, J. Carrasquilla, M. T. Fishman, R. G. Melko, and M. P. A. Fisher, Wave-function positivization via automatic differentiation, Phys. Rev. Res. **2**, 032060(R) (2020).

[5] C. Hubig, Use and implementation of autodifferentiation in tensor network methods with complex scalars, arXiv:1907.13422

[6] Z.-Q. Wan and S.-X. Zhang, Automatic differentiation for complex valued SVD, arXiv:1909.02659.

[7] S. Mohamed, M. Rosca, M. Figurnov, and A. Mnih, Monte Carlo gradient estimation in machine learning, J. Mach. Learn. Res. **21**, 1 (2020).

[8] W. K. Hastings, Monte Carlo sampling methods using Markov chains and their applications, Biometrika **57**, 97 (1970).

[9] J.-W. van de Meent, B. Paige, H. Yang, and F. Wood, An introduction to probabilistic programming, arXiv:1809.10756.

[10] J. N. Foerster, G. Farquhar, M. Al-Shedivat, T. Rocktäschel, E. P. Xing, and S. Whiteson, DICE: The infinitely differentiable Monte Carlo estimator, in *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden 10-15 July 2018*, edited by J. G. Dy and A. Krause, Proceedings of Machine Learning Research, Vol. 80 (PMLR, 2018), pp. 1524–1533.

[11] J. Schulman, N. Heess, T. Weber, and P. Abbeel, Gradient estimation using stochastic computation graphs, Adv. Neural Inf. Process. Syst. **28**, 3528 (2015).

[12] E. Y. Loh, J. E. Gubernatis, R. T. Scalettar, S. R. White, D. J. Scalapino, and R. L. Sugar, Sign problem in the numerical simulation of many-electron systems, Phys. Rev. B **41**, 9301 (1990).

[13] C. Wu and S.-C. Zhang, Sufficient condition for absence of the sign problem in the fermionic quantum Monte Carlo algorithm, Phys. Rev. B **71**, 155115 (2005).

[14] E. Berg, M. A. Metlitski, and S. Sachdev, Sign-problem-free quantum Monte Carlo of the onset of antiferromagnetism in metals, Science **338**, 1606 (2012).

[15] E. F. Huffman and S. Chandrasekharan, Solution to sign problems in half-filled spin-polarized electronic systems, Phys. Rev. B **89**, 111101(R) (2014).

[16] Z.-X. Li, Y.-F. Jiang, and H. Yao, Solving the fermion sign problem in quantum Monte Carlo simulations by Majorana representation, Phys. Rev. B **91**, 241117(R) (2015).

[17] L. Wang, Y.-H. Liu, M. Iazzi, M. Troyer, and G. Harcos, Split Orthogonal Group: A Guiding Principle for Sign-Problem-Free Fermionic Simulations, Phys. Rev. Lett. **115**, 250601 (2015).

[18] Z.-X. Li, Y.-F. Jiang, and H. Yao, Majorana-Time-Reversal Symmetries: A Fundamental Principle for Sign-Problem-Free Quantum Monte Carlo Simulations, Phys. Rev. Lett. **117**, 267002 (2016).

[19] Z. C. Wei, C. Wu, Y. Li, S. Zhang, and T. Xiang, Majorana Positivity and the Fermion Sign Problem of Quantum Monte Carlo Simulations, Phys. Rev. Lett. **116**, 250601 (2016).

[20] P. Broecker, J. Carrasquilla, R. G. Melko, and S. Trebst, Machine learning quantum phases of matter beyond the fermion sign problem, Sci. Rep. **7**, 8823 (2017).

[21] Z.-X. Li and H. Yao, Sign-problem-free fermionic quantum Monte Carlo: Developments and applications, Annu. Rev. Condens. Matter Phys. **10**, 337 (2019).

[22] R. Blankenbecler, D. J. Scalapino, and R. L. Sugar, Monte Carlo calculations of coupled boson-fermion systems. I, Phys. Rev. D **24**, 2278 (1981).

[23] D. Ceperley and B. Alder, Quantum Monte Carlo, Science **231**, 555 (1986).

[24] A. W. Sandvik, Stochastic method for analytic continuation of quantum Monte Carlo data, Phys. Rev. B **57**, 10287 (1998).

