

RESEARCH ARTICLE | NOVEMBER 21 2025

Load-balanced diffusion Monte Carlo method with lattice regularization

Special Collection: [2025 JCP Emerging Investigators Special Collection](#)

Kousuke Nakano   ; Sandro Sorella  ; Michele Casula 

 Check for updates

J. Chem. Phys. 163, 194117 (2025)

<https://doi.org/10.1063/5.0296986>



Articles You May Be Interested In

TURBORVB: A many-body toolkit for *ab initio* electronic simulations by quantum Monte Carlo

J. Chem. Phys. (May 2020)

Space-warp coordinate transformation for efficient ionic force calculations in quantum Monte Carlo

J. Chem. Phys. (January 2022)

Size-consistent variational approaches to nonlocal pseudopotentials: Standard and lattice regularized diffusion Monte Carlo methods revisited

J. Chem. Phys. (April 2010)

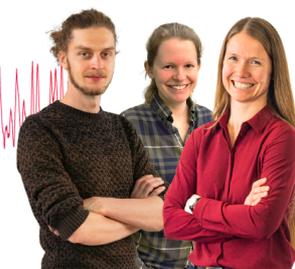
Webinar From Noise to Knowledge

May 13th – Register now



Zurich
Instruments

Universität
Konstanz



Load-balanced diffusion Monte Carlo method with lattice regularization

Cite as: J. Chem. Phys. 163, 194117 (2025); doi: 10.1063/5.0296986

Submitted: 16 August 2025 • Accepted: 23 October 2025 •

Published Online: 21 November 2025



View Online



Export Citation



CrossMark

Kousuke Nakano,^{1,a)} Sandro Sorella,² and Michele Casula³

AFFILIATIONS

¹Center for Basic Research on Materials (CBRM), National Institute for Materials Science (NIMS), 1-2-1 Sengen, Tsukuba, Ibaraki 305-0047, Japan

²International School for Advanced Studies (SISSA), Via Bonomea 265, 34136 Trieste, Italy

³Institut de Minéralogie, de Physique des Matériaux et de Cosmochimie (IMPMC), Sorbonne Université, CNRS UMR 7590, IRD UMR 206, MNHN, 4 Place Jussieu, 75252 Paris, France

Note: This paper is a part of the 2025 JCP Emerging Investigators Special Collection.

a) Author to whom correspondence should be addressed: kousuke_1123@icloud.com

ABSTRACT

Ab initio quantum Monte Carlo (QMC) is a stochastic approach for solving the many-body Schrödinger equation without resorting to one-body approximations. QMC algorithms are readily parallelizable via ensembles of N_w walkers, making them well suited to large-scale high-performance computing. Among the QMC techniques, diffusion Monte Carlo (DMC) is widely regarded as the most reliable since it provides the projection onto the ground state of a given Hamiltonian under the fixed-node approximation. One practical realization of DMC is the lattice regularized diffusion Monte Carlo (LRDMC) method, which discretizes the Hamiltonian within the Green's function Monte Carlo framework. DMC methods—including LRDMC—employ the so-called branching technique to stabilize walker weights and populations. At the branching step, walkers must be synchronized globally; any imbalance in per-walker workload can leave central processing unit (CPU) or graphics processing unit (GPU) cores idle, thereby degrading overall hardware utilization. The conventional LRDMC algorithm intrinsically suffers from such load imbalance, which grows as $\log(N_w)$, rendering it less efficient on modern parallel architectures. In this work, we present an LRDMC algorithm that inherently addresses the load imbalance issue and achieves significantly improved weak-scaling parallel efficiency. Using the binding energy calculation of a water–methane complex as a test case, we demonstrated that the conventional and load-balanced LRDMC algorithms yield consistent results. Furthermore, by utilizing the Leonardo supercomputer equipped with NVIDIA A100 GPUs, we demonstrated that the load-balanced LRDMC algorithm can maintain extremely high parallel efficiency ($\sim 98\%$) up to 512 GPUs (corresponding to $N_w = 51\,200$), together with a speedup of $\times 1.24$ if directly compared with the conventional LRDMC algorithm with the same number of walkers. The speedup stays sizable, i.e., $\times 1.18$, even if the number of walkers is reduced to $N_w = 400$.

© 2025 Author(s). All article content, except where otherwise noted, is licensed under a Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>). <https://doi.org/10.1063/5.0296986>

I. INTRODUCTION

Understanding and predicting the quantum behavior of electrons and nuclei remains a fundamental challenge in physics and chemistry. Despite decades of theoretical and computational developments, solving the many-body Schrödinger equation exactly is still intractable for systems of practical interest due to its exponential complexity. As a result, approximate methods have become indispensable. Among them, density functional theory (DFT)¹ has emerged as a standard approach by mapping the complex many-electron problem to a system of non-interacting

electrons subject to an effective potential, derived from the so-called eXchange–Correlation functional (XC).² However, the accuracy of DFT hinges on the choice of XC functionals, for which no exact form is known. Thus, DFT often struggles in systems with strong electron correlation or non-local interactions, and systematic improvements remain elusive.^{3,4}

Ab initio quantum Monte Carlo (QMC) methods aim at solving the many-body Schrödinger equation using stochastic techniques, without relying on one-body approximations. Their intrinsic scalability and accuracy have always been appealing since their first appearance. Among various QMC implementations, diffusion

Monte Carlo (DMC) is particularly notable for its ability to compute the exact ground state energy for a Hamiltonian with boundaries usually defined by the nodal surface of a trial wavefunction, via imaginary-time projection.⁵ This constraint is known as “fixed-node approximation” (FNA), a remedy for the Fermionic sign problem appearing in the Hamiltonian quantum evolution.⁶ These boundaries are in principle improvable, according to the quality of the trial wavefunction nodes, thanks to the FNA variational property. Owing to its quantitative reliability, DMC has been widely applied to challenging systems where DFT often fails, such as solid and liquid hydrogen under high pressure,^{7–10} molecular crystals,^{11,12} and two-dimensional materials.^{13,14} The most commonly used DMC implementation employs the Suzuki–Trotter decomposition to iteratively perform projection steps.⁵ However, it suffers from numerical instability in the extrapolation to the zero time step limit, where the total energy does not converge monotonically.¹⁵ To address this issue, an alternative approach based on a lattice regularized Hamiltonian, and thus dubbed lattice regularized diffusion Monte Carlo (LRDMC), was proposed.¹⁶ LRDMC exploits the Green’s Function Monte Carlo (GFMC) technique,^{17–19} which was traditionally used in lattice models, to continuous-space systems by discretizing the Laplacian on a real-space mesh. In LRDMC, the projection operator is implemented as powers of a Green’s function, and unlike standard DMC, it does not require Suzuki–Trotter factorization. This eliminates the time step bias inherent in the conventional DMC implementation. Instead, a bias arises due to the lattice discretization of the Hamiltonian, but this is of the order of $\mathcal{O}(a^2)$ and can be systematically removed by extrapolation to the continuous space limit ($a \rightarrow 0$),²⁰ which turns out to be much smoother than the time step extrapolation of standard DMC.¹⁵

In both DMC and LRDMC, the so-called branching step plays a central role in enhancing computational efficiency. Each walker carries out *independent* projection steps and accumulates a statistical weight, which is used to perform weight and population control through branching: walkers with large weights are spawned, while those with small weights are removed at periodic intervals. Except for the branching steps, all walkers evolve independently, making these methods highly parallelizable. In practice, large-scale DMC simulations typically involve a large number of walkers distributed across many central processing units (CPUs) or graphical processing units (GPUs). However, branching steps require synchronization: all walkers must complete their projection steps before branching can proceed. If computational load becomes imbalanced

across walkers—for example, due to the stochastic nature of the projection—the faster walkers must wait for the slower ones, causing some working units (on CPUs or GPUs) to remain idle. As detailed in Sec. II, the conventional LRDMC algorithm inherently leads to this kind of load imbalance, especially as the number of walkers increases. Consequently, its parallel efficiency degrades on modern hardware platforms that rely on a large walker population. In this work, we address this scalability issue by developing and presenting a new LRDMC algorithm that eliminates load imbalance, thereby enabling efficient use of large number of walkers on modern parallel architectures.

This paper is organized as follows: in Sec. II, we review the conventional LRDMC algorithm and we introduce the new one satisfying load-balance among walkers; in Sec. III, we verify that the proposed load-balanced LRDMC algorithm gives consistent DMC energies if compared with the conventional LRDMC algorithm for the methane-water dimer; in Sec. IV, we show how the load balance fulfilled by the new algorithm improves the weak-scaling properties of the LRDMC algorithm; and in Sec. V, we summarize the main outcome of this work.

II. METHODS

In this section, we outline the conventional LRDMC algorithm that has been used in previous studies,^{21–25} as well as the load-balanced LRDMC algorithm proposed in this work. The essential difference between the two algorithms lies in whether load balance among walkers is inherently guaranteed when processed in parallel. A schematic comparison of the two algorithms is provided in Fig. 1. The mathematical background of both algorithms and the details of their convergence properties are presented in Appendixes A and B.

A. Overview of the lattice regularized diffusion Monte Carlo method

Lattice regularized diffusion Monte Carlo (LRDMC)¹⁶ is a projection technique that allows one to improve a variational ansatz systematically in the continuous space. This method is based on Green’s function Monte Carlo (GFMC),^{17–19} filtering out the ground state wavefunction (WF) $|\Psi_0\rangle$ from a given trial (or guiding) WF, $|\Psi_G\rangle$. Since the eigenstates of the Hamiltonian \hat{H} are a complete basis set, the trial WF can be expanded as

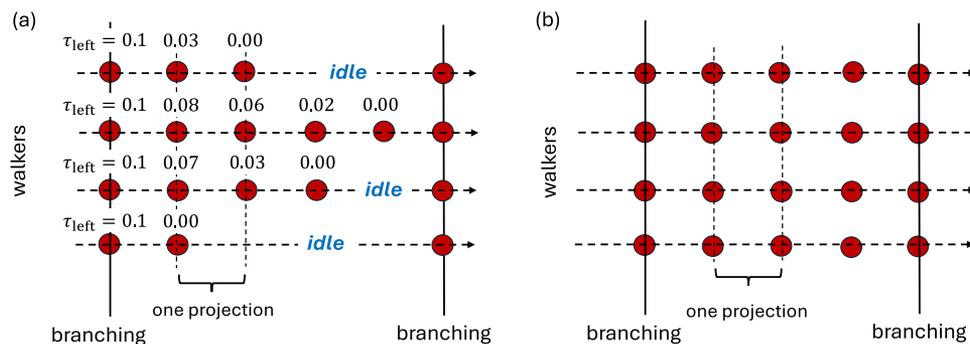


FIG. 1. Schematic figure of the difference between two LRDMC algorithms: (a) Conventional and (b) load-balanced LRDMC algorithms. In algorithm (a), each walker maintains its own remaining time τ_{left} and undergoes projection steps drawn from independent random numbers until $\tau_{\text{left}} = 0.0$, which can leave some walkers idle. In contrast, in algorithm (b), every walker performs the same number of projection steps, thereby preserving load balancing.

$$|\Psi_G\rangle = \sum_{i \geq 0} a_i |Y_i\rangle, \quad (1)$$

where a_i is the coefficient for the i th eigenvector $|Y_i\rangle$. Therefore, by applying $(\Lambda - \hat{H})^M$, one can obtain

$$\begin{aligned} |Y_0\rangle &\propto \lim_{M \rightarrow \infty} (\Lambda - \hat{H})^M |\Psi_G\rangle \\ &= \lim_{M \rightarrow \infty} (\lambda - E_0)^M \left[a_0 |Y_0\rangle + \sum_{i \neq 0} \left(\frac{\lambda - E_i}{\lambda - E_0} \right)^M a_i |Y_i\rangle \right], \end{aligned} \quad (2)$$

where Λ is a diagonal matrix with $\Lambda_{x,x'} = \lambda \delta_{x,x'}$ and E_i is i th eigenvalue of \hat{H} . Assuming a non-degenerate ground state and $\left| \frac{\lambda - E_i}{\lambda - E_0} \right| < 1$ for $i \geq 1$, the projection filters out the ground state WF $|Y_0\rangle$ from a given trial WF $|\Psi_G\rangle$, as long as the trial WF is not orthogonal to the true ground state (i.e., $a_0 \equiv \langle \Psi_G | Y_0 \rangle \neq 0$). The condition $\left| \frac{\lambda - E_i}{\lambda - E_0} \right| < 1$ can be verified on a lattice for a large enough λ . On the continuum, the spectrum of the *ab initio* \hat{H} is not bounded from above. Thus, the above-mentioned condition is verified only in the $\lambda \rightarrow \infty$ limit, which can be easily taken, as shown later on. This projection scheme is called the power method. In practice, Eq. (2) can be implemented iteratively,

$$|\Psi_{n+1}\rangle = (\Lambda - \hat{H}) |\Psi_n\rangle. \quad (3)$$

By expanding over the given basis set with the guiding function $\Psi_G(x)$, we have

$$\Psi_G(x') \Psi_{n+1}(x') = \sum_x \mathcal{G}_{x',x} \Psi_G(x) \Psi_n(x), \quad (4)$$

where $\Psi_n(x) \equiv \langle x | \Psi_n \rangle$ and

$$\mathcal{G}_{x',x} \equiv \langle x' | \Lambda - \hat{H} | x \rangle \frac{\langle x' | \Psi_G \rangle}{\langle x | \Psi_G \rangle} \equiv \lambda \delta_{x',x} - \mathcal{H}_{x',x} \frac{\Psi_G(x')}{\Psi_G(x)} \quad (5)$$

is the so-called importance sampling Green's function. For sake of simplicity, here and henceforth, we assume $\mathcal{H}_{x',x}$ to be a real matrix. Within a statistical implementation of the power method, we would be tempted to interpret Eq. (4) as a master equation for a stochastic variable, where $\Pi_n(x) \equiv \Psi_G(x) \Psi_n(x)$ represents the probability distribution at the iteration n and \mathcal{G} is the transition probability. However, some important points must be elucidated in order to reach a final statistical interpretation of this quantum evolution.

