Journal of Computational Physics 228 (2009) 3390–3404

Contents lists available at ScienceDirect

Journal of Computational Physics

journal homepage: www.elsevier.com/locate/jcp

QWalk: A quantum Monte Carlo program for electronic structure

Lucas K. Wagner [a,b,*], Michal Bajdich [a], Lubos Mitas [a]

[a] Center for High Performance Simulation and Department of Physics, North Carolina State University, Raleigh, NC 27695, United States
[b] The CNS Group, 366 Le Conte Hall #3700, Berkeley, CA 94720-3700, United States

ARTICLE INFO

Article history:
Received 5 December 2007
Received in revised form 3 November 2008
Accepted 20 January 2009
Available online 29 January 2009

Keywords:
Monte Carlo
Stochastic methods
Quantum mechanics

ABSTRACT

We describe QWalk, a new computational package capable of performing quantum Monte Carlo electronic structure calculations for molecules and solids with many electrons. We describe the structure of the program and its implementation of quantum Monte Carlo methods. It is open-source, licensed under the GPL, and available at the web site http://www.qwalk.org.

1. Introduction

The solution of the stationary Schrödinger equation for interacting systems of quantum particles is one of the key challenges in quantum chemistry and condensed matter physics. Many problems in electronic structure of atoms, molecules, clusters and solids require the ground and excited eigenstates of the electron–ion (or electron–nucleus) Born–Oppenheimer Hamiltonian

$$H = -\frac{1}{2}\sum_i \nabla_i^2 - \sum_{il}\frac{Z_I}{r_{il}} + \sum_{i>j}\frac{1}{r_{ij}}, \tag{1}$$

where upper/lower cases indicate nuclei/electrons. Due to the Coloumb interaction, the eigenstates are very complicated functions in the in $3N_e$-dimensional space where N_e is the number of electrons. Over the past six decades or so, physicists and chemists have developed many powerful approaches and theories that attempt to solve the electronic structure problem with varying degrees of accuracy. Among these are the wave function methods such as Hartree–Fock (HF) and post Hartree–Fock (post-HF), and also methodologies which are based on functionals of electron density such as density functional theories (DFT). Because none of them are exact in practice, each of these methods occupies its place in the computational toolbox. DFT represents an excellent tradeoff between accuracy and computational efficiency, allowing thousands of electrons to be treated, usually getting qualitative trends correctly for many quantities and materials such as cohesive/binding energies, many (but not all) energy differences between different systems, and can even be quantitatively accurate for some quantities (such as geometries), especially for the systems of atoms from the first two rows of the periodic table. Many systems and effects are, however, not accurately described (van der Waals systems, systems with transition metal atoms, many excitations, among others) and require treatment of quantum many-body effects more accurately. One can then turn to post-Hartree–Fock methods

* Corresponding author. Address: Center for High Performance Simulation and Department of Physics, North Carolina State University, Raleigh, NC 27695, United States.
E-mail addresses: lkwagner@berkeley.edu, lucas.wagner@gmail.com (L.K. Wagner), mbajdic@ncsu.edu (M. Bajdich), lmitas@ncsu.edu (L. Mitas).

0021-9991/$ - see front matter © 2009 Elsevier Inc. All rights reserved.
doi:10.1016/j.jcp.2009.01.017

based on sophisticated expansions of wave functions in one-particle basis sets. These methods can be made formally exact at the cost of exponential scaling with system size, and the most accurate approximate methods scale quite poorly with the system size, say, $O(N_e^{5-7})$. It is very difficult to find a method that scales well, at most $O(N_e^3)$, and also offers higher accuracy than DFT.

Quantum Monte Carlo methods fill this gap by using stochastic algorithms to treat the many-body wave function in the full $3N_e$-dimensional space. This approach has several advantages – good scaling in the number of electrons ($O(N_e^{2-3})$, depending on the quantity of interest) and is amenable to parallel implementations at 99% efficiency. Over the past ~20 years, QMC has been applied to a host of systems such as model systems, atoms, molecules and solids, with impressive accuracy across this wide range [1–6]. For extended systems, particularly, it is the most accurate method available for total energies on the materials that have been tested. Since these calculations represent rather recent developments, the packages for QMC are currently in development and only a few options are available for the community at large. We have developed a new program QWalk for general purpose QMC calculations written in C++ with modern programming techniques and incorporating state of the art quantum Monte Carlo algorithms including variational, diffusion and reptation Monte Carlo in a fast and flexible code. QWalk has already been used in several publications [7–12], and we would like to present a summary of its current capabilities.

There are several packages in existence that perform QMC calculations [13–16]; however, QWalk offers unique capabilities in the design of the software and the attitude towards third party contributions. The code is centered around principles of separation of data and abstraction of types, in that each piece of code knows as little as possible about the context in which it is used and the objects that it uses. This structure has three major effects on code development: (1) avoiding bugs by keeping the number of interactions between different parts of the code small, (2) enabling new developers to quickly add new features into the code, and (3) enabling interchangeability of different pieces of code that perform equivalent tasks. This structure allows us to achieve the two goals of adding developers to the QWalk community and experimenting with new QMC methodologies.

2. Methods

QWalk implements three flavors of quantum Monte Carlo methods: variational Monte Carlo, diffusion Monte Carlo, and reptation Monte Carlo. They each offer different tradeoffs in speed, accuracy, and which quantities are available. We give a summary here.

2.1. Variational Monte Carlo

The expectation value for an arbitrary operator $\mathcal{O}$ and a given trial wave function Ψ_T is given by

$$\langle \mathcal{O} \rangle = \frac{\langle \Psi_T | \mathcal{O} | \Psi_T \rangle}{\langle \Psi_T | \Psi_T \rangle} = \frac{\int \Psi_T^2(\mathbf{R}) [\mathcal{O}\Psi_T(\mathbf{R})/\Psi_T(\mathbf{R})] d\mathbf{R}}{\int \Psi_T^2(\mathbf{R}) d\mathbf{R}},$$

where $\mathbf{R} = (\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_{N_e})$ denotes a set of N_e electron coordinates in 3D space. Typically, such integrals are evaluated by reducing the multi-dimensional integral into a sum of products of low-dimensional integrals. Unfortunately, this either restricts the functional form of $\Psi_T(R)$ or makes the calculations undo-able for more than a few electrons. One of the key motivations for employing stochastic approaches is to eliminate this restriction and to gain qualitatively new variational freedom for describing many-body effects.

In order to evaluate the expectation value integral stochastically we first generate a set $\{\mathbf{R}_m\}$ of statistically independent sampling points distributed according to $\Psi_T^2(\mathbf{R})$ using the Metropolis [17,18] algorithm. In this algorithm, there is some freedom in choosing the moves. For atoms with effective core potentials, we use the moves as outlined in Ref. [1], modified with a delayed rejection step similar to Ref. [19], although developed independently, and for full-core calculations, we use the accelerated Metropolis method from Ref. [20]. The total energy and its components are evaluated, as well as other properties. The expectation value is then estimated by averaging over the samples $\{\mathbf{R}_m\}$. For example, the VMC energy is given by the average of the quantity called local energy

$$E_{\mathrm{VMC}} = \frac{1}{M}\sum_{m=1}^{M} \frac{H\Psi_T(\mathbf{R}_m)}{\Psi_T(\mathbf{R}_m)} + \varepsilon = \frac{1}{M}\sum_{m=1}^{M} E_{loc}(\mathbf{R}_m) + \varepsilon$$

with the statistical error ε proportional to $1/\sqrt{M}$.

It is straightforward to apply the variational theorem in this framework. Consider a variational wave function $\Psi_T(R,P)$, where R is the set of all the electron positions and P is the set of variational parameters in the wave function

$$E(P) = \frac{\int \Psi_T(\mathbf{R},P) H \Psi_T(\mathbf{R},P) d\mathbf{R}}{\int \Psi_T^2(\mathbf{R},P) d\mathbf{R}}. \tag{2}$$

A (hopefully) good approximation to the ground state is then the wave function with the set of parameters P that minimizes $E(P)$. The stochastic method of integration allows us to use explicitly correlated trial wave functions defined in the next

section such as the compact Slater–Jastrow form. In fact, as long as the trial function and its derivatives can be evaluated quickly, any functional form can be used.

