Fourth-order diffusion Monte Carlo algorithms for solving quantum many-body problems

Harald A. Forbert and Siu A. Chin
Center for Theoretical Physics, Department of Physics, Texas A&M University, College Station, Texas 77843
(Received 9 September 2000; revised manuscript received 20 December 2000; published 21 March 2001)

By decomposing the important sampled imaginary time Schrödinger evolution operator to fourth order with positive coefficients, we derived a number of distinct fourth-order diffusion Monte Carlo algorithms. These sophisticated algorithms require higher derivatives of the drift velocity and local energy and are more complicated to program. However, they allowed very large time steps to be used, converged faster with lesser correlations, and virtually eliminated the step size error. We demonstrated the effectiveness of these quartic algorithms by solving for the ground-state energy and radial density distribution of bulk liquid helium.

DOI: 10.1103/PhysRevB.63.144518 PACS number(s): 67.40.Db, 02.70.Rr, 05.30.Jp

I. INTRODUCTION

The basic idea of the diffusion Monte Carlo (DMC) algorithm is to solve for the ground state of the Hamiltonian H by evolving the imaginary time Schrödinger equation

$$-\frac{\partial}{\partial t}\psi(\mathbf{x},t)=H\psi(\mathbf{x},t)=\left[-\frac{1}{2}\nabla^2+V(\mathbf{x})\right]\psi(\mathbf{x},t) \quad (1)$$

to large time.[1–3] Here, $\mathbf{x}$ and ∇^2 denote the coordinate and the Laplacian of the N-particle system. In order for the algorithm to be practical, capable of handling rapidly varying potentials, it is essential to implement important sampling as suggested by Kalos *et al.*[4] This means that instead of solving for $\psi(\mathbf{x})$, one evolves the product wave function $\rho(\mathbf{x})=\phi(\mathbf{x})\psi(\mathbf{x})$ according to[2,3]

$$-\frac{\partial}{\partial t}\rho(\mathbf{x},t)=\phi(\mathbf{x})H\phi^{-1}(\mathbf{x})\rho(\mathbf{x},t),$$

$$=-\tfrac{1}{2}\nabla^2\rho(\mathbf{x},t)+\nabla_i[G_i(\mathbf{x})\rho(\mathbf{x},t)]+E_L(\mathbf{x})\rho(\mathbf{x},t), \quad (2)$$

$$=[T+D+E_L]\rho(\mathbf{x},t)=\tilde{H}\rho(\mathbf{x},t), \quad (3)$$

where

$$E_L(\mathbf{x})=\phi(\mathbf{x})^{-1}H\phi(\mathbf{x}) \quad (4)$$

is the *local energy*,

$$G_i(\mathbf{x})=\phi(\mathbf{x})^{-1}\nabla_i\phi(\mathbf{x})=-\nabla_i S(\mathbf{x}) \quad (5)$$

is the *drift velocity*, and $\phi(\mathbf{x})=\exp[-S(\mathbf{x})]$ is the trial ground-state wave function.

Equation (3) has the formal operator solution

$$\rho(t)=e^{-t(T+D+E_L)}\rho(0)=[e^{-\epsilon(T+D+E_L)}]^n\rho(0). \quad (6)$$

Various DMC algorithms correspond to different approximations of the short-time evolution operator $e^{-\epsilon(T+D+E_L)}$. Initial implementations[1–3] of the DMC algorithm correspond to essentially approximating

$$e^{-\epsilon(T+D+E_L)}\approx e^{-\frac{1}{2}\epsilon E_L}e^{-\epsilon T}e^{-\epsilon D}e^{-\frac{1}{2}\epsilon E_L}, \quad (7)$$

which is at most first order in ϵ. By using various clever tricks, this error can be reduced substantially in specific applications.[5] However, it was recognized by Chin[6] that in order to have a general second-order DMC algorithm, one must simulate the embedded Fokker-Planck evolution operator $e^{-\epsilon(T+D)}$, i.e., the Fokker-Planck equation

$$-\frac{\partial}{\partial t}\rho(\mathbf{x},t)=-\frac{1}{2}\nabla^2\rho(\mathbf{x},t)+\nabla_i[G_i(\mathbf{x})\rho(\mathbf{x},t)]=L\rho(\mathbf{x},t) \quad (8)$$

correctly to second order. The reason for this is clear. In the limit when the trial function is the exact ground-state wave function $\phi(\mathbf{x})\to\psi_0(\mathbf{x})$, the local energy is the exact ground-state energy, which is just a constant. The convergence of the DMC algorithm would then coincide with the convergence of the Langevin algorithm for simulating the Fokker-Planck equation. Thus in order to have a second-order DMC algorithm,[6] one must have a second-order Langevin algorithm, for example, by approximating

$$e^{-\epsilon L}=e^{-\epsilon(T+D)}\approx e^{-\frac{1}{2}\epsilon T}e^{-\epsilon D}e^{-\frac{1}{2}\epsilon T}. \quad (9)$$

This idea of operator factorization seemed promising for generating higher-order DMC algorithms. However, Suzuki[7] proved in 1991 that, beyond second order, it is impossible to factorize

$$\exp[\epsilon(A+B)]=\prod_{i=1}^{N}\exp[a_i\epsilon A]\exp[b_i\epsilon B] \quad (10)$$

without having some coefficients a_i and b_i being negative. Since $e^{-a_i\epsilon T}$ is the diffusion kernel, a negative a_i would imply a diffusion process backward in time, which is impossible to simulate. Thus higher than second-order DMC algorithms cannot be based on obvious factorizations of the form (10).