[25] N. Prokof'ev, B. Svistunov, and I. Tupitsyn, "Worm" algorithm in quantum Monte Carlo simulations, Phys. Lett. A **238**, 253 (1998).

[26] E. Gull, A. J. Millis, A. I. Lichtenstein, A. N. Rubtsov, M. Troyer, and P. Werner, Continuous-time Monte Carlo methods for quantum impurity models, Rev. Mod. Phys. **83**, 349 (2011).

[27] C. C. Margossian, A review of automatic differentiation and its efficient implementation, Wiley Interdiscip. Rev.: Data Min. Knowl. Discovery **9**, e1305 (2019).

[28] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, Learning representations by back-propagating errors, Nature (London) **323**, 533 (1986).

[29] Y. LeCun, Y. Bengio, and G. Hinton, Deep learning, Nature (London) **521**, 436 (2015).

[30] J. P. Kleijnen and R. Y. Rubinstein, Optimization and sensitivity analysis of computer simulation models by the score function method, Eur. J. Oper. Res. **88**, 413 (1996).

[31] R. J. Williams, Simple statistical gradient-following algorithms for connectionist reinforcement learning, Mach. Learn. **8**, 229 (1992).

[32] D. P. Kingma and M. Welling, Auto-encoding variational bayes, in *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, 14-16 April 2014*, edited by Y. Bengio and Y. LeCun, Conference Track Proceedings (ICLR, 2014).

[33] D. J. Rezende, S. Mohamed, and D. Wierstra, Stochastic backpropagation and approximate inference in deep generative models, in *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*, JMLR Workshop and Conference Proceedings, Vol. 32 (JMLR.org, 2014), pp. 1278–1286.

[34] R. Y. Rubinstein, Sensitivity analysis of discrete event systems by the "push out" method, Ann. Oper. Res. **39**, 229 (1992).

[35] W. L. McMillan, Ground state of liquid He^4, Phys. Rev. **138**, A442 (1965).

[36] D. Ceperley, G. V. Chester, and M. H. Kalos, Monte Carlo simulation of a many-fermion study, Phys. Rev. B **16**, 3081 (1977).

[37] H. Robbins and S. Monro, A stochastic approximation method, Ann. Math. Stat. **22**, 400 (1951).

[38] J. Kiefer and J. Wolfowitz, Stochastic estimation of the maximum of a regression function, Ann. Math. Stat. **23**, 462 (1952).

[39] L. Bottou, F. E. Curtis, and J. Nocedal, Optimization methods for large-scale machine learning, SIAM Rev. **60**, 223 (2018).

[40] A. Harju, B. Barbiellini, S. Siljamäki, R. M. Nieminen, and G. Ortiz, Stochastic Gradient Approximation: An Efficient Method to Optimize Many-Body Wave Functions, Phys. Rev. Lett. **79**, 1173 (1997).

[41] D. P. Kingma and J. Ba, Adam: A method for stochastic optimization, in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, 7-9 May 2015*, edited by Y. Bengio and Y. LeCun, Conference Track Proceedings (ICLR, San Diego, 2015).

[42] S.-i. Amari, Natural gradient works efficiently in learning, Neural Comput. **10**, 251 (1998).

[43] R. Pascanu and Y. Bengio, Revisiting natural gradient for deep networks, arXiv:1301.3584.

[44] J. Martens, New insights and perspectives on the natural gradient method, J. Mach. Learn. Res. **21**, 146 (2020).

[45] R. Karakida, S. Akaho, and S.-i. Amari, Pathological spectra of the Fisher information metric and its variants in deep neural networks, Neural Comput. **33**, 2274 (2021).

[46] R. B. Grosse and R. Salakhutdinov, Scaling up natural gradient by sparsely factorizing the inverse fisher matrix, in *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, edited by F. R. Bach and D. M. Blei, JMLR Workshop and Conference Proceedings, Vol. 37 (JMLR.org, 2015), pp. 2304–2313.

[47] J. Martens and R. Grosse, Optimizing neural networks with Kronecker-factored approximate curvature, in *Proceedings of the 32nd International Conference on Machine Learning, Lille, France*, Vol. 37 (JMLR.org, 2015), pp. 2408–2417.

