Random Batch Algorithms for Quantum Monte Carlo simulations

Shi Jin*, Xiantao Li†

Abstract. Random batch algorithms are constructed for quantum Monte Carlo simulations. The main objective is to alleviate the computational cost associated with the calculations of two-body interactions, including the pairwise interactions in the potential energy, and the two-body terms in the Jastrow factor. In the framework of variational Monte Carlo methods, the random batch algorithm is constructed based on the over-damped Langevin dynamics, so that updating the position of each particle in an N-particle system only requires $\mathcal{O}(1)$ operations, thus for each time step the computational cost for N particles is reduced from $\mathcal{O}(N^2)$ to $\mathcal{O}(N)$. For diffusion Monte Carlo methods, the random batch algorithm uses an energy decomposition to avoid the computation of the total energy in the branching step. The effectiveness of the random batch method is demonstrated using a system of liquid ^{4}He atoms interacting with a graphite surface.

Key words: Quantum Monte Carlo Method, Random Batch Methods, Langevin equation.

1 Introduction

One of the fundamental problems in chemistry is the computation of the ground state energy of a many-body quantum system. Although this major difficulty has been circumvented to some extent by the density-functional theory [27], the quantum Monte Carlo (QMC) method [2,3,11,37,42] still remains an important approach to determine the ground state energy and electron correlations.

This paper is concerned with the implementation of the QMC for many-body systems. More specifically, we consider the Hamiltonian,

$$\widehat{H}=\sum_{i=1}^{N}-\frac{\hbar^2}{2m}\triangle_{\boldsymbol{r}_i}+\sum_{i\neq j}W(\boldsymbol{r}_i-\boldsymbol{r}_j)+\sum_{i=1}^{N}V_{ext}(\boldsymbol{r}_i). \tag{1.1}$$

Here we use $\boldsymbol{r}=(\boldsymbol{r}_1,\boldsymbol{r}_2,\cdots,\boldsymbol{r}_N)$ to denote the particle coordinates with N being the total number of particles. and the Laplacian $(-\triangle)$ in the first term of the Hamiltonian

*School of Mathematical Sciences, Institute of Natural Sciences, MOE-LSEC and SHL-MAC, Shanghai Jiao Tong University, Shanghai, China (shijin-m@sjtu.edu.cn)
†Department of Mathematics, Pennsylvania State University, University Park, PA 16802, USA (Xiantao.Li@psu.edu)

indicates the kinetic energy. The second term in the Hamiltonian, which is a double sum, embodies the pairwise interactions, e.g., Coulomb, while the last term includes the external potential, namely,

$$V_{ext}(\boldsymbol{r}_i) = \sum_{\alpha=1}^{M} U(\boldsymbol{r}_i - R_\alpha), \tag{1.2}$$

where R_α, for instance, can be the position of an atom.

In principle, the ground state can be obtained by computing the smallest eigenvalue and the corresponding eigenfunction. It can be expressed in terms of a Rayleigh quotient,

$$E = \min_{\Phi} \frac{\int_{\mathbb{R}^{3N}} \Phi \widehat{H} \Phi d\boldsymbol{r}_1 \cdots d\boldsymbol{r}_N}{\int_{\mathbb{R}^{3N}} |\Phi|^2 d\boldsymbol{r}_1 \cdots d\boldsymbol{r}_N}, \tag{1.3}$$

and the minimizer Φ corresponds to the ground state wave function. However, due to the high dimensionality, a direct numerical approach, e.g., using finite difference or finite element methods together with numerical quadrature for the integrals suffers from the curse of dimensionality, thus is typically prohibitively expensive.

Within the variational Monte Carlo (VMC) framework, this issue is addressed by selecting an appropriate ansatz, denoted here by $\Phi \approx \Psi_0$, for the many-body wave function. Then the multi-dimensional integral is interpreted as a statistical average, which can be sampled using a Monte Carlo procedure. Traditionally, Ψ_0 is constructed using the one-body wave functions, with the effect of particle correlations described by Jastrow factors [11]. Recently, artificial neural networks from machine learning have also been used to represent the many-body wave function [6, 15, 16, 35]. In fact, the recent surge of interest in applying machine-learning algorithms to scientific computing problems has been a strong motivation for the current work.

The first part of this paper is concerned with the numerical implementation of VMC. Since VMC formulates the energy calculation as a sampling problem, the most natural approach is the Metropolis-Hastings (MH) algorithm which, in general, falls into the category of Markov chain Monte Carlo (MCMC) algorithms in statistics. At each step, the chain is updated by calculating the energy change. As can be seen from (1.1) and (1.3), this requires visiting all particles in the system. A direct treatment would involve $\mathcal{O}\big(N(N+M)\big)$ operations in each time step. The presence of the Jastrow factor further complicates the computation. To alleviate the computational cost, we propose a random batch method (RBM), originated from emerging machine learning algorithms [4, 5, 44], and recently introduced to classical interacting particle systems in [21] and extended to various applications in both classical and quantum N-body systems [13, 19, 20, 26, 28–30]. In particular, [21] established an error of RBM to be of $O(\sqrt{\Delta t})$, where Δt is the time step, *uniformly* in N. For the present problem, the objective is to use such an idea to quickly relax the quantum system and sample the energy in the VMC method.

To this end, we first formulate the sampling problem using an over-damped Langevin equation, where the particles are driven by a drift and a stochastic force. The idea of using a Langevin dynamics to construct a VMC algorithm has been pursued in [40].

Rather than computing the particle interactions directly, our proposed RBM algorithm divides the system into random batches and only the interactions within each batch are computed. As a result, on average, updating all N particles only requires $\mathcal{O}(N+M)$ operations. We justify the method by examining the transition density and show that at each step the density induced by the RBM is consistent with the exact transition kernel up to $\mathcal{O}(\Delta t^2)$, the same order as the Euler-Maruyama method.

The other important approach in QMC is the diffusion Monte Carlo (DMC) method [2,37], which starts with the time-dependent Schrödinger equation (TDSE), and evolves the quantum system in an imaginary time scale, leading to a parabolic equation [37],

$$\partial_t \Psi = (E_T - \widehat{H})\Psi. \tag{1.4}$$

The energy shift E_T is adjusted on-the-fly based on the change of the magnitude of the wave function. The key observation is that the dynamics (1.4) can be associated with a stochastic process. In particular, the wave function $|\Psi|^2$ can be interpreted as the empirical measure of a particle system, in which the particles are driven by drift velocity and diffusion. The growth/decay of the wave function is treated by introducing multiple copies of the system, each of which is called a walker or a diffuser [2, 37]. The number of walkers, which reflects the change of the norm of the wave function, is realized by using a birth/death process. The movement of the walkers is driven by the same over-damped Langevin dynamics. Therefore, the RBM is again a natural fit. On the other hand, the probability associated with the birth/death process depends on the total energy. To avoid the computation of the total energy E, especially before the ground state is reached, we propose to decompose the energy into one-, two-, and three-body terms. We construct an RBM where at each step a batch with three particles are selected and we only compute the energy within the batch.

Speeding up QMC simulations has been an important focus in computational chemistry. Various software packages have been developed to this end [24,33,39]. For instance, Kim et al. [24] demonstrated how DMC algorithms can be efficiently implemented on high-performance computer clusters. They showed that when the dynamics of walkers is distributed among the OPENMP threads or MPI units, one can achieve an almost ideal speedup. Toward this end, we implemented the RBM algorithm by moving the walkers in parallel, and we are able to perform QMC simulations of a Helium system with 5016 particles using only 60 cores.

The rest of the paper is organized as follows. We first consider the RBM in the VMC setting in section 2, and justify the method in terms of the transition density. Numerical results are presented for the Helium system. In section 3, we show the RBM in the DMC setting, followed by numerical results. The paper is concluded in section 4.

2 The Random Batch Algorithm for the Variational Monte Carlo Methods

The crucial observation that motivated the VMC framework is that the ground state energy can be viewed as an average with respective to a probability density,

$$E = \left\langle E_{\text{tot}}(\cdot) \right\rangle = \int p(\boldsymbol{r}) E_{\text{tot}}(\boldsymbol{r}) d\boldsymbol{r}, \tag{2.1}$$

where $p(\boldsymbol{r})$ is regarded as a probability density function (PDF),

$$p(\boldsymbol{r}) \propto |\Phi_0(\boldsymbol{r})|^2, \tag{2.2}$$

and the energy E_{tot}, given by,

$$E_{\text{tot}}(\boldsymbol{r}) = \frac{\widehat{H}\Phi_0}{\Phi_0}, \tag{2.3}$$

will be regarded as a random variable.

The ground state wave function is usually sought in a Slater determinant form with a Jastrow factor [11, 18],

$$\Phi_0 = e^{-J(\boldsymbol{r})} \Pi_{i=1}^N S\big(\phi(\boldsymbol{r}_1), \ldots, \phi(\boldsymbol{r}_N)\big), \quad J(\boldsymbol{r}) = \sum_{i<j} u(|\boldsymbol{r}_i - \boldsymbol{r}_j|). \tag{2.4}$$

Here S is the Slater determinant with $\phi(\boldsymbol{r})$ being the single-particle wave function, and we assume a common pairwise form $u(|\boldsymbol{r}_i - \boldsymbol{r}_j|$ for the Jastrow factor J. It is also possible to include three-body terms. For simplicity, we do not consider the spin orbitals.

We will consider Boson systems, which allow us to neglect the sign problem [37] and focus exclusively on the sampling procedure. In addition, to have a class of explicit trial wave functions to work with, we follow the QMC methods for liquid Helium interacting with a graphite surface [34, 43], where the following ansatz has been proven successful,

$$\Phi_0 = e^{-J(\boldsymbol{r})} \Pi_{i=1}^N \phi(\boldsymbol{r}_i), \quad u(r) = \left(\frac{a}{r}\right)^5 + \frac{b^2}{r^2 + c^2}. \tag{2.5}$$

For homogeneous Hellium systems, the ansatz with only the Jastor factor has been widely used in QMC simulations [23, 32]. The ansatz in (2.5) includes orbitals centered around the graphite atoms.

From (2.5), we can write the density (2.2) in an exponential form,

$$p(\boldsymbol{r}) \propto e^{-2V}, \quad V = -\ln \Phi_0 = -\sum_i \log \phi(\boldsymbol{r}_i) + \frac{1}{2} \sum_i \sum_{j \neq i} u(|\boldsymbol{r}_i - \boldsymbol{r}_j|). \tag{2.6}$$

The PDF is reminiscent of a Gibbs distribution with temperature $\beta^{-1} = 1/2$.

The goal of VMC is to create samples according to such a probability density function, from which the ground state energy can be computed from (2.1) by averaging over those

samples. Most VMC methods are of Markov chain Monte Carlo (MCMC) type. Namely, one constructs a Markov chain, which equilibrates to the PDF given by (or close to) (2.6).