2.2. Diffusion Monte Carlo

To obtain accuracy beyond a given variational ansatz, we employ another method which projects out the ground state of a given symmetry from any trial wave function. To do this, we simulate the action of the operator $e^{-(H-E_0)\tau}$ on the trial function, where τ is the projection time and E_0 is the self-consistently determined energy of the ground state. As $\tau \to \infty, e^{-(H-E_0)\tau}\Psi_T \to \Phi_0$, where Φ_0 is the ground state of a given symmetry. For large τ, there is no general expansion for $e^{-(H-E_0)\tau}$, but for small τ, we can write the projection operator in **R**-representation

$$G(\mathbf{R}', \mathbf{R}, \tau) = \exp(-(\mathbf{R}' - \mathbf{R})^2/2\tau) \times \exp[-\tau(V(\mathbf{R}) + V(\mathbf{R}') - 2E_0)/2] + \mathcal{O}(\tau^3)$$

using the Trotter's expansion formula for small τ. The resulting Green's function can be interpreted as a product of a diffusion kernel $G_D(\mathbf{R}', \mathbf{R}, \tau) = \exp(-(\mathbf{R}' - \mathbf{R})^2/2\tau)$ and a branching kernel $G_B(\mathbf{R}', \mathbf{R}, \tau) = [-\tau(V(\mathbf{R}) + V(\mathbf{R}') - 2E_0)/2]$.

The basic idea of diffusion Monte Carlo (DMC) is to sample a path

$$G(\mathbf{R}_N, \mathbf{R}_{N-1}, \tau) \cdots G(\mathbf{R}_2, \mathbf{R}_1, \tau)\Psi_T(\mathbf{R}_1). \tag{3}$$

For N large enough (for a long enough path), the distribution of $\mathbf{R}_N$ will approach Φ_0. However, to interpret this as a stochastic process, the path distribution must be positive; that is, the product of all G's with Ψ_T must be positive. This gives rise to the fixed-node approximation [21–24], where the nodes (the places where the trial function equals zero) of the trial wave function are used as approximation to the nodes of the ground state wave function. One can avoid this restriction by performing a released-node calculation [25], although the price is a change from polynomial to exponential scaling with system size. With the nodal constraint, the diffusion Monte Carlo approach typically obtains 90–95% of the correlation energy in an amount of time proportional to N_e^α where $\alpha = 2, 3$ depending on actual implementation and type of the system. In what follows we therefore assume that the fixed-node condition is enforced and therefore Φ_0 is the antisymmetric ground state for a given fixed-node boundary condition.

In actual calculations, we perform an importance-sampling transformation, where $G(\mathbf{R}', \mathbf{R}, \tau)$ is replaced by the importance sampled Green's function

$$\tilde{G}(\mathbf{R}', \mathbf{R}, \tau) = \Psi_T(\mathbf{R}')G(\mathbf{R}', \mathbf{R}, \tau)/\Psi_T(\mathbf{R}). \tag{4}$$

The diffusion part of the Green's function then becomes a diffusion-drift kernel

$$G_D(\mathbf{R}', \mathbf{R}, \tau) = \exp(-(\mathbf{R}' - \mathbf{R} - \tau\nabla ln\Psi_T(\mathbf{R}))^2/2\tau), \tag{5}$$

while the branching part is transformed into

$$G_B(\mathbf{R}', \mathbf{R}, \tau) = \exp[-\tau(E_L(\mathbf{R}) + E_L(\mathbf{R}') - 2E_0)/2], \tag{6}$$

where $E_L(\mathbf{R}) = [H\Psi_T(\mathbf{R})]/\Psi_T(\mathbf{R})$ is the local energy. Both statistical sampling and averaging efficiency are vastly improved since the 'force' $\nabla \ln \Psi_T(\mathbf{R})$ biases the walk to where the wave function is large, and the local energy $E_L(\mathbf{R})$ is much smoother than the potential energy. Assuming N is large, the path $\tilde{G}(\mathbf{R}_N, \mathbf{R}_{N-1}, \tau) \cdots \tilde{G}(\mathbf{R}_2, \mathbf{R}_1, \tau)\Psi_T^2(\mathbf{R}_1)$, samples the distribution $\Psi_T(\mathbf{R}_N)\Phi_0$ $(\mathbf{R}_N)$, which is called the mixed distribution. The ground state energy is obtained from the integral $\int \Psi_T\Phi_0 H\Psi_T/\Psi_T d\mathbf{R} = \int \Phi_0 H\Psi_T d\mathbf{R} = E_0$, since Φ_0 is an eigenstate of H within the nodal boundaries.

Diffusion Monte Carlo attains the mixed distribution by starting with a distribution of Ψ_T^2 and interpreting the action of the Green's function as a stochastic process with branching and elimination, eventually ending up with $\Psi_T\Phi_0$. It has the advantage that the $\tau \to \infty$ limit is easy to achieve, but the disadvantage of not having access to the pure distribution. A more subtle limitation is that the branching process spoils any imaginary-time data and can decrease the efficiency of the simulation if there is too much branching. Even with these limitations, in current implementations DMC is probably the most efficient way to obtain the fixed-node approximation to the ground state energy.

DMC is implemented almost identically to VMC, except that the time step is typically much smaller and each configuration accumulates a weight equal to $\exp[-\tau_{eff}(E_L(\mathbf{R}') + E_L(\mathbf{R}) - 2E_{ref})/2]$. Since we use an acceptance/rejection step, τ_{eff} is chosen somewhat smaller than τ as $\tau_{eff} = q\tau$, where q is the acceptance ratio. To control the fluctuations in the weights, we employ a branching algorithm that keeps the number of configurations constant, which improves the parallel load balancing properties of DMC. Every few steps we choose a set of configurations that have large weights (w_1) for branching. Each one of these configurations is matched with a smaller weight configuration (w_2) which is due for killing. The large weight configuration is branched and the small weight configuration is killed with probability $\frac{w_1}{w_1+w_2}$, with each copy gaining a weight of $\frac{w_1+w_2}{2}$. Otherwise, the small weight configuration is branched and the large weight configuration is killed, with the copies having the same weight as before. Configurations are then exchanged between nodes to keep the number of configurations on each node constant, and thus preserve high parallel efficiency. QWalk keeps track of two numbers: E_{ref} and E_0. E_{ref} is first set to the VMC average energy, and then to the energy of the last block. The energy that goes into the weights, E_0, is then calculated every few steps as

$$E_0 = E_{ref} - f \ln\left(\frac{\sum w_i}{N_{conf}}\right), \tag{7}$$

where N_{conf} is the number of configurations in the simulation and f is a positive constant that controls how quickly the weights return to the average value.

During the DMC calculation, the local energy will very occasionally fluctuate down significantly, causing the weight to increase too much. Of course, this is very much dependent on the quality of the trial function and the studied system. This can be fixed by limiting the growth of the weights. For fluctuations beyond ten standard deviations of the energy, we smoothly bring the effective time step to zero for the weights, which avoids the efficiency problem without introducing a noticeable error. However, if this weight limiting appears too often it can bias the final results and decreasing of the time step is necessary. The bias due to this cutoff goes to zero as the time step goes to zero or as the trial function approaches the exact one.

2.3. Reptation Monte Carlo

For operators that do not commute with the Hamiltonian, we use Reptation Monte Carlo [26] with the bounce algorithm [27]. We sample the path distribution

$$\Pi(s) = \Psi_T(\mathbf{R}_0)G(\mathbf{R}_0, \mathbf{R}_1, \tau)\cdots G(\mathbf{R}_{N-1}, \mathbf{R}_N, \tau)\Psi_T(\mathbf{R}_N), \tag{8}$$

where $s = [\mathbf{R}_0, \mathbf{R}_1, \ldots, \mathbf{R}_{N-1}, \mathbf{R}_N]$ is a projection path. In the limit as $\tau \to \infty$, $\exp(-H\tau)|\Psi_T\rangle \to |\Phi_0\rangle$, the ground state, and, since it is a Hermitian operator, the conjugate equation also holds. Therefore, the distribution of $\mathbf{R}_0$ and $\mathbf{R}_N$ is the mixed distribution $\Psi_T(\mathbf{R})\Phi_0(\mathbf{R})$, and the distribution of $\mathbf{R}_{N/2}$ is $\Phi_0^2(\mathbf{R}_{n/2})$ in the limit as $n \to \infty$. We evaluate the energy as $E_{RMC} = \langle [E_L(\mathbf{R}_0) + E_L(\mathbf{R}_N)]/2\rangle$ and operators non-commuting with H as $\mathcal{O}_{RMC} = \langle \mathcal{O}(\mathbf{R}_{N/2})\rangle$ Reptation Monte Carlo does not include branching, instead it uses an acceptance/rejection step. This is a tradeoff, allowing us to project only for a finite τ, since otherwise the probability distribution function is not normalizable, but allowing access to the pure distribution. The path can sometimes get stuck due to rejections even with the bounce algorithm, which is a well-known limit on the efficiency of the algorithm. In QWalk, RMC is approximately as efficient as DMC until the rejection rate begins to increase, making the path move very slowly. In our current implementation we empirically find that this slowdown occurs at approximately 150 electrons, although it also depends on the type and size of the system and on the quality of the trial wave function.