In this work, we show how to derive a number of distinct quartic DMC algorithms by factorizing the operator $e^{-\epsilon(T+D+E_L)}$ to fourth order with positive coefficients. We first review how each factorized operator can be simulated in Sec. II, followed by a derivation of a fourth-order DMC algorithm in Sec. III. The backbone of this algorithm is a fourth-order Langevin algorithm that is important in its own right. In Sec. IV we examine the working details of this algorithm and check its quartic convergence on various systems including the practical case of liquid helium. In Sec. V we discuss alternative quartic algorithms by considering the

unrestricted factorization of $e^{-\epsilon(T+D+E_L)}$. The convergences of two alternative fourth-order algorithms are also tested on liquid helium. Our conclusions and suggestions for future work are contained in Sec. VI.

II. SIMULATING THE BASIC OPERATORS

The method of operator factorization depends on the fact that each component factor can be simulated exactly or to the required order. The effect of $e^{-\epsilon T}$ on $\rho(\mathbf{x},t)$ is to evolve the latter forward in time according to the *diffusion* equation

$$-\frac{\partial}{\partial t}\rho(\mathbf{x},t)=-\frac{1}{2}\nabla^2\rho(\mathbf{x},t). \tag{11}$$

For a set of points $\{x_i\}$ distributed according to $\rho(\mathbf{x},t)$, this can be exactly simulated by updating each point according to

$$x_i'=x_i+\sqrt{\epsilon}\xi_i, \tag{12}$$

where $\{\xi_i\}$ is a set of Gaussian distributed random numbers with zero mean and unit variance. The operator $e^{-\epsilon D}$ evolves $\rho(\mathbf{x},t)$ forward in time according to the *continuity* equation

$$-\frac{\partial}{\partial t}\rho(\mathbf{x},t)=\partial_i[G_i(\mathbf{x})\rho(\mathbf{x},t)], \tag{13}$$

where $G_i(\mathbf{x})\rho(\mathbf{x},t)=J_i(\mathbf{x})$ is the particle current density with drift velocity field $G_i(\mathbf{x})$. This can also be exactly simulated by setting

$$x_i'=x_i(\epsilon), \tag{14}$$

where $x_i(\epsilon)$ is the exact trajectory determined by

$$\frac{d\mathbf{x}}{dt}=\mathbf{G}(\mathbf{x}), \tag{15}$$

with the initial condition $x_i(0)=x_i$. In practice, one can only solve this trajectory equation to the required order of accuracy. The operator $e^{-\epsilon E_L}$ evolves $\rho(\mathbf{x},t)$ forward in time according to the *rate* equation

$$-\frac{\partial}{\partial t}\rho(\mathbf{x},t)=E_L(\mathbf{x})\rho(\mathbf{x},t). \tag{16}$$

The exact solution

$$\rho(\mathbf{x},t+\epsilon)=e^{-\epsilon E_L(\mathbf{x})}\rho(\mathbf{x},t) \tag{17}$$

can be simulated by updating the *weight* W_k associated with the configuration $\mathbf{x}_k$ by

$$W_k'=e^{-\epsilon[E_L(\mathbf{x}_k)-E]}W_k. \tag{18}$$

A uniform constant E is usually added to keep the weights near unity.

There are various methods[8–11] of keeping track of weights, the original and simplest method[8] is just to replicate the configuration $\mathbf{x}_i$ on the average $e^{-\epsilon[E_L(\mathbf{x}_i)-E]}$ times. We use a method that is intermediate between that of Refs. 9 and 10. Our algorithm is, however, independent of any specific method of weight tracking.

A first-order factorization of

$$e^{-\epsilon(T+D+E_L)}\approx e^{-\epsilon E_L}e^{-\epsilon T}e^{-\epsilon D}+o(\epsilon^2), \tag{19}$$

leads to the following first-order algorithm DMC1:

$$x_i'=x_i(\epsilon)+\xi_i\sqrt{\epsilon}, \tag{20}$$

where $x_i(\epsilon)$ needs to be solved at least to first order [such as $x_i(\epsilon)=x_i+\epsilon G_i(\mathbf{x})$]. The final position $\mathbf{x}'$ is then weighted by

$$W'=e^{-\epsilon[E_L(\mathbf{x}')-E]}. \tag{21}$$

A second-order factorization of

$$e^{-\epsilon(T+D+E_L)}\approx e^{-\frac{1}{2}\epsilon E_L}e^{-\frac{1}{2}\epsilon D}e^{-\epsilon T}e^{-\frac{1}{2}\epsilon D}e^{-\frac{1}{2}\epsilon E_L}+o(\epsilon^3), \tag{22}$$

leads to the second-order DMC algorithm

$$y_i=x_i(\epsilon/2)+\xi_i\sqrt{\epsilon},$$
$$x_i'=y_i(\epsilon/2), \tag{23}$$

where the final position $\mathbf{x}'$ is weighted according to

$$W'=e^{-\frac{1}{2}\epsilon[E_L(\mathbf{x})+E_L(\mathbf{x}')-2E]}. \tag{24}$$

The trajectories $x_i(\epsilon/2)$ and $y_i(\epsilon/2)$ must now be solved at least to second order. Following Ref. 6 we will refer to this as algorithm DMC2a. The alternative second-order factorization

$$e^{-\epsilon(T+D+E_L)}\approx e^{-\frac{1}{2}\epsilon E_L}e^{-\frac{1}{2}\epsilon T}e^{-\epsilon D}e^{-\frac{1}{2}\epsilon T}e^{-\frac{1}{2}\epsilon E_L}+o(\epsilon^3) \tag{25}$$

leads to the algorithm

$$y_i=x_i+\xi_i\sqrt{\epsilon/2},$$
$$x_i'=y_i(\epsilon)+\xi_i'\sqrt{\epsilon/2}, \tag{26}$$

where ξ_i and ξ_i' are independent unit Gaussian random numbers. The final position $\mathbf{x}'$ is again weighted according to Eq. (24). This is algorithm DMC2b of Ref. 6. For more details on these second-order algorithms, we refer readers to Ref. 6 for further discussions.