Thanks to the explicit ansatz (2.5) for the wave function, the total energy can be explicitly expressed as follows,

$$E_{\text{tot}}(\boldsymbol{r}) = \frac{1}{2}K_i^2 + \sum_{i\neq j} W(\boldsymbol{r}_i - \boldsymbol{r}_j) + \sum_{i=1}^{N} V_{ext}(\boldsymbol{r}_i) \tag{2.7}$$

Since the computational cost is of primary concern here, let us write out all the relevant terms. The first term comes from the kinetic energy,

$$K_i^2 = -\frac{\hbar^2}{m}\frac{\triangle_i \Phi_0}{\Phi_0} = -\frac{\hbar^2}{m}\triangle_i \ln\Phi_0 - \frac{\hbar^2}{m}|\nabla_i \ln\Phi_0|^2. \tag{2.8}$$

The actual form of the kinetic energy depends on the choice of the ansatz for Φ. For instance, with the choice (2.6),the total energy is given by

$$E_{\text{tot}}(\boldsymbol{r}) = -\frac{\hbar^2}{2m}\triangle V - \frac{\hbar^2}{2m}\|\nabla V\|^2 + \sum_{i\neq j} W(\boldsymbol{r}_i - \boldsymbol{r}_j) + \sum_{i=1}^{N}\sum_{\alpha=1}^{M} U(\boldsymbol{r}_i - R_\alpha). \tag{2.9}$$

Since the one-particle wave function is non-negative, we express it as exponential functions,

$$\phi(\boldsymbol{r}_i) = \sum_{\alpha=1}^{M} e^{-\theta(\boldsymbol{r}_i - R_\alpha)}, \tag{2.10}$$

for some function θ. This form has been used in [43] and the parameters were obtained by solving a one-dimensional Schrödinger equation.

In light of (2.9), the calculation of the total energy, which will be part of both the variational and diffusion Monte Carlo algorithms, scales *quadratically* in terms of the number of particles N.

2.1 The classical Metropolis-Hastings Algorithm

A classical algorithm in VMC is the Metropolis-Hastings algorithm. This algorithm is usually implemented by randomly displacing one particle as a time. With the observation that,

$$V = \sum_i V_i, \quad V_i = -\log\phi(\boldsymbol{r}_i) + \sum_{\substack{j=1 \\ j\neq i}}^{N} u(|\boldsymbol{r}_i - \boldsymbol{r}_j|), \tag{2.11}$$

only V_i needs to be computed to determine the energy change due to the change of $\boldsymbol{r}_i$, which subsequently determines the rejections/acceptance of this move. The MH algorithm is standard in computational chemistry for both classical and quantum systems [1], so we keep the discussion brief and summarize the algorithm in **Algorithm** 1. Notice that the only parameters in the algorithm are the size of the trial moves, denoted by Δx, Δy and Δz in each of the three spatial directions, respectively.

It is clear from (2.10) and (2.11) that updating the position of one particle requires $\mathcal{O}(N+M)$ operations. Our goal is to reduce the cost of this computation to $\mathcal{O}(1)$.

Algorithm 1 Metropolis-Hastings (MH) algorithm for variational Monte Caro

```
for nt=1, num_steps do
  for np=1, num_particles do
    Randomly pick an atom i
    e_old = V_i in (2.11);
    r_old = r_i;
    r_i ← r_i + ( (rand() -0.5)*Δx, (rand() -0.5)*Δy, (rand() -0.5)*Δz );
    Compute the energy e_new= V_i and ΔE = e_new - e_old;
    if exp[-2ΔE] > rand() then
      r_i= r_old
    end if
  end for
end for
```

2.2 A random batch algorithm based on the over-damped Langevin Dynamics

The idea behind the random batch algorithm can be best explained in terms of an over-damped Langevin dynamics,

$$d\boldsymbol{r}_i = \nabla \log \phi(\boldsymbol{r}_i)dt - \sum_{\substack{j=1 \\ j\neq i}}^{N} \nabla_{\boldsymbol{r}_i} u(|\boldsymbol{r}_i - \boldsymbol{r}_j|)dt + dW_i(t), \quad 1 \leq i \leq N. \tag{2.12}$$

Here $W_i(t)$'s are independent Wiener processes. Its empirical measure $f(\boldsymbol{r},t)$ corresponds to the Fokker Planck equation (FPE),

$$\partial_t f = -\nabla \cdot (\boldsymbol{v} f) + \frac{1}{2} \triangle f, \tag{2.13}$$

where $\boldsymbol{v} = (\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_N)$ and

$$\boldsymbol{v}_i = \nabla \log \phi(\boldsymbol{r}_i) - \sum_{\substack{j=1 \\ j\neq i}}^{N} \nabla_{\boldsymbol{r}_i} u(|\boldsymbol{r}_i - \boldsymbol{r}_j|), \tag{2.14}$$

is interpreted as a drift velocity. Under suitable conditions [31], the dynamical system with potential given by (2.6) is ergodic, and the PDF $p(\boldsymbol{r})$ in (2.6) is the unique equilibrium measure of this stochastic system. Therefore the numerical integration of the SDEs (2.12) offers a route to navigate to (2.6) and sample the energy.

Using the over-dampled Langevin equation to sample the Gibb distribution has been a widely known method. In the context of VMC, this approach has been adopted by Scemama et al. [40] to improve standard methods. In addition, they combined the Langevin dynamics with the Metropolis-Hastings algorithm to accept/reject the produced samples.

A direct discretization, e.g., the Euler-Maruyama method [25], would involve the following step [25],

$$\boldsymbol{r}_i(t+\Delta t)=\boldsymbol{r}_i(t)+\nabla\log\phi(\boldsymbol{r}_i)\Delta t-\sum_{j\neq i}\nabla_{\boldsymbol{r}_i}u(|\boldsymbol{r}_i(t)-\boldsymbol{r}_j(t)|)\Delta t+\Delta W_i,\quad 1\leq i\leq N. \tag{2.15}$$

Here we assume that the step size Δt is uniform, and the discrete time is given by $\mathcal{T}:=\{n\Delta t, n\geq 0\}$. The method (2.15) is applied to each time step $t\in\mathcal{T}$. At each step, ΔW_i is sampled from a normal random distribution with zero mean and variance Δt.

Although the Euler-Maruyama method is completely different from the Metropolis-Hastings algorithm, they nevertheless have a similar computational cost for updating the position of each particle. More specifically, one has to compute the interactions with all other particles ($u(|\boldsymbol{r}_i(t)-\boldsymbol{r}_j(t)|)$), for all $j\neq i$. In addition, one needs to compute $\log\phi(\boldsymbol{r}_i)$, which is given by,

$$\log\phi(\boldsymbol{r}_i)=\log\sum_{\alpha=1}^{M}e^{-\theta(\boldsymbol{r}_i-R_\alpha)}. \tag{2.16}$$

Together, they contribute to $\mathcal{O}(M+N)$ operations for each particle at each time step.

To reduce the cost of evaluating the two-body interactions, the RBM proceeds as follows (this corresponds to the RBM with replacement in [21]): At each step, one randomly picks out two particles, i and j, and compute their interactions, $\nabla_{\boldsymbol{r}_i}u(|\boldsymbol{r}_i-\boldsymbol{r}_j|)$, then updates their positions as follows,

$$\begin{cases}\boldsymbol{r}_i(t+\Delta t)=\boldsymbol{r}_i(t)+\nabla\log\phi(\boldsymbol{r}_i)\Delta t+(N-1)\nabla_{\boldsymbol{r}_i}u(|\boldsymbol{r}_i-\boldsymbol{r}_j|)\Delta t+\Delta W_i,\\ \boldsymbol{r}_j(t+\Delta t)=\boldsymbol{r}_j(t)+\nabla\log\phi(\boldsymbol{r}_j)\Delta t+(N-1)\nabla_{\boldsymbol{r}_j}u(|\boldsymbol{r}_i-\boldsymbol{r}_j|)\Delta t+\Delta W_j.\end{cases} \tag{2.17}$$

Notice that $\nabla_{\boldsymbol{r}_j}u(|\boldsymbol{r}_i-\boldsymbol{r}_j|)=-\nabla_{\boldsymbol{r}_i}u(|\boldsymbol{r}_i-\boldsymbol{r}_j|)$, thus only one of them needs to be computed. The factor $(N-1)$ accounts for the fact that we are using *one* term $u(|\boldsymbol{r}_i-\boldsymbol{r}_j|)$ to account for the interactions with all $(N-1)$ particles. In general, it is also possible to pick larger random batches. Choosing batches with two particles is most popular.

In light of (2.16), the computation of the one-body term still involves $\mathcal{O}(M)$ operations. However, since

$$\nabla\log\phi(\boldsymbol{r}_i)=\sum_{\alpha=1}^{M}-\nabla\theta(\boldsymbol{r}_i-R_\alpha)q_\alpha^i,\quad q_\alpha^i=\frac{e^{-\theta(\boldsymbol{r}_i-R_\alpha)}}{\sum_{\beta=1}^{M}e^{-\theta(\boldsymbol{r}_i-R_\beta)}}, \tag{2.18}$$

where the coefficients q_α^i's are non-negative and $\sum_\alpha q_\alpha^i=1$, thus the log-gradient term can be viewed as a statistical average with discrete probability given by $\{q_\alpha^i\}_{\alpha=1}^M$. So a simple idea is to pick *just one* term α randomly, e.g., by using a direct Monte Carlo method for one step. The implementation is straightforward: Assume that one starts with α and computes $e_{old}=\theta(\boldsymbol{r}_i-R_\alpha)$, and then we randomly pick $1\leq\beta\leq M$, and compute $e_{new}=\theta(\boldsymbol{r}_i-R_\beta)$. We accept β with probability $\exp\left[e_{new}-e_{old}\right]$. We summarize the random batch algorithm in **Algorithm 2**.

Algorithm 2 Random batch algorithm for variational Monte Carlo

for nt=1, num_steps **do**
  for np=1, num_particles/2 **do**
    Randomly pick two particles i and j with $i \neq j$.
    Perform one step of the Monte Carlo algorithm with respect to $\{q_\alpha^i\}$ and select α.
    Compute $\boldsymbol{b}_i = -\nabla\theta(\boldsymbol{r}_i - R_\alpha)$.
    Perform one step of the Monte Carlo algorithm with respect to $\left\{q_\alpha^j\right\}$ and select β.
    Compute $\boldsymbol{b}_j = -\nabla\theta(\boldsymbol{r}_j - R_\beta)$.
    Evaluate $\boldsymbol{u}_{ij} = -\boldsymbol{u}_{ji} = (N-1)\nabla_{\boldsymbol{r}_i} u(|\boldsymbol{r}_i - \boldsymbol{r}_j|)$.
    Update the particle positions,

$$\begin{aligned} \boldsymbol{r}_i &\longleftarrow \boldsymbol{r}_i + \boldsymbol{b}_i \Delta t + \boldsymbol{u}_{ij}\Delta t + \Delta W_i, \\ \boldsymbol{r}_j &\longleftarrow \boldsymbol{r}_j + \boldsymbol{b}_j \Delta t + \boldsymbol{u}_{ji}\Delta t + \Delta W_j. \end{aligned} \tag{2.19}$$

  end for
end for

As a result of the random sampling of the one- and two-body interactions, updating the position of each particle *only requires* $\mathcal{O}(1)$ *operations*. In the next section, we will study the transition density of the random algorithm, which in turn serves as a validation of the algorithms.