[48] R. B. Grosse and J. Martens, A kronecker-factored approximate fisher matrix for convolution layers, in *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, 19-24 June 2016*, edited by M. Balcan and K. Q. Weinberger, JMLR Workshop and Conference Proceedings, Vol. 48 (JMLR.org, 2016), pp. 573–582.

[49] J. Ba, R. B. Grosse, and J. Martens, Distributed second-order optimization using kronecker-factored approximations, in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, 24-26 April 2017*, Conference Track Proceedings, (OpenReview.net, 2017).

[50] J. Martens, J. Ba, and M. Johnson, Kronecker-factored curvature approximations for recurrent neural networks, in *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, 30 April-3 May 2018*, Conference Track Proceedings (OpenReview.net, 2018).

[51] K. Osawa, Y. Tsuji, Y. Ueno, A. Naruse, R. Yokota, and S. Matsuoka, Large-scale distributed second-order optimization using kronecker-factored approximate curvature for deep convolutional neural networks, in *IEEE Conference on Computer*

Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20 June 2019 (Computer Vision Foundation/IEEE, 2019), pp. 12359–12367.

[52] S. Sorella, Green Function Monte Carlo with Stochastic Reconfiguration, Phys. Rev. Lett. **80**, 4558 (1998).

[53] S. Sorella, Generalized Lanczos algorithm for variational quantum Monte Carlo, Phys. Rev. B **64**, 024512 (2001).

[54] Y. Nomura, A. S. Darmawan, Y. Yamaji, and M. Imada, Restricted Boltzmann machine learning for solving strongly correlated quantum systems, Phys. Rev. B **96**, 205152 (2017).

[55] I. Glasser, N. Pancotti, M. August, I. D. Rodriguez, and J. I. Cirac, Neural-Network Quantum States, String-Bond States, and Chiral Topological States, Phys. Rev. X **8**, 011006 (2018).

[56] D. Pfau, J. S. Spencer, A. G. D. G. Matthews, and W. M. C. Foulkes, *Ab initio* solution of the many-electron Schrödinger equation with deep neural networks, Phys. Rev. Res. **2**, 033429 (2020).

[57] C.-Y. Park and M. J. Kastoryano, Geometry of learning neural quantum states, Phys. Rev. Res. **2**, 023232 (2020).

[58] G. Carleo and M. Troyer, Solving the quantum many-body problem with artificial neural networks, Science **355**, 602 (2017).

[59] D.-L. Deng, X. Li, and S. Das Sarma, Machine learning topological states, Phys. Rev. B **96**, 195145 (2017).

[60] G. Carleo, Y. Nomura, and M. Imada, Constructing exact representations of quantum many-body systems with deep neural networks, Nat. Commun. **9**, 5322 (2018).

[61] Z. Cai and J. Liu, Approximating quantum many-body wave functions using artificial neural networks, Phys. Rev. B **97**, 035116 (2018).

[62] D. Kochkov and B. K. Clark, Variational optimization in the AI era: Computational graph states and supervised wave-function optimization, arXiv:1811.12423.

[63] R. Kaubruegger, L. Pastori, and J. C. Budich, Chiral topological phases from artificial neural networks, Phys. Rev. B **97**, 195136 (2018).

[64] K. Choo, G. Carleo, N. Regnault, and T. Neupert, Symmetries and Many-Body Excitations with Neural-Network Quantum States, Phys. Rev. Lett. **121**, 167204 (2018).

[65] X. Liang, W.-Y. Liu, P.-Z. Lin, G.-C. Guo, Y.-S. Zhang, and L. He, Solving frustrated quantum many-particle models with convolutional neural networks, Phys. Rev. B **98**, 104426 (2018).

[66] T. Vieijra, C. Casert, J. Nys, W. De Neve, J. Haegeman, J. Ryckebusch, and F. Verstraete, Restricted Boltzmann Machines for Quantum States with Non-Abelian or Anyonic Symmetries, Phys. Rev. Lett. **124**, 097201 (2020).

[67] O. Sharir, Y. Levine, N. Wies, G. Carleo, and A. Shashua, Deep Autoregressive Models for the Efficient Variational Simulation of Many-Body Quantum Systems, Phys. Rev. Lett. **124**, 020503 (2020).

[68] L. Yang, Z. Leng, G. Yu, A. Patel, W.-J. Hu, and H. Pu, Deep learning-enhanced variational Monte Carlo method for quantum many-body physics, Phys. Rev. Res. **2**, 012039(R) (2020).