III. A FOURTH-ORDER ALGORITHM

For a fourth-order factorization of $\exp[\epsilon(A+B)]$ with positive coefficients, Suzuki[7] has shown that it is necessary to retain as a factor the exponential of either double commutators $[A,[B,A]]$ or $[B,[A,B]]$. Recently, Chin[12] has derived three such factorization schemes, two of which were also found previously by Suzuki.[13] To decompose $e^{-\epsilon(T+D+E_L)}=e^{-\epsilon(L+E_L)}$ to fourth order, one possibility is to keep the Langevin operator L intact. In this case, the double commutator

$$[E_L,[L,E_L]]=[E_L,[T,E_L]]=(\partial_i E_L)(\partial_i E_L) \tag{27}$$

is the square of the gradient of the local energy, which is a manageable coordinate function. Since the Langevin operator is complicated to simulate, we must choose a fourth-order factorization of $e^{-\epsilon(L+E_L)}$ that minimizes the appearance of L. We choose the following factorization as given by Refs. 12 and 13:

$$e^{-\epsilon(L+E_L)}=e^{-\frac{1}{6}\epsilon E_L}e^{-\frac{1}{2}\epsilon L}e^{-\frac{2}{3}\epsilon\tilde{E}_L}e^{-\frac{1}{2}\epsilon L}e^{-\frac{1}{6}\epsilon E_L}+o(\epsilon^5), \tag{28}$$

with $\tilde{E}_L$ given by

$$\tilde{E}_L=E_L+\tfrac{1}{48}\epsilon^2[E_L,[L,E_L]]=E_L+\tfrac{1}{48}\epsilon^2|\nabla E_L|^2. \tag{29}$$

Thus to the extent that the local energy $E_L(\mathbf{x})$ is a smooth function, the double commutator correction will be negligible.

The weights in Eq. (28) have a simple structure. If $\mathbf{x}_0$ is the initial configuration, $\mathbf{x}_{1/2}$ the Langevin evolved configuration time step $\epsilon/2$ later, and $\mathbf{x}_1$ the Langevin evolved configuration a time step $\epsilon/2$ later still, then we assign the final configuration $\mathbf{x}_1$ a weight of

$$W_1=e^{-\epsilon[\frac{1}{6}E_L(\mathbf{x}_1)+\frac{2}{3}\tilde{E}_L(\mathbf{x}_{1/2})+\frac{1}{6}E_L(\mathbf{x}_0)-E]}. \tag{30}$$

The demanding part of this DMC algorithm is the simulation of the Fokker-Planck Eq. (8). The resulting Langevin algorithm is an important simulation algorithm with numerous applications in statistical and chemical physics.[14] Since we have recently given a detailed derivation of a fourth-order Langevin algorithm,[15] we will be brief here in summarizing its essential features. To obtain a fourth-order Langevin algorithm, we again seek to decompose $e^{-\epsilon L}=e^{-\epsilon(T+D)}$ to fourth order. In this case, we keep the double commutator $[D,[T,D]]$, which is at most a second-order differential operator, and factorize the Fokker-Planck operator as[12]

$$\begin{aligned}e^{-\epsilon L}&=e^{-\epsilon(T+D)}\\&=e^{-\frac{1}{2}(1-\frac{1}{\sqrt{3}})\epsilon T}e^{-\frac{1}{2}\epsilon D}e^{-\frac{1}{\sqrt{3}}\epsilon\tilde{T}}\\&\quad\times e^{-\frac{1}{2}\epsilon D}e^{-\frac{1}{2}(1-\frac{1}{\sqrt{3}})\epsilon T}+o(\epsilon^5),\end{aligned} \tag{31}$$

$$\begin{aligned}\frac{1}{\sqrt{3}}\tilde{T}&=\frac{1}{\sqrt{3}}T+\frac{\epsilon^2}{24}(2-\sqrt{3})[D,[T,D]]\\&=\frac{1}{\sqrt{3}}T+\frac{\epsilon^2}{24}(2-\sqrt{3})[\partial_i\partial_j f_{i,j}+\partial_i v_i],\end{aligned} \tag{32}$$

where subscripts indicate partial differentiations, and

$$f_{i,j}\equiv 2S_{i,k}S_{j,k}-S_{i,j,k}S_k, \tag{33}$$

$$v_i\equiv-\tfrac{1}{2}(2S_{i,j,k}S_{j,k}+S_{i,j}S_{j,k,k}-S_{i,j,k,k}S_j). \tag{34}$$

By appropriate normal ordering, the double commutator term can be regarded as a nonuniform Gaussian random walk. However, in order to be able to sample the nonuniform Gaussian in cases where $f_{i,j}$ has negative eigenvalues, we implement the normal ordering as follows so that the full covariance matrix is always positive definite in the limit of small ϵ:

$$\begin{aligned}\exp\left(\frac{\epsilon}{\sqrt{3}}\tilde{T}\right)&=\exp\left(\frac{\epsilon}{2\sqrt{3}}T\right)\mathcal{N}\left\{\exp\left[\frac{\epsilon^3}{24}(2-\sqrt{3})(\partial_i\partial_j f_{i,j}+\partial_i v_i)\right]\right\}\exp\left(\frac{\epsilon}{2\sqrt{3}}T\right),\\&=\mathcal{N}\left\{\exp\left[\frac{\epsilon}{2\sqrt{3}}\left(-\frac{1}{2}\partial_i\partial_j\delta_{i,j}\right)+\frac{\epsilon^3}{24}(2-\sqrt{3})(\partial_i\partial_j f_{i,j}+\partial_i v_i)\right]\right\}\exp\left(\frac{\epsilon}{2\sqrt{3}}T\right),\end{aligned} \tag{35}$$

where $\mathcal{N}$ denotes the normal ordering of all derivative operators to the left. Factorization (31) can now be simulated as