Another practical issue emerges when the interaction $u(|\boldsymbol{r}|)$ has a singularity near zero. In this case, a direct implementation of the random batch algorithm would often require much smaller step sizes in the integration of the Langevin dynamics (2.12) [30]. The issue can be mitigated by separating $u(|\boldsymbol{r}|)$ into a singular, but short-ranged part, and a long-ranged, but smooth part [30]. The short-range interactions can be efficiently computed using Verlet's cell list method which, for each particle, still involves $\mathcal{O}(1)$ operations. This is a common practice in classical molecular simulations [1, 12]. Meanwhile, the long-range part, which is where most computations are involved, can be simulated by the random batch algorithm. Here we use a simple approach to separate out the singularity by introducing a cut-off distance r_{cut}, then replacing the short-range part by an extrapolation using a Taylor expansion, namely,

$$u_L(r) = \begin{cases} u(r) & r > r_{\text{cut}}, \\ u(r_{\text{cut}}) + u'(r_{\text{cut}})(r - r_{\text{cut}}) + \frac{1}{2}u''(r_{\text{cut}})(r - r_{\text{cut}})^2, & \text{Otherwise.} \end{cases} \tag{2.20}$$

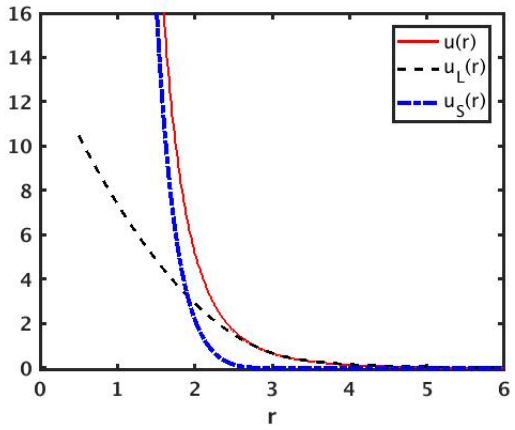

Figure 1: Separation of the interaction $u(r)=r^{-5}$ with singularity at $r=0$ (solid line) into a long range interaction $u_L(r)$ (dashed) without singularity, and a short range interaction $u_S(r)$ (dot-dashed).

The short-range part is then defined as $u_S(r)=u(r)-u_L(r)$. Figure 1 shows an example of how such a decomposition can be easily constructed.

2.3 The transition kernel of the random batch algorithm

2.3.1 The random batch algorithm for the one-body term

We will first consider the Monte-Carlo sampling of the one-body term (2.18), and for clarity we place the problem in the setting of solving a d-dimensional SDE system,

$$d\boldsymbol{r}(t)=\boldsymbol{a}(\boldsymbol{r}(t))dt+\sigma dW_t. \tag{2.21}$$

Here $\sigma\geq 0$ is a constant, which is also allowed to be zero. In light of (2.18), we consider a vector field $\boldsymbol{a}$ that can be expressed as,

$$\boldsymbol{a}(\boldsymbol{r})=\sum_{\alpha=1}^{M} q_\alpha \boldsymbol{a}_\alpha(\boldsymbol{r}), \tag{2.22}$$

where the coefficients q_α's represent a discrete probability density, that is, $q_\alpha\geq 0$ and $\sum_\alpha q_\alpha=1$. We examine the random algorithm,

$$\boldsymbol{r}(t+\Delta t)=\boldsymbol{r}(t)+\boldsymbol{a}_\alpha\big(\boldsymbol{r}(t)\big)\Delta t+\sigma\Delta W, \tag{2.23}$$

where the index α is selected at random according to the discrete density. We consider uniform step size Δt, and the equation will be applied to each step t.

Clearly, the corresponding transition density is given by,

$$p\big(\boldsymbol{r}(t+\Delta t)=\boldsymbol{y}|\boldsymbol{r}(t)=\boldsymbol{x}\big)=\sum_{\alpha=1}^{M} q_\alpha \frac{1}{(2\pi\sigma^2)^{d/2}}\exp\left[-\frac{\big(\boldsymbol{y}-\boldsymbol{x}-\boldsymbol{a}_\alpha(\boldsymbol{x})\Delta t\big)^2}{2\sigma^2}\right]. \tag{2.24}$$

For any function $A(\boldsymbol{r}) \in C^4(\mathbb{R}^d)$ with suitable growth conditions [25], one has,

$$\begin{aligned}
&\int_{\mathbb{R}^d} A(\boldsymbol{y}) p\big(\boldsymbol{x}(t+\Delta t)=\boldsymbol{y}|\boldsymbol{x}(t)=\boldsymbol{x}\big) d\boldsymbol{y} \\
&= \sum_{\alpha=1}^{M} q_\alpha \big[A(\boldsymbol{x}) + \boldsymbol{a}_\alpha(\boldsymbol{x})\cdot\nabla A(\boldsymbol{x})\Delta t + \frac{1}{2}\triangle A(x)\Delta t + \mathcal{O}(\Delta t^2)\big] \\
=& A(\boldsymbol{x}) + \boldsymbol{a}(\boldsymbol{x})\cdot\nabla A(\boldsymbol{x})\Delta t + \frac{1}{2}\triangle A(x)\Delta t + \mathcal{O}(\Delta t^2).
\end{aligned} \tag{2.25}$$

Therefore, this random algorithm has a first weak-order of accuracy, which is comparable to the Euler-Maruyama method. Even though the drift term $\boldsymbol{a}(\boldsymbol{r})$ is only sampled once at each step, the method is still convergent. To our knownledge, this surprising property was first noticed by E et al. in the context of multiscale methods for SDEs [9], where the weak convergence is proved in a more general (multiscale) setting.

2.3.2 The random batch algorithm for pair-wise interactions

We now turn to the SDE system (2.12) with pair-wise interactions,

$$d\boldsymbol{r}_i(t) = \nabla \log\phi(\boldsymbol{r}_i)dt - \sum_{j\neq i}\nabla u(|\boldsymbol{r}_i-\boldsymbol{r}_j|)dt + dW_t. \tag{2.26}$$

By letting $\boldsymbol{u}_{ij} = \nabla u(|\boldsymbol{r}_i-\boldsymbol{r}_j|)$, we can write the pair-wise terms as,

$$\boldsymbol{u}_i = \sum_{j\neq i}\boldsymbol{u}_{ij},\ \boldsymbol{u}_{ij} = -\boldsymbol{u}_{ji}. \tag{2.27}$$

To study the weak convergence, one may consider the conditional expectation,

$$\mathbb{E}\Big[A(\boldsymbol{r}(t+\Delta t)|\boldsymbol{r}(t)=\boldsymbol{x}\Big]. \tag{2.28}$$

This is represented by the transition density as follows,

$$\mathbb{E}\big[A(\boldsymbol{r}(t+\Delta t)|\boldsymbol{r}(t)=\boldsymbol{x}\big] = \int A(\boldsymbol{y}) p\big(\boldsymbol{r}(t+\Delta t)=\boldsymbol{y}|\boldsymbol{r}(t)=\boldsymbol{x}\big) d\boldsymbol{y}. \tag{2.29}$$

The transition density for the SDEs (2.26) follows the Fokker-Planck equation [25]. The explicit form of the solution is often unknown. But with the approximation by the Euler-Maruyama method,

$$\boldsymbol{r}_i(t+\Delta t) = \boldsymbol{r}_i(t) + \nabla\log\phi(\boldsymbol{r}_i)\Delta t + \boldsymbol{u}_i\Delta t + \Delta W_i, \tag{2.30}$$

we can identify an approximate transition kernel,

$$\begin{aligned}
&p^{EM}(\boldsymbol{r}(t+\Delta t)=\boldsymbol{y}|\boldsymbol{r}(t)=\boldsymbol{x}) \\
&= \frac{1}{(2\pi\sigma^2\Delta t)^{d/2}} \exp\Big[-\big(\boldsymbol{y}-\boldsymbol{x}-\nabla\log\phi(\boldsymbol{x})\Delta t - \boldsymbol{u}(\boldsymbol{x})\Delta t\big)^2/(2\sigma^2\Delta t)\Big].
\end{aligned} \tag{2.31}$$

By the weak Itô-Taylor expansion [25], we have from the density induced by the Euler-Maruyama method,

$$\mathbb{E}\big[A(X(t+\Delta t)|X(t)=x\big]=A(x)+\mathcal{L}A(x)\Delta t+\mathcal{O}(\Delta t^2), \tag{2.32}$$

where $\mathcal{L}$ is the generator,

$$\mathcal{L}A(\boldsymbol{x})=\sum_i\big(\nabla\log\phi(\boldsymbol{x}_i)+\boldsymbol{u}_i\big)\cdot\nabla_{x_i}A(\boldsymbol{x})+\frac{1}{2}\triangle A(\boldsymbol{x}). \tag{2.33}$$

The expansion (2.32) is consistent with that of the exact transition density up to $\mathcal{O}(\Delta t^2)$, making the Euler-Maruyama method first order in the weak sense [25].

We now turn to the random batch algorithm 2.17 with replacement [21]. The convergence property has recently been proved in [20]:

Theorem 2.1. The random batch algorithm over $N/2$ steps has weak order 1.

Here we illustrate the weak convergence in terms of the transition density. This also helps us to construct RBM for diffusion Monte Carlo. Since we randomly pick a pair of components to update, the transition density, denoted here by p^{RB}, is given by,

$$p^{RB}\big(\boldsymbol{r}(t+\Delta t)=\boldsymbol{y}|\boldsymbol{r}=\boldsymbol{x}\big)=\frac{2}{(N-1)N}\sum_{i>j}q_{ij}(\boldsymbol{y}|\boldsymbol{x}), \tag{2.34}$$

where,

$$\begin{aligned}q_{ij}(\boldsymbol{y}|\boldsymbol{x})=&\frac{1}{(2\pi\Delta t)^{3N/2}}\exp\Big[-\big(\boldsymbol{y}_i-\boldsymbol{x}_i-\nabla\log\phi(\boldsymbol{x}_i)\Delta t-(N-1)\boldsymbol{u}_{ij})\Delta t\big)^2/(2\Delta t)\Big]\\&\times\exp\Big[-\big(\boldsymbol{y}_j-\boldsymbol{x}_j-\nabla\log\phi(\boldsymbol{x}_j)\Delta t-(N-1)\boldsymbol{u}_{ji}\Delta t\big)^2/(2\Delta t)\Big]\\&\times\Pi_{k\neq i,j}\delta(\boldsymbol{y}_k-\boldsymbol{x}_k).\end{aligned} \tag{2.35}$$

The delta functions were included to ensure that when the pair (i,j) is selected, other components are not updated. In the following discussions, we will simply write the transition density as $p^{RB}(\boldsymbol{y}|\boldsymbol{x})$.