3. Hamiltonians and model potentials

The methods described in the previous section are very general approximations to the solution of the Schrödinger equation. They can easily apply not only to the common atoms and molecules, but also to important model systems such one-dimensional rings and the homogeneous electron gas. As long as the potential is local, it can be implemented into the QMC framework with a minimum of extra work. Various effective interactions are easy to introduce as well.

Also important for application to heavier nuclei are pseudopotentials, which replace the core electrons with a nonlocal potential

$$\hat{V}_{\mathrm{PSP}} = V_{local}(r) + \sum_{l=0}^{lmax} V_l(r)|l\rangle\langle l|, \tag{9}$$

where $|l\rangle\langle l|$ is the projector onto angular momentum l and r is the electron–pseudoion distance. While any pseudopotential of this form can be used, we use potentials which are bounded for $r \to 0$ unlike several pseudoptentials which contain $-Z_{eff}/r$ Coulomb term. These bounded pseudopotentials have been created specifically for QMC and are available in the literature [28–31], although more traditional Hartree–Fock or DFT pseudopotentials in the Troullier–Martins form work as well. The removal of the divergence of the potential energy near the nucleus increases the efficiency of the QMC calculation, often by orders of magnitude. The nonlocality can be treated exactly in VMC[32], and approximately[33] or variationally [34] in DMC/RMC approaches.

4. Forms of the wave function

The most commonly used trial wave function in QMC calculations is of the Slater–Jastrow product

$$\Psi_T = \Psi_A e^U, \tag{10}$$

where Ψ_A is the antisymmetric part while e^U is the Jastrow correlation factor (explained in the next subsection). The antisymmetric part is given as a single Slater determinant or a linear combination of Slater determinants

$$\Psi_A = \sum c_i D_i^{\uparrow} D_i^{\downarrow}, \tag{11}$$

where $D_i^{\uparrow(\downarrow)}$ is a determinant of the spin up (down) one-particle orbitals calculated in Hartree–Fock (HF) or density functional theory (DFT) approaches.

4.1. Jastrow factor

The Jastrow factor is written as e^U, where

$$U = \sum_{ilk} c_k^{en} a_k(r_{il}) + \sum_{ijk} c_k^{ee} b_k(r_{ij}) + \sum_{ijlklm} c_{klm}^{een}[a_k(r_{il})a_l(r_{jl}) + a_k(r_{jl})a_l(r_{il})]b_m(r_{ij}), \tag{12}$$

i,j are electron indexes, and I is a nuclear index. Both the coefficients and parameters within the basis functions can be optimized. In addition, the $\{c^{ee}\}$ and $\{c^{een}\}$ coefficients can be made spin-dependent. For the basis functions, we satisfy the exact electron–electron cusp conditions with the function $b(r) = cp(r/r_{\text{cut}})/(1+\gamma p(r/r_{\text{cut}}))$, where $p(y) = y - y^2 + y^3/3$, γ is the curvature, which is optimized, and c is the cusp (1/4 for like spins and 1/2 for unlike spins). Further correlation is added by including functions of the form $b_k(r) = a_k(r) = \frac{1-z(r/r_{\text{cut}})}{1+\beta_k z(r/r_{\text{cut}})}$ where $z(x) = x^2(6 - 8x + 3x^2)$ and β_k is an optimized parameter. These functions have several favorable properties, going smoothly to zero at a finite cutoff radius, and covering the entire functional space between 0 and r_{cut}. This allows the Jastrow factor to be very compact, typically requiring optimization of around 25 parameters while still coming close to saturating the functional form. While these are the standard basis functions, they can be replaced or augmented by any in the program by a simple change to the Jastrow input. The third term in Eq. (12), which sums over two electron indices and ionic indexes, can be expensive to evaluate for large systems and is sometimes excluded. A Jastrow factor with only the first two terms is called a two-body Jastrow, and with the *een* term included is called a three-body Jastrow.

One of the key advantages of QMC is that, since all the integrals are done by Monte Carlo, almost any ansatz can be used, as long as it is reasonably quick to evaluate. We generally begin with a Slater determinant or sum thereof, then multiply by a symmetric Jastrow correlation factor. This Jastrow factor improves the efficiency of DMC, but not the fixed-node approximation, since it does not change the nodes. It is quite remarkable that this simple form, even with only single term in the Slater determinant sum, proved to be rather accurate for whole range of systems and between 90% and 95% of the correlation energy is typically obtained with this trial wave function in the fixed-node DMC method.

The accuracy can be further improved by replacing the Slater determinant with the compact pfaffian or backflow functions, that allow electron correlation to change the nodes and thus improve the fixed-node DMC accuracy.

4.2. Pfaffian pairing wave function

Pairing wave functions with a Jastrow factor for molecules were first investigated by Casula and coworkers [35], who studied the constant number of particles projection of the BCS wave function. The general Jastrow-BCS pairing wave function can be expressed as $\Psi_{\text{BCS}} = e^U \det[\mathbf{\Phi}]$, where e^U is the Jastrow factor of above and the matrix $\mathbf{\Phi}_{ij} = \phi(\mathbf{ri}_i, \mathbf{r}_j)$ is the pairing function between opposite-spin electrons (the function is easily extended for $N_{up} \neq N_{down}$). This function contains the Slater determinant as a special case (for singlet spin state) when ϕ is written as the sum over the occupied single-particle orbitals: $\phi(r_i, r_j) = \sum_k^{N_e} \varphi_k(r_i)\varphi_k(r_j)$. We have also implemented the more general pfaffian [8] pairing wave function, which allows for singlet (unlike) spin pairing, triplet (like) spin pairing in addition to possible unpaired single-particle states. The general pfaffian pairing wave function Ψ_{PF} is written as the pfaffian of the antisymmetric matrix

$$\Phi_{PF} = \text{pf}\begin{bmatrix} \boldsymbol{\xi}^{\uparrow\uparrow} & \mathbf{\Phi}^{\uparrow\downarrow} & \boldsymbol{\varphi}^{\uparrow} \\ -\mathbf{\Phi}^{\uparrow\downarrow T} & \boldsymbol{\xi}^{\downarrow\downarrow} & \boldsymbol{\varphi}^{\downarrow} \\ -\boldsymbol{\varphi}^{\uparrow T} & -\boldsymbol{\varphi}^{\downarrow T} & 0 \end{bmatrix} e^U, \tag{13}$$

where $\mathbf{\Phi}^{\uparrow\downarrow}, \boldsymbol{\xi}^{\uparrow\uparrow(\downarrow\downarrow)}$ and $\boldsymbol{\varphi}^{\uparrow(\downarrow)}$ represent the following block matrices. The $\mathbf{\Phi}^{\uparrow\downarrow}$ is the singlet pairing matrix as in the above BCS wave function, the $\boldsymbol{\varphi}^{\uparrow(\downarrow)}$ includes additional unpaired one-particle orbitals for a spin-polarized system. Finally, the $\boldsymbol{\xi}^{\uparrow\uparrow(\downarrow\downarrow)}$ are antisymmetric triplet pairing matrices. The operation of the pfaffian ensures that the entire wave function is antisymmetric. The pfaffian wave function contains the BCS wave function as a special case without triplet pairing, and therefore contains the Slater determinant wave function as well. The general expansion for Φ is

$$\mathbf{\Phi}^{\uparrow\downarrow}(\mathbf{r}_1, \mathbf{r}_2) = \sum_{kl} c_{kl}\varphi_k^{\uparrow}(\mathbf{r}_1)\varphi_l^{\downarrow}(\mathbf{r}_2) \tag{14}$$

under the constraint that $c_{kl} = c_{lk}$.$\boldsymbol{\xi}$ is written in a very similar way

$$\boldsymbol{\xi}^{\uparrow\uparrow}(\mathbf{r}_1, \mathbf{r}_2) = \sum_{kl} d_{kl}^{\uparrow\uparrow}\varphi_k^{\uparrow}(\mathbf{r}_1)\varphi_l^{\uparrow}(\mathbf{r}_2) \tag{15}$$

under the constraint that $d_{kl}^{\uparrow\uparrow} = -d_{lk}^{\uparrow\uparrow}$ and similarly for the $\downarrow\downarrow$ channel. The sum extends over the space of occupied and virtual orbitals. All pairing functions as well as unpaired orbitals are fully optimizable within VMC. The extensions of pfaffian pairing wave function to linear combinations of pfaffians with one or many sets of different pairing functions are also possible. For more information about the performance and implementation of the pfaffian wave function, see Refs. [8,36].

