QWalk: A Quantum Monte Carlo Program for Electronic Structure

Lucas K. Wagner*, Michal Bajdich, Lubos Mitas

Center for High Performance Simulation and Department of Physics,
North Carolina State University, Raleigh, NC 27695.
(Dated: May 28, 2018)

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.

I. INTRODUCTION

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. In particular, many problems in electronic structure of atoms, molecules, clusters and solids require the ground and excited eigenstates of the electron-ion Born-Oppenheimer Hamiltonian

$$H = -\frac{1}{2} \sum_i \nabla_i^2 - \sum_{iI} \frac{Z_I}{r_{iI}} + \sum_{i>j} \frac{1}{r_{ij}}, \qquad (1)$$

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, etc.) and require treatment of quantum many-body effects more accurately. One can turn to post-Hartree-Fock methods based on sophisticated expansions of wave functions in one-particle basis sets. These methods can be made formally exact,

unfortunately, the computational cost is substantial and the most accurate approaches 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 offer 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. It 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, 2]. 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 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 algorithms in a fast and flexible code. QWalk has already been used in several publications [3, 4, 5, 6, 7, 8], and we would like to present a summary of its current capabilities.

II. METHOD

A. Variational Monte Carlo

The expectation value for an arbitrary operator $\mathcal{O}$ and a given trial variational 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 $\mathbf{R} = (\mathbf{r}_1, \mathbf{r}_2, ..., \mathbf{r}_{N_e})$ 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

---

*Current address: 366 Le Conte Hall #3700; Berkeley, CA 94720

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 algorithm. 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

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

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 wavefunction

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

A (hopefully) good approximation to the ground state is then the wavefunction with the set of parameters P that minimizes $E(P)$. The stochastic method of integration allows us to use explicitly correlated trial wave functions such as the Slater-Jastrow form, along with other functional forms as explained later. In fact, as long as the trial function and its derivatives can be evaluated quickly, any functional form can be used.

Within the program, this procedure is broken down into two parts: sampling Ψ_T^2 while evaluating energy and other properties, and optimizing the wave function. The first part, sampling Ψ_T^2, is carried out using the Metropolis-Hastings [9, 10] algorithm. We start with a point $\mathbf{R}$ in $3N_e$ dimensional space and generate a second point $\mathbf{R}'$ according to the transition probability $T(\mathbf{R}' \leftarrow \mathbf{R})$. T is a completely arbitrary function as long as $T(\mathbf{R}' \leftarrow \mathbf{R}) \neq 0 \Leftrightarrow T(\mathbf{R} \leftarrow \mathbf{R}') \neq 0$; that is, all moves are reversible. We then accept the move with probability

$$a = min\left(1, \frac{\Psi_T^2(\mathbf{R}')T(\mathbf{R}' \leftarrow \mathbf{R})}{\Psi_T^2(\mathbf{R})T(\mathbf{R} \leftarrow \mathbf{R}')}\right). \qquad (3)$$

After a few steps, the distribution converges to Ψ_T^2, and we continue making the moves until the statistical uncertainties are small enough. For atoms with effective core potentials, we use the moves as outlined in Ref. [1], modified with a delayed rejection step similar to Ref. [11], although developed independently, and for full-core calculations, we use the accelerated Metropolis method from Ref. [12]. The total energy and its components are evaluated, as well as other properties.

We then optimize the wave function using a fixed set of sample points. Since the samples are then correlated,

small energy differences can be determined with much greater precision than the total energy. There are many quantities other than energy that, upon being minimized, will provide a good approximation to the ground state wave function. One important one is the variance of the local energy; that is

$$\sigma^2 = \frac{\int d\mathbf{R}\Psi_T^2(\mathbf{R})(E_{loc} - \langle E_{loc}\rangle)^2}{\int d\mathbf{R}\Psi_T^2(\mathbf{R})}. \qquad (4)$$

Since E_{loc} is a constant when $|\Psi_T\rangle = |\Phi_0\rangle$, the variance will go to zero for an exact eigenstate. There are several other possible functions, listed in Sec. IV B, but variance and energy are the most common quantities to minimize.