With direct Taylor expansions, one finds that, for any observable $A(\boldsymbol{x})$,

$$\begin{aligned}\int A(\boldsymbol{y})q_{ij}(\boldsymbol{y}|\boldsymbol{x})d\boldsymbol{y}=&A(\boldsymbol{x})+\nabla\log\phi(\boldsymbol{x}_i)\cdot\nabla_{\boldsymbol{x}_i}A(\boldsymbol{x})\Delta t+\nabla\log\phi(\boldsymbol{x}_j)\cdot\nabla_{\boldsymbol{x}_j}A(\boldsymbol{x})\Delta t\\&+(N-1)\boldsymbol{u}_{ij}\cdot\nabla_{\boldsymbol{x}_i}A(\boldsymbol{x})\Delta t+(N-1)\boldsymbol{u}_{ji}\cdot\nabla_{\boldsymbol{x}_j}A(x)\Delta t\\&+\frac{1}{2}\triangle_{\boldsymbol{x}_i}A(\boldsymbol{x})\Delta t+\frac{1}{2}\triangle_{\boldsymbol{x}_j}A(\boldsymbol{x})\Delta t+\mathcal{O}(\Delta t^2).\end{aligned} \tag{2.36}$$

Combining this with (2.34), we have,

$$\begin{aligned}\int A(\boldsymbol{y})p^{RB}(\boldsymbol{y}|\boldsymbol{x})d\boldsymbol{y}=&A(\boldsymbol{x})+\frac{2\Delta t}{N}\Big\{\sum_i\nabla\log\phi(\boldsymbol{x}_i)\cdot\nabla_{\boldsymbol{x}_i}A(\boldsymbol{x})\\&+\sum_i\sum_{j\neq i}\boldsymbol{u}_{ij}\cdot\nabla_{\boldsymbol{x}_i}A(\boldsymbol{x})+\frac{1}{2}\triangle A(\boldsymbol{x})\Big\}+\mathcal{O}(\Delta t^2).\end{aligned} \tag{2.37}$$

Therefore, the random batch algorithm with replacement, when applied to one batch of two particles, has the same accuracy as the Euler-Maruyama method over a time step of $2\Delta t/N$. Note one full time step in Euler-Maruyama method corresponds to $N/2$ such steps in the RBM with replacement.

2.4 Numerical Results

We conduct numerical experiments with ^{4}He atoms interacting with a two-dimensional lattice. The ^{4}He atoms, due to the fact that the total spin is zero, are bosons. Driven by its superfluid properties and many observed quantum effects, ^{4}He atoms have been extensively studied by computer simulations. Acting as a substrate, the lattice has a triangular structure with lattice spacing given by $a_0 = 4.2576$ Å. Such a lattice can be generated using rectangular unit cells, each of which contains two atoms. For example, Figure 2 shows such a system with 12×7 unit cells and a total of 168 atoms. The model is adapted from [22]. We choose Å as the length unit and k_BKelvin as the unit of energy.

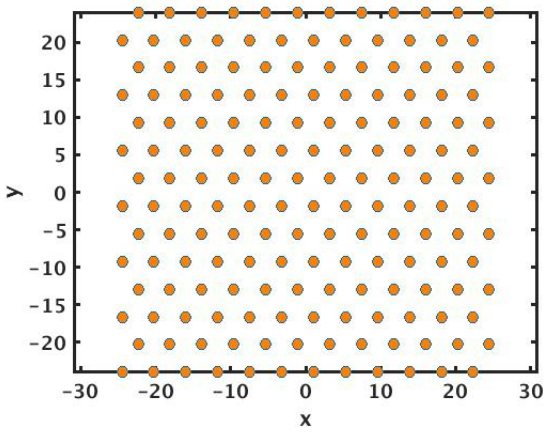

Figure 2: A two dimensional lattice with Helium atoms.

Particles that represent the wave function Φ_0 are created randomly near the nuclei. We follow the setup in [34]. In particular, in the wave function ansatz (2.5), the one-particle wave function is assumed to be,

$$\phi(\boldsymbol{r}_i) = \exp-\left((z_i - z_e)^2/z_0^2\right) \sum_{\alpha=1}^{M} \exp\left(-(\boldsymbol{r}_i - R_\alpha)^2/r_0^2\right). \tag{2.38}$$

Here z_i indicates the third component of the coordinate $\boldsymbol{r}_i$. In addition, the two-body terms in the Jastrow factor are chosen to consist of both short and long range terms,

$$u(r) = \left(\frac{a}{r}\right)^5 + \frac{b^2}{c^2 + r^2}. \tag{2.39}$$

Although the first term decays rather quickly, we do not use an abrupt truncation of the function. Instead, we follow the construction (2.20), and split it into a function that

vanishes beyond a cut-off distance r_{cut}. The remaining part is merged into the second term in (2.39) and regarded as a long-range interaction. The parameters, with unit Å, are given in Table 1.

Table 1: Model parameters in the QMC simulations of ^{4}He.

z_e	z_0	r_0	a	b	c	r_{cut}
2.85	0.521	15	2.771	5.0	10.0	8.0

We first carry out VMC simulations using RBM-VMC (**Algorithm** 2) and the Euler-Maruyama method (2.15). In the simulations, we run the algorithms with 300 ensembles and the average energy at each step will be computed as an average over these ensembles. In principle, the algorithms can be implemented with just one realization, and the ground state energy would be computed entirely from the time series. But multiple ensembles can be easily implemented in parallel. In addition, the ensembles can later be turned into walkers in the DMC simulations.

Figure 3 shows the average energy computed from the RBM-VMC and the Euler-Maruyama methods in the time interval [0,150]. The step size is $\Delta t = 10^{-3}$. We observe that both methods relax to equilibrium around $t = 25$. Since the time scale is fictitious, we do not assign a unit for the time variable.

We also show the time correlation of the sampled energy after the system has reached equilibrium. To obtain a more quantitative comparison, we implemented an MCMC diagnostics. In this context, the relaxation is known as the burn-in period, and a thinning parameter can be used to indicate correlations. More specifically, we use the Raftery and Lewis criteria [36] ($q = 0.025, r = 0.0125, s = 0.95$) and find that the burn-in period is 23.49 and 38.54, with thinning parameters 0.058 and 0.066, for the Euler-Maruyama and RBM, respectively. One can see that the random batch method has slightly longer burn-in time, and longer correlation. Since both of these methods are constructed by integrating SDEs in time, we have factored in the step size Δt in estimating these parameters. We also show the energy sampled from the Metropolis-Hastings algorithm in Figure 4. The average energy is 2.361113×10^4 with standard statistical error 1.698. Note that it is not straightforward to compare the previous two algorithms to the Metropolis-Hastings algorithm, since the latter method does not have an associated time scale.

We now compare the CPU time that is needed to move the 300 Markov chains for 1000 steps. In this comparison, we have excluded the cost associated with the energy calculations in the random batch and Euler-Maruyama methods, since they are not needed in the burn-in period, and even upon equilibrium, it is a good practice to sample it every few steps to obtain less correlated samples. From Table 2, one clearly sees that the RBM is more efficient than the Euler-Maruyama method, mainly due to the random sampling of the pairwise interactions in the Jastrow factor in the wave function (2.5). It is much more efficient than the Metropolis-Hastings algorithm, mainly because the latter method requires the calculation of the energy at *every* step.

Finally, we examine the effect of the time discretization. Unlike the metropolis-Hastings algorithm, the RBM and Euler-Maruyama methods are biased, and the results

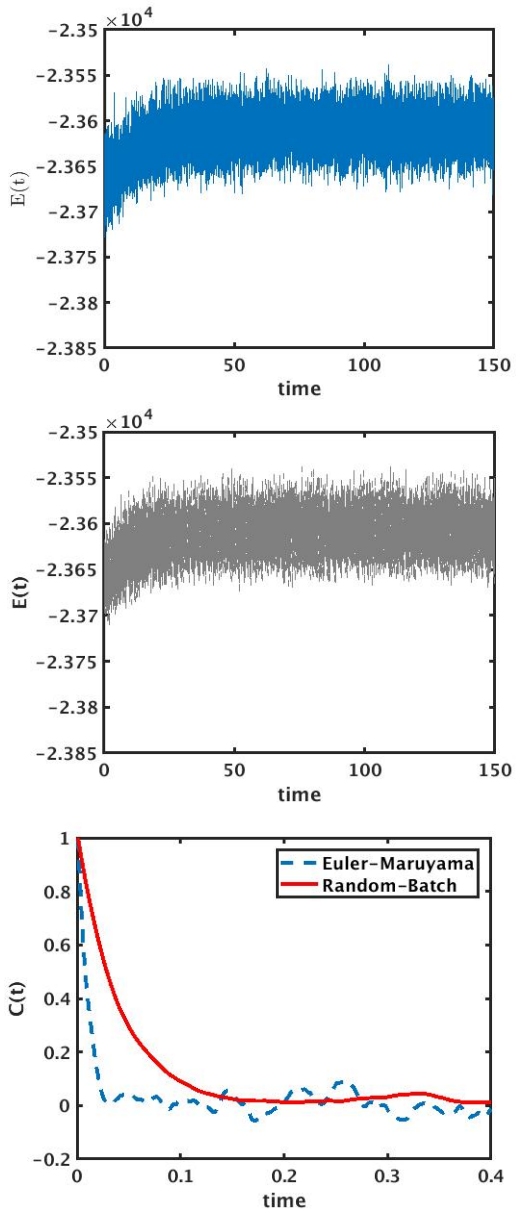

Figure 3: A comparison of the random batch Algorithm 2 (top) to the Euler-Maruyama method (middle). The bottom panel shows the time correlation.

Table 2: Comparison of the CPU time (measured in seconds) for several VMC methods.

	Metropolis-Hastings	Euler-Maruyama	Random Batch
CPU time for a 1000-step sampling period	1503	469	54

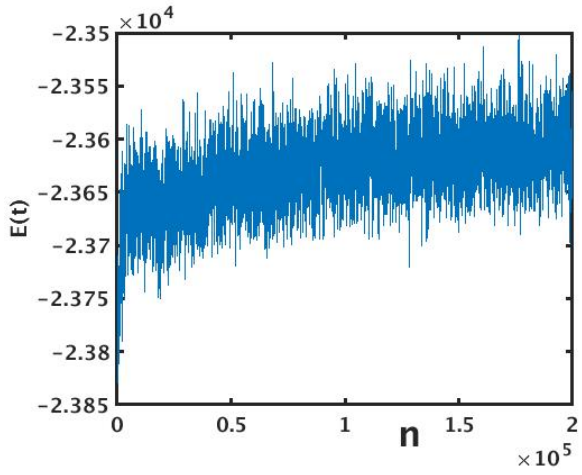

Figure 4: The energy sampled from 200,000 steps of the Metropolis-Hastings algorithm.

depend on the step size. Figure 5 shows the averages computed from the two methods for different choices of Δt. We choose 10^5 samples from equilibrium in the estimation. Compared to the values from the MH algorithm, it can be observed that the Euler-Maruyama method over-estimates the ground state energy, while the random batch method under-estimates it.