$$\begin{aligned}w_i&=x_i+\xi_i\sqrt{\frac{\epsilon}{2}\left(1-\frac{1}{\sqrt{3}}\right)},\\y_i&=w_i(\epsilon/2)+\xi_i'\sqrt{\frac{\epsilon}{2\sqrt{3}}},\\z_i&=y_i-\frac{\epsilon^3}{24}(2-\sqrt{3})v_i(\mathbf{y})+\sqrt{\frac{\epsilon}{2\sqrt{3}}}\\&\quad\times\left[\delta_{i,j}+\frac{1}{2}\left(\frac{1}{\sqrt{3}}-\frac{1}{2}\right)\epsilon^2 f_{i,j}(\mathbf{y})\right]\xi_j'',\\x_i'&=z_i(\epsilon/2)+\xi_i'''\sqrt{\frac{\epsilon}{2}\left(1-\frac{1}{\sqrt{3}}\right)},\end{aligned} \tag{36}$$

where ξ_i to ξ_i''' are four sets of independent Gaussian random numbers with zero mean and unit variance. Here, the two trajectory equations $w_i(\epsilon/2)$, $z_i(\epsilon/2)$ must be solved correctly to at least fourth order. Empirically one observes that the more accurately one solves the trajectory equation, the smaller is the fourth-order error coefficient. However, in practice one must weigh improved convergence, which allows larger time steps to be used, against greater computational effort. In the present case, we solve the trajectory equation by the standard fourth-order Runge-Kutta algorithm.

Equation (28) is our basic fourth-order DMC algorithm and will be referred to as DMC4. We will first explore its workings in some detail before considering alternative algorithms.

IV. APPLYING THE FOURTH-ORDER ALGORITHM

We begin by verifying that DMC4 is indeed quartic by solving the dimensionless three-dimensional (3D) harmonic oscillator in the form

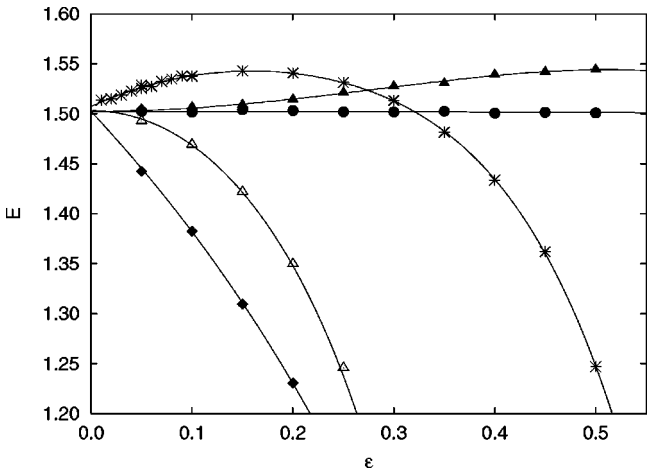

FIG. 1. The convergence of the ground-state energy of a dimensionless 3D harmonic oscillator as a function of the time step size ϵ. The diamonds are first-order DMC1 simulation results. The filled and open triangles are second-order DMC2a and DMC2b results. The asterisks indicate results of a linear algorithm with rejection. The filled circles are DMC4 results. See text for details. The error bars are smaller than the symbol size. The various lines are the corresponding exact analytical results except in the case of the rejection algorithm. For the latter case the line is just a sixth-order polynomial fit to the simulation data.

$$H = -\tfrac{1}{2}\nabla^2 + \tfrac{1}{2}r^2 \tag{37}$$

both analytically with the help of MATHEMATICA and by direct Monte Carlo simulation. The trial function used is

$$\phi(\mathbf{r}) = \exp(-\tfrac{1}{2}\alpha r^2) \tag{38}$$

with a deliberate poor choice of the trial parameter $\alpha = 1.8$. In Fig. 1 we plot the ground-state energy from the mixed expectation

$$E = \frac{\langle \phi | H | \psi_0 \rangle}{\langle \phi | \psi_0 \rangle} \tag{39}$$

as a function of the step size ϵ used. The lines are analytical functions from MATHEMATICA and the plotting symbols are Monte Carlo simulation results. We have included one first-order, two second-order, and one first-order rejection DMC algorithm for comparison. The detailed descriptions of DMC1, DMC2a, and DMC2b can be found in Ref. 6. The rejection algorithm uses a first-order Langevin algorithm together with a generalized Metropolis acceptance-rejection step so that the square of the trial function is exactly sampled at all step sizes.[2] For this case, we have no analytical result and the plotted line is just a sixth-order polynomial fit. The point of this exercise is not that we can solve the harmonic oscillator successfully, but that we have verified the quartic convergence of our fourth-order algorithm *analytically*. Moreover, we have verified numerically that our Monte Carlo implementation exactly matches theoretical expectations; that even in Monte Carlo simulations, the quartic convergence is very distinct from quadratic convergence.

We next test DMC4 by solving the dimensionless 3D Morse potential with the Hamiltonian

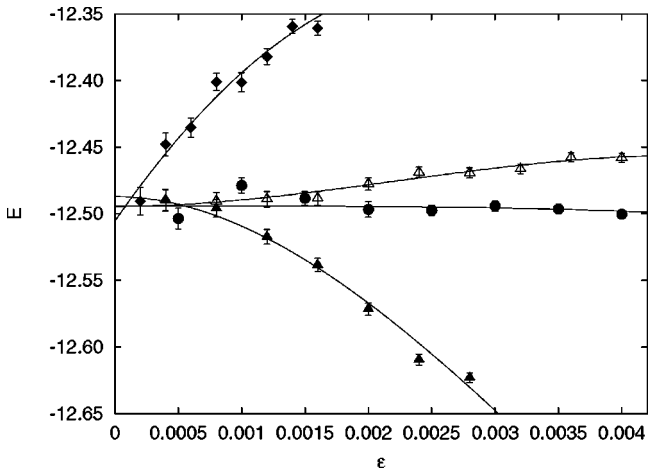

FIG. 2. The convergence of the ground-state energy of a dimensionless 3D Morse oscillator as a function of the time step size ϵ. The diamonds are first-order DMC1 simulation results. The filled and open triangles are second-order DMC2a and DMC2b results. The filled circles are the fourth-order results of DMC4. The various lines are least-square fits to the simulation data. The first-order results are fitted with a parabola, the second-order results by a cubic polynomial, and the fourth-order results by just a constant plus a fourth-order term in ϵ.