B. Projector 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. 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 $\mathbf{R}$-representation as

$$G(\mathbf{R}', \mathbf{R}, \tau) \simeq exp(-(\mathbf{R}' - \mathbf{R})^2/2\tau)$$
$$\times exp(-\frac{\tau}{2}(V(\mathbf{R}) + V(\mathbf{R}') - 2E_0))$$

which can be interpreted as a dynamic diffusion kernel $G_D(\mathbf{R}', \mathbf{R}, \tau) = exp(-(\mathbf{R}' - \mathbf{R})^2/2\tau)$ times a branching kernel $G_B(\mathbf{R}', \mathbf{R}, \tau) = exp(-\frac{1}{2}(V(\mathbf{R}) + V(\mathbf{R}') - 2E_0))$.

The basic idea of projector Monte Carlo is to sample a path $G(\mathbf{R}_N, \mathbf{R}_{N-1}, \tau)...G(\mathbf{R}_2, \mathbf{R}_1, \tau)\Psi_T(\mathbf{R}_1)$. 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 [13, 14, 15, 16], 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 [17], although the price is a change from polynomial to exponential scaling with system size. With the nodal constraint, the projector 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

$$\tilde{G}(\mathbf{R}', \mathbf{R}, \tau) = \Psi_T(\mathbf{R}')G(\mathbf{R}', \mathbf{R}, \tau)/\Psi_T(\mathbf{R}) \qquad (5)$$

The dynamic part of the Green's function then becomes

$$G_D(\mathbf{R}', \mathbf{R}, \tau) = exp(-(\mathbf{R}' - \mathbf{R} - \tau \nabla ln\Psi_T(\mathbf{R}))^2/2\tau) \quad (6)$$

and the branching part becomes

$$G_B(\mathbf{R}', \mathbf{R}, \tau) = exp(-\frac{1}{2}(E_L(\mathbf{R}) + E_L(\mathbf{R}') - 2E_0)), \quad (7)$$

both of which are much better-behaved stochastically, since the 'force' $\nabla ln\Psi_T(\mathbf{R})$ biases the walk to where the wavefunction is large, and the local energy $E_L(\mathbf{R})$ is much smoother than the potential energy. Then if we generate the path $\tilde{G}(\mathbf{R}_N, \mathbf{R}_{N-1}, \tau)...\tilde{G}(\mathbf{R}_2, \mathbf{R}_1, \tau)\Psi_T^2(\mathbf{R}_1)$, for large enough N, the distribution of $\mathbf{R}_N$ is $\Psi_T(\mathbf{R}_N)\Phi_0(\mathbf{R}_N)$, which is called the mixed distribution. The ground state energy is obtainable by evaluating 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. In QWalk, two versions of the projector method are implemented: Diffusion Monte Carlo, which has the advantage that the large N limit is easily obtained, and Reptation Monte Carlo, which makes the 'pure' distribution Φ_0^2 available.

Diffusion Monte Carlo has been discussed by many authors [1, 12], and suffice it to say that it 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 killing and branching, 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.

For quantities that do not commute with the Hamiltonian, we use Reptation Monte Carlo [18] with the bounce algorithm [19]. We sample the path distribution

$$\Pi(s) = \Psi_T(\mathbf{R}_0)G(\mathbf{R}_0, R_1, \tau)\ldots G(\mathbf{R}_{n-1}, \mathbf{R}_n, \tau)\Psi_T(\mathbf{R}_n) \quad (8)$$

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 $O_{RMC} = \langle O(\mathbf{R}_{N/2})\rangle$ 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

TABLE I: The central objects of the code and their physical correspondents

Module 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	$\Psi_T(\mathbf{R})$, $\nabla\Psi_T(\mathbf{R})$, $\nabla^2\Psi_T(\mathbf{R})$
Dynamics generator	Metropolis trial move (Green's function)

probability distribution function is not normalizable, but allowing access to the pure distribution and imaginary time correlations. 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 quality of the trial wave function.