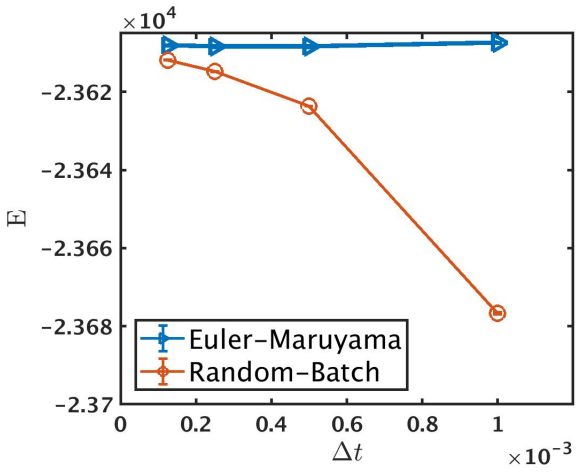

Figure 5: The average of the energy computed from the random batch and Euler-Maruyama methods for various choices of the step size Δt.

3 The Random Batch Algorithm in Diffusion Quantum Monte Carlo Methods

The accuracy of the VMC method is limited by the ansatz of the wave function (2.5). The idea of the DMC is to go back to the time-dependent Schrödinger equation and evolve the system along the imaginary time,

$$\partial_t \Psi = (E_T - \widehat{H})\Psi. \tag{3.1}$$

Here a rescaling of time scale $it/\hbar \to t$ has been introduced and t now represents a fictitious time scale. Since the transient is not of interest here, we will not keep track of the time scales.

Depending on the choice of the reference energy E_T, the solution would either decay or grow exponentially, unless E_T coincides with the ground state energy, at which point, the wave function converges to the ground state as $t \to +\infty$.

Instead of solving (3.1) directly, it is often more practical to find $f(\boldsymbol{r},t)$ with

$$f(\boldsymbol{r},t) = \Psi(\boldsymbol{r},t)\Phi_0(\boldsymbol{r}). \tag{3.2}$$

This ansatz has the flavor of the importance sampling. In addition, if one chooses $\Psi(\boldsymbol{r},0) = \Phi_0(\boldsymbol{r})$, then $f(\boldsymbol{r},0) = |\Phi_0|^2 \propto p(\boldsymbol{r})$ in (2.6). Therefore, we can use a VMC method to initialize $f(\boldsymbol{r},t)$.

Direct calculations yield the following differential equation [37],

$$\partial_t f = -\nabla \cdot \Big(\frac{\hbar^2}{m}\boldsymbol{v}(\boldsymbol{r})f\Big) + \frac{\hbar^2}{2m}\nabla^2 f - \big(E_T - E_{\text{tot}}(\boldsymbol{r})\big)f. \tag{3.3}$$

The average energy $E(t)$ is defined as a weighted average,

$$E(t) = \frac{\int f(\boldsymbol{r},t)E_{\text{tot}}(\boldsymbol{r})d\boldsymbol{r}}{\int f(\boldsymbol{r},t)d\boldsymbol{r}}. \tag{3.4}$$

Without the last term on the right hand side of (3.3), the equation above, with a time rescaling $\tau \to \tau\hbar^2/m$, would be reduced to the Fokker-Planck equation (2.13) associated with the SDE (2.12), with the additional term that embodies the influence of the choice of the energy shift on the change of total mass.

Within a short time step, Δt, the solution of (3.3) can be approximated by [37],

$$f(\boldsymbol{r},t+\Delta t) = \int_{\mathbb{R}^{3N}} G(\boldsymbol{r},\boldsymbol{r}',\Delta t) f(\boldsymbol{r}',t)d\boldsymbol{r}', \tag{3.5}$$

where the function G, often referred to as Green's function, is given by [37],

$$G(\boldsymbol{r},\boldsymbol{r}',\Delta t) = \frac{1}{\big(2\pi\sigma^2\big)^{3N/2}} \exp\left[-\frac{\big(\boldsymbol{r}' - \boldsymbol{r} - \frac{\Delta t\hbar}{m}\boldsymbol{v}(\boldsymbol{r})\big)^2}{2\sigma^2}\right] \exp\big[\Delta t\big(E_T - E_{\text{tot}}(\boldsymbol{r})\big)\big]. \tag{3.6}$$

The parameter $\sigma = \sqrt{\Delta t}\hbar/\sqrt{m}$ and the vector field v is given by (2.14).

This Green's function can be interpreted as a transition kernel in a general sense. In terms of an observable A, the action of the Green's function is expressed as follows,

$$\int A(\boldsymbol{r}')G(\boldsymbol{r}',\boldsymbol{r},\Delta t)d\boldsymbol{r}' = A(\boldsymbol{r}) + \frac{\hbar^2}{m}\boldsymbol{v}(\boldsymbol{r})\cdot\nabla A(\boldsymbol{r})\Delta t + \frac{\hbar^2}{m}\frac{\Delta t}{2}\triangle A(\boldsymbol{r}) + \Delta t\big(E_T - E_{\text{tot}}(\boldsymbol{r})\big)A(\boldsymbol{r}) + \mathcal{O}(\Delta t^2). \tag{3.7}$$

One can write $G(\boldsymbol{r}',\boldsymbol{r},\Delta t) = G_1(\boldsymbol{r}',\boldsymbol{r},\Delta t)G_2(\boldsymbol{r}',\boldsymbol{r},\Delta t)$, with

$$\begin{aligned} G_1(\boldsymbol{r}',\boldsymbol{r},\Delta t) &= \frac{1}{\big(2\pi\sigma^2\big)^{3N/2}}\exp\left[-\frac{\big(\boldsymbol{r}' - \boldsymbol{r} - \frac{h\hbar\Delta t}{m}\boldsymbol{v}(\boldsymbol{r})\big)^2}{2\sigma^2}\right], \\ G_2(\boldsymbol{r}',\boldsymbol{r},\Delta t) &= \exp\big[\Delta t\big(E_T - E_{\text{tot}}(\boldsymbol{r})\big)\big]. \end{aligned} \tag{3.8}$$

Computationally, the two operations are carried out in two steps, which can be viewed as an operator-splitting method. Better results are often obtained with a symmetric splitting, which corresponds to redefining,

$$G_2(\boldsymbol{r}',\boldsymbol{r},\Delta t) = \exp\big[\Delta t\big(E_T - \tfrac{1}{2}(E_{\text{tot}}(\boldsymbol{r}) + E_{\text{tot}}(\boldsymbol{r}'))\big)\big]. \tag{3.9}$$

A typical DMC algorithm begins with an ensemble of L copies of the system, also known as walkers [2]. For each realization, one first solves the SDEs,

$$d\boldsymbol{r}_i(t) = \frac{\hbar^2}{m}\nabla\log\phi(\boldsymbol{r}_i)dt + \frac{\hbar^2}{m}\sum_{j\neq i}\boldsymbol{v}_{ij}dt + \sigma dW_i(t). \tag{3.10}$$

This step corresponds to the action of the first Green's function G_1. Specifically, $\boldsymbol{r}$ and $\boldsymbol{r}'$ in G_1 refer to, respectively, the positions of the particles before and after these SDEs are solved for one time step. As alluded to at the beginning of this section, these SDEs coincide with the over-damped Langevin equations (2.12) after a simple rescaling of the time variable.

One can think of the approximations by these SDEs as an approximation of the function $f(\boldsymbol{r},t)$ using a sum of delta functions,

$$f(\boldsymbol{r},t) \approx \frac{1}{L}\sum_{\ell=1}^{L}\delta(\boldsymbol{r} - \boldsymbol{r}^{(\ell)}(t)). \tag{3.11}$$

The Green's function G_1 is precisely the transition kernel. In particular, the number of walkers will not be changed by this step.

After the particles at the step $t+\Delta t$ are updated by G_1, the Green's function G_2 in (3.9) needs to be incorporated. This is done by using a birth/death process to determine whether a realization should be removed or duplicated. For each walker, one computes a weight factor,

$$w(t+\Delta t) = \exp\big[\Delta t\big(E_T - \tfrac{1}{2}(E_{\text{tot}}(\boldsymbol{r}) + E_{\text{tot}}(\boldsymbol{r}'))\big)\big], \tag{3.12}$$

which corresponds to the Green's function G_2 in (3.9). To apply Green's function G_2, the walkers are duplicated (removed) based on the magnitude of $w(t+\Delta t)$. The overall algorithm is summarized on **Algorithm** 3, which will be later referred to as the direct DMC method.

Algorithm 3 Diffusion Monte Carlo (Direct DMC)

Sample the initial num_walkers walkers using the VMC algorithm. Set $M(1)$ as the number of walkers initially. Set E_T to be the average energy computed from the VMC.

for nt=1, num_steps **do**
  for n=1, num_walkers **do**
    Compute the energy $E_{\text{tot}}(\boldsymbol{r})$.
    Drift and diffuse the nth walker according to (3.10).
    Compute the energy $E_{\text{tot}}(\boldsymbol{r}')$.
    Determine the probability of the branching process:

$$w_n = \exp\left[\Delta t\left(E_T - \left(E_{\text{tot}}(\boldsymbol{r}) + E_{\text{tot}}(\boldsymbol{r}')\right)/2\right)\right].$$

  end for
  for n=1, num_walkers **do**
    if $w_n < 1$ **then**
      The walker survives with probability w_n.
    else
      The walker is duplicated $\lfloor w_n \rfloor$ times. A new walker is created with probability $w_n - \lfloor w_n \rfloor$.
    end if
  end for
  Recount the number of walkers num_walkers, and set it to $M(\text{nt}+1)$.

  Adjust the energy shift: $E_T \leftarrow E_T + \kappa \ln \frac{M(\text{nt}+1)}{M(\text{nt})}$.
end for

3.1 The random batch algorithm for DMC

Since the initialization, as well as the drift-diffusion step of the DMC involves the solution of the over-damped Langevin dynamics (2.12) (or (3.10)), our random batch algorithm for VMC can be directly applied to this part of the DMC method, to mitigate the same issue encountered in the Metropolis-Hastings algorithm.