To interpret Green's function as a transition probability, all matrix elements of the Green's function must be non-negative. However, it is well known that for Fermions, $\mathcal{G}_{x',x}$ is not always positive. This issue is known as the *sign problem* and is circumvented via the FNA by defining an effective FN Hamiltonian, as detailed in [Appendix D](#). In the main text, we assume that all matrix elements are non-negative, but the discussion can be easily generalized to FN cases.

Even when Green's function is non-negative, $\mathcal{G}_{x',x}$ is generally not normalized. Therefore, we need to introduce a normalization factor $b_x = \sum_{x'} \mathcal{G}_{x',x}$ so that we can define the p matrix as

$$p_{x',x} = \frac{\mathcal{G}_{x',x}}{b_x}. \quad (6)$$

$p_{x',x}$ can now be interpreted as transition probability since it is non-negative and row-wise normalized ($\sum_{x'} p_{x',x} = 1$). Then, the corresponding Markov process considers the presence of the normalization factor b_x by adding a weight w in the statistical implementation of the projection, which is thus defined by a dyad (x_n, w_n) . Its evolution from (x_n, w_n) is described by

$$x_{n+1} = x' \quad \text{with probability } p_{x',x}, \quad (7)$$

$$w_{n+1} = w_n \cdot b_x. \quad (8)$$

The Markov process, comprising the diffusion of the configuration as well as its weight, can be cast in the following master equation:

$$\mathcal{P}_{n+1}(x', w') = \sum_x \int dw K(x', w' | x, w) \mathcal{P}_n(x, w), \quad (9)$$

where the transition kernel K is defined as $K(x', w' | x, w) = p_{x',x} \delta(w' - w b_x)$. Multiplying both sides of Eq. (9) by w' and integrating over it yields

$$\begin{aligned} \int dw' w' \mathcal{P}_{n+1}(x', w') &= \sum_x p_{x',x} b_x \int dw w \mathcal{P}_n(x, w) \\ &= \sum_x \mathcal{G}_{x',x} \int dw w \mathcal{P}_n(x, w). \end{aligned} \quad (10)$$

The marginal weighted probability is nothing but the mixed many-body distribution, i.e., $\int dw w \mathcal{P}_n(x, w) = \Pi_n(x)$ at step n . Thus, the desired ground-state energy E_0 is computed at the projection step n as

$$E_0 = \frac{\langle \Psi_G | \hat{H} | \Psi_0 \rangle}{\langle \Psi_G | \Psi_0 \rangle} = \frac{\sum_x e_L(x) \Psi_G(x) \Psi_n(x)}{\sum_{x'} \Psi_G(x') \Psi_n(x')}, \quad (11)$$

where the local energy, $e_L(x)$, is defined as

$$e_L(x) = \frac{\langle \Psi_G | \hat{H} | x \rangle}{\langle \Psi_G | x \rangle} = \sum_{x'} \mathcal{H}_{x,x'} \cdot \frac{\Psi_G(x')}{\Psi_G(x)}. \quad (12)$$

To implement the Markov process in a computable scheme, we define its basic element, the so-called *walker*, determined by the dyad (x, w) . Note how the weight w is associated to the amplitude of the mixed distribution at x . The walker changes its configuration and weight by performing a Markovian process with a discrete iteration time n : the dyad (x_n, w_n) is distributed according to $\mathcal{P}_n(x, w)$. Most importantly, the walker determines, in a statistical sense, the mixed quantum state $\Psi_G(x) \Psi_n(x)$ as

$$\Psi_G(x) \Psi_n(x) = \int dw w \mathcal{P}_n(x, w) \sim \langle \delta_{x,x_n} w_n \rangle, \quad (13)$$

where $\langle \dots \rangle$ denotes the statistical average over *independent* Markov chains at the n th step. Thus, the desired ground-state energy E_0 is computed as

$$\begin{aligned} E_0 &= \frac{\sum_x e_L(x) \Psi_G(x) \Psi_n(x)}{\sum_{x'} \Psi_G(x') \Psi_n(x')} = \frac{\sum_x e_L(x) \int dw w \mathcal{P}_n(x, w)}{\sum_{x'} \int dw' w' \mathcal{P}_n(x', w')} \\ &\sim \frac{\langle e_L(x_n) w_n \rangle}{\langle w_n \rangle}. \end{aligned} \quad (14)$$

However, in practice, the strategy of simply computing independent Markov chains and averaging their results is not adopted. This is because walkers with extremely large or small weights may appear, leading to numerical instability. While keeping a multi-walker strategy with N_w walkers, we instead reconfigure their weights via the so-called branching technique for a more efficient computation.²⁰ Every N_{proj} projections, the code performs a branching step as follows.

- (1) Select the new walkers from the previous ones with a probability that is proportional to the former walkers' weights before the branching step,

$$p_{\alpha,n} = \frac{w_{\alpha,n}}{\sum_{\beta} w_{\beta,n}}. \quad (15)$$

- (2) Set the new weights equal to the weights average of the former walkers,

$$w_{\alpha,n+1} = \bar{w}_n \equiv \frac{1}{N_w} \sum_{\beta} w_{\beta,n} \quad \forall \alpha. \quad (16)$$

Note that, in the multi-walker formalism, the dyad acquires the additional walker index α : $(x_{\alpha,n}, w_{\alpha,n})$. As rigorously proved in Refs. 18 and 20, this walkers' reconfiguration does not change their statistical average and suppresses their fluctuations by dropping (spawning) walkers having small (large) weights. The selection of new configurations in a branching step is implemented by an efficient reconfiguration scheme deciding which walker survives or dies by extracting N_w correlated random numbers,²⁰

$$Z_{\alpha} = \frac{\xi + \alpha - 1}{N_w}, \quad \alpha = 1, \dots, N_w, \quad (17)$$

where ξ is a uniform variate in $[0,1)$. This set of numbers is then used to select the new configuration by comparing it with the normalized weights $w_{\alpha,n}/\bar{w}_n$. This reconfiguration provides the same stabilization effect as the conventional branching scheme,⁵ but with the advantage that the total number of walkers remains fixed throughout the simulation.^{18,20} Since the total number of walkers is fixed, the reconfiguration method is known to alleviate the population-control bias, which is problematic in the standard DMC branching scheme.¹⁸ However, the intrinsic exponential scaling of computational cost in projection methods for large systems,²⁶ originating from weight fluctuations of walkers, should remain, although it is not problematic in practical applications.²⁶

To make full use of the available samples, observables are evaluated using all samples obtained after the Markov chain has reached equilibrium. After thermalization time p expressed in branching step units, the configurations x along the Markov process will be equilibrated according to $\pi_{\text{eq}}(x)$; then, at each branching step n , we can imagine to start the projection $n-p$ steps backward with initial weights $w_{\alpha,n-p} = 1 \forall \alpha$. In this case, the initial probability is given by $\mathcal{P}_{n-p}(x, w) = \delta(w-1)\pi_{\text{eq}}(x)$. To perform the Monte Carlo averages with the branching technique, we can accumulate the average weights in the global population factor \tilde{G}_n^p developed after $n-p$ branching steps,

$$\tilde{G}_n^p = \prod_{j=1}^p \bar{w}_{n-j}, \quad (18)$$

while setting the weights to 1 after each walkers reconfiguration. In this way, the ground state energy is computed as

$$E_0 \approx \frac{\sum_n \tilde{G}_n^p \bar{e}_L(x_n)}{\sum_n \tilde{G}_n^p}, \quad (19)$$

where $\bar{e}_L(x_n)$ is the mean local energy averaged over the walkers, which reads

$$\bar{e}_L(x_n) = \frac{\sum_{\alpha} w_{\alpha,n} e_L(x_{\alpha,n})}{\sum_{\alpha} w_{\alpha,n}} \quad (20)$$

and is evaluated just before each reconfiguration, with weights $w_{\alpha,n}$ acquired only during the evolution after the $n-p$ branching step.

We notice that the ground-state energy has the zero-variance-zero-bias property. In other words, the fluctuation of the local energy becomes zero for $|\Psi_G\rangle = |\Psi_0\rangle$ since $|\Psi_0\rangle$ is an eigenstate of the (fixed-node) Hamiltonian \hat{H} . Even with a variational trial wavefunction $|\Psi_G\rangle \neq |\Psi_0\rangle$, the energy variance is always finite because the divergence of the local energy in the vicinity of the nodal surface is always canceled out by the zeros of the trial and fixed-node WFs.²⁷ Let us denote the distance from the nodal surface as δ . The trial wave function Ψ_G and the fixed-node solution Ψ_0 vanish linearly with this distance, i.e., $\Psi_G \propto \delta$ and $\Psi_0 \propto \delta$, and the local energy e_L diverges as $e_L \propto \delta^{-1}$ with a trial wavefunction Ψ_G that is not an exact eigenfunction of the fixed-node Hamiltonian.²⁷ Therefore, $\Pi(x) \equiv \Psi_G(x)\Psi_0(x)$ and $e_L(x)$ behave as $\propto \delta^2$ and δ^{-1} , respectively. The calculation of the energy variance requires the integral $\int_{\epsilon}^1 d\delta \Pi(\delta)(e_L(\delta))^2 \sim \int_{\epsilon}^1 d\delta (\delta^2)(\delta^{-2}) = \int_{\epsilon}^1 d\delta$, where ϵ is an infinitesimal positive number. This is always finite. However, for derivatives, such as $\partial e_L(x) \propto \delta^{-2}$, we have $\int_{\epsilon}^1 d\delta \Pi(\delta)(\partial e_L(\delta))^2 \sim \int_{\epsilon}^1 d\delta (\delta^2)(\delta^{-2})^2 = \int_{\epsilon}^1 d\delta \delta^{-2}$. Thus, the variance of derivatives intrinsically diverges in the vicinity of nodal surface. A reweighting^{28,29} or regularization³⁰ technique is needed to cure the divergence.

When applying the GFMC in the context of *ab initio* calculation, the original continuous Hamiltonian is regularized by allowing electron hopping with step size a . The corresponding Hamiltonian $\hat{\mathcal{H}}^a$ is then defined such that $\hat{\mathcal{H}}^a \rightarrow \hat{\mathcal{H}}$ for $a \rightarrow 0$. After discretizing the space through a lattice spacing a , the hopping elements $H'_{x',x}$ of the Hamiltonian can be computed by considering only the hoppings of each electron to its neighboring grid points in the current electronic configuration. As a result, $H'_{x',x}$ becomes sparse, allowing the GFMC method to be applied to a continuous Hamiltonian. This approach is referred to as LRDMC.¹⁶ Then, the energy obtained by LRDMC exhibits a bias of the order of $\mathcal{O}(a^2)$ with respect to a , and therefore, to access unbiased total energies (or other observables), LRDMC calculations are performed for several values of a , and an extrapolation to $a \rightarrow 0$ is carried out. The technical details are explained in Appendix D.