$$H = -\tfrac{1}{2}\nabla^2 + D_e[e^{-2\alpha(r-r_0)} - 2e^{-\alpha(r-r_0)}], \tag{40}$$

with $D_e = 50$, $r_0 = 1$ and $\alpha = 10$. These values ensure that the ground state is high up in the potential and that the ground-state wave function is not well approximated by a Gaussian. Again for testing purposes, we shall simply regard this as a dimensionless problem. We use a trial function of the form

$$\phi(\mathbf{r}) = \exp(-ar - br^{-3}), \tag{41}$$

with $a = 15.29$, $b = 6.82$, and variational energy -11.1774. This is to be compared with the exact ground-state energy of -12.5. The convergence of various DMC algorithms are compared in Fig. 2. The quartic convergence of DMC4 is again verified. Its convergence is clearly distinct from lower-order results and is nearly flat. In this case, we have no analytical results and all lines are just least-square fits to the data.

To demonstrate that DMC4 can be used to solve realistic physical problems, we use it to solve for ground-state properties of bulk liquid helium described by the many-body Hamiltonian

$$H = \sum_i -\frac{\hbar^2}{2m}\nabla_i^2 + \sum_{i<j} V(r_{ij}), \tag{42}$$

where $\hbar^2/m = 12.12$ Å^2 K with potential V determined by Aziz *et al.*[16] Instead of the usual McMillan trial function, we use a trial function of the form

$$\phi(\mathbf{x}) = \prod_{i<j} \exp\{-\ln(2)\exp[-(r_{ij} - c_0)/d_0]\}. \tag{43}$$

With $c_0 = 2.8$ and $d_0 = 0.48$ Å, this trial function gives a slightly better energy of 5.886(5) K/particle. Since the stan-

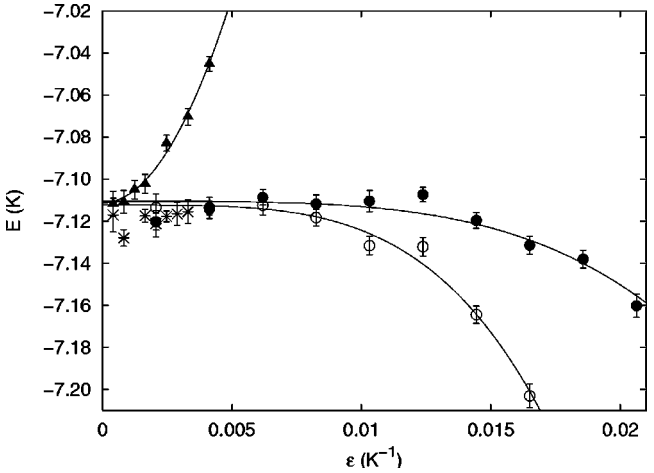

FIG. 3. The time step convergence of the ground-state energy per particle for bulk liquid helium in a 128-particle simulation. The solid circles are the result of our fourth-order algorithm DMC4. The open circles and asterisks are for algorithm DMC4a and DMC4b, respectively. For comparison, we also show as triangles second-order results from algorithm DMC2a. The lines are least-square fits to the data.

dard calculation details[17] are well known, we will just describe the results as summarized in Fig. 3. Again, the convergence of our fourth-order algorithm is clearly quartic. The extrapolated values are $-7.114(2)$ K for our fourth-order algorithm and $-7.111(2)$ K for the second-order algorithm DMC2a. Both are in agreement with Boronat's and Casulleras's[18] second-order DMC result of $-7.121(10)$ K. For 128 particles with a population of 400, each data point of DMC2a required $T_2=11.1$ h on a single processor of an Origin 2000 machine programmed in *C*. We will use this time T_2 as a standard for comparing other algorithms running the same number of iterations. Algorithm DMC4, whose details will be further described below, requires 4.6 T_2 per data point. Note that very large time steps can be used with algorithm DMC4, roughly ten times as large as those of second-order algorithm DMC2a.

In Fig. 4, we show the resulting radial density distribution $g(r)$ from our fourth-order calculation at $\epsilon=0.008\,25$ K^{-1} and $\epsilon=0.0165$ K^{-1}. The distribution is virtually unchanged even at these large time steps and both are in excellent agreement with the experimental distribution of Svensson *et al.*[19]

In Fig. 5 we show DMC4's thermalization toward the exact ground state from the variational trial wave function. Starting from the initial variational energy, we plot the population averaged energy as a function of iterated time for various time step sizes. This plot shows that each iteration of the algorithm at $\epsilon=0.0165$ K^{-1} is indistinguishable from multiple iterations at smaller time steps having the same time interval. Moreover, it demonstrates that the algorithm converges to the ground state inversely proportional to the step size used, up to $\epsilon=0.0165$ K^{-1}. That is, only five iterations are needed at $\epsilon=0.0165$ K^{-1} to reach the exact ground state near $t=0.08$ K^{-1} and 20 iterations at $\epsilon=0.004\,125$ K^{-1}, etc. Thus our fourth-order algorithm can project out the ground state with ten times fewer updates than a second-order algorithm.