It remains to treat the transition kernel $G_2(\boldsymbol{r}',\boldsymbol{r},\Delta t)$ (3.9). The primary challenge is that computing the energy at each step requires $\mathcal{O}((N+M)N)$ operations in order to update the position of N particles. To reduce this part of the computation, we propose to write the total energy (2.9) as follows,

$$E_{\text{tot}}(\boldsymbol{r}) = \sum_{i=1}^{N} E_1(\boldsymbol{r}_i) + \sum_{1\le i<j\le N} E_2(\boldsymbol{r}_i,\boldsymbol{r}_j) + \sum_{1\le i<j<k\le N} E_3(\boldsymbol{r}_i,\boldsymbol{r}_j,\boldsymbol{r}_k). \tag{3.13}$$

These three terms are onsite, two-body, and three-body contributions. The on-site energy comes from the one-particle wave function and the external potential,

$$E_1(\boldsymbol{r}_i) = -\frac{\hbar^2}{2m}\nabla^2 \ln\phi(\boldsymbol{r}_i) - \frac{\hbar^2}{2m}|\nabla \ln\phi(\boldsymbol{r}_i)|^2 + \sum_{\alpha=1}^{M} U(\boldsymbol{r}_i - R_\alpha). \tag{3.14}$$

To ensure that this part of the energy is evaluated with $\mathcal{O}(1)$ operations, we pick one atom α in the external potential randomly in the last term, and compute,

$$E_1(\boldsymbol{r}_i) = -\frac{\hbar^2}{2m}\nabla^2 \ln\phi(\boldsymbol{r}_i) - \frac{\hbar^2}{2m}|\nabla \ln\phi(\boldsymbol{r}_i)|^2 + MU(\boldsymbol{r}_i - R_\alpha). \tag{3.15}$$

Let $\boldsymbol{r}_{ij} = \boldsymbol{r}_i - \boldsymbol{r}_j$ be the relative position and $r_{ij} = |\boldsymbol{r}_{ij}|$ be its distance. The two-body term consists of the following terms,

$$E_2(\boldsymbol{r}_i,\boldsymbol{r}_j) = -\frac{\hbar^2}{m}\nabla^2 \ln u(r_{ij}) + \frac{\hbar^2}{m}\left(\nabla \ln\phi(\boldsymbol{r}_i) - \nabla \ln\phi(\boldsymbol{r}_j)\right)\cdot \nabla u(r_{ij}) + \frac{\hbar^2}{m}|\nabla u(r_{ij})|^2 + W(r_{ij}). \tag{3.16}$$

The three-body term can be derived from the first term in the kinetic energy (2.8), and it is given by,

$$E_3(\boldsymbol{r}_i,\boldsymbol{r}_j,\boldsymbol{r}_k) = \frac{\hbar^2}{m}\Big[\nabla u(r_{ij})\cdot\nabla u(r_{ik}) + \nabla u(r_{ji})\cdot\nabla u(r_{jk}) + \nabla u(r_{ki})\cdot\nabla u(r_{kj})\Big]. \tag{3.17}$$

These three-body terms arise due to the $\|\nabla V\|^2$ term in (2.7).

This partition of the energy is structured in the same manner as in molecular dynamics models [1]. In the random batch algorithm, we randomly pick a batch C_I with three particles: $C_I = \{i,j,k\}$. We first update the position of the three particles (drift and diffuse) by solving the over-damped Langevin dynamics (3.10) using the random batch algorithm with batch size 3. This is demonstrated in (3.22) in **Algorithm 4**. We then

define a *local* energy,

$$\begin{aligned} E_I(\boldsymbol{r}_i,\boldsymbol{r}_j,\boldsymbol{r}_k) =& E_1(\boldsymbol{r}_i)+E_1(\boldsymbol{r}_j)+E_1(\boldsymbol{r}_k) \\ &+\tfrac{N-1}{2}\Big[E_2(\boldsymbol{r}_i,\boldsymbol{r}_j)+E_2(\boldsymbol{r}_j,\boldsymbol{r}_k)+E_2(\boldsymbol{r}_k,\boldsymbol{r}_i)\Big], \\ &+\tfrac{(N-1)(N-2)}{2}E_3(\boldsymbol{r}_i,\boldsymbol{r}_j,\boldsymbol{r}_k). \end{aligned} \tag{3.18}$$

In light of (3.15), (3.16), and (3.17), the cost for evaluating this local energy (3.18) remains $\mathcal{O}(1)$.

In the branching step of our new DMC method, we assign a batch with a weight,

$$w_I = \exp\left[\Delta t\left(\tfrac{3}{N}E_T - E_I(\boldsymbol{r}_i,\boldsymbol{r}_j,\boldsymbol{r}_k)\right)\right], \tag{3.19}$$

which helps to determine whether a walker should be continued/duplicated/deleted. This amounts to an approximation of Green's function G_2. To see this, note, on average, the effect of this random procedure on $f(\boldsymbol{r},t)$ is given by,

$$\begin{aligned} &\frac{6}{N(N-1)(N-2)}\sum_{i<j<k} w_I(\boldsymbol{r}_i,\boldsymbol{r}_j,\boldsymbol{r}_k)f(\boldsymbol{r},t) \\ &\quad= \frac{6}{N(N-1)(N-2)}\sum_{i<j<k}\Big[1+E_1(\boldsymbol{r}_i)\Delta t+E_1(\boldsymbol{r}_j)\Delta t+E_1(\boldsymbol{r}_k)\Delta t\Big]f(\boldsymbol{r},t) \\ &\qquad+\frac{3}{N(N-2)}\sum_{i<j<k}\big(E_2(\boldsymbol{r}_i,\boldsymbol{r}_j)+E_2(\boldsymbol{r}_j,\boldsymbol{r}_k)+E_2(\boldsymbol{r}_k,\boldsymbol{r}_i)\big)\Delta t f(\boldsymbol{r},t) \\ &\qquad+\frac{3}{N}\sum_{i<j<k}E_3(\boldsymbol{r}_i,\boldsymbol{r}_j,\boldsymbol{r}_k)\Delta t f(\boldsymbol{r},t)+\mathcal{O}(\Delta t^2), \\ &\quad= f(\boldsymbol{r},t)+(E_T-E_{\text{tot}}(\boldsymbol{r}))\frac{3\Delta t}{N}f+\mathcal{O}(\Delta t^2). \end{aligned} \tag{3.20}$$

Therefore the random batch algorithm is consistent with Green's function G_2 in (3.9) up to order $\mathcal{O}(\Delta t^2)$. Note that the evaluation of E_I only requires $\mathcal{O}(1)$ operations.

In the implementation, to avoid frequent removal and duplication of walkers, we apply the branching process after $N/3$ batches of particles are updated. In this case, the weight function is defined by collecting the local energy from each batch (denoted by I_m here),

$$w(\boldsymbol{r}) = \exp\left[\Delta t\left(E_T-\widetilde{E}_{\text{tot}}\right)\right], \quad \widetilde{E}_{\text{tot}} = \sum_{m=1}^{N/3} E_{I_m}. \tag{3.21}$$

Similar to (3.20), one can verify with direct calculations that the branching process with probability $w(\boldsymbol{r})$ is also consistent with Green's function G_2 in (3.9). Overall, the algorithm is summarized in **Algorithm 4**.

3.2 Numerical Results

Now we test the RBM-DMC (**Algorithm** 4) and compare the results with the direct DMC method (**Algorithm** 3). For the initialization, we first apply a VMC method using the

Algorithm 4 Diffusion Monte Carlo using Random Batch (RBM-DMC)

Sample the initial num_walkers walkers using a VMC algorithm. Set $M(1)$ to be the number of walkers initially. Set E_T to be the average energy computed from the VMC.

for nt=1, num_steps **do**

for n=1, num_walkers **do**

for m=1, N/3 **do**

Randomly pick a batch I_m with three particles (i,j,k).

Perform one step of the Monte Carlo algorithm with respect to $\{q_\alpha^i\}$ and select α. Compute $\boldsymbol{b}_i = -\nabla\theta(\boldsymbol{r}_i - R_\alpha)$. Similarly compute $\boldsymbol{b}_j$ and $\boldsymbol{b}_k$.

Evaluate $\boldsymbol{u}_{ij} = -\boldsymbol{u}_{ji} = (N-1)\nabla_{\boldsymbol{r}_i} u(|\boldsymbol{r}_i - \boldsymbol{r}_j|)$. Similarly evaluate $\boldsymbol{u}_{ik}$ and $\boldsymbol{u}_{jk}$.

Update the position of the three particles,

$$\begin{aligned}
\boldsymbol{r}_i &\longleftarrow \boldsymbol{r}_i + \frac{\hbar^2}{m}\boldsymbol{b}_i\Delta t + \frac{\hbar^2}{m}(\boldsymbol{u}_{ij}+\boldsymbol{u}_{ik})\Delta t + \sigma\Delta W_i, \\
\boldsymbol{r}_j &\longleftarrow \boldsymbol{r}_j + \frac{\hbar^2}{m}\boldsymbol{b}_j\Delta t + \frac{\hbar^2}{m}(\boldsymbol{u}_{ji}+\boldsymbol{u}_{jk})\Delta t + \sigma\Delta W_j, \\
\boldsymbol{r}_k &\longleftarrow \boldsymbol{r}_k + \frac{\hbar^2}{m}\boldsymbol{b}_k\Delta t + \frac{\hbar^2}{m}(\boldsymbol{u}_{ki}+\boldsymbol{u}_{kj})\Delta t + \sigma\Delta W_k.
\end{aligned} \tag{3.22}$$

Compute the local batch energy $E_{I_m}\big(\boldsymbol{r}(t+\Delta t)\big)$ from (3.18).

end for

Determine the probability of the branching process from E_n, $E_n = \sum_{m=1}^{N/3} E_{I_m}$,

$$w_n = \exp\big[\Delta t\big(E_T - E_n\big)\big].$$

end for

Branch the walkers and adjust the energy E_T as in the direct DMC algorithm

end for

ansatz (2.5) for the wave function Φ_0. The Metropolis-Hastings Monte Carlo method is used in both methods so that they start at the same states. 300 ensembles are created by sub-sampling one sample out of every 500 steps from the VMC runs to avoid correlations among the ensembles. For both methods, we use $\Delta t = 10^{-4}$ and run 200,000 steps of simulations.

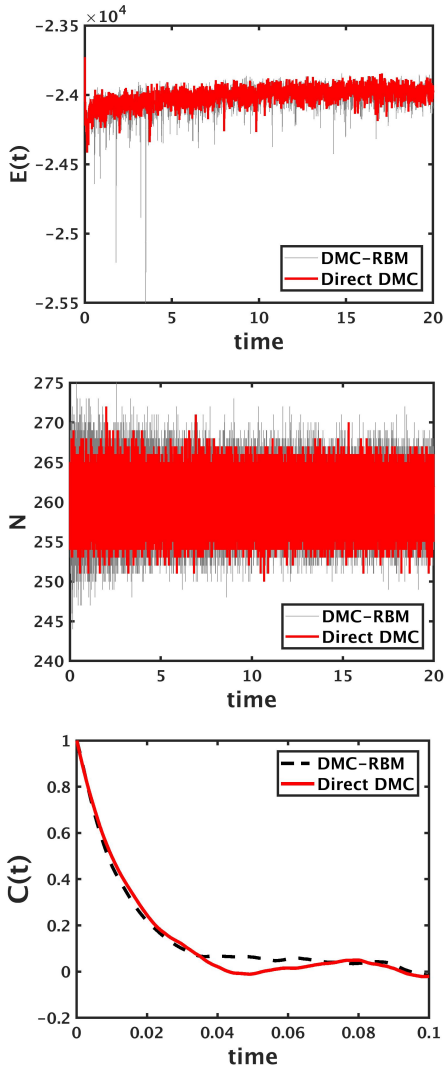

Figure 6: A comparison of the RBM-DMC (**Algorithm** 4) to the direct DMC method (**Algorithm** 3). Top: time series; Middle: The number of walkers; Bottom: time correlation.