B. The conventional LRDMC algorithm and its intrinsic load-imbalance

As we have seen above, in the LRDMC framework, the projection of the wavefunction is governed by the repeated application of a Green's function, interpreted as a stochastic transition matrix

Eqs. (6)–(8). To ensure a probabilistic interpretation, Green's function must be non-negative, which often requires shifting the Hamiltonian by a constant Λ such that all diagonal elements become positive [Eq. (5)]. While on a lattice this shift can always be applied, on the continuum this is prevented from the fact that the local Hamiltonian terms can become arbitrarily large due to the divergence of the Coulomb potential. Moreover, choosing an excessively large shift degrades the efficiency, as it increases the probability for a walker to remain in the same configuration. This follows from $\mathcal{G}_{x,x} \gg \mathcal{G}_{x' \neq x,x}$, thus $p_{x,x} \gg p_{x' \neq x,x}$, leading to long autocorrelation times and inefficient sampling.

To circumvent this issue, the projection steps Eqs. (7) and (8) are reformulated in the $\lambda \rightarrow \infty$ limit, so defining a *continuous-time* stochastic process. Rather than evolving the walker with discrete steps governed by the bare $p_{x',x}$ Eq. (6), the algorithm considers the number of successive diagonal moves *stochastically*, followed by a single off-diagonal move driven by $p_{x' \neq x,x}$. Within this formalism, the number k of successive diagonal moves is drawn from a uniformly distributed random number $\xi \in [0, 1)$, resulting in^{19,31}

$$k = \left\lfloor -\frac{\ln(1-\xi)}{1-p_{x_n,x_n}} \right\rfloor. \quad (21)$$

The corresponding walker weight is updated as

$$w_{n+1} = w_n \cdot b^k. \quad (22)$$

This approach is generalized to the continuous-time limit, namely, $\lambda \rightarrow \infty$, i.e., with an infinitesimal time step defined as $\delta\tau \equiv 1/\lambda \rightarrow 0$.^{20,32} In this limit, the projection in Eq. (2), $(\Lambda - \hat{H})^M$, is evaluated by keeping $M/\lambda = \tau$ finite, such that the imaginary time evolution is cast in $\exp(-\tau\hat{H})$, apart from an irrelevant constant λ^M . At the same time, the denominator of Eq. (21) can be written as

$$1 - p_{x,x} = \sum_{x'(\neq x)} p_{x',x} = \frac{\sum_{x'(\neq x)} \mathcal{G}_{x',x}}{\lambda - e_L(x)} \rightarrow \delta\tau \sum_{x'(\neq x)} \mathcal{G}_{x',x} + O((\delta\tau)^2). \quad (23)$$

By plugging Eq. (23) into Eq. (21), one obtains at the leading order in $\delta\tau$,

$$\tau\xi = -\frac{\ln(1-\xi)}{\sum_{x'(\neq x_n)} \mathcal{G}_{x',x_n}}, \quad (24)$$

which corresponds to $\tau\xi = k\delta\tau$, the persistence time at configuration x in the continuous-time limit. Correspondingly, Eq. (22) becomes

$$w_{n+1} = w_n \cdot \exp(-\tau\xi e_L(x_n)). \quad (25)$$

Therefore, in the continuous-time limit formulation, one estimates the persistence time at configuration x_n via Eq. (24) and updates the walkers weights according to Eq. (25), before performing an off-diagonal move *exclusively* driven by the off-diagonal elements \mathcal{G}_{x',x_n} appropriately normalized with $\sum_{x'(\neq x_n)} \mathcal{G}_{x',x_n}$. The imaginary time is then updated according to $\tau_{\text{left}} \leftarrow \tau_{\text{left}} - \tau\xi$ at each off-diagonal update until τ_{left} becomes 0. Each branching (reconfiguration) step is performed after an integrated imaginary time evolution τ that is an input parameter defined *a priori*. Importantly, this continuous-time formalism allows simulations of the imaginary-time propagator

$\exp(-\tau H)$ without requiring the Suzuki–Trotter decomposition, thereby eliminating the systematic Suzuki–Trotter error inherent in traditional DMC implementations. The price to pay is the lattice discretization bias that has to be extrapolated, as we have seen in Sec. II A.^{16,20} The pseudocode of the conventional LRDMC algorithm is shown in Algorithm 1.

The conventional LRDMC algorithm faces a specific challenge in massively parallel environments, particularly with many walkers. In the conventional algorithm, the projection time evolution is governed by stochastic sampling the propagation intervals, or persistence times τ_ξ , using uniform variates Eq. (24). As a result, the computational effort required per walker to complete the total time evolution τ between two branching steps becomes unbalanced: the number of steps necessary for each walker to reach $\tau_{\text{left}} = 0$ varies randomly. This imbalance poses a problem because the branching process must wait until *all* walkers have completed their projection steps. Consequently, walkers that finish earlier have to remain idle until the slowest walker completes its projection. Consequently, the parallel efficiency of the conventional LRDMC algorithm deteriorates as the number of walkers increases. This degradation can be shown to grow logarithmically with the number of walkers N_w , as detailed in Appendix C. Indeed, the maximum number of moves during a fixed time interval between two branching steps increases as $\log(N_w)$, indicating that the actual computation time also increases as $\log(N_w)$, even though the per-walker workload remains unchanged on average.

One might consider alleviating the load-imbalance problem within the framework of the conventional LRDMC algorithm by choosing a sufficiently large projection time τ between two consecutive branching steps. However, employing a large value of τ has two main limitations: a degradation of computational efficiency and a possible numerical overflow arising from the accumulated walker weights. Regarding the first issue, if τ is taken too large, the weight distribution among walkers becomes highly imbalanced. As a result, at the branching step, a substantial number of walkers are eliminated and the walkers' survival rate diminishes. This, in turn, decreases the effective sample size, thereby increasing the statistical error. Consequently, a too large τ value severely undermines the efficiency of the LRDMC calculation, thus a trade-off must be found between the load-imbalance reduction and the increase in the statistical error bar. As for the second issue, in the conventional LRDMC algorithm, the weights are updated according to Eq. (25), and as evident from this expression, they grow exponentially at each step. Therefore, when a large τ value is used, some weights could lead to numerical overflow. Furthermore, the accumulation of weights using Eq. (18) can become infeasible for the same reason.

To overcome the load-imbalance of the conventional LRDMC algorithm, we propose a new LRDMC algorithm that ensures a perfectly load-balanced evolution of walkers between two branching steps, independently of their number. The details of the new algorithm are described in Sec. II C.

C. The load-balanced LRDMC algorithm

The idea behind a load-balanced LRDMC algorithm relies on the possibility to define a branching rate determined by a fixed number of off-diagonal moves, set equal for every walker, rather than by the persistence time, which depends instead on each walker path,

ALGORITHM 1. Conventional LRDMC algorithm.

Input: Given $N_{\text{branching}}$, N_{walker} , and τ

Output: Computed averages and variances

- 1 Generate initial configurations \vec{x} for all walkers;
- 2 **for** $j \leftarrow 1$ to $N_{\text{branching}}$ **do**
- 3 **for** $i \leftarrow 1$ to N_{walker} **do**
- 4 $w = 1.0$; $\tau_{\text{left}} = \tau$;
- 5 **while** $\tau_{\text{left}} > 0$ **do**
- 6 Compute the diagonal Green function matrix element $\mathcal{G}_{x',x}$ and $e_L(x)$;
- 7 Draw a random number, $\xi = [0, 1)$;
- 8 Compute the persistence time τ_ξ , during which the walker stays in configuration x :

$$\tau_\xi = \min \left[\frac{-\log(1 - \xi)}{\sum_{x' \neq x} \mathcal{G}_{x',x}}, \tau_{\text{left}} \right];$$
- 10 Update the weight: $w \leftarrow w \exp[-\tau_\xi e_L(x)]$;
- 11 Update the remaining time: $\tau_{\text{left}} \leftarrow \tau_{\text{left}} - \tau_\xi$;
- 12 **if** $\tau_{\text{left}} = 0$ **then**
- 13 **break**;
- 14 Move the walker i to a new configuration ($x' \neq x$) with probability:

$$p_{x',x} = \frac{\mathcal{G}_{x',x}}{\sum_{x' \neq x} \mathcal{G}_{x',x}};$$
- 16 Store the accumulated weight w and local energy e_L after the projection;
- 17 $i \leftarrow i + 1$;
- 18 Compute the average weight \bar{w} and the average local energy \bar{e}_L before branching;
- 19 Use branching scheme to control walker weight fluctuations;
- 20 $j \leftarrow j + 1$;
- 21 Compute averages and variances using appropriate estimators;

finally responsible of the load imbalance issue. In Sec. II B, we have seen that the persistence time is proportional to the magnitude of the diagonal Green's function elements of the LRDMC Hamiltonian. Thus, we need to define a projection process that is driven by an auxiliary Hamiltonian with zero diagonal matrix elements, in such a way that the corresponding walkers will never stop; i.e., they will constantly hop from site to site. This can be realized with the strategy described as follows. Given the FN LRDMC Hamiltonian H on a lattice, we can formally divide its matrix elements in off-diagonal $\tilde{\mathcal{H}}_{x',x}$ and diagonal $\mathcal{W}(x) \equiv \mathcal{H}_{x,x}$ ones, such that

$$\mathcal{H}_{x',x} = \tilde{\mathcal{H}}_{x',x} + \delta_{x',x} \mathcal{W}(x), \quad (26)$$

where $\tilde{\mathcal{H}}_{x,x} = 0$ by construction. Notice that the above-mentioned partition can be realized $\forall x$ thanks to the potential regularization

provided by the definition of the LRDMC Hamiltonian, which cuts off any divergence of the bare *ab initio* potential by means of the length scale a (see Appendix D for a detailed description of the regularized potential). From Eq. (26), it follows that the wavefunction Ψ_0 , ground state of the FN LRDMC Hamiltonian with energy E_0 , will fulfill the following Schrödinger equation:

$$\sum_x \tilde{\mathcal{H}}_{x',x} \Psi_0(x) = (E_0 - \mathcal{W}(x')) \Psi_0(x'). \quad (27)$$

In analogy with the original Schrödinger equation, one can define a modified Green function (with importance sampling) based on $\tilde{\mathcal{H}}_{x',x}$

$$\mathcal{G}_{x',x} = -\frac{1}{\mathcal{W}(x) - E_0} \tilde{\mathcal{H}}_{x',x} \cdot \frac{\Psi_G(x')}{\Psi_G(x)}. \quad (28)$$

According to Eq. (28), one can implement a stochastic process, in close analogy with what has been derived for conventional LRDMC algorithm based on Eq. (5) by using the decomposition,

$$\begin{aligned} \mathcal{G}_{x',x} &= b_x p_{x',x}, \\ b_x &= \frac{1}{\mathcal{W}(x) - E_0} \bar{b}_x, \\ \bar{b}_x &= -\sum_{x'} \Psi_G(x') \bar{\mathcal{H}}_{x',x} / \Psi_G(x), \\ p_{x',x} &= -\frac{\Psi_G(x') \bar{\mathcal{H}}_{x',x} / \Psi_G(x)}{\bar{b}_x}. \end{aligned} \quad (29)$$

In order to interpret b_x as a legitimate weight while $p_{x',x}$ is the normalized transition probability, b_x must be non-negative in the FNA. This follows from the positiveness of $\mathcal{W}(x) - E_0 \quad \forall x$. To show this, take the minimum of the potential $\mathcal{W}(x_0)$ occurring for some configuration x_0 . Since the trial state $\Psi_G(x) = \delta_{x,x_0}$ has variational energy $\mathcal{W}(x_0)$, this implies immediately that $\mathcal{W}(x) - E_0 > 0 \quad \forall x$. Then, as we have already seen in Sec. II A, the algorithm built on the set of Eq. (29), performs the conventional lattice update,

$$\begin{aligned} x_{n+1} &= x' \quad \text{with } p_{x',x}, \\ w_{n+1} &= w_n b_x. \end{aligned} \quad (30)$$

A master equation formally equivalent to Eq. (4) can then be drawn for the above-mentioned stochastic process. One can show (see Appendix B) that a *stationary* distribution of this process is the marginal weighted probability given by

$$\int dw w \mathcal{P}(x, w) = (\mathcal{W}(x) - E_0) \Psi_G(x) \Psi_0(x), \quad (31)$$

which is the conventional LRDMC mixed distribution Eq. (13) modulated by the factor $(\mathcal{W}(x) - E_0)$. As derived in Appendix B, we can prove that for a good enough mixed distribution $(\mathcal{W}(x) - E_0) \Psi_G(x) \Psi(x)$; namely, with $\Psi(x)$ close enough to the true FN ground-state wavefunction $\Psi_0(x)$, the initial state converges to the final distribution in Eq. (31) for $M \rightarrow \infty$ with the modified Green function in Eq. (28). Owing to these convergence properties, for averaging an observable, we need to consider the additional factor $(\mathcal{W}(x) - E_0)$ in the distribution, which must be compensated by an extra inverse factor in the weights. For instance, the total energy is evaluated as

$$E_0 = \frac{\sum_x e_L(x) \Psi_G(x) \Psi_n(x)}{\sum_x \Psi_G(x') \Psi_n(x')}, \quad (32)$$

ALGORITHM 2. Load-balanced LRDMC algorithm.