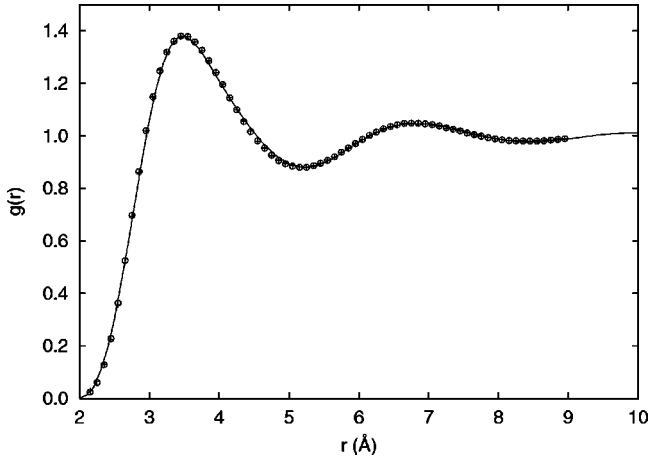

FIG. 4. The radial density distribution of bulk liquid helium. The circles are DMC4 results at $\epsilon=0.008\,25$ K^{-1} and the crosses are DMC4 results at $\epsilon=0.0165$ K^{-1}. The solid line is the experimentally extracted $g(r)$ of Svensson *et al.* (Ref. 19) at 1 K.

More important than the thermalization time is the observable correlation time. In a Monte Carlo calculation, it is highly desirable to have uncorrelated configurations for an accurate estimate of the statistical errors. The correlation coefficient for an observable O is defined by

$$c_O(\Delta t)=\frac{\langle O(t+\Delta t)O(t)\rangle-\langle O(t)\rangle^2}{\langle O(t)O(t)\rangle-\langle O(t)\rangle^2}. \tag{44}$$

In Fig. 6, we show the ground-state energy correlation function of liquid helium as computed by our fourth-order algorithm. The correlation time is roughly $\Delta t\approx 0.15$ K^{-1}, at which point the correlation coefficient dropped to zero. This plot shows that the correlation time depends only on the total

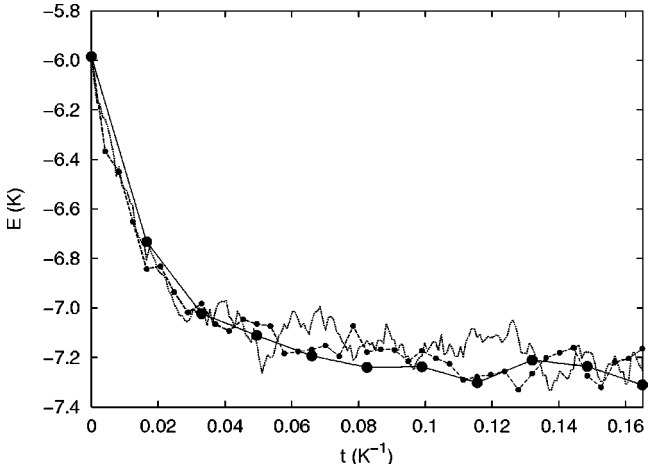

FIG. 5. The relaxation of liquid helium's ground-state energy toward its exact value as simulated by DMC4 at three time step sizes. The large circles are at $\epsilon=0.0165$ K^{-1} and the small circles are at $\epsilon=0.004\,125$ K^{-1}. They are connected by straight-line segments to guide the eye. The dotted line corresponds to results at $\epsilon=0.000\,825$ K^{-1}.

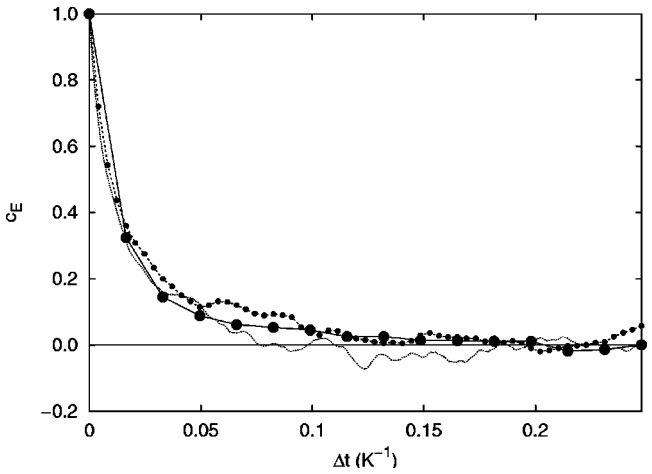

FIG. 6. The correlation coefficient function, Eq. (44), for the ground-state energy of liquid helium as computed by DMC4 at the three time step size of $\epsilon=0.0165\,\mathrm{K}^{-1}$ (large circles), $\epsilon=0.004\,125\,\mathrm{K}^{-1}$ (smaller circles) and $\epsilon=0.000\,825\,\mathrm{K}^{-1}$ (dotted line). The connecting line segments are for guiding the eye only.

time separation. Thus if the algorithm remains accurate at large time steps, then fewer iterations are needed to produce uncorrelated configurations.

In implementing the fourth-order Langevin algorithm, we used the standard fourth-order Runge-Kutta algorithm to solve the trajectory Eq. (15). When the step size is large, the fourth-order error in the Runge-Kutta algorithm can greatly overshadow the intrinsic fourth-order step size error of the Langevin and that of the DMC algorithm, causing both to fail prematurely. To guard against this, we monitor the difference between the results of the fourth-order Runge Kutta and its embedded second-order algorithm. If the square of this difference is larger than some tolerance, say 0.01, we recalculate the trajectory twice at half the time step size. Even at the largest step size used, only a few percent of trajectories need to be recalculated, incurring only a small additional overhead. This additional effort greatly extended the flatness of the convergence curve as shown in Fig. 3.