4.3. Backflow correlated wave function

Another way to systematically improve the nodal structure of trial wave function is through the introduction of backflow transformation [37–45]. Given a trial wave function of form $\Psi_T(\mathbf{R}) = \Psi_A(\mathbf{R})e^U$, the nodal structure is completely defined by

the nodes of its antisymmetric part $\Psi_A(\mathbf{R})$. The backflow transformation replaces $\Psi_A(\mathbf{R})$ by $\Psi_A(\mathbf{X})$, where $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \ldots)$ are some quasi-coordinates dependent on all electron positions $\mathbf{R}$, such that overall antisymmetry is preserved. The nodes of $\Psi_A(\mathbf{X})$ can then differ from nodes of $\Psi_A(\mathbf{R})$ and improve the fixed-node approximation.

The quasi-coordinate of ith electron at position $\mathbf{r}_i$ is given as

$$\mathbf{x}_i = \mathbf{r}_i + \xi_i(\mathbf{R}) = \mathbf{r}_i + \xi_i^{en}(\mathbf{R}) + \xi_i^{ee}(\mathbf{R}) + \xi_i^{een}(\mathbf{R}), \tag{16}$$

where ξ_i is the ith electron's backflow displacement divided to the contributions from one-body (electron–nucleus), two-body (electron–electron) and three-body (electron–electron–nucleus) terms. They can be further expressed as

$$\xi_i^{en}(\mathbf{R}) = \sum_I \left[\sum_k c_k^{en} a_k(r_{iI})\right] \mathbf{r}_{iI}, \tag{17}$$

$$\xi_i^{ee}(\mathbf{R}) = \sum_{j\neq i} \left[\sum_k c_k^{ee} b_k(r_{ij})\right] \mathbf{r}_{ij}, \tag{18}$$

$$\xi_i^{een}(\mathbf{R}) = \sum_{I,j\neq i} \left[\sum_{klm} c_{klm}^{een} [a_k(r_{iI}) a_l(r_{jI}) + a_k(r_{jI}) a_l(r_{iI})] b_m(r_{ij})\right] \mathbf{r}_{ij} + \left[\sum_{klm} d_{klm}^{een} [a_k(r_{iI}) a_l(r_{jI}) + a_k(r_{jI}) a_l(r_{iI})] b_m(r_{ij})\right] \mathbf{r}_{iI}, \tag{19}$$

where $\mathbf{r}_{ij} = \mathbf{r}_i - \mathbf{r}_j$ and $\mathbf{r}_{iI} = \mathbf{r}_i - \mathbf{r}_I$. The terms in the large square brackets are identical to the previously introduced one, two and two three-body Jastrow terms from Eq. (12). The implementation of backflow transformation therefore takes great advantage of already existent Jastrow. The improvement in nodal structure and gains in correlation energies can be achieved by optimizing all the Jastrow parameters within backflow transformation. For more details about implementation and performance of backflow transformation in QWalk see Ref. [36].

5. Correlated sampling approach to small energy differences

Correlated sampling is a technique where one samples two very similar systems with the same sets of samples. The variance in the difference of two random variables A and B is given by $\mathrm{Var}(A-B) = \mathrm{Var}(A) + \mathrm{Var}(B) - 2\mathrm{Cov}(A,B)$, so if the fluctuations in A and B are highly correlated, the variance of the difference will be decreased significantly. For example, suppose we have generated a set of samples distributed according to some probability distribution $P_1(X)$. Averages are obtained as usual by calculating the integral $\langle O_1\rangle_{P_1} = \int P_1(X) O_1 dX$. Suppose we wish to find $\langle O_2\rangle_{P_2} - \langle O_1\rangle_{P_1}$. This can be written as

$$\int P_2(X) O_2 - P_1(X) O_1 dX = \int P_1(X) \left[\frac{P_2}{P_1} O_2 - O_1\right] dX. \tag{20}$$

Since the samples are distributed according $P_1(X)$ this integral is evaluated by averaging the following weighted difference:

$$\sum_i^M \left[\frac{w_i(X_i) O_2(X_i)}{\sum_j w_i(X_i)} - \frac{O_1(X_i)}{M}\right], \tag{21}$$

where $w_i(X_i) = P_2(X_i)/P_1(X_i)$.

We can thus optimize the wave function using the same set of sample points [46] to evaluate the differences in energy between separate parameter sets. Since the samples are then correlated (they are the same!), small energy differences can be determined with much greater precision than the total energy. However, the expectation value of the energy is not bounded by below for a finite number of samples (as it is for an infinite sample set), so there are numerical problems with simply optimizing the energy using this technique. However, there are many quantities other than energy that can be minimized to provide a good approximation to the ground state wave function. A classical example is the variance of the local energy given by

$$\sigma^2 = \frac{\int d\mathbf{R} \Psi_T^2(\mathbf{R}) (E_{loc} - \langle E_{loc}\rangle)^2}{\int d\mathbf{R} \Psi_T^2(\mathbf{R})}. \tag{22}$$

Since E_{loc} is a constant when $|\Psi_T\rangle = |\Phi_0\rangle$, the variance vanishes for an exact eigenstate [46]. There are several other possible functions that can be optimized, summarized in Table 1. In order to optimize the energy efficiently, one can use the same fixed sample set, but evaluate a low-variance estimator of the Hessian matrix [47] and use Newton's method, bypassing the lower-bound problem.

Beyond using a fixed set of sample points to optimize the wave function parameters, one can also perform on-the-fly correlated sampling between two similar systems. This may be a small change in the nuclear coordinates or the electric field. For VMC, this can be done exactly by setting the weights equal to $w(X) = \Psi_2^2(X)/\Psi_1^2(X)$. DMC and RMC both require some approximation to the Green's function to weight the secondary averages correctly. We use the approximation of Filippi and Umrigar [48], who discuss the subject in a greater detail.

Table 1
Optimization objective functions implemented.

Function	Minimized quantity
Variance	$\langle (E_L(\mathbf{R}) - E_{ref})^2 \rangle$
Energy	$\langle E_L(\mathbf{R}) \rangle$
Mixed	aEnergy $+ (1-a)$Variance, $0 < a < 1$
Absolute value	$\langle \lvert E_L(\mathbf{R}) - E_{ref} \rvert \rangle$
Lorentz	$\langle ln(1 + (E_L(\mathbf{R}) - E_{ref})^2/2) \rangle$

6. Organization and implementation

In Fig. 1, we provide a representation of the data flow for a QMC calculation. The common thread of all calculations is that one starts with a model Hamiltonian, defines a trial wave function to describe the solution to the eigenstate of the Hamiltonian, and then applies one or several QMC methods to either evaluate the properties of the trial wave function (VMC) or to approximately solve for the eigenstate given the nodes of the trial wave function (DMC and RMC). QWalk casts these mathematical objects directly into C++ base classes as listed in Table 2. Each of these base classes has a well-defined interface, with which each of the other classes work with. For example, the VMC method does not need to know the actual form of the wave function; only what its value and derivatives are. Thus, it remains agnostic and the user can plug in whatever wave function that can provide those quantities.

The code is written in a coarse-grained object-oriented approach, creating independent sections of code that are written efficiently using procedural programming techniques. It is extremely modular; almost every piece can be removed and replaced with another without modifying other sections of code. The modular structure also allows for partial rewrites of the code without worrying about other parts. In fact, each major module has been rewritten several times in this manner as we add new features and refactor the code. For the user, this structure shows itself in flexibility.