Input: Given $N_{\text{branching}}$, N_{walker} , and N_{proj}

Output: Computed averages and variances

- 1 Generate initial configurations \vec{x} for all walkers;
- 2 **for** $j \leftarrow 1$ **to** $N_{\text{branching}}$ **do**
- 3 **for** $i \leftarrow 1$ **to** N_{walker} **do**
- 4 $w = 1.0$;
- 5 **for** $k \leftarrow 1$ **to** N_{proj} **do**
- 6 Compute b_x and \bar{b}_x (diagonal Green function matrix elements);
- 7 Update weight: $w \leftarrow w \cdot b_x$, where $b_x = \frac{1}{(\mathcal{W}(x) - E_0)} \cdot \bar{b}_x$;
- 8 Move walker $x \rightarrow x'$ according to probability $p_{x',x}$;
- 9 Store the accumulated weight w and local energy e_L after the projection;
- 10 $i \leftarrow i + 1$;
- 11 Compute the average weight \bar{w} and the average local energy \bar{e}_L before branching;
- 12 Use branching scheme to control walker weight fluctuations;
- 13 Update E_0 ;
- 14 $j \leftarrow j + 1$;
- 15 Compute averages and variances using appropriate estimators;

which can be computed by

$$E_0 \sim \frac{\left\langle e_L(x_n) \frac{w_n}{\mathcal{W}(x_n) - E_0} \right\rangle}{\left\langle \frac{w_n}{\mathcal{W}(x_n) - E_0} \right\rangle}, \quad (33)$$

where $e_L(x) = -\bar{b}_x + \mathcal{W}(x)$ is the local energy of the FN LRDMC Hamiltonian $\mathcal{H}_{x',x}$ in Eq. (26) and $\langle \dots \rangle$ is the statistical average over independent Markov chains at the n th step, denoted as (x_n, w_n) , and distributed according to $\mathcal{P}(x, w)$ Eq. (31). From Eq. (33), it is clear that the ground state estimate E_0 , which is the value appearing in the modified Green's function of Eq. (28), can be self-consistently determined by averaging the local energy.

The sampling strategy implied by $\langle \dots \rangle$ in Eq. (33) and adopted by the load-balanced LRDMC algorithm follows exactly the same multi-walkers framework described in Subsection II A, together with the same walkers reconfiguration scheme [see Eqs. (15)–(19), which perfectly hold also in this case]. A difference between the conventional and the load-balanced LRDMC algorithms shows up in the weighted averages of the multi-walker framework. Indeed, the mean local energy averaged over the walkers, corresponding to Eq. (20) in the conventional algorithm, now becomes

$$\tilde{e}_L(x_n) = \frac{\sum_{\alpha} \frac{w_{\alpha,n}}{\mathcal{W}(x_{\alpha,n}) - E_0} \cdot e_L(x_{\alpha,n})}{\sum_{\alpha} \frac{w_{\alpha,n}}{\mathcal{W}(x_{\alpha,n}) - E_0}}, \quad (34)$$

where the weights accumulated according to Eqs. (29) and (30) must acquire an extra factor $1/(\mathcal{W}(x) - E_0)$ in the averages to compensate for the modified distribution of Eq. (31). This is the same difference that links Eq. (33) with Eq. (14). In the load-balanced LRDMC algorithm, E_0 is updated after each branching step.

In addition, in the load-balanced algorithm, the ground-state energy has the zero-variance-zero-bias property since $|\Psi_0\rangle$ is an eigenstate of the (fixed-node) Hamiltonian \hat{H} . However, the variance behavior in the vicinity of the nodal surface is different from the conventional one. Let us denote again the distance from the nodal surface as δ . In the load-balanced LRDMC algorithm, we have the additional factor $(\mathcal{W}(x) - E_0)$ in the local observable part, which diverges as δ^{-1} in the vicinity of nodal surface. As in the conventional LRDMC algorithm, the trial wave function Ψ_G and the fixed-node solution Ψ_0 vanish linearly with this distance, i.e., $\Psi_G \propto \delta$ and $\Psi_0 \propto \delta$, and the local energy e_L diverges as $e_L \propto \delta^{-1}$ with a trial wavefunction Ψ_G that is not an exact eigenfunction of the fixed-node Hamiltonian \hat{H} . Therefore, $\Pi(x)$, $e_L(x)$, and $(\mathcal{W}(x) - E_0)$ behave as $\propto \delta$, δ^{-1} , and δ^{-1} , respectively. The calculation of the energy variance requires the integral $\int_{\epsilon}^1 d\delta \Pi(\delta) (e_L(\delta) / (\mathcal{W}(\delta) - E_0))^2 \sim \int_{\epsilon}^1 d\delta (\delta) (1)^2 = \int_{\epsilon}^1 d\delta \delta$, where ϵ is an infinitesimal positive number. This is always finite, as in the conventional LRDMC algorithm. For derivatives, such as $\partial e_L \propto \delta^{-2}$, we have $\int_{\epsilon}^1 d\delta \Pi(\delta) (\partial e_L(\delta) / (\mathcal{W}(\delta) - E_0))^2 \sim \int_{\epsilon}^1 d\delta (\delta) (\delta^{-2} / \delta^{-1})^2 = \int_{\epsilon}^1 d\delta \delta^{-1}$. Thus, despite the divergence of the variance of derivatives being milder than that in the conventional algorithm (i.e., $\int_{\epsilon}^1 d\delta \delta^{-2}$), a reweighting or regularization technique is still needed to cure the divergence. Nevertheless, we can expect the error bar on energy derivatives, such as ionic forces, in the load-balanced algorithm to be smaller than the one computed by the conventional algorithm at fixed statistics due to reduced fluctuations of the reweighted operators close to the nodes.

In addition to the differences in the weighted averages, the main change with respect to the conventional LRDMC algorithm concerns the way the branching rate is set. Indeed, in this modified LRDMC algorithm, the number of steps between two branchings, denoted N_{proj} , is no longer stochastically determined as in the conventional algorithm. In the conventional LRDMC algorithm, as described in Subsection II B, N_{proj} corresponds to the randomly determined number of projections required to reach the projection time τ between two branching steps. Instead, in the load-balanced LRDMC, N_{proj} is defined *a priori* as a fixed input parameter. Indeed, no continuous-time limit is needed, as no $\lambda \rightarrow \infty$ limit is taken. The electrons instantaneously jump from site to site, with a sequence determined by N_{proj} , equally set for all walkers. As a result, the computational cost associated with the projection steps becomes strictly uniform across all walkers. This uniformity is expected to prevent any degradation in parallel efficiency, even when a large number of walkers is employed. Indeed, as we will demonstrate later, the weak-scaling results clearly show that the conventional algorithm suffers from a decline in weak-scaling performance, which goes like $\ln(N_w)$. In contrast, the new algorithm exhibits excellent weak-scaling behavior. This indicates that the proposed modification is crucial for maintaining a high computational throughput, especially in LRDMC calculations that aim to leverage GPU hardware architectures. Moreover, we have numerically verified to high precision that both the conventional and load-balanced algorithms yield consistent results in the limit $a \rightarrow 0$, provided the average number of projection steps is the same. This confirms the equivalence of the two algorithms in our testbed systems, as far as the convergence properties are concerned.

The pseudocode of the load-balanced LRDMC algorithm is reported in Algorithm 2.

III. VALIDATION OF THE ALGORITHM AND ITS IMPLEMENTATION

We first demonstrate that the conventional algorithm of LRDMC and the load-balanced version yield identical results. In the QMC community, recent efforts have aimed at systematically evaluating the reproducibility of results across independently developed codes.¹⁵ A notable example is a collaborative study involving 11 different community-developed QMC packages, in which the total energy and binding energy of a water-methane dimer were computed using a standardized basis set and a common effective core potential, thereby allowing for a controlled comparison of code implementations. In the present work, we adopt the same water-methane system as our benchmark case. The two LRDMC algorithms—conventional and load-balanced—were implemented using the jQMC³³ and TurboRVB³⁴ packages. For both oxygen and hydrogen atoms, we employed correlation-consistent effective core potentials (ccECPs)^{35,36} and used the associated cc-pVTZ basis sets, as in the aforementioned benchmark study. The angular part of the basis functions was expressed using spherical (solid) harmonics. The trial wavefunction was generated using PySCF^{37,38} and converted into the jQMC and TurboRVB wavefunction formats via TREX-IO.³⁹ We adopted the same functional form and variational parameters of the Jastrow factor used specifically in the TurboRVB calculations of the benchmark study¹⁵ for both the jQMC and TurboRVB calculations. A central technical challenge in

TABLE I. LRDMC energies of the water molecule for discretized lattice spaces (a), computed using both the load-balanced and conventional algorithms implemented in jQMC and TurboRVB.

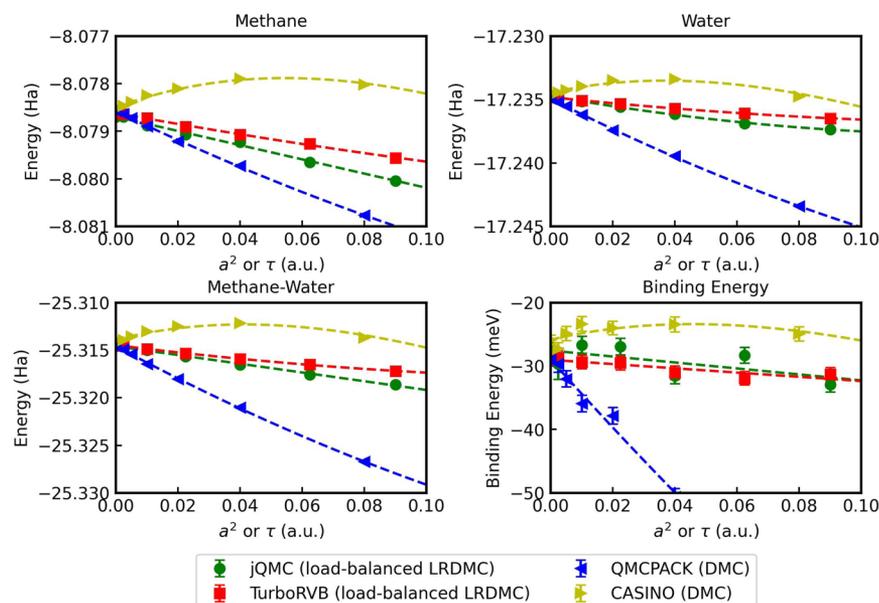
a (bohr)	LRDMC energy (Ha)			
	jQMC		TurboRVB	
	Load-balanced	Conventional	Load-balanced	Conventional
0.05	-17.234 83(3)	-17.234 93(5)	-17.234 85(2)	-17.234 87(1)
0.10	-17.235 16(2)	-17.235 14(4)	-17.235 09(2)	-17.235 07(1)
0.15	-17.235 59(3)	-17.235 62(5)	-17.235 35(2)	-17.235 40(1)
0.20	-17.236 16(2)	-17.236 22(5)	-17.235 71(2)	-17.235 73(1)
0.25	-17.236 88(2)	-17.236 80(6)	-17.236 08(2)	-17.236 11(1)
0.30	-17.237 35(2)	-17.237 23(5)	-17.236 48(2)	-17.236 48(1)

TABLE II. Total energies (Ha) of the methane, water, methane–water, and its binding energy (meV), obtained with the extrapolation to $\tau \rightarrow 0$ (DMC) and $a \rightarrow 0$ (LRDMC). The CASINO (DMC), QMCPACK (DMC), and TurboRVB (conventional LRDMC) results are taken from Ref. 15. The numbers in parentheses indicate 1σ in the last digit(s).

Package	Methane (Ha)	Water (Ha)	Methane–Water (Ha)	Binding energy (meV)
CASINO (DMC)	-8.078 56(1)	-17.234 73(2)	-25.314 32(3)	-26.8(1.0)
QMCPACK (DMC)	-8.078 58(2)	-17.234 82(3)	-25.314 43(7)	-29.0(1.1)
TurboRVB (conventional LRDMC)	-8.078 60(1)	-17.234 79(1)	-25.314 45(1)	-28.1(3)
TurboRVB (load-balanced LRDMC)	-8.078 62(2)	-17.234 82(2)	-25.314 47(2)	-29.0(7)
jQMC (load-balanced LRDMC)	-8.078 70(2)	-17.234 72(2)	-25.314 59(4)	-27.6(1.2)

DMC calculations using ECPs is the treatment of the non-local part of the ECP operator. In this work, we adopt the determinant locality approximation (DLA)⁴⁰ within the LRDMC framework, which corresponds to using the DLA with the T-move⁴¹ scheme in standard DMC.²⁴

To begin with, we verified that the conventional and load-balanced LRDMC algorithms yield identical results for the water molecule, which is as one of the benchmark systems in the previous reproducibility study.¹⁵ Table I shows the total energies computed with both algorithms for $a = 0.05, 0.10, 0.15, 0.20, 0.25,$ and 0.30 a.u.