V. ALTERNATIVE FOURTH-ORDER ALGORITHMS

Our DMC4 algorithm (28), which preserves the Fokker-Planck operator L intact, may not be the most efficient fourth-order algorithm possible. Consider the limit in which the trial function approaches the exact ground state $\phi(\mathbf{x})\rightarrow\psi_0(\mathbf{x})$. In this ideal case the local energy becomes an irrelevant constant $E_L(\mathbf{x})\rightarrow E_0$, and the algorithm is just

$$e^{-\epsilon(L+E_L)}\propto e^{-\frac{1}{2}\epsilon L}e^{-\frac{1}{2}\epsilon L}, \tag{45}$$

which is the running of the fourth-order Langevin algorithm twice, at half the time step. It seems plausible that one should be able to derive a fourth-order DMC algorithm that reduces to a single run of the fourth-order Langevin algorithm in the same limit.

We are thus led to consider the general factorization, to fourth order, of a three-operator exponential $e^{-\epsilon(T+D+E_L)}$. There are now nine double commutators to be considered: 6 are the generalizations of the two-operator case,

$$[T,[D,T]],\quad [D,[T,D]],\quad [D,[E_L,D]],$$

$$[E_L,[D,E_L]],\quad [E_L,[T,E_L]],\quad [T,[E_L,T]], \tag{46}$$

and three new ones are related by the Jacobi identity

$$[T,[D,E_L]]+[D,[E_L,T]]+[E_L,[T,D]]=0. \tag{47}$$

Thus only two of the last three commutators are independent. Note also that for the present form of the operators, $[E_L,[D,E_L]]=0$. We have examined all these commutators in the case of liquid helium to determine which one is doable and can be kept. To explore the many possible factorizations, we have devised a MATHEMATICA program to combine the exponential of operators symbolically. With the help of this program, we have explored an extensive list of distinct fourth-order algorithms. Since there are many operators in each such factorization, it is too cumbersome to write out the explicit exponential form. Moreover, since the factorization will always be left-right symmetric, there is no need to repeat operators on the left side. In the following, we will only indicate the exponential operators symbolically beginning with the *central* one and list only operators to the *right*. For example, algorithm DMC4 (28) will be denoted as

$$\begin{aligned} T+D+E_L &\approx \frac{2}{3}\tilde{E}_L+\frac{1}{2}L+\frac{1}{6}E_L, \\ &\approx \frac{2}{3}\tilde{E}_L+\frac{1}{2}\left[\frac{1}{2}\left(1-\frac{1}{\sqrt{3}}\right)T+\frac{1}{2}D+\frac{1}{\sqrt{3}}\tilde{T}+\frac{1}{2}D+\frac{1}{2}\left(1-\frac{1}{\sqrt{3}}\right)T\right]+\frac{1}{6}E_L. \end{aligned} \tag{48}$$

Each update of this algorithm requires the evaluation of, in decreasing order of computational complexity, four D's, two $\tilde{T}$'s, one $\tilde{E}_L$, one E_L, and four T's. (The last E_L from the last update can be used as the first E_L of the current update.) Since D is the most computationally intensive operator, followed by $\tilde{T}$, $\tilde{E}_L$, etc., we would like to minimize their appearance in that order. Below, we will describe two alternative algorithms that are computationally more economical than DMC4 in solving for the ground state of liquid helium.

One possible fourth-order algorithm is to retain the same

double commutators $[D,[T,D]]$ and $[E_L,[T,E_L]]$ as in DMC4, but allow $L=T+D$ to be broken up:

$$T+D+E_L \approx \tfrac{1}{3}\tilde{T}+\tfrac{1}{6}D+\tfrac{3}{8}E_L+\tfrac{1}{6}T+\tfrac{1}{3}D+\tfrac{1}{6}T+\tfrac{1}{8}\tilde{E}_L. \tag{49}$$

Here, $\tilde{T}$ and $\tilde{E}$ are given by

$$\tilde{T}=T+\frac{\epsilon^2}{72}[D,[T,D]], \tag{50}$$

$$\tilde{E}_L=E_L+\frac{\epsilon^2}{12}[E_L,[T,E_L]]. \tag{51}$$

This algorithm requires four D's, but only one $\tilde{T}$, one $\tilde{E}_L$, two E_L's, and four T's. We will denote this algorithm as DMC4a. This algorithm is roughly 10% faster than DMC4 and its quartic convergence is clearly demonstrated in Fig. 3. However, its convergence range is only about half of DMC4. The running time for this algorithm is 4.1 T_2.

To reduce the number of D operators, one must pay the price of retaining additional double commutators. We will refer to the following algorithm with only two D operators as DMC4b:

$$T+D+E_L \approx \frac{1}{\sqrt{3}}a_0\tilde{T}+\frac{1}{2\sqrt{3}}E_L+\frac{1}{2\sqrt{3}}(1-a_0)T+\frac{1}{2}D + \frac{1}{4}c_0T+\frac{1}{2}c_0E_L+\frac{1}{4}c_0T, \tag{52}$$

where $a_0=1/\sqrt{1+\sqrt{3}}$, $c_0=1-1/\sqrt{3}$, and

$$\frac{1}{\sqrt{3}}a_0\tilde{T}=\frac{1}{2\sqrt{3}}a_0T+\frac{\epsilon^2}{24}\left[(2-\sqrt{3})([D,[T,D]]+[D,[E_L,D]])+\left(c_0-\frac{a_0}{\sqrt{3}}\right)[E_L,[T,E_L]]\right]+\frac{1}{2\sqrt{3}}a_0T. \tag{53}$$

The additional commutator

$$[D,[E_L,D]]=-G_i\nabla_i[G_j\nabla_jE_L(\mathbf{x})] \tag{54}$$

is a calculable function involving a higher derivative of G_i and E_L. In this algorithm, we have placed all the double commutators at the center so that they are evaluated only once per update. This is done by splitting $(1/\sqrt{3})a_0T \rightarrow (1/2\sqrt{3})a_0T+\ldots(1/2\sqrt{3})a_0T$ in Eq. (53), meaning that we first do half of the required Gaussian random walk, evaluate all double commutators, then complete the remaining half of the Gaussian walk including the effect of $[D,[T,D]]$ as it is done in the Langevin algorithm. The ubiquitous irrational coefficients are roots of quadratic equations that force unwanted double commutators to vanish. One can check by inspection that as $\phi(\mathbf{x})\rightarrow\psi_0(\mathbf{x})$ and $E_L(\mathbf{x})\rightarrow E_0$, Eq. (52) reduces to just the fourth-order Langevin algorithm (31).