[69] H. He, Y. Zheng, B. A. Bernevig, and G. Sierra, Multi-layer restricted Boltzmann machine representation of 1D quantum many-body wave functions, arXiv:1910.13454.

[70] H. Saito, Solving the Bose-Hubbard model with machine learning, J. Phys. Soc. Jpn. **86**, 093001 (2017).

[71] K. McBrian, G. Carleo, and E. Khatami, Ground state phase diagram of the one-dimensional Bose-Hubbard model from restricted Boltzmann machines, J. Phys.: Conf. Ser. **1290**, 012005 (2019).

[72] J. Hermann, Z. Schätzle, and F. Noé, Deep-neural-network solution of the electronic schrödinger equation, Nat. Chem. **12**, 891 (2020).

[73] S. Sorella and L. Capriotti, Algorithmic differentiation and the calculation of forces by quantum Monte Carlo, J. Chem. Phys. **133**, 234111 (2010).

[74] See https://github.com/tensorflow/tensorflow.

[75] See https://github.com/pytorch/pytorch.

[76] Code implementation of the applications can be found at https://github.com/refraction-ray/admc.

[77] L. Onsager, Crystal statistics. I. A two-dimensional model with an order-disorder transition, Phys. Rev. **65**, 117 (1944).

[78] G. E. Hinton, A practical guide to training restricted Boltzmann machines, in *Neural Networks: Tricks of the Trade - Second Edition*, edited by G. Montavon, G. B. Orr, and K. Müller, Lecture Notes in Computer Science, Vol. 7700 (Springer, 2012), pp. 599–619.

[79] S. R. White, Density Matrix Formulation for Quantum Renormalization Groups, Phys. Rev. Lett. **69**, 2863 (1992).

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

[81] V. Nair and G. E. Hinton, Rectified linear units improve restricted Boltzmann machines, in *Proceedings of the 27th International Conference on Machine Learning (ICML-10), 21-24 June 2010, Haifa, Israel*, edited by J. Fürnkranz and T. Joachims (Omnipress, 2010), pp. 807–814.

[82] A. W. Sandvik, Finite-size scaling of the ground-state parameters of the two-dimensional Heisenberg model, Phys. Rev. B **56**, 11678 (1997).

[83] F. Mezzacapo, N. Schuch, M. Boninsegni, and J. I. Cirac, Ground-state properties of quantum many-body systems: Entangled-plaquette states and variational Monte Carlo, New J. Phys. **11**, 083026 (2009).

[84] L. Wang, I. Pizorn, and F. Verstraete, Monte Carlo simulation with tensor network states, Phys. Rev. B **83**, 134421 (2011).

[85] M. Lubasch, J. I. Cirac, and M.-C. Bañuls, Algorithms for finite projected entangled pair states, Phys. Rev. B **90**, 064425 (2014).

[86] M. Troyer and U.-J. Wiese, Computational Complexity and Fundamental Limitations to Fermionic Quantum Monte Carlo Simulations, Phys. Rev. Lett. **94**, 170201 (2005).

[87] D. Hangleiter, I. Roth, D. Nagaj, and J. Eisert, Easing the Monte Carlo sign problem, Sci. Adv. **6**, eabb8341 (2020).

[88] R. Levy and B. K. Clark, Mitigating the Sign Problem Through Basis Rotations, Phys. Rev. Lett. **126**, 216401 (2021).

[89] A. J. Kim, P. Werner, and R. Valentí, Alleviating the sign problem in quantum Monte Carlo simulations of spin-orbit-coupled multiorbital Hubbard models, Phys. Rev. B **101**, 045108 (2020).

[90] Z.-Q. Wan, S.-X. Zhang, and H. Yao, Mitigating the fermion sign problem by automatic differentiation, Phys. Rev. B **106**, L241109 (2022).

[91] J. Mao, J. N. Foerster, T. Rocktäschel, M. Al-Shedivat, G. Farquhar, and S. Whiteson, A baseline for any order gradient estimation in stochastic computation graphs, in *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, edited by K. Chaudhuri and R. Salakhutdinov, Proceedings of Machine Learning Research, Vol. 97 (PMLR, 2019), pp. 4343–4351.