**FIG. 2.** Total energy of methane, water, methane–water dimer, and binding energy. The data for QMCPACK and CASINO are taken from Ref. 15. The error bars in the graphs represent one standard deviation (1σ). The corresponding numbers are shown in Table II.

This verification was carried out using both jQMC and TurboRVB packages. The conventional LRDMC results obtained with TurboRVB were taken from Ref. 15, while all other data were newly computed in the present work. As shown in Table I, for all values of a , the total energies obtained with both jQMC and TurboRVB agree within the error bars between the conventional and load-balanced LRDMC algorithms. This demonstrates that, given the same trial wavefunction, both algorithms sample the same fixed-node ground state at fixed a , confirming the numerical equivalence of the two projection algorithms. A more detailed discussion of the computational performance and parallelization efficiency (i.e., weak-scaling) of these algorithms is provided in Sec. IV.

Next, we compare the DMC and LRDMC results obtained from different packages. As demonstrated in Ref. 15, the energies at finite time step discretization (τ) in DMC and finite lattice-space discretization (a) in LRDMC are dependent on the details of each package's implementation and thus are not suitable for direct comparison. Therefore, we perform calculations analogous to those in the reproducibility study¹⁵ to assess the consistency of our implementation. In particular, for the methane molecule, water molecule, and methane-water dimer, we compute the total energies in the $a \rightarrow 0$ limit using the load-balanced LRDMC implementation in both jQMC and TurboRVB. These results are then compared with the $\tau \rightarrow 0$ DMC energies reported in Ref. 15, obtained using QMCPACK^{42,43} and CASINO,⁴⁴ as well as with the $a \rightarrow 0$ extrapolated values from the conventional LRDMC implementation in TurboRVB.³⁴ The results are summarized in Fig. 2 and Table II. As clearly shown in these comparisons, the computed total energies and binding energies exhibit excellent agreement with the literature values, thereby validating the correctness of our load-balanced LRDMC implementations in both jQMC and TurboRVB packages.

IV. WEAK SCALING BENCHMARK AND DIRECT COMPARISON BETWEEN THE CONVENTIONAL AND LOAD-BALANCED LRDMC ALGORITHMS

In this study, the weak-scaling benchmark was conducted using the jQMC package on Leonardo,⁴⁵ a supercomputer operated by CINECA in Italy. Leonardo is equipped with four NVIDIA[®] A100 GPUs per node. By assigning a large number of walkers to each GPU and utilizing inter-node parallelization, we performed benchmarks for the two different LRDMC algorithms. For this scaling analysis, we chose the benzene molecule, which has 30 valence electrons ($N_e = 30$). For both carbon and hydrogen atoms, we employed ccECPs pseudopotentials,^{35,36} along with the corresponding aug-cc-pVTZ basis sets. For the angular part, the polynomial (Cartesian) function was employed (i.e., the d and f orbitals are composed of 6 and 10 orbitals). The trial wavefunction was generated using PySCF^{37,38} and subsequently converted to the jQMC wavefunction format via TREX-IO.³⁹ The Jastrow factor included both two-body and three-body terms³⁴ and was variationally optimized by minimizing the energy using the stochastic reconfiguration (SR) method.⁴⁶ The conventional LRDMC calculations have been performed at an optimal branching time $\tau = 0.30$ a.u., a good compromise between reduction of load-imbalance and increase in stochastic error bars due to walkers fluctuations, yielding an *average* projection number of $N_{\text{proj}} = 294$. The load-balanced LRDMC calculations employed the same projection number ($N_{\text{proj}} = 294$) to have a clearer idea on the relative efficiency between the two algorithms.

Figure 3(a) presents the results of the weak-scaling test for the conventional and load-balanced LRDMC algorithms, each one normalized with respect to the corresponding numerical cost of 400 walkers on 4 GPUs. These benchmarks provide a quantitative assessment of the parallel efficiency of the implementations with

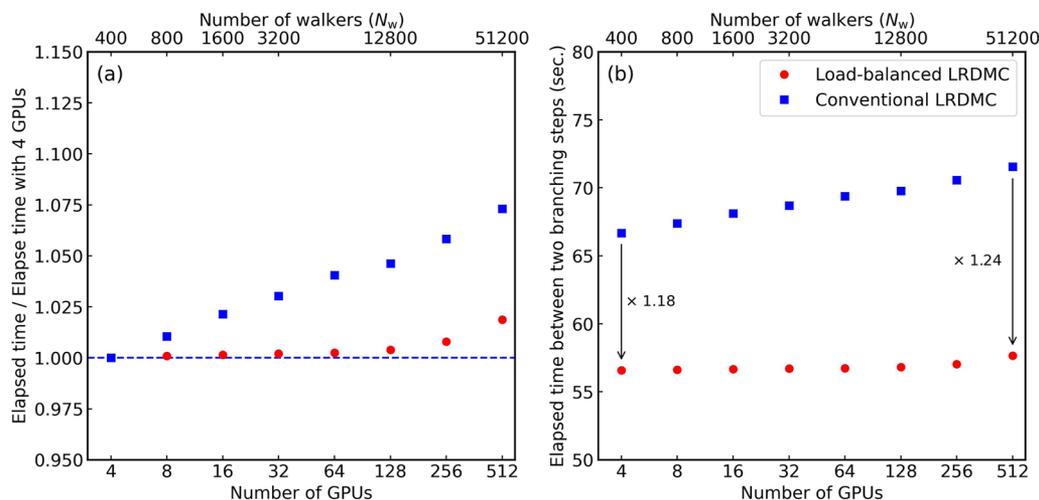


FIG. 3. Comparison of the weak-scaling benchmark between the conventional and load-balanced LRDMC algorithms, measured on Leonardo using benzene molecule ($N_e = 30$). A Hamiltonian discretization parameter of $a = 0.30$ a.u. was applied in all LRDMC calculations. In the conventional LRDMC runs, τ was set to 0.3 a.u., yielding an *average* projection number of $N_{\text{proj}} = 294$. The load-balanced LRDMC calculations employed the same projection number ($N_{\text{proj}} = 294$).

multiple walkers and serve as critical indicators of its suitability for large-scale LRDMC simulations. As clearly shown in Fig. 3(a), the conventional algorithm exhibits a steep decline in computational efficiency as a function of the logarithm of the number of walkers (i.e., the degree of GPU parallelization). As explained in Sec. II, this is because the conventional algorithm determines the length of each projection step using a stochastic estimate of the persistence time, resulting in a sizable load imbalance among parallel walkers. Since all walkers must wait until the slowest projection operation is completed, some walkers remain idle in the last projection periods. This behavior leads to an increased likelihood of encountering “slow” walkers with long projection times as the number of walkers grows, resulting in a linear degradation of weak-scaling efficiency as a function of $\ln(N_w)$. In contrast, the load-balanced LRDMC algorithm ensures, by design, the computational workload is uniformly distributed among walkers. Consequently, the benchmark results demonstrate that the proposed algorithms maintains nearly optimal weak scaling even for very large number of walkers (up to 51 200 in our test).

Figure 3(b) presents the actual elapsed times between two consecutive branching steps for the conventional and load-balanced LRDMC algorithms. This benchmark demonstrates the extent to which the load-balanced LRDMC algorithm accelerates the calculation compared to the conventional LRDMC algorithm. As clearly shown in Fig. 3(b), the load-balanced LRDMC algorithm consistently outperforms the conventional algorithm for any number of N_w . The speed-up becomes more pronounced as N_w increases: for $N_w = 400$, the speed-up factor is $\times 1.18$, while for the largest number of walkers tested in this work, $N_w = 51\,200$, the speed-up reaches $\times 1.24$. It should be noted that the conventional and load-balanced LRDMC calculations were carried out with the same number of branching steps and they provided consistent error bars. These results indicate that the load-balanced LRDMC algorithm not only improves the scaling with respect to N_w compared to the conventional LRDMC algorithm but also yields a substantial reduction in the actual computational cost.

V. CONCLUSIONS

In this work, we have introduced a load-balanced algorithm for the lattice regularized diffusion Monte Carlo (LRDMC) method, which is based on the Green’s Function Monte Carlo (GFMC) framework. We implemented this algorithm in the *ab initio* quantum Monte Carlo packages, TurboRVB and jQMC, and demonstrated that (1) it reproduces the same results as the conventional LRDMC algorithm and (2) it achieves superior parallel efficiency compared to the conventional algorithm as the number of walkers (N_w) increases up to $\sim 10^5$. A direct comparison between the conventional and load-balanced LRDMC calculations reveals that the latter is more efficient by $\times 1.18$ and $\times 1.24$ for $N_w = 400$ and $N_w = 51\,200$, respectively, always due to a more optimal management of the walkers evolution between two branching steps. Achieving an appropriate load balance among walkers is increasingly critical, as many modern supercomputers are equipped with many-core GPUs. Maximizing GPU utilization requires processing a large number of walkers on each GPU, which in turn necessitates simulations with many walkers. The present work addresses this

challenge, enabling more efficient DMC simulations on GPU-based architectures.

While our benchmarks were carried out in the context of *ab initio* QMC, the same load-balanced implementation can be applied wherever GFMC-based simulations require large walker populations. GFMC is not only widely used in electronic-structure calculations but also in lattice-model studies in condensed-matter physics¹⁸ and in *ab initio* nuclear-physics⁴⁷ computations. In any such application where performance is limited by walker imbalance, the load-balanced LRDMC (GFMC) algorithm presented here should yield significant speedups.

ACKNOWLEDGMENTS

K.N. and M.C. are grateful for computational resources from EuroHPC under computational Grant No. EHPC-EXT-2024E01-064 allocated on Leonardo (booster partition). K.N. and M.C. acknowledge EPICURE, a EuroHPC Joint Undertaking initiative for supporting this project on the booster partition of Leonardo through the EuroHPC JU 2024E01 call for proposals for extreme scale access mode. K.N. acknowledges financial support from the Ministry of Education, Culture, Sports, Science and Technology (MEXT) through Leading Initiative for Excellent Young Researchers (Grant No. JPMXS0320220025) and from the Japan Science and Technology Agency (JST) through PRESTO (Grant No. JPMJPR24J9). M.C. acknowledges the European High Performance Computing Joint Undertaking (JU) for the partial support through the “EU-Japan Alliance in HPC” HANAMI project (Hpc AlliaNce for Applications and supercoMputing Innovation: the Europe-Japan collaboration). We dedicate this paper to the memory of Professor Sandro Sorella (SISSA), who passed away in 2023. He was one of the most influential researchers in the QMC community and profoundly inspired this work through his development of the *ab initio* QMC code, TurboRVB. He was the first to propose the load-balanced algorithm described here and to write its early version implementation in the TurboRVB package. His former intuition has become even more relevant now, with the advent of Python-based codes, jQMC, written in JAX,⁴⁸ which favors computations with many walkers on GPUs, such as the one reported in this work.

AUTHOR DECLARATIONS

Conflict of Interest

The authors have no conflicts to disclose.

Author Contributions

Kousuke Nakano: Conceptualization (equal); Data curation (lead); Formal analysis (equal); Funding acquisition (equal); Investigation (lead); Methodology (equal); Project administration (supporting); Resources (supporting); Software (lead); Supervision (supporting); Validation (lead); Visualization (lead); Writing – original draft (equal); Writing – review & editing (equal). **Sandro Sorella:** Conceptualization (lead); Formal analysis (supporting); Methodology (equal); Software (equal). **Michele Casula:** Conceptualization

(equal); Data curation (supporting); Formal analysis (equal); Funding acquisition (equal); Investigation (supporting); Methodology (equal); Project administration (lead); Resources (lead); Software (equal); Supervision (lead); Validation (supporting); Visualization (supporting); Writing – original draft (equal); Writing – review & editing (lead).

DATA AVAILABILITY

The code and data supporting the findings of this study are available from the jQMC and TurboRVB GitHub repositories: (<https://github.com/kousuke-nakano/jQMC>) and (<https://github.com/sissaschool/turborvb>), respectively.