The results of this algorithm for liquid helium are shown in Fig. 3 as asterisks. The ground-state energy is correctly obtained, but because these higher derivatives are rapidly varying functions, we have not been able to stabilize the population of weights beyond $\epsilon\approx 0.004$. While this algorithm may not work as well as DMC4 and DMC4a for liquid helium, its economy of requiring only two trajectories per update may be of utility in other applications. The calculation time per data point shown is 3.2 T_2.

VI. CONCLUSIONS

In this work, we have derived a number of distinct fourth-order DMC algorithms by factorizing the imaginary time Schrödinger evolution operator to fourth order. This is a notable advance in algorithm development, made possible only by the recent progress in understanding positive coefficient operator factorization. Our work illustrates a global view of algorithms as products of factorized operators. Such a perspective gives order and insight into the working of the diffusion Monte Carlo algorithm. It would have been very difficult to derive such a high-order algorithm without such a conceptual structure. We have further demonstrated the practicality of these algorithms by using them to solve for the ground state of liquid helium. The quartic convergences of DMC4 and DMC4a have been verified and both yielded ground-state energy and radial density distribution in excellent agreement with experiment. Despite the fact that these algorithms are rather complicated to program requiring higher-order derivatives, they allow very large step sizes to be used, virtually eliminate the time step size error, and greatly reduce statistical correlations between successive updated configurations.

Since this is only the first demonstration of quartic algorithms, there is room for further improvements. For example, we have shown how the factorization of a three-operator exponential can lead to a number of distinct fourth-order DMC algorithms. A more systematic categorization of various fourth-order factorizations would help in obtaining the most efficient algorithm. Secondly, the retainment of some double commutators is necessary, however, it has not been studied in detail where they should be placed so as to minimize the fourth-order error coefficient or computational effort. It is observed that the step size convergence curve is flatter when the trajectory equation is solved more exactly. In this work, we have only used the fourth-order Runge-Kutta algorithm in solving for the deterministic trajectory. Future study may explore the effects of using alternative numerical methods, such as symplectic algorithms[20] for solving the trajectory equation.

Recently, there has been a breakthrough in applying the diffusion Monte Carlo method for solving strongly interact-

ing Fermi systems. Casulleras and Boronat[21] have shown that normal liquid ^{3}He can be solved to experimental accuracy by a combination of a better backflowing trial function, fixed-node approximation, and analytical nodal surface improvements. The implication here is that physical, strongly interacting Fermion problems can be solved without having an ideal ''Fermion algorithm.'' To the extent that this work can solve the fixed-node energy at larger time steps, it will contribute to the efficiency of solving physical Fermion problems in general. Furthermore, Casulleras and Boronat only improved the nodal surface to first order; our method suggests that improvements to fourth order are possible, depending on the complexity of the trial wave function used.

ACKNOWLEDGMENT

This research was funded, in part, by the United States National Science Foundation Grant Nos. PHY-9512428, PHY-9870054, and DMR-9509743.

[1] J. B. Anderson, J. Chem. Phys. **63**, 1499 (1975).
[2] P. Reynolds, D. Ceperley, B. Alder, and W. Lester, J. Chem. Phys. **77**, 5593 (1982).
[3] J. Moskowitz, K. Schmidt, M. Lee, and M. Kalos, J. Chem. Phys. **77**, 349 (1982).
[4] M. H. Kalos, D. Levesque, and L. Verlet, Phys. Rev. A **9**, 2178 (1974).
[5] C. J. Umrigar, M. P. Nightingale, and K. J. Runge, J. Chem. Phys. **99**, 2865 (1993).
[6] S. A. Chin, Phys. Rev. A **42**, 6991 (1990).
[7] M. Suzuki, J. Math. Phys. **32**, 400 (1991).
[8] M. H. Kalos, Phys. Rev. **128**, 1791 (1962).
[9] J. H. Hetherington, Phys. Rev. A **30**, 2713 (1984).
[10] M. P. Nightingale and H. W. J. Blöte, Phys. Rev. Lett. **60**, 1562 (1988).
[11] N. Cerf and O. C. Martin, Phys. Rev. E **51**, 3679 (1995).
[12] S. A. Chin, Phys. Lett. A **226**, 344 (1997).
[13] M. Suzuki, in *Computer Simulation Studies in Condensed Matter Physics VIII*, edited by D. P. Landau, K. K. Mon, and H.-B. Shüttler (Springer, Berlin, 1996).
[14] H. Risken, *The Fokker-Planck Equation, Methods of Solution and Applications*, 2nd ed. (Springer, New York, 1989).
[15] H. A. Forbert and S. A. Chin, Phys. Rev. E **63**, 016703 (2001).
[16] R. A. Aziz *et al.*, J. Chem. Phys. **70**, 4330 (1979).
[17] M. H. Kalos, M. A. Lee, P. A. Whitlock, and G. V. Chester, Phys. Rev. B **24**, 115 (1981).
[18] J. Boronat and J. Casulleras, Phys. Rev. B **49**, 8920 (1994).
[19] E. C. Svensson, V. F. Sears, A. D. B. Woods, and P. Martel, Phys. Rev. B **21**, 3638 (1980).
[20] E. Forest and R. D. Ruth, Physica D **43**, 105 (1990).
[21] J. Casulleras and J. Boronat, Phys. Rev. Lett. **84**, 3121 (2000).