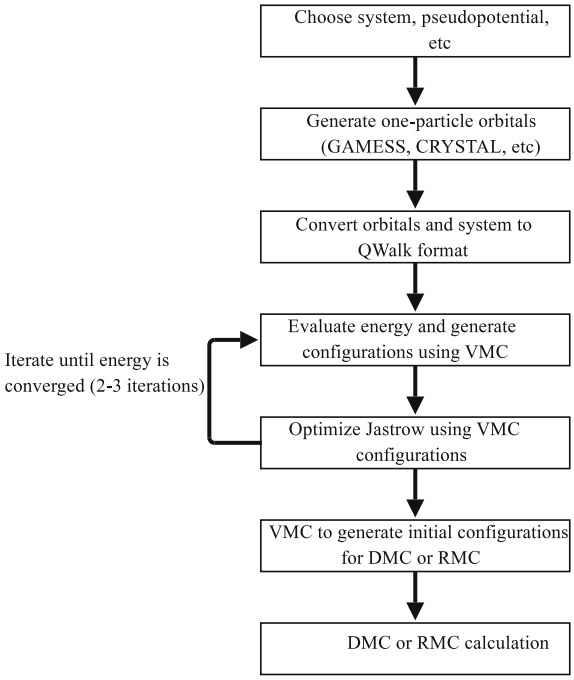

Fig. 1. Flow of a QMC calculation.

Table 2
The central objects of the code and their physical correspondents.

Class name	Mathematical object
System	Parameters and form of the Hamiltonian
Sample point	$\mathbf{R}$, the integration variables
Wave function type	Wave function ansatz
Wave function	$\Psi_T(\mathbf{R}), \nabla\Psi_T(\mathbf{R}), \nabla^2\Psi_T(\mathbf{R})$
Method	Monte Carlo algorithm

We will now provide a listing of the available instances for the major classes, along with some details of their implementation.

7. Base classes of objects

The features of QWalk are determined by the list of implemented objects. We list the main classes as well as currently available types within the classes.

7.1. System

This class determines the Hamiltonian of the system.
Currently implemented:

- Open boundary conditions (molecules).
- 3D periodic boundary conditions (solids). Slabs and 1D systems are treatable with large cells in the non-periodic direction.
- 1D ring with a magnetic field penetrating the center.
- Simple harmonic oscillator potential.
- Homogeneous electron gas.

7.2. Method

Perform some calculation using the System and Wave function defined.
Currently implemented:

- Variational Monte Carlo evaluation of expectation values using delayed rejection moves [19].
- Optimization of wave function parameters including:
  - coefficients of Slater determinants,
  - backflow parameters,
  - pairing functions,
  - Jastrow coefficients using one of several optimization methods including:
  - Correlated sampling optimization of the variance, mean absolute deviation, or Lorentzian of the local energy [46,49].
  - Correlated sampling optimization of the average energy using Newton's method and a modified Hessian [47].
- Diffusion Monte Carlo using either standard moves [50] or specially adapted moves for full-core atoms [20].
- Reptation Monte Carlo [26] using bounce moves [27].

7.3. Wave function

Functional form of the wave function. The properties of these wave functions can be evaluated directly using variational Monte Carlo, or they can be used as trial functions for diffusion Monte Carlo and reptation Monte Carlo. These functions can be multiplied or added together by using meta-wave functions.
Currently implemented:

- A single or multiple Slater determinants, with several different kernels for orbital evaluation.
- Jastrow correlation factor as described in Ref. [11].
- Pfaffians [8].
- Backflow wave functions [45].

7.4. Basis function

A function of a particle absolute position (for plane waves) or relative position from the basis function center location. At the time of this writing, there are ten written basis functions ranging from Gaussian functions to plane waves to Padé approximants. For brevity, we do not provide a complete list. These functions are used to represent the wave function as a linear or nonlinear combination of functions, evaluate pseudopotentials, and for other uses throughout the code.

7.5. Average generator

Evaluation of expectation values of given quantities (quantum mechanical operators). This can be done for any averaging method (variational, diffusion, or reptation Monte Carlo). Total energy and its components are averaged automatically. The user can further request any of the following:

- Dipole moment of a molecule.
- Two-body correlation function between like and unlike spins (e.g., between unlike spins $(N_{e\uparrow}N_{e\downarrow})^{-1}\sum_{ij}\delta(r-r_{ij})$).
- The static structure factor $N_e^{-1}\sum_i \exp(\mathbf{k}\cdot\mathbf{r}_i)\sum_i \exp(-\mathbf{k}\cdot\mathbf{r}_i)$.

7.6. Miscellaneous features

- Automated conversion from GAMESS [51], CRYSTAL [52], Gaussian [53], SIESTA [54]. Each of these converters prepares a starting QWalk setup with the atoms, pseudopotentials, trial function from the DFT/quantum chemistry code, and an appropriate starting Jastrow correlation function.
- Nonlocal pseudopotentials are supported, both in the localization approximation [33] and variationally in diffusion Monte Carlo [34].
- Correlated sampling between two similar systems using space warping in both VMC [55] and DMC/RMC using an approximation [48].
- A powerful averaging utility that supports error estimation for user-written average generators.

8. Examples

8.1. Example of implementation: VMC

The classes form a tree of successive abstractions (Fig. 2). At the top of the tree is the QMC method, VMC in this case. It works only in terms of the objects directly below it, which are the concepts of System, Wave function data, etc. (see Table 2). These in turn may have further abstractions below them, as we have shown for the wave function object. The highest wave function object is of type 'Multiply', which uses two wave function types to create a combined wave function. In this case, it multiplies a Slater determinant with a Jastrow correlation factor to form a Slater–Jastrow function. Since the wave functions are interchangeable, the Slater determinant can be replaced with any antisymmetric function, as well as the Jastrow factor. The type is listed along with the specific instance of that type in parenthesis. At each level, the part in parenthesis could be replaced with another module of the same type.

We present an implementation of the VMC algorithm as an example of how the code is organized (Fig. 3). For reasons of space, we do not write the function line-by-line, which includes monitoring variables, etc., but instead give a sketch of the algorithm. The VMC method works at the highest level of abstraction, only in terms of the wave function, system, and random dynamics. It does not care what kind of system, wave function, etc. are plugged in, only that they conform to the correct interfaces. In Appendix A, we give an example of how to create a new module.