APPENDIX A: THE STATIONARY DISTRIBUTION OF THE CONVENTIONAL LATTICE REGULARIZED MONTE CARLO METHOD

Let Green's function with importance sampling be defined as

$$\mathcal{G}_{x',x} = (\Lambda_{x',x} - \mathcal{H}_{x',x}) \cdot \frac{\Psi_G(x')}{\Psi_G(x)}, \quad (\text{A1})$$

with $\Lambda \equiv \lambda \delta_{x',x}$ being a diagonal matrix. Since $\mathcal{G}_{x',x}$ is generally not normalized, we introduce a normalization factor $b_x = \sum_{x'} \mathcal{G}_{x',x}$ so that the transition probability can be written as

$$p_{x',x} = \frac{\mathcal{G}_{x',x}}{b_x}. \quad (\text{A2})$$

Here, we assume that all elements of $p_{x',x}$ are non-negative real numbers after the FN-approximation. In the GFMC framework, both position x and weight w are updated as follows.

1. Generate $x_{n+1} = x'$ with probability $p_{x',x}$.
2. Update the weight: $w_{n+1} = w_n \cdot b_x$.

They imply that we are tracking not only the probability of position, denoted as $\pi(x)$, but also a joint probability distribution over both position and weight, denoted as $\mathcal{P}(x, w)$. First, we derive the equilibrium probability of position, $\pi(x)$. The master equation for the probability distribution $\pi(x)$ over the state space becomes

$$\pi_{n+1}(x_j) = \sum_i \pi_n(x_i) p_{x_j, x_i}. \quad (\text{A3})$$

Defining the vector,

$$\vec{\pi}_n = [\pi_n(x_1), \pi_n(x_2), \dots, \pi_n(x_M)], \quad (\text{A4})$$

the transition matrix $P \in \mathbb{R}^{M \times M}$ with elements $P_{i,j} \equiv p_{x_j, x_i}$, where M is the number of basis, the above-mentioned master equation can be expressed compactly as

$$\vec{\pi}_{n+1} = \vec{\pi}_n P. \quad (\text{A5})$$

The matrix P is a right stochastic matrix: a square matrix with non-negative entries and rows summing to 1 ($\sum_j P_{ij} = 1$). If P is irreducible (i.e., any state is reachable from any other in a finite number of steps) and aperiodic (i.e., the greatest common divisor of the return times to a state is one), then it is a *primitive matrix* and satisfies the conditions of the Perron–Frobenius Theorem.⁴⁹

By Gershgorin's circle theorem, the maximum eigenvalue of any right stochastic matrix is 1,⁵⁰ and the Perron–Frobenius theorem guarantees that all other eigenvalues have modulus less than 1. Consequently, the iteration of the projections by P filters out the eigenstate with the eigenvalue of 1 and there exists a unique stationary distribution $\vec{\pi}_{\text{eq}}$, such that

$$\vec{\pi}_{\text{eq}} = \vec{\pi}_{\text{eq}} P. \quad (\text{A6})$$

This describes the mathematical structure behind the power method for iteratively updating probability distributions. In other words, repeatedly applying the update $\vec{\pi}_{k+1} = \vec{\pi}_k P$ from any initial distribution leads to convergence toward the stationary state $\vec{\pi}_{\text{eq}}$, as all non-dominant eigenmodes decay exponentially. We can immediately show that $\pi_{\text{eq}}(x) \propto b_x |\Psi_G(x)|^2$ is indeed a stationary distribution by plugging it into Eq. (A3),

$$\begin{aligned} \sum_x b_x |\Psi_G(x)|^2 \cdot p_{x',x} &= \sum_x |\Psi_G(x)|^2 \cdot \mathcal{G}_{x',x} \\ &= \sum_x (\lambda \delta_{x',x} - \mathcal{H}_{x',x}) \Psi_G(x) \Psi_G(x') \\ &= \sum_x (\lambda \delta_{x,x'} - \mathcal{H}_{x,x'}) \Psi_G(x) \Psi_G(x') \\ &= \sum_x (\lambda \delta_{x,x'} - \mathcal{H}_{x,x'}) \Psi_G(x) / \Psi_G(x') \cdot |\Psi_G(x')|^2 \\ &= \sum_x \mathcal{G}_{x,x'} |\Psi_G(x')|^2 \\ &= b_{x'} |\Psi_G(x')|^2, \end{aligned} \quad (\text{A7})$$

where the hermiticity of the Hamiltonian $\mathcal{H}_{x,x'} = \mathcal{H}_{x',x}$ is exploited, by also assuming for simplicity that the Hamiltonian matrix elements are real. Then, by the Perron–Frobenius theorem, this left eigenvector with eigenvalue 1 is unique, confirming that $b_x |\Psi_G(x)|^2$ corresponds to the stationary distribution [also see Eq. (8.40) of Ref. 20].

Next, we derive the equilibrium distribution of position and weight, $\mathcal{P}(x, w)$. Notice that the stationary distribution $\pi(x)$ obtained earlier is the unweighted marginal of $\mathcal{P}(x, w)$, i.e., $\pi(x) = \int dw \mathcal{P}(x, w)$. The transition kernel K for the weighted process is defined as

$$K(x', w' | x, w) = p_{x',x} \delta(w' - w b_x). \quad (\text{A8})$$

The corresponding master equation becomes

$$\mathcal{P}_{n+1}(x', w') = \sum_x \int dw K(x', w' | x, w) \mathcal{P}_n(x, w). \quad (\text{A9})$$

Multiplying both sides by w' and integrating yields

$$\begin{aligned} \int dw' w' \mathcal{P}_{n+1}(x', w') &= \sum_x p_{x',x} b_x \int dw w \mathcal{P}_n(x, w) \\ &= \sum_x \mathcal{G}_{x',x} \int dw w \mathcal{P}_n(x, w). \end{aligned} \quad (\text{A10})$$

The marginal weighted probability $\Pi(x) = \int dw w \mathcal{P}(x, w)$ is nothing but the many-body wavefunction at step n , i.e., $\Psi_n(x) \Psi_G(x) \equiv \Pi_n(x) = \int dw w \mathcal{P}_n(x, w)$. In the vector notation, this leads to

$$\vec{\Pi}_{n+1} = \vec{\Pi}_n G, \quad (\text{A11})$$

where $G_{ij} \equiv \mathcal{G}_{x_j, x_i}$. Since $\mathcal{G}_{x', x} = p_{x', x} b_x$, we can define a diagonal matrix B with elements $B_{i,i} = b_{x_i}$, so that $G = BP$. If B is strictly positive and P is primitive, then G is also primitive. Therefore, according to the Perron–Frobenius theorem, the master equation has a stationary (unnormalized) solution $\Pi_{\text{eq}}(x)$, and the spectral radius $\rho(G) > 0$ ensures the existence of a dominant eigenvector. In fact, the eigenvector of G is $\Psi_0(x)\Psi_G(x)$, and the corresponding eigenvalue is $\lambda - E_0$, which can be easily verified by substitution,

$$\sum_x \Psi_0(x)\Psi_G(x) \cdot \mathcal{G}_{x', x} = \sum_x \Psi_0(x)\Psi_G(x)(\lambda\delta_{x', x} - \mathcal{H}_{x', x}) \frac{\Psi_G(x')}{\Psi_G(x)}, \quad (\text{A12})$$

$$= \sum_x \Psi_0(x)(\lambda\delta_{x', x} - \mathcal{H}_{x', x})\Psi_G(x'), \quad (\text{A13})$$

$$= (\lambda - E_0)\Psi_0(x')\Psi_G(x'). \quad (\text{A14})$$

The eigenvalue $(\lambda - E_0)$ is apparently the dominant eigenvalue because $E_0 < E_i$ for $\forall i$, where i refers to the i th excited state. Thus, applying G repeatedly to any initial distribution yields

$$\bar{\Pi}_{\text{init}} G^n = \bar{\Pi}_n \sim (\lambda - E_0)^n \Psi_0(x)\Psi_G(x), \quad (\text{A15})$$

where the prefactor $(\lambda - E_0)^n$ is canceled out in computing an observable.

APPENDIX B: THE STATIONARY DISTRIBUTION OF THE LOAD-BALANCED LATTICE REGULARIZED MONTE CARLO METHOD

Given the definition of the modified Green's function in the load-balanced LRDMC algorithm,

$$\mathcal{G}_{x', x} = -\frac{1}{\mathcal{W}(x) - E_0} \tilde{\mathcal{H}}_{x', x} \frac{\Psi_G(x')}{\Psi_G(x)}, \quad (\text{B1})$$

with $\tilde{\mathcal{H}}_{x', x}$ being a hermitian (and real for the sake of simplicity) matrix, we can follow the same steps carried out in Appendix A to show that the marginal weighted probability, which reads

$$\Pi_{\text{eq}}(x) = \int dw w \mathcal{P}(x, w) = (\mathcal{W}(x) - E_0)\Psi_G(x)\Psi_0(x) \quad (\text{B2})$$

is stationary in the stochastic Markov chain process driven by $\mathcal{G}_{x', x}$ in Eq. (B1), such that

$$\bar{\Pi}_{\text{eq}} = \bar{\Pi}_{\text{eq}} G. \quad (\text{B3})$$

This can be proven as follows:

$$\begin{aligned} \sum_x (\mathcal{W}(x) - E_0)\Psi_0(x)\Psi_G(x) \cdot \mathcal{G}_{x', x} \\ = -\sum_x \Psi_0(x)\Psi_G(x) \tilde{\mathcal{H}}_{x', x} \frac{\Psi_G(x')}{\Psi_G(x)}, \end{aligned} \quad (\text{B4})$$

$$= -\sum_x \Psi_0(x) \tilde{\mathcal{H}}_{x', x} \Psi_G(x'), \quad (\text{B5})$$

$$= (\mathcal{W}(x') - E_0)\Psi_0(x')\Psi_G(x'). \quad (\text{B6})$$

This does not guarantee that the dominant eigenvalue of the master equation is 1 because \mathcal{G} in Eq. (B1) [Eq. (28) of the main text] is not normalized. However, Π_{eq} is certainly a saddle point in the space of distribution functions for those processes driven by \mathcal{G} . Moreover, by performing an eigenvalue decomposition of the initial state Ψ_{init} ,

$$|\Psi_{\text{init}}\rangle = \sum_{i \geq 0} a_i |\Psi_i\rangle, \quad (\text{B7})$$

where a_i is the coefficient for the i th eigenvectors (Ψ_i) with energy ε_i , and $\varepsilon_{i+1} \geq \varepsilon_i$, one can show that the initial distribution $\Pi_{\text{init}}(x) = (\mathcal{W}(x) - E_0)\Psi_G(x)\Psi_{\text{init}}(x)$, with E_0 being the best guess for the ground state energy ε_0 (assumed for simplicity non degenerate), is transformed by the application of \mathcal{G} in such a way that the distribution at the next iteration reads

$$\begin{aligned} \Pi_{\text{next}}(x) = (\mathcal{W}(x) - \varepsilon_0) \left[a_0 \Psi_G(x)\Psi_0(x) \right. \\ \left. + \sum_{i > 0} a_i \frac{\mathcal{W}(x) - \varepsilon_i}{\mathcal{W}(x) - \varepsilon_0} \Psi_G(x)\Psi_i(x) \right]. \end{aligned} \quad (\text{B8})$$

The remaining part beyond $a_0 \Psi_G(x)\Psi_0(x)$ will have renormalized coefficients $a_i \rightarrow a_i \frac{\mathcal{W}(x) - \varepsilon_i}{\mathcal{W}(x) - \varepsilon_0}$. If $\left| \frac{\mathcal{W}(x) - \varepsilon_i}{\mathcal{W}(x) - \varepsilon_0} \right| < 1 \quad \forall x$, then the amplitude of the higher energy coefficients will be damped and the distribution will converge to $(\mathcal{W}(x) - \varepsilon_0)\Psi_G(x)\Psi_0(x)$ for a large enough number of applications of \mathcal{G} together with the update $E_0 \rightarrow \varepsilon_0$. The condition $\left| \frac{\mathcal{W}(x) - \varepsilon_i}{\mathcal{W}(x) - \varepsilon_0} \right| < 1$ holds for $\varepsilon_i < 2\mathcal{W}(x) - \varepsilon_0$. If the initial state is good enough (namely, with higher energy components suppressed, i.e., $a_k \ll 1$ for k such that $\varepsilon_k > 2\mathcal{W}(x) - \varepsilon_0$, and with a non-zero overlap with the ground state, i.e., $a_0 \neq 0$), then the distribution will converge toward the mixed ground state one Eqs. (B2) and (31). The convergence can be easily monitored during the simulation by looking at the decrease (or stability) in E_0 , estimated during the projection [see Eq. (33)]. Indeed, breaking the convergence condition will mean introducing very high energy states, leading to an energy blow-up. In practice, we never found such a case in all systems studied so far, starting from “regular” variational trial states. This is related to the fact that $\mathcal{W}(x)$ turns out to be always large compared to the energy scales of the projected wavefunction, yielding $\forall x$ a sufficiently wide energy stability window $[\varepsilon_0, 2\mathcal{W}(x) - \varepsilon_0]$.