Figure 6 shows the time series (top panel) generated by the two algorithms. We observe that the random batch method generates samples with slightly larger fluctuations during the burn in period. But the fluctuations eventually become comparable to those from the direct DMC simulations. The population of the walkers (middle panel) exhibits a similar behavior. We also examined the time correlation of the total energy

(2.9). This is done by using the time series within the time interval $(10,20)$ and regard it as a stationary process.

We conduct simulations with various choices of the step size Δt to monitor the convergence. Figure 7 shows the energy computed from each instance. We decreased Δt from 10^{-4} to 0.5×10^{-4}, and then further to 0.25×10^{-4}. We observe that the results from the direct DMC and the random batch DMC methods both exhibit linear convergences. The extrapolated energy values at $\Delta t=0$ are -2.39723×10^4 and -2.39756×10^4, respectively.

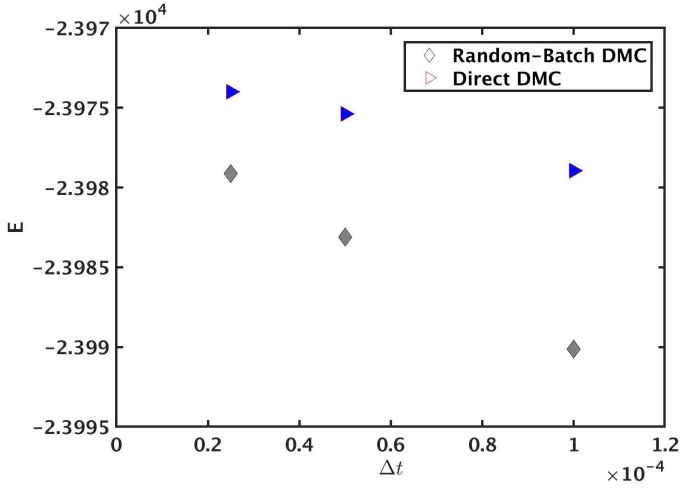

Figure 7: The computed average energy for several choices of the step size Δt .

Since our primary focus is on the speedup of the computation, We examine the CPU runtime for various system sizes. More specifically, we increase the system size from the original 168 particles, to $N=378$, $N=672$ and $N=1050$ particles, and in each case, we run the direct DMC and the RBM-DMC for 1000 steps. For the initial system $N=168$, the runtimes are 129.29 and 474.44 (seconds) for RBM-DMC and direct DMC, respectively. In this case, the random batch algorithm requires $1/4$ of the CPU time, which is a moderate speedup. But as shown in Figure 8, the CPU time for the direct DMC method increases much more rapidly as N increases.

With the advent of modern high-performance computer clusters, QMC methods have become a leading candidate for computing electronic structures of relatively large systems. As demonstrated in [24], direct DMC methods can be implemented in multi-core processors, by distributing the random walkers among different units. As a first step toward this goal, we study the ^{4}He system on a graphite lattice with non-homogeneous deformation. More specifically, by mimicking an external load, we displace the atoms in the third direction according to a Gaussian profile:

$$z_j=z_e+h_0\exp\left[-(x_j^2+y_j^2)/1000\right], \tag{3.23}$$

with h_0 indicating the height of the sheet at the origin. To establish such a spatial profile, a much larger system is needed. We consider a system with 5016 atoms, as shown in

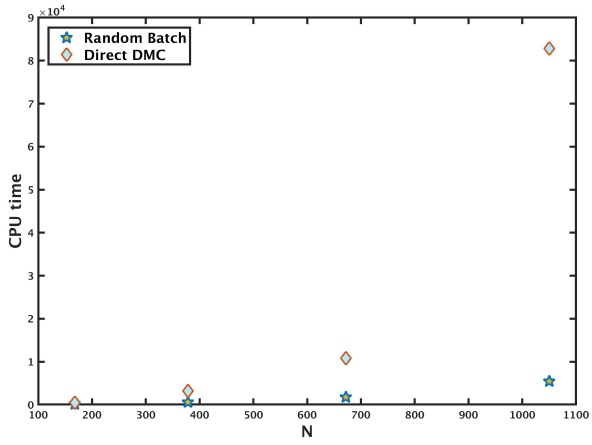

Figure 8: A comparison of the CPU runtime (in seconds) for running 1000 steps of DMC.

Figure 9. We implemented RBM-DMC (**Algorithm** 4) on 60 CPUs by distributing the walkers among the CPUs. After each branching step, the walkers are re-distributed to maintain a load balance.

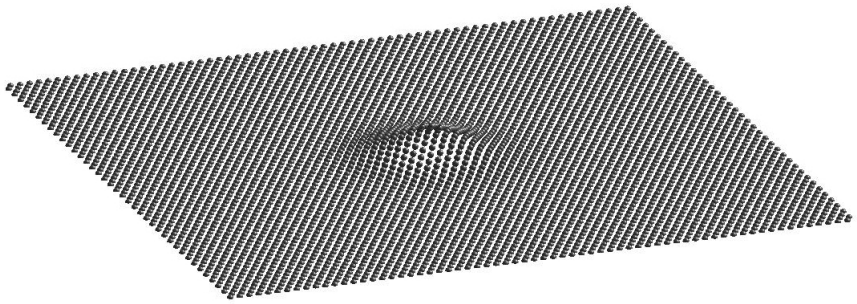

Figure 9: The out-of-plane displacement of the atoms on the graphite lattice.

We first perform the VMC simulations with 180 ensembles on the two systems, including the homogeneous lattice ($h_0 = 0$), and the deformed lattice (we pick $h_0 = 2a_0$). This is done by using the RBM-DMC (**Algorithm** 4) with the branching process turned off. We choose $\Delta t = 10^{-4}$ and run the algorithms for 160,000 steps. Figure 10 shows the energy computed from the iterations and averaged over the 180 ensembles. In both cases, the energy exhibits a sharp relaxation before reaching a steady profile. We notice that the deformation leads to higher ground state energy. Each of the VMCs simulations take about 30 hours.

At the end of the VMC run, we computed the particle density, from the 180 ensembles. For visualization purpose, we use the smoothed-kernel density estimator (mvksdensity in MATLAB) with width 1.5Å to obtain the density. In this method, the position of each

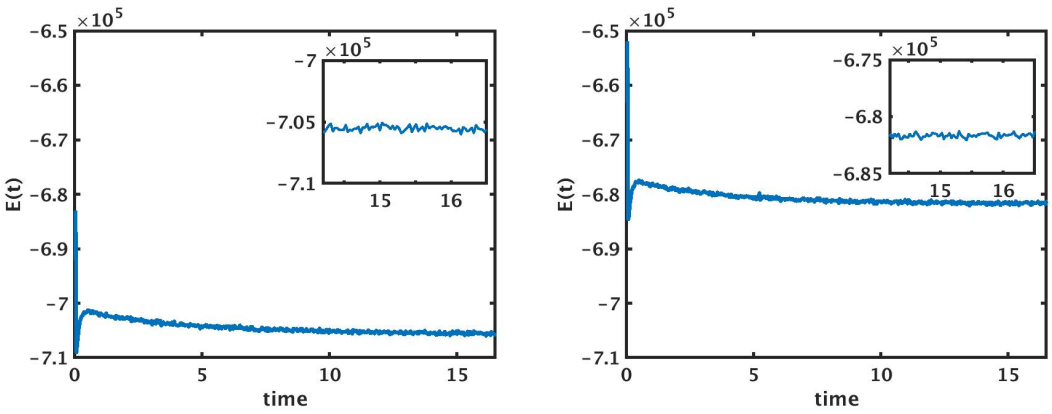

Figure 10: The energy from the VMC simulations. Left: undeformed lattice; Right: with deformation (3.23). The insets show the energy after the system reaches equilibrium.

particle (out of 5016) is interpreted as a data point, and the kernel density includes the contribution from all particles and all the ensembles. Figure 11 shows the density plots for both cases. An interesting observation is that in the deformed case, higher density is found in an annulus region, where the deformation is the largest.

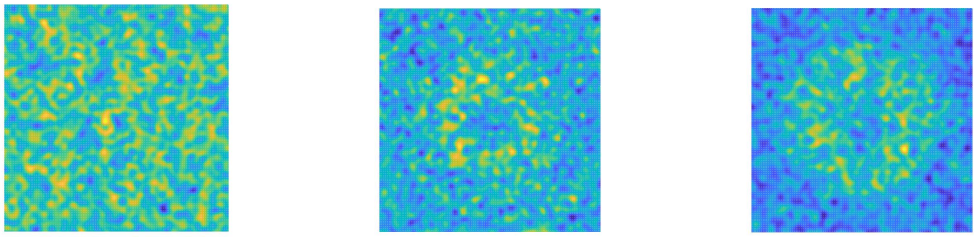

Figure 11: The particle density. Left: undeformed lattice after the VMC sampling; Middle: System with deformation after the VMC sampling; Right: System with deformation after the DMC sampling.

With the walkers prepared by the VMC simulation, we perform DMC simulations with the RBM-DMC method (**algorithm** 4). Again we use $\Delta t = 10^{-4}$ and we ran 240,000 steps of the algorithm. We monitor the energy and Figure 12 shows how the energy changes during the simulations. The system with homogeneous lattice takes slightly longer to reach the steady state, and therefore we run the simulation for an extended period (360,000 steps).