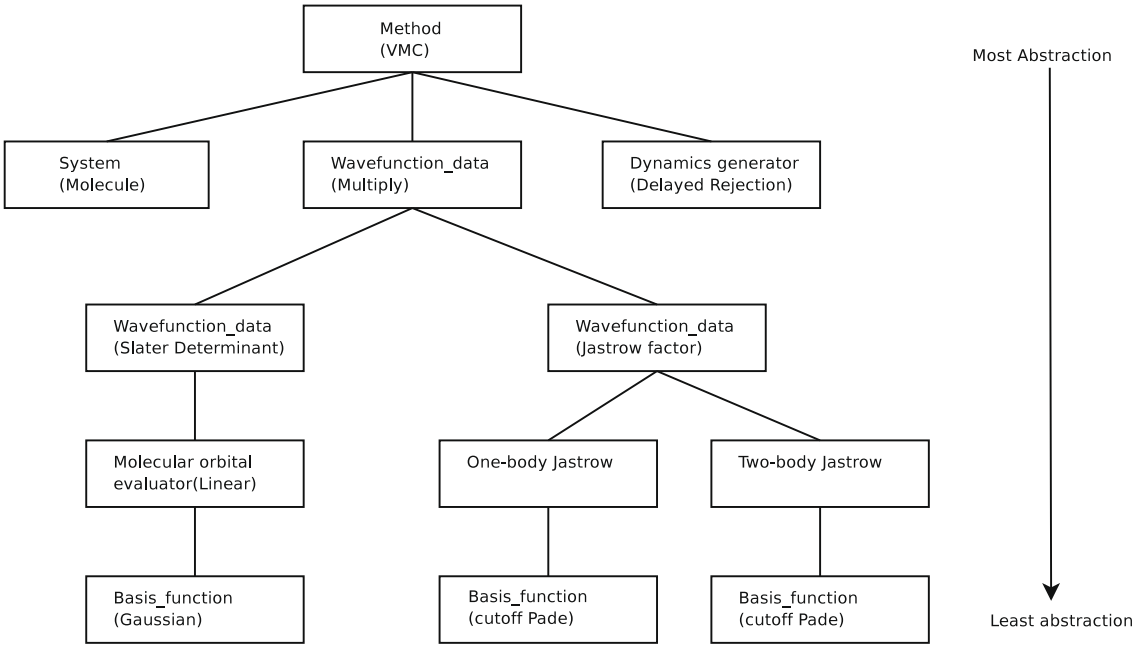

Fig. 2. Partial calculation structure for the VMC method on a molecule using a Slater–Jastrow wave function. Each vertical level represents a level of abstraction. The VMC method uses a dynamics generator to move electronic configurations using the Hamiltonian defined in the System object and the variational wave function defined in the Wavefunction_data object. Also shown is the object structure for the wave function calculation.

```
Vmc_method::run(vector <string> & vmc_section,
                vector <string> & system_section,
                vector <string> & wavefunction_section) {

  //Allocate the objects we will be working with
  System * sys=NULL;
  allocate(sys, system_section);

  Wavefunction_data * wfdata=NULL;
  allocate(wfdata, sys, wavefunction_section);

  Sample_point * sample=NULL;
  sys->generateSample(sample);
  Wavefunction * wf=NULL;
  wfdata->generateWavefunction(wf);

  //the Sample_point will tell the Wavefunction
  //when we move an electron
  sample->attachWavefunction(wf);
  sample->randomGuess();

  //This is the entire VMC algorithm
  for(int s=0; s< nsteps; s++) {
    for(int e=0; e < nelectrons; e++) {
       dynamics_generator->sample(e,timestep,wf,sample);
    } //end electron loop
    //gather averages
  } //end step loop

  //report final averages
```

Fig. 3. Simple VMC code.

8.2. Example calculation

To give a feeling for the flow of the program, we will go through a simple calculation. A schematic of the procedure is given in Fig. 1. The first two steps are to choose the system and use a code such as GAMESS or CRYSTAL to prepare the one-particle orbitals, which is done as usual for the code. The converter program included with QWalk then creates the `system`, `slater`, and `jastrow` files automatically, so all the user must do is use the include directive to use them. In Fig. 4, we evaluate the properties of the starting Slater wave function by creating 500 electron configurations, and then propagating them for 16 blocks, each of which consists of 10 Monte-Carlo steps with a time step of 1.0 a.u. The final set of configurations

```
#load the converted pseudo-nuclei, number of electrons
include sysfile

#load the Slater determinant of one-particle orbitals
trialfunc { include slaterfile }

#
method {  VMC
   nconfig 500            #number of configurations
   nblock 16              #averaging blocks
   nstep 10               #steps to take per block
   timestep 1.0           #timestep to use
   storeconfig configfile #save configurations to
                          #a file
}
```

Fig. 4. Example input file for VMC evaluation of properties. This corresponds to the fourth box in Fig. 1.

```
include sysfile

trialfunc { multiply
  wf1 { include slaterfile }
  wf2 { include jastrowfile }
}

method {  OPTIMIZE nconfig 500 iterations 30 readconfig configfile
}
```

Fig. 5. Example input file for optimization of variational parameters. This is the fifth box in Fig. 1.

```
include sysfile
trialfunc { include optfile.wfout }

method {  VMC nconfig 500 timestep 1.0 nstep 10  nblock 16
   readconfig configfile   storeconfig configfile
}

method { DMC nconfig 500 timestep 0.02 nstep 50 nblock 16
    readconfig configfile  storeconfig configfile
}
```

Fig. 6. Example input file for evaluation of properties of the correlated wave function, plus a DMC calculation. This corresponds to the sixth and seventh block in Fig. 1.

is then stored in `configfile`, and QWalk outputs the total energy and other properties that have been accumulated along the way.

We then wish to correlate the wave function by including the Jastrow factor (Fig. 5). The converter has already created a starting Jastrow wave function with all coefficients equal zero except the exact electron–electron cusps, which have values $1/2$ and $1/4$ as explained before. Therefore we request a Slater–Jastrow product form. The first factor is the Slater determinant that we used before while the second is the Jastrow created by the converter. We request optimization using a fixed set of configurations that we generated in the previous VMC run.

Finally, we wish to evaluate properties of the new correlated wave function using the VMC routine (Fig. 6). This input file is similar to the last, except that we include the output wave function from the optimization in the trialfunc section. Also in the example, we perform a DMC calculation immediately after the VMC calculation. Its input is nearly identical to that already discussed.

9. Performance and portability

It is clearly important for a program to be able to both run efficiently on many architectures and to be portable to many compilers. To this end, we have carefully kept the code in standard C++, with a minimum of external libraries. It compiles using the Intel C++ compiler, GNU Compiler Collection, Portland Group, IBM's xlC compiler, and Pathscale compilers on all systems which support those compilers. QWalk can optionally use BLAS linear algebra libraries and MPI for parallel operation. The program, however, will compile without these libraries, so the barrier to entry is quite low.

QWalk is designed to be memory efficient as much as possible. The amount of memory required scales as $\mathcal{O}(N_e^2)$, primarily from the distance matrix between electrons, the orbital coefficients, and the updates for the Jastrow factor. Using a Siesta trial function and a system of 256 Si atoms (1024 valence electrons), the program uses approximately 590 MB of memory per processor core, well within the capabilities of most systems.

The cost of a QMC calculation can be estimated as $T = T_s N_d \sigma^2/\epsilon^2$, where T is the total computer time, T_s is the time taken for one Monte Carlo step, N_d is the number of steps to decorrelate sequential samples, σ^2 is the variance of the desired quantity, and ϵ is the desired stochastic uncertainty. While the scaling is very favorable, the prefactor is large, approximately by a factor of 100 to 1000, depending on the type of the system, over that of DFT. It is informative to compare Coupled Cluster with single and double excitations (CCSD) and a triple-zeta basis to DMC (Table 3), since they often offer similar accuracy. While CCSD does not obtain as much of the correlation energy as DMC, it is much faster for small systems; however, DMC quickly overtakes CCSD at around 10 valence electrons due to superior scaling. There are of course many variables that affect the relative performance of the two methods, and we make no attempt to account for all of them; these calculations are just to give an idea of how one should choose between them.

Table 3
Computer time needed to calculate two simple systems using CCSD and DMC. The time is normalized to the CCSD calculation on methane. E_c is defined as the Hartree–Fock energy minus the calculated energy for the method. QWalk was used to perform the DMC calculation and GAMESS was used for the CCSD calculation. The DMC result uses a Slater–Jastrow wave function and includes all steps of the QMC calculation, including optimization of the Jastrow correlation factor. Pseudopotentials were used for both methods to remove the core electrons.

System	Method	E_c	Computer time
Methane	CCSD	0.193	1
Methane	DMC	0.240(1)	43.3
Ethane	CCSD	0.404	273.3
Ethane	DMC	0.438(2)	123.3

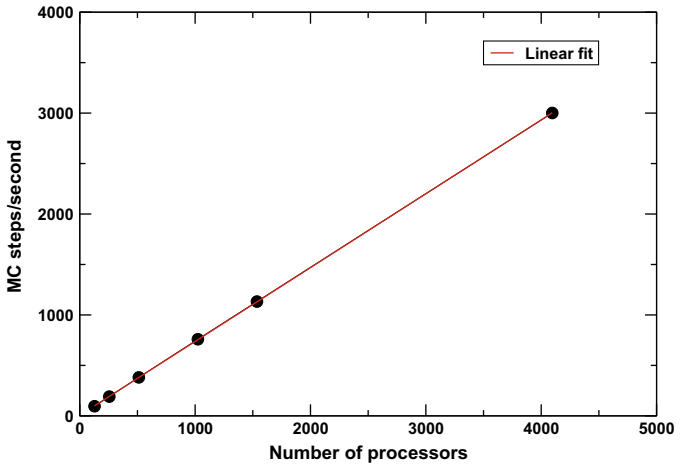

Fig. 7. Scaling of QWalk code performing a diffusion Monte Carlo calculation by distributing independent configurations across nodes. This calculation was performed on the Franklin supercomputer at NERSC.

Parallel performance is particularly important for accurate but computationally demanding methods such as quantum Monte Carlo. While systems with a few tens of electrons are currently feasible on a single processor, larger systems require parallel platforms. Fortunately, for small and moderately sized systems which fit into local memory, Monte Carlo methods are quite easy to parallelize using the replicated configuration strategy, where each processor owns one or more electronic configurations and the simulations proceed practically independently. Diffusion Monte Carlo requires some special care, however, because it contains a branching/killing step that can change the total number of configurations in the calculation. We have implemented an algorithm that keeps the total number of configurations constant while keeping the communication to a minimum. This leads to 99% efficient scaling up to 4096 processor cores, the maximum number that we have tested (Fig. 7).

We have presented the number of Monte Carlo steps per second; the actual efficiency for a calculation is slightly more complicated. All Monte Carlo methods require a warmup period, and increasing the number of replicated configurations increases the number of warmup periods that need to be thrown away (one period per configuration). Suppose that to obtain the stochastic uncertainities desired, we require N_p production Monte Carlo steps, the warmup period is N_w steps, and we are using N_c configurations (processor cores). The proportion of time spent in warmup (and thus wasted) is then $N_w N_c (N_p + N_w N_c)^{-1}$. For typical values of $N_p = 10^7$ and $N_w = 10^2$, this fraction is 0.09 for 10,000 processor cores, an overall parallel efficiency of 91%.

10. Development strategy

The source code is kept in a public Mercurial [56] (a distributed revision control system) repository. This provides a relatively easy path for incorporation of code from contributors, since it allows the following development strategy:

(1) The researcher downloads the source code repository with the history of all changes. The repository is now local to the developer's computer.
(2) The researcher then programs the new feature, and can keep up to date with the mainline development by pulling from the public repository. This does not require sharing of his or her revisions.

(3) Simultaneously with publishing a paper on the new results, the researcher can make his or her work immediately available publicly by sending a patch to the mainline maintainer. There are tools and a tutorial on the QWalk website that make this process very simple.

11. Conclusion

QWalk is a state of the art, usable, and extensible program for performing quantum Monte Carlo calculations on electronic systems. It is able to handle medium to large systems of electrons; the maximum size is mostly limited by the available computer time. It works in parallel very efficiently (Fig. 7), so it can take advantage of large clusters and multi-core computers. Since QWalk is available without charge and under the GNU Public license, it is hoped that it will help bring both development and use of quantum Monte Carlo methods to a wide audience. Due to its modular form it is straightforward to expand the QWalk's applicability to quantum systems beyond the electron–ion Hamiltonians in continuous space such as models of BEC/BCS condensates and other quantum models. It is easy to modify the system module to incorporate other types of interactions and to expand the one-particle and pair orbitals using the coded basis functions.

There are two major directions forward for development. The first is performance. While QWalk is efficiently coded, QMC methods are nevertheless very computationally demanding, and any performance improvements are welcome and desirable. Strategies for parallelization will soon need to be improved as well, since the distributed configuration strategy begins to lose efficiency at around 10,000 processors, particularly when large stochastic uncertainties are acceptable. Work is currently progressing on this front.

The second direction is the addition of new features. QMC has tremendous flexibility in the choice of wave function; it must only be reasonably fast to evaluate. QWalk provides the developer with the opportunity to easily write and try out new wave functions. A similar advantage also exists for the choice of Hamiltonian, expectation value, and Monte Carlo method. With a moderate amount of developer effort, QWalk has the opportunity to become a large collection of models and wave functions, where a wave function developed for one Hamiltonian can immediately be tried with another, with access to many QMC methods for analysis.

Acknowledgments

We would like to extend our thanks to Zack Helms, David Sulock, Prasenjit Sen, Ji-Woo Lee, Jindřich Kolorenč, Jeffrey Grossman, and Pavel Vagner for early testing of the code, and in the case of Zack Helms, Pavel Vagner, Jindřich Kalorenč and David Sulock, contributions to some parts. This has been a long-term project and funding has been provided by an NSF Graduate Research Fellowship for L. Wagner and further by EAR-0530110 and DMR-0804549 NSF grants, by DOE DE-FG05-08OR23336 Endstation grant, by INCITE, CNMS allocations at ORNL and by allocation at NCSA.

Appendix A. Adding a module

We provide an example of how to add a new module. In this case, we look at a Basis_function object, which has the fewest functions to fill in. All modules can be added in exactly the same way, differing only in what functions need to be defined. Suppose we wish to add a Gaussian function $\exp(-\alpha x^2)$. First we declare the new module in a header file:

```
class Gaussian_basis:public Basis_function {
public:
  //read the input
  void read(vector <string> & words);
  //the distance at which the function is zero double cutoff();
  //The work functions. Given a distance,
  //getVal returns the values
  //and getLap returns the values, first
  //derivatives with respect to x,y, and z,
  //and the Laplacian
  void getVal(Arrayl <doublevar> & r,
           Arrayl <doublevar> & vals);
  void getLap(Arrayl <doublevar> & r,
           Array2 <doublevar> & vals);
private:
  //put local variables here
  double alpha;3
  double cut;
};
```

Then we define the new functions:

```
Gaussian_basis::read(vector <string> & words) {
  unsigned int pos=0;
  if(!readvalue(words, pos,alpha,"ALPHA"))
    error("Need ALPHA in gaussian basis");
  const double m=1e-18;
  cut=sqrt(-log(m/alpha));
}
Gaussian_basis::cutoff() {
  return cut;
}
Gaussian_basis::getVal(Arrayl <doublevar> & r,
           Arrayl <doublevar> & vals) {
  //The basis function module can represent several
  //functions, which are put into the vals array.
  //Here we only have one.
  //r is an array of form r,r^2,x,y,z
  vals(0)=exp(-alpha*r(1));
}
//getLap is omitted for space reasons
```

The programmer then adds the source file to the Makefile and into a single if statement in the Basis_function.cpp file. The module can now be used anywhere another Basis_function can be used. All the modules follow this basic procedure, just with different functions.

Appendix B. Plane wave to LCAO converter

Gaussian basis sets have been used in quantum chemistry for years and have been developed to the point that there are well-defined sets which saturate the one-body Hilbert space surprisingly quickly. They are localized, which improves the scaling of QMC, and allow a very compact expression of the one-particle orbitals, so less basis functions need to be calculated. Overall, a gaussian representation can improve the performance of the QMC code by orders of magnitude over the plane-wave representation. We have developed a simple method to do this conversion that is fast and accurate. We start with the plane-wave representation of the kth orbital $\Phi_k(\vec{r}) = \sum_{\vec{G}} c_{k\vec{G}} e_{\vec{G}}(\vec{r})$, and wish to find the LCAO equivalent $\Phi_k^{\mathrm{LCAO}}(\vec{r}) = \sum_j a_{kj}\phi_j(\vec{r})$, where $e_{\vec{G}}$ is a plane-wave function and ϕ_j is a Gaussian function. Maximizing the overlap between Φ_k and Φ_k^{LCAO}, we obtain $Sa_k = Pc_k$, where $S_{ij} = \langle\phi_i|\phi_j\rangle$ and $P_{i\vec{G}} = \langle\phi_i|e_{\vec{G}}\rangle$. Then the Gaussian coefficients are given as $a_k = S^{-1}Pc_k$. All the overlap integrals are easily written in terms of two-center integrals for S, and P is easily evaluated in terms of a shifted Gaussian integral. The limiting part of the conversion is the calculation of the inverse of S, which can be done with fast LAPACK routines.

References

[1] W.M.C. Foulkes, L. Mitas, R.J. Needs, G. Rajagopal, Rev. Mod. Phys. 73 (2001) 33.
[2] J. Grossman, J. Chem. Phys. 117 (2002) 1434.
[3] P.R.B.L. Hammond, W.A. Lester Jr., Monte Carlo Methods in Ab Initio Quantum Chemistry: Quantum Monte Carlo for Molecules, World Scientific, 1994.
[4] A. Lüchow, J.B. Anderson, Ann. Rev. Phys. Chem. 51 (2000) 501–526.
[5] A. Aspuru-Guzik, W.A. Lester Jr., Handbook of Numerical Analysis, Special Volume: Computational Chemistry, vol. X, Elsevier, 2003. p. 485.
[6] D. Ceperley, L. Mitas, Adv. Chem. Phys. 93 (1996) 1.
[7] J. Grossman, L. Mitas, Phys. Rev. Lett. 94 (2005) 056403.
[8] M. Bajdich, L. Mitas, G. Drobny, L. Wagner, K. Schmidt, Phys. Rev. Lett. 96 (2006) 130201.
[9] M. Bajdich, L. Mitas, G. Drobny, L. Wagner, Phys. Rev. B 72 (2005) 075131.
[10] P. Vagner, M. Mosko, R. Nemeth, L. Wagner, L. Mitas, Physica E (2006) 350.
[11] L.K. Wagner, L. Mitas, J. Chem. Phys. 126 (3) (2007) 034105.
[12] M. Bajdich, L. Mitas, L.K. Wagner, K.E. Schmidt, Phys. Rev. B (Condens. Mat. Mater. Phys.) 77 (11) (2008) 115112.
[13] R.J. Needs, M.D. Towler, N.D. Drummond, P. López Ríos, CASINO version 2.3 User Manual, University of Cambridge, Cambridge, 2008.
[14] A. Aspuru-Guzik, R. Salomón-Ferrer, B. Austin, R. Perusquía-Flores, M.A. Griffin, R.A. Oliva, D. Skinner, D. Domin, W.A. Lester Jr., J. Comput. Chem. 26 (8) (2006) 856.
[15] J. Kim et al., Qmcpack, materials computation center. URL: <http://www.mcc.uiuc.edu>.
[16] C. Umrigar, C. Filippi, Cornell–Holland Ab-initio Materials Package. URL: <http://pages.physics.cornell.edu/*sim*cyrus/champ.html>.
[17] N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, E. Teller, J. Chem. Phys. 21 (1953) 1087–1092.
[18] W. Hastings, Biometrika 57 (1970) 97–109.
[19] D. Bressanini, G. Morosi, S. Tarasco, A. Mira, J. Chem. Phys. 121 (8) (2004) 3446–3451.
[20] C. Umrigar, M. Nightingale, K. Runge, J. Chem. Phys. 99 (1993) 2865.
[21] J.B. Anderson, J. Chem. Phys. 63 (4) (1975) 1499–1503.
[22] J.B. Anderson, J. Chem. Phys. 65 (10) (1976) 4121–4127.
[23] J.W. Moskowitz, K.E. Schmidt, M.A. Lee, M.H. Kalos, J. Chem. Phys. 77 (1) (1982) 349–355.

[24] P.J. Reynolds, D.M. Ceperley, B.J. Alder, W.A. Lester Jr., J. Chem. Phys. 77 (11) (1982) 5593–5603.
[25] D. Ceperley, B. Adler, Phys. Rev. Lett. 45 (1980) 566.
[26] S. Baroni, S. Moroni, Phys. Rev. Lett. 82 (1999) 4745.
[27] C. Pierleoni, D.M. Ceperley, ChemPhysChem 6 (2005) 1872.
[28] I. Ovcharenko, A. Aspuru-Guzik, W. Lester Jr., J. Chem. Phys. 114 (2001) 7790.
[29] Y. Lee, P. Kent, M. Towler, R. Needs, G. Rajagopal, Phys. Rev. B and private communication 62 (2000) 13347.
[30] J. Trail, R. Needs, J. Chem. Phys. 122 (2005) 174109.
[31] M. Burkatzki, C. Filippi, M. Dolg, J. Chem. Phys. 126 (23) (2007) 234105.
[32] S. Fahy, X.W. Wang, S.G. Louie, Phys. Rev. Lett. 61 (14) (1988) 1631–1634.
[33] L. Mitas, E. Shirley, D. Ceperley, J. Chem. Phys. 95 (1991) 3467.
[34] M. Casula, Phys. Rev. B (Condens. Mat. Mater. Phys.) 74 (16) (2006) 161102.
[35] M. Casula, C. Attaccalite, S. Sorella, J. Chem. Phys. 121 (2004) 7110.
[36] M. Bajdich, Ph.D. Thesis, North Carolina State University, 2007. URL: <http://altair.physics.ncsu.edu/bajdich/Phd_thesis_m_bajdich.pdf>.
[37] R.P. Feynman, M. Cohen, Phys. Rev. 102 (5) (1956) 1189–1204.
[38] K.E. Schmidt, M.A. Lee, M.H. Kalos, G.V. Chester, Phys. Rev. Lett. 47 (11) (1981) 807–810.
[39] R.M. Panoff, J. Carlson, Phys. Rev. Lett. 62 (10) (1989) 1130–1133.
[40] J.W. Moskowitz, K.E. Schmidt, J. Chem. Phys. 97 (5) (1992) 3382–3385.
[41] Y. Kwon, D.M. Ceperley, R.M. Martin, Phys. Rev. B 48 (16) (1993) 12037–12046.
[42] Y. Kwon, D.M. Ceperley, R.M. Martin, Phys. Rev. B 50 (3) (1994) 1684–1694.
[43] Y. Kwon, D.M. Ceperley, R.M. Martin, Phys. Rev. B 53 (11) (1996) 7376–7382.
[44] N.D. Drummond, P.L. Rios, A. Ma, J.R. Trail, G.G. Spink, M.D. Towler, R.J. Needs, J. Chem. Phys. 124 (22) (2006) 224104.
[45] P.L. Rios, A. Ma, N.D. Drummond, M.D. Towler, R.J. Needs, Phys. Rev. E (Stat., Nonlinear, Soft Mat. Phys.) 74 (6) (2006) 066701.
[46] C. Umrigar, K. Wilson, J. Wilkins, Phys. Rev. Lett. 60 (1988) 1719.
[47] C.J. Umrigar, C. Filippi, Phys. Rev. Lett. 94 (2005) 150201.
[48] C. Filippi, C.J. Umrigar, Phys. Rev. B 61 (2000) R16291.
[49] D. Bressanini, G. Morosi, M. Mella, J. Chem. Phys. 116 (13) (2002) 5345–5350.
[50] P. Reynolds, D. Ceperley, B. Alder, W. Lester Jr., J. Chem. Phys. 77 (1982) 5593.
[51] M.W. Schmidt, K.K. Baldridge, J.A. Boatz, S.T. Elbert, M.S. Gordon, J.H. Jensen, S. Koseki, N. Matsunaga, K.A. Nguyen, S.J. Su, T.L. Windus, M. Dupuis, J.A. Montgomery, J. Comput. Chem. 14 (1993) 1347.
[52] V. Saunders, R. Dovesi, C. Roetti, R. Orlando, C. Zicovich-Wilson, N. Harrison, K. Doll, B. Civalleri, I. Bush, P. DArco, M. Llunell, CRYSTAL 06 User's Manual, 2008.
[53] M.J. Frisch, G.W. Trucks, H.B. Schlegel, G.E. Scuseria, M.A. Robb, J.R. Cheeseman, J.A. Montgomery Jr., T. Vreven, K.N. Kudin, J.C. Burant, J.M. Millam, S.S. Iyengar, J. Tomasi, V. Barone, B. Mennucci, M. Cossi, G. Scalmani, N. Rega, G.A. Petersson, H. Nakatsuji, M. Hada, M. Ehara, K. Toyota, R. Fukuda, J. Hasegawa, M. Ishida, T. Nakajima, Y. Honda, O. Kitao, H. Nakai, M. Klene, X. Li, J.E. Knox, H.P. Hratchian, J.B. Cross, V. Bakken, C. Adamo, J. Jaramillo, R. Gomperts, R.E. Stratmann, O. Yazyev, A.J. Austin, R. Cammi, C. Pomelli, J.W. Ochterski, P.Y. Ayala, K. Morokuma, G.A. Voth, P. Salvador, J.J. Dannenberg, V.G. Zakrzewski, S. Dapprich, A.D. Daniels, M.C. Strain, O. Farkas, D.K. Malick, A.D. Rabuck, K. Raghavachari, J.B. Foresman, J.V. Ortiz, Q. Cui, A.G. Baboul, S. Clifford, J. Cioslowski, B.B. Stefanov, G. Liu, A. Liashenko, P. Piskorz, I. Komaromi, R.L. Martin, D.J. Fox, T. Keith, M.A. Al-Laham, C.Y. Peng, A. Nanayakkara, M. Challacombe, P.M.W. Gill, B. Johnson, W. Chen, M.W. Wong, C. Gonzalez, J.A. Pople, Gaussian 03, Revision C.02, Gaussian, Inc., Wallingford, CT, 2004.
[54] J.M. Soler, E. Artacho, J.D. Gale, A. García, J. Junquera, P. Ordejón, D. Sánchez-Portal, J. Phys.: Condens. Mat. 14 (11) (2002) 2745–2779.
[55] C.J. Umrigar, Int. J. Quant. Chem. Symp. 23 (1989) 217.
[56] Mercurial revision control software. URL: <http://www.selenic.com/mercurial>.