APPENDIX C: EMULATION OF THE PARALLELIZATION EFFICIENCY OF THE CONVENTIONAL LRDMC ALGORITHM

The degradation of the parallelization efficiency of the conventional LRDMC algorithm can be easily demonstrated using a simple emulation of the implementation (i.e., a toy model). If the total off-diagonal transition probability $b_x = \sum_{x' \neq x} \mathcal{G}_{x', x}$ is assumed to follow a normal distribution (or is approximated as a constant), then the number of projection steps required until $\tau_{\text{left}} = 0$ can be simulated using a simple random sampling program, corresponding lines 3–5, 7–9, and 11–13, 17 of Algorithm 1. In Fig. 4, we plot the minimum, maximum, and average number of projection steps required to reach $\tau_{\text{left}} = 0$ as a function of the number of walkers used in the emulation. In this emulation, the projection time τ was set 5.0, and the off-diagonal sum of the Green's function, $b_x \equiv \sum_{x' \neq x} \mathcal{G}_{x', x}$

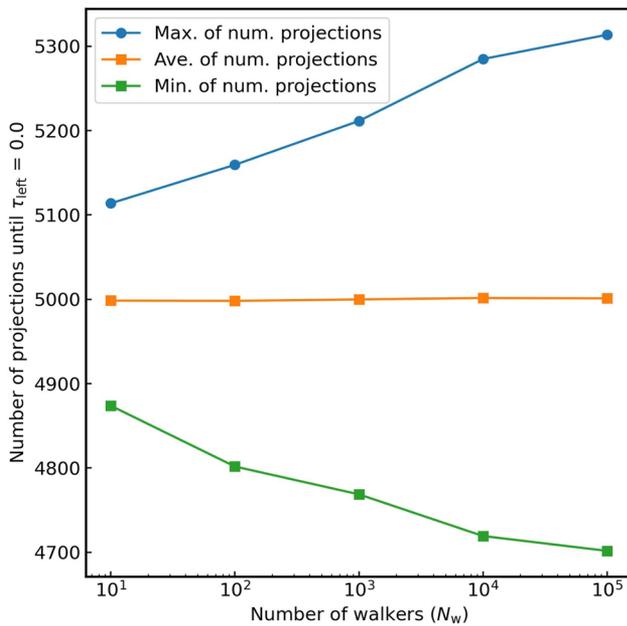


FIG. 4. Emulation of the evolution of τ in the conventional LRDMC implementation with respect to the number of walkers (see the main text for details). Simulations were carried out for walker populations of $N_w = 10, 100, 1000, 10\,000,$ and $100\,000$.

in Eq. (24), was sampled from a normal distribution with mean of 1×10^3 and standard deviation of 0.5. As the figure shows, while the average number of steps remains constant regardless of the number of walkers, the branching algorithm must wait for the slowest walker to finish. The outcome of this emulation reproduces the $\log(N_w)$ scaling of the actual benchmark results shown in Fig. 3.

APPENDIX D: FIXED-NODE APPROXIMATION AND LATTICE DISCRETIZATION IN LRDMC

As already mentioned, Green's function cannot be made strictly positive for fermions; therefore, the fixed-node approximation (FNA) has to be introduced²⁰ in order to avoid the sign problem. The FNA is implemented by defining an effective fixed-node (FN) Hamiltonian $\mathcal{H}_{x',x}^{\text{FN}}$, with modified off-diagonal matrix elements and with the inclusion of the spin-flip term $\mathcal{V}_{\text{sf}}(x) = \sum_{x':s_{x',x}>0} \mathcal{H}_{x',x} \Psi_G(x')/\Psi_G(x)$,

$$\mathcal{H}_{x',x}^{\text{FN}} = \begin{cases} \mathcal{H}_{x,x} + \mathcal{V}_{\text{SF}}(x) & \text{for } x' = x, \\ \mathcal{H}_{x',x} & \text{for } x' \neq x, s_{x',x} < 0, \\ 0 & \text{for } x' \neq x, s_{x',x} > 0, \end{cases} \quad (\text{D1})$$

where $s_{x',x} = \Psi_G(x') \mathcal{H}_{x',x} \Psi_G(x)$. In other words, for off-diagonal matrix elements $\mathcal{H}_{x',x}$ coming from the Laplacian discretization (kinetic terms), $s_{x',x} > 0$ indicates that the hopping $x \rightarrow x'$ crosses the nodal surface of the guiding function. For more complicated off-diagonal elements, coming, e.g., for the non-local pseudopotentials, the sign $s_{x',x}$ has a less trivial interpretation, and it is chosen

simply to guarantee the non-negativity of the FN $\mathcal{G}_{x',x}^{\text{FN}}$. Indeed, the use of the FN Green's function with the so-called guiding function,

$$\mathcal{G}_{x',x}^{\text{FN}} = (\Lambda_{x',x} - \mathcal{H}_{x',x}^{\text{FN}}) \frac{\Psi_G(x')}{\Psi_G(x)} = \lambda \delta_{x',x} - \frac{\Psi_G(x')}{\Psi_G(x)} \mathcal{H}_{x',x}^{\text{FN}} \quad (\text{D2})$$

can prevent the crossing of regions where the configuration space yields a sign flip of Green's function; therefore, the walkers are constrained in the same nodal pockets to avoid the sign problem.

The Hamiltonian in the spin flip term is composed of two contributions. $\mathcal{H}_{x',x} \Psi_G(x')/\Psi_G(x) = \mathcal{K}_{x',x} \Psi_G(x')/\Psi_G(x) + \mathcal{V}_{x',x}^{\text{nl}} \Psi_G(x')/\Psi_G(x)$, where $\mathcal{K}_{x',x}$ and $\mathcal{V}_{x',x}^{\text{nl}}$ are the non-local elements in the kinetic and potential terms, respectively. The non-local potential term is considered only for calculations with non-local (semi-local) effective core potentials. With the non-local (semi-local) ECPs, the above-mentioned fixed-node Hamiltonian corresponds to the T-move⁴¹ in the standard DMC framework. If one replaces $\mathcal{V}_{x',x}^{\text{nl}} \Psi_G(x')/\Psi_G(x)$ with $\mathcal{V}_{x',x}^{\text{nl}} D_G(x')/D_G(x)$, where D_G represents the determinant part of the trial wavefunction Ψ_G , the resultant fixed-node Hamiltonian corresponds to the T-move with determinant locality approximation (DTM)⁴⁰ in the standard DMC framework.

In LRDMC, the original continuous Hamiltonian is regularized by allowing electron hoppings with step size a , in order to mimic the electronic kinetic energy on the continuum. The corresponding regularized Hamiltonian $\hat{\mathcal{H}}^a$ is then defined such that $\hat{\mathcal{H}}^a \rightarrow \hat{\mathcal{H}}$ for $a \rightarrow 0$. Namely, the kinetic and potential parts are approximated by a finite difference form: $\hat{\mathcal{H}}^a \equiv \hat{K}^a + \hat{V}^a$.

We start from the kinetic part \hat{K}^a . The discretized Laplacian ∇_a^2 acts on a function $f(x_i, y_i, z_i)$ as

$$\nabla_{a,i}^2 f(x_i, y_i, z_i) = \frac{1}{a^2} \{ [f(x_i + a) - f(x_i)] + [f(x_i - a) - f(x_i)] \} \\ \leftrightarrow y_i \leftrightarrow z_i. \quad (\text{D3})$$

Therefore, the discretized kinetic operator $\hat{K}^a \equiv -\frac{1}{2} \sum_i \nabla_{a,i}^2$ acts on $|x\rangle$ as

$$\hat{K}^a |x\rangle \equiv -\frac{1}{2} \sum_{i=1}^{N_e} \nabla_{a,i}^2 |x\rangle = \frac{3N_e}{a^2} |x\rangle - \frac{1}{2a^2} \sum_{j=1}^{6N_e} |x_j\rangle, \quad (\text{D4})$$

leading to

$$\hat{K}_{x',x}^a \frac{\Psi_G(x')}{\Psi_G(x)} = \frac{3N_e}{a^2} \delta_{x',x} - \frac{1}{2a^2} \frac{\Psi_G(x')}{\Psi_G(x)} \delta_{x',x}^{ad}, \quad (\text{D5})$$

where $\delta_{x',x}^{ad} = 1$ only if x' is adjacent to x , otherwise 0. Here, one should consider *only* $6N_e x'$ adjacent to x . Thus, the GFMC method is applicable to the continuous *ab initio* Hamiltonian, thanks to the sparsity of the discretized kinetic operator. The three versors, upon which the displacements in Eq. (D3) are defined, are set to rotate with random angles along the LRDMC simulation, in such a way that the underlying lattice is *randomized* and the lattice space extrapolation reach the continuous space limit faster.

The potential term \hat{V}^a is divided into local and non-local terms, $\hat{V}^a \equiv \hat{V}_{\text{loc}}^a + \hat{V}_{\text{nl}}$, where only the local term depends on the discretized mesh a , through the following definition:

$$V_{\text{loc}}^a(x) \equiv V_{\text{loc}}(x) + \frac{1}{2} \left[\frac{\sum_i (\nabla_{a,i}^2 - \nabla_i^2) \Psi_G(x)}{\Psi_G(x)} \right], \quad (\text{D6})$$

with $V_{\text{loc}}(x) = \sum_i v_{\text{ei}}(r_i) + \sum_{i,j} v_{\text{ee}}(r_{i,j}) + V_{\text{ii}}$ electron-ion (ei), electron-electron (ee), and ion-ion (ii) interactions. With the modified potential in Eq. (D6), the local energy of the original Hamiltonian coincides with that of the regularized one,

$$\begin{aligned} e_L^a(x) &= \frac{\langle \Psi_G | -\frac{1}{2} \nabla_a^2 + \hat{V}_{\text{loc}}^a + \hat{V}_{\text{nl}} | x \rangle}{\langle \Psi_G | x \rangle} \\ &= \frac{\langle \Psi_G | -\frac{1}{2} \nabla^2 + \hat{V}_{\text{loc}} + \hat{V}_{\text{nl}} | x \rangle}{\langle \Psi_G | x \rangle} \equiv e_L(x). \end{aligned} \quad (\text{D7})$$

Together with the randomized mesh, the condition in Eq. (D7) also accelerates the convergence of the FN $\hat{\mathcal{H}}^a$ ground-state energy to the continuous $a \rightarrow 0$ limit.

With the aim at defining a rigorously stable and robust method that works for any finite value of a , we need to further modify the local potential in Eq. (D6), to avoid negative divergences that can arise from the nodes of the guiding function, from electron configurations too close to the nuclei, or from both, where the bare electron-ion potential is strongly attractive. To proceed, we notice that the regularization of Eq. (D6) can be interpreted as the modification of the bare electron-ion potential involving the i th electron $v_{\text{ei}}(r_i) \rightarrow v_{\text{zv},i}^a(x)$, where

$$v_{\text{zv},i}^a(x) = v_{\text{ei}}(r_i) + \frac{(\nabla_{a,i}^2 - \nabla_i^2) \Psi_G(x)}{2\Psi_G(x)} \quad \text{for } i \in [1, N_e], \quad (\text{D8})$$

such that $V_{\text{loc}}^a(x) = \sum_i v_{\text{zv},i}^a(x) + \sum_{i,j} v_{\text{ee}}(r_{i,j}) + V_{\text{ii}}$. The above-mentioned modification cancels out most of the singularities of the attractive potential thanks to the electron-ion cusp conditions fulfilled by Ψ_G . However, we can still have unbounded negative values on the nodal surface of the guiding function. A safe possibility to protect the simulation from all these instabilities is given by the following modification of the electron-ion potential for each electron i :

$$v_{\text{max},i}^a(x) = \max [v_{\text{zv},i}^a(x), v_{\text{ei}}^a(r_i)], \quad (\text{D9})$$

where $v_{\text{ei}}^a(r_i)$ is defined as

$$v_{\text{ei}}^a(r_i) = -\sum_I \frac{Z_I}{\max(|r_i - R_I|, a)} \quad (\text{D10})$$

for all-electron calculations, which introduces a lower bound $\propto -1/a$, while for calculations with ECPs, $v_{\text{ei}}^a(r_i) = v_{\text{ei}}(r_i) = -\sum_I Z_I^{\text{pseudo}}/|r_i - R_I|$ because in the latter case, the bare Coulomb divergence is perfectly canceled by the localized pseudopotential channels for smooth pseudopotentials, implicitly added to \mathcal{V}_{sf} in Eq. (D1) and coming from the partial localization of \hat{V}_{nl} . Since the condition $v_{\text{ei}}^a(r_i) > v_{\text{zv},i}^a(x)$ is often satisfied for electrons within a lattice-space range from the nuclei, Eq. (D9) is applied, in practice,

only when the electrons cross the nodal surface with the discretized Laplacian (and the discretized ECP mesh),

$$v_{\text{opt},i}^a(x) = \begin{cases} v_{\text{max},i}^a(x) & \text{if } \Psi_G(x)\Psi_G(x_j) < 0, \text{ for at least one } x_j \\ v_{\text{zv},i}^a(x) & \text{otherwise,} \end{cases} \quad (\text{D11})$$

where x_j indicates the possible configurations proposed by the discretized Laplacian (and ECP). Within this formulation, and by replacing $v_{\text{zv},i}^a(x)$ in Eq. (D8) with $v_{\text{opt},i}^a(x)$ in Eq. (D11), the final regularized potential, $\mathcal{W}(x) = \sum_i v_{\text{opt},i}^a(x) + \sum_{i,j} v_{\text{ee}}(r_{i,j}) + \mathcal{V}_{\text{sf}}(x) + V_{\text{ii}}$, is protected from the possible negative energy instabilities close to the nodal surface and/or close to the nuclei, and guarantees the existence of a finite ground-state energy for any $a > 0$.