4 Summary and Discussions

We have constructed random batch algorithms for quantum Monte Carlo simulations. The main objective is to alleviate the computational cost associated with the calculations of two-body interactions, including the particle interactions in the potential energy,

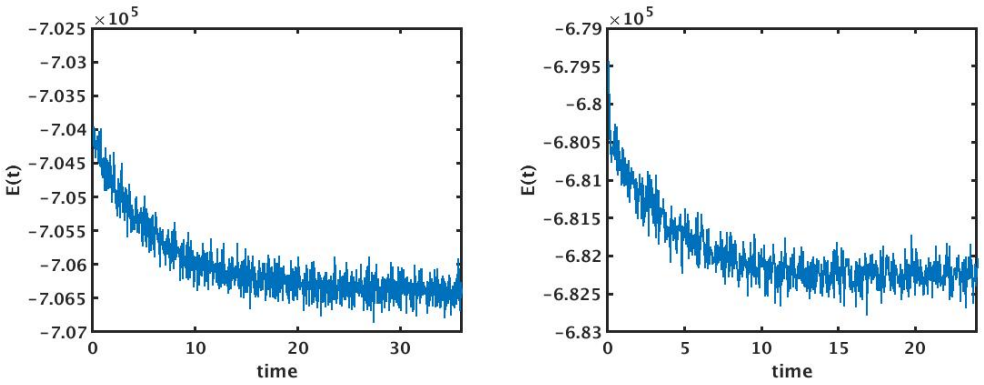

Figure 12: The energy from the DMC simulations. Left: undeformed lattice; Right: with deformation.

and the pairwise terms in the Jastrow factor. In the framework of variational Monte Carlo methods, the random batch algorithm is constructed based on the over-damped Langevin dynamics, so that updating the position of each particle only requires $\mathcal{O}(1)$ operations per time step. Consequently for the N-particle system the computational cost per time step is reduced from $O(N^2)$ to $O(N)$. For the diffusion Monte Carlo method, we proposed to decompose the total energy into on-site, two-body, and three-body terms, which can be evaluated within a random batch of three particles. This still guarantees $\mathcal{O}(N)$ operations per time step for the N-body particle system.

We have placed the main emphasis on the speedup of the computation. The speedup is more significant for larger systems, where the asymptotic scaling kicks in. In terms of the accuracy, we have shown that the random algorithms have first-order accuracy, comparable to the Euler-Maruyama method. This is certainly a low-order method. For instance, in the VMC simulations, we observed that the random batch algorithm remains stable when $\Delta t = 0.05$, but the step size has to be reduced to at least $\Delta t = 0.001$ to ensure a good accuracy. In this case, high-order diffusion Monte Carlo methods [10] would be helpful, and the construction of random batch algorithms with higher accuracy is certainly an open issue. Another common practice to correct the bias is to combine the algorithm with an Metropolis-Hastings step to accept/reject samples generated by the random batch method [37, 40]. Maintaining detailed balance in the random batch algorithm is another interesting direction.

In principle, some of these interactions in QMC can be (and have been) treated using fast summation methods, e.g., the fast multipole methods for Coulomb interactions or Gaussian functions [8, 14]. But compared to the fast summation methods, the implementation of RBM is much easier.

This paper only focuses on the VMC and DMC methods. Another important methodology is the path-integral quantum Monte Carlo [7, 17, 38], which works with the density-matrix at finite temperature. The formulation of path integral method using molecular dynamics techniques [41] seems to be an appropriate platform to implement the RBM.

Acknowledgment

Jin's research is partly supported by NSFC grant No. 11871297. Li's research is supported by NSF under grant DMS-1819011 and DMS-1953120.

References

[1] Michael P Allen and Dominic J Tildesley. *Computer Simulation of Liquids*. Oxford university press, 2017.

[2] James B Anderson. A random-walk simulation of the Schrödinger equation: H+3. *The Journal of Chemical Physics*, 63(4):1499–1503, 1975.

[3] James B Anderson. *Quantum Monte Carlo: origins, development, applications*. Oxford University Press, 2007.

[4] Léon Bottou. Online learning and stochastic approximations. *On-line learning in neural networks*, 17(9):142, 1998.

[5] Sébastien Bubeck. Convex optimization: Algorithms and complexity. *arXiv preprint arXiv:1405.4980*, 2014.

[6] Giuseppe Carleo and Matthias Troyer. Solving the quantum many-body problem with artificial neural networks. *Science*, 355(6325):602–606, 2017.

[7] David M Ceperley. Path integrals in the theory of condensed helium. *Reviews of Modern Physics*, 67(2):279, 1995.

[8] Hongwei Cheng, Leslie Greengard, and Vladimir Rokhlin. A fast adaptive multipole algorithm in three dimensions. *Journal of computational physics*, 155(2):468–498, 1999.

[9] Weinan E, Di Liu, and Eric Vanden-Eijnden. Analysis of multiscale methods for stochastic differential equations. *Communications on Pure and Applied Mathematics*, 58(11):1544–1585, 2005.

[10] Harald A Forbert and Siu A Chin. Fourth-order diffusion Monte Carlo algorithms for solving quantum many-body problems. *Physical Review B*, 63(14):144518, 2001.

[11] WMC Foulkes, Lubos Mitas, RJ Needs, and G Rajagopal. Quantum Monte Carlo simulations of solids. *Reviews of Modern Physics*, 73(1):33, 2001.

[12] Daan Frenkel and Berend Smit. *Understanding molecular simulation: from algorithms to applications*, volume 1. Elsevier, 2001.

[13] Francois Golse, Shi Jin, and Thierry Paul. The random batch method for n-body quantum dynamics. *arXiv:1912.07424*, 2020.

[14] Leslie Greengard and John Strain. The fast Gauss transform. *SIAM Journal on Scientific and Statistical Computing*, 12(1):79–94, 1991.

[15] Jiequn Han, Jianfeng Lu, and Mo Zhou. Solving high-dimensional eigenvalue problems using deep neural networks: A diffusion Monte Carlo like approach. *arXiv preprint arXiv:2002.02600*, 2020.

[16] Jiequn Han, Linfeng Zhang, and E Weinan. Solving many-electron Schrödinger equation using deep neural networks. *Journal of Computational Physics*, 399:108929, 2019.

[17] MF Herman, EJ Bruskin, and BJ Berne. On path integral Monte Carlo simulations. *The Journal of Chemical Physics*, 76(10):5150–5155, 1982.

[18] Robert Jastrow. Many-body problem with strong forces. *Physical Review*, 98(5):1479, 1955.

[19] Shi Jin and Lei Li. On the mean field limit of random batch method for interacting particle systems. *arXiv preprint arXiv:2005.11740*, 2020.

[20] Shi Jin, Lei Li, and Jian-Guo Liu. Convergence of random batch method for interacting particles with disparate species and weights. *arXiv:2003.13064*, 2020.

[21] Shi Jin, Lei Li, and Jian-Guo Liu. Random batch methods (RBM) for interacting particle systems. *Journal of Computational Physics*, 400:108877, 2020.

[22] F Joly, C Lhuillier, and B Brami. The helium-graphite interaction. *Surface science*, 264(3):419–422, 1992.

[23] Malvin H Kalos, Dominique Levesque, and Loup Verlet. Helium at zero temperature with hard-sphere and other forces. *Physical Review A*, 9(5):2178, 1974.

[24] Jeongnim Kim, Andrew D Baczewski, Todd D Beaudet, Anouar Benali, M Chandler Bennett, Mark A Berrill, Nick S Blunt, Edgar Josué Landinez Borda, Michele Casula, David M Ceperley, et al. QMCPACK: an open source ab initio quantum monte carlo package for the electronic structure of atoms, molecules and solids. *Journal of Physics: Condensed Matter*, 30(19):195901, 2018.

[25] Peter E Kloeden and Eckhard Platen. *Numerical solution of stochastic differential equations*, volume 23. Springer Science & Business Media, 2013.

[26] Dongnam Ko and Enrique Zuazua. Model predictive control with random batch methods for a guiding problem. *arXiv:2004.14834*, 2020.

[27] W. Kohn and L. J. Sham. Self-consistent equations including exchange and correlation effects. *Physical Review*, 140(4A):A1133–A1138, 1965.

[28] Lei Li, Yingzhou Li, Jian-Guo Liu, Zibu Liu, and Jianfeng Lu. A stochastic version of stein variational gradient descent for efficient sampling. *Communications in Applied Mathematics and Computational Science*, 15(1):37–63, 2020.

[29] Lei Li, Jian-Guo Liu, and Yijia Tang. A direct simulation approach for the Poisson-Boltzmann equation using the random batch method. *arXiv:2004.05614*, 2020.

[30] Lei Li, Zhenli Xu, and Yue Zhao. A random-batch Monte Carlo method for many-body systems with singular kernels. *SIAM Journal on Scientific Computing*, 42(3):A1486–A1509, 2020.

[31] Jonathan C Mattingly, Andrew M Stuart, and Desmond J Higham. Ergodicity for SDEs and approximations: locally Lipschitz vector fields and degenerate noise. *Stochastic processes and their applications*, 101(2):185–232, 2002.

[32] William Lauchlin McMillan. Ground state of liquid he4. *Physical Review*, 138(2A):A442, 1965.

[33] RJ Needs, MD Towler, ND Drummond, Pablo Lopez Rios, and JR Trail. Variational and diffusion quantum Monte Carlo calculations with the casino code. *The Journal of Chemical Physics*, 152(15):154106, 2020.

[34] Tao Pang. Diffusion Monte Carlo: a powerful tool for studying quantum many-body systems. *American Journal of Physics*, 82(10):980–988, 2014.

[35] David Pfau, James S Spencer, Alexander G de G Matthews, and W Matthew C Foulkes. Ab-initio solution of the many-electron Schrödinger equation with deep neural networks. *arXiv preprint arXiv:1909.02487*, 2019.

[36] Adrian E Raftery and Steven M Lewis. Practical Markov Chain Monte Carlo: one long run with diagnostics: implementation strategies for Markov Chain Monte Carlo. *Statistical science*, 7(4):493–497, 1992.

[37] Peter J Reynolds, David M Ceperley, Berni J Alder, and William A Lester Jr. Fixed-node quantum Monte Carlo for molecules. *The Journal of Chemical Physics*, 77(11):5593–5603, 1982.

[38] A Sarsa, KE Schmidt, and WR Magro. A path integral ground state method. *The Journal of Chemical Physics*, 113(4):1366–1371, 2000.

[39] Anthony Scemama, Michel Caffarel, Emmanuel Oseret, and William Jalby. Qmc= chem: A quantum monte carlo program for large-scale simulations in chemistry at the petascale level and beyond. In *International Conference on High Performance Computing for Computational Science*, pages 118–127. Springer, 2012.

[40] Anthony Scemama, Tony Lelièvre, Gabriel Stoltz, Eric Cancès, and Michel Caffarel. An efficient sampling algorithm for variational Monte Carlo. *The Journal of chemical physics*, 125(11):114105, 2006.

[41] Mark E Tuckerman, Bruce J Berne, Glenn J Martyna, and Michael L Klein. Efficient molecular dynamics and hybrid Monte Carlo algorithms for path integrals. *The Journal of Chemical Physics*, 99(4):2796–2808, 1993.

[42] Wolfgang von der Linden. A quantum Monte Carlo approach to many-body physics. *Physics Reports*, 220(2-3):53–162, 1992.

[43] PA Whitlock, GV Chester, and B Krishnamachari. Monte carlo simulation of a helium film on graphite. *Physical Review B*, 58(13):8704, 1998.

[44] Stephen J Wright. Coordinate descent algorithms. *Mathematical Programming*, 151(1):3–34, 2015.