III. ORGANIZATION AND IMPLEMENTATION

The code is written in a combination of object-oriented and procedural techniques. The object-oriented approach is coarse-grained, creating independent sections of code that are written efficiently in a procedural fashion. It is extremely modular; almost every piece can be removed and replaced with another. A contributor of a module only has to change one line in the main code to allow use of a new module. This allows for flexibility while keeping the code base relatively simple and separable. 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.

The modules form a tree of successive abstractions (Fig. 1). 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 I). These in turn may have further abstractions below them, as we've 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 pluggable, the Slater determinant can be replaced with any antisymmetric function, as well as the Jastrow factor. The type is listed along with the spe-
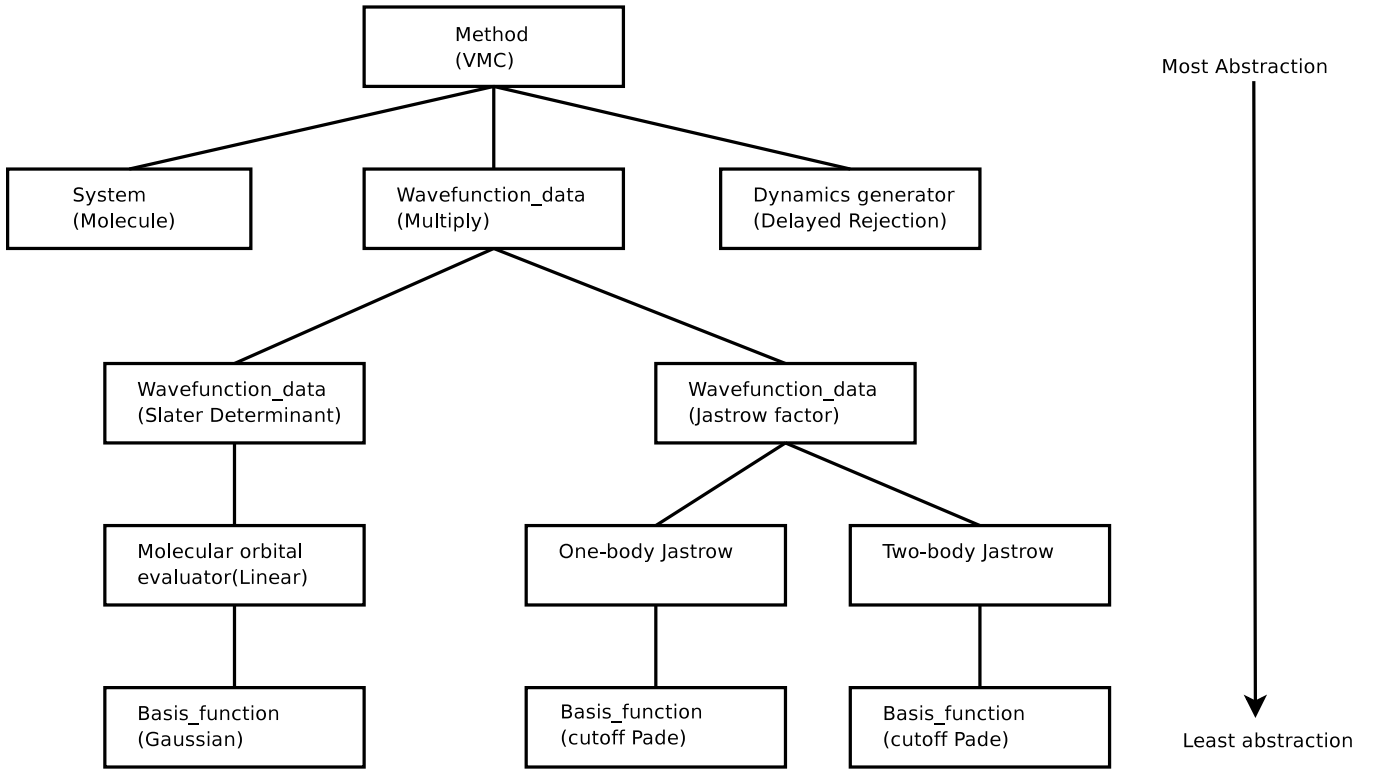
FIG. 1: Calculation structure for the