Finally, we notice that, specifically for all-electron calculations, multiple lattice discretization can be introduced to deal with different characteristic lengths in the real space, such as the ones affecting core and valence electrons; see Refs. 16, 22, and 32 for details. The multiple lattice discretization can be applied both to the conventional and load-balanced GFMC algorithms. For the sake of clarity, we used the single lattice discretization throughout the discussion.

REFERENCES

- W. Kohn and L. J. Sham, *Phys. Rev.* **140**, A1133 (1965).
- R. M. Martin, *Electronic Structure: Basic Theory and Practical Methods* (Cambridge University Press, 2004).
- J. P. Perdew and K. Schmidt, *AIP Conf. Proc.* **577**, 1–20 (2001).
- M. G. Medvedev, I. S. Bushmarinov, J. Sun, J. P. Perdew, and K. A. Lyssenko, *Science* **355**, 49 (2017).
- W. M. C. Foulkes, L. Mitas, R. J. Needs, and G. Rajagopal, *Rev. Mod. Phys.* **73**, 33 (2001).
- P. J. Reynolds, D. M. Ceperley, B. J. Alder, W. A. Lester, Jr, and A. William, *J. Chem. Phys.* **77**, 5593 (1982).
- N. D. Drummond, B. Monserrat, J. H. Lloyd-Williams, P. L. Ríos, C. J. Pickard, and R. J. Needs, *Nat. Commun.* **6**, 7794 (2015).
- H. Niu, Y. Yang, S. Jensen, M. Holzmann, C. Pierleoni, and D. M. Ceperley, *Phys. Rev. Lett.* **130**, 076102 (2023).
- L. Monacelli, M. Casula, K. Nakano, S. Sorella, and F. Mauri, *Nat. Phys.* **19**, 845 (2023).
- G. Tenti, K. Nakano, A. Tirelli, S. Sorella, and M. Casula, *Phys. Rev. B* **110**, L041107 (2024).
- A. Zen, J. G. Brandenburg, J. Klimeš, A. Tkatchenko, D. Alfé, and A. Michaelides, *Proc. Natl. Acad. Sci. U. S. A.* **115**, 1724 (2018).
- F. Della Pia, A. Zen, D. Alfé, and A. Michaelides, *Phys. Rev. Lett.* **133**, 046401 (2024).
- E. Mostaani, N. D. Drummond, and V. I. Fal'ko, *Phys. Rev. Lett.* **115**, 115501 (2015).
- Y. Nikaido, T. Ichibha, K. Hongo, F. A. Reboredo, K. C. H. Kumar, P. Mahadevan, R. Maezono, and K. Nakano, *J. Phys. Chem. C* **126**, 6000 (2022).
- F. Della Pia, B. X. Shi, Y. S. Al-Hamdani, D. Alfé, T. A. Anderson, M. Barborini, A. Benali, M. Casula, N. D. Drummond, M. Dubecký, C. Filippi, P. R. C. Kent, J. T. Krogel, P. López Ríos, A. Lüchow, Y. Luo, A. Michaelides, L. Mitas, K. Nakano, R. J. Needs, M. C. Per, A. Scemama, J. Schultze, R. Shinde, E. Sliotman, S. Sorella, A. Tkatchenko, M. Towler, C. J. Umrigar, L. K. Wagner, W. A. Wheeler, H. Zhou, and A. Zen, *J. Chem. Phys.* **163**, 104110 (2025).
- M. Casula, C. Filippi, and S. Sorella, *Phys. Rev. Lett.* **95**, 100201 (2005).
- D. F. B. ten Haaf, H. J. M. van Bommel, J. M. J. van Leeuwen, W. Van Saarloos, and D. M. Ceperley, *Phys. Rev. B* **51**, 13039 (1995).
- M. Calandra Buonaura and S. Sorella, *Phys. Rev. B* **57**, 11446 (1998).
- S. Sorella and L. Capriotti, *Phys. Rev. B* **61**, 2599 (2000).

- ²⁰F. Becca and S. Sorella, *Quantum Monte Carlo Approaches for Correlated Systems* (Cambridge University Press, 2017).
- ²¹K. Nakano, R. Maezono, and S. Sorella, *J. Chem. Theory Comput.* **15**, 4044 (2019).
- ²²K. Nakano, R. Maezono, and S. Sorella, *Phys. Rev. B* **101**, 155106 (2020).
- ²³A. Raghav, R. Maezono, K. Hongo, S. Sorella, and K. Nakano, *J. Chem. Theory Comput.* **19**, 2222 (2023).
- ²⁴K. Nakano, S. Sorella, D. Alfè, and A. Zen, *J. Chem. Theory Comput.* **20**, 4591 (2024).
- ²⁵K. Nakano, B. X. Shi, D. Alfè, and A. Zen, *J. Chem. Theory Comput.* **21**, 4426 (2025).
- ²⁶N. Nemec, *Phys. Rev. B* **81**, 035119 (2010).
- ²⁷A. Zen, Y. Luo, S. Sorella, and L. Guidoni, *J. Chem. Theory Comput.* **9**, 4332 (2013).
- ²⁸C. Attaccalite and S. Sorella, *Phys. Rev. Lett.* **100**, 114501 (2008).
- ²⁹J. Van Rhijn, C. Filippi, S. De Palo, and S. Moroni, *J. Chem. Theory Comput.* **18**, 118 (2021).
- ³⁰S. Pathak and L. K. Wagner, *AIP Adv.* **10**, 085213 (2020).
- ³¹N. Trivedi and D. M. Ceperley, *Phys. Rev. B* **41**, 4552 (1990).
- ³²M. Casula, “New QMC approaches for the simulation of electronic systems: a first application to aromatic molecules and transition metal compounds,” Ph.D. thesis, SISSA, 2005.
- ³³jQMC developers, “jQMC,” GitHub, <https://github.com/kousuke-nakano/jQMC> (accessed 16 June 2025).
- ³⁴K. Nakano, C. Attaccalite, M. Barborini, L. Capriotti, M. Casula, E. Coccia, M. Dagrada, C. Genovese, Y. Luo, G. Mazzola, A. Zen, and S. Sorella, *J. Chem. Phys.* **152**, 204121 (2020).
- ³⁵M. C. Bennett, C. A. Melton, A. Annaberdiyev, G. Wang, L. Shulenburger, and L. Mitás, *J. Chem. Phys.* **147**, 224106 (2017).
- ³⁶M. C. Bennett, G. Wang, A. Annaberdiyev, C. A. Melton, L. Shulenburger, and L. Mitás, *J. Chem. Phys.* **149**, 104108 (2018).
- ³⁷Q. Sun, T. C. Berkelbach, N. S. Blunt, G. H. Booth, S. Guo, Z. Li, J. Liu, J. D. McClain, E. R. Sayfutyarova, S. Sharma, S. Wouters, and G. K.-L. Chan, *WIREs Comput. Mol. Sci.* **8**, e1340 (2018).
- ³⁸Q. Sun, X. Zhang, S. Banerjee, P. Bao, M. Barbry, N. S. Blunt, N. A. Bogdanov, G. H. Booth, J. Chen, Z.-H. Cui, J. J. Eriksen, Y. Gao, S. Guo, J. Hermann, M. R. Hermes, K. Koh, P. Koval, S. Lehtola, Z. Li, J. Liu, N. Mardirossian, J. D. McClain, M. Motta, B. Mussard, H. Q. Pham, A. Pulkin, W. Purwanto, P. J. Robinson, E. Ronca, E. R. Sayfutyarova, M. Scheurer, H. F. Schurkus, J. E. T. Smith, C. Sun, S.-N. Sun, S. Upadhyay, L. K. Wagner, X. Wang, A. White, J. D. Whitfield, M. J. Williamson, S. Wouters, J. Yang, J. M. Yu, T. Zhu, T. C. Berkelbach, S. Sharma, A. Y. Sokolov, and G. K.-L. Chan, *J. Chem. Phys.* **153**, 024109 (2020).
- ³⁹E. Posenitskiy, V. G. Chilkuri, A. Ammar, M. Hapka, K. Pernal, R. Shinde, E. J. Landinez Borda, C. Filippi, K. Nakano, O. Kohulák, S. Sorella, P. de Oliveira Castro, W. Jalby, P. L. Ríos, A. Alavi, and A. Scemama, *J. Chem. Phys.* **158**, 174801 (2023).
- ⁴⁰A. Zen, J. G. Brandenburg, A. Michaelides, and D. Alfè, *J. Chem. Phys.* **151**, 134105 (2019).
- ⁴¹M. Casula, *Phys. Rev. B* **74**, 161102 (2006).
- ⁴²J. Kim, A. D. Baczewski, T. D. Beaudet, A. Benali, M. C. Bennett, M. A. Berrill, N. S. Blunt, E. J. L. Borda, M. Casula, D. M. Ceperley, S. Chiesa, B. K. Clark, R. C. Clay, K. T. Delaney, M. Dewing, K. P. Esler, H. Hao, O. Heinonen, P. R. C. Kent, J. T. Krogel, I. Kylänpää, Y. W. Li, M. G. Lopez, Y. Luo, F. D. Malone, R. M. Martin, A. Mathuriya, J. McMinis, C. A. Melton, L. Mitás, M. A. Morales, E. Neuscamman, W. D. Parker, S. D. Pineda Flores, N. A. Romero, B. M. Rubenstein, J. A. R. Shea, H. Shin, L. Shulenburger, A. F. Tillack, J. P. Townsend, N. M. Tubman, B. Van Der Goetz, J. E. Vincent, D. C. Yang, Y. Yang, S. Zhang, and L. Zhao, *J. Phys.: Condens. Matter* **30**, 195901 (2018).
- ⁴³P. R. C. Kent, A. Annaberdiyev, A. Benali, M. C. Bennett, E. J. Landinez Borda, P. Doak, H. Hao, K. D. Jordan, J. T. Krogel, I. Kylänpää, J. Lee, Y. Luo, F. D. Malone, C. A. Melton, L. Mitás, M. A. Morales, E. Neuscamman, F. A. Reboredo, B. Rubenstein, K. Saritas, S. Upadhyay, G. Wang, S. Zhang, and L. Zhao, *J. Chem. Phys.* **152**, 174105 (2020).
- ⁴⁴R. J. Needs, M. D. Towler, N. D. Drummond, P. López Ríos, and J. R. Trail, *J. Chem. Phys.* **152**, 154106 (2020).
- ⁴⁵M. Turisini, G. Amati, and M. Cestari, *JLSRF* **8**, A186 (2024).
- ⁴⁶S. Sorella, *Phys. Rev. Lett.* **80**, 4558 (1998).
- ⁴⁷J. E. Lynn, I. Tews, S. Gandolfi, and A. Lovato, *Annu. Rev. Nucl. Part. Sci.* **69**, 279 (2019).
- ⁴⁸J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, and Q. Zhang (2025). “JAX: Composable transformations of Python+NumPy programs,” GitHub, V.0.6.0. <http://github.com/jax-ml/jax>
- ⁴⁹C. D. Meyer, *Matrix Analysis and Applied Linear Algebra* (SIAM, 2023).
- ⁵⁰R. A. Horn and C. R. Johnson, *Matrix Analysis*, 2nd ed. (Cambridge University Press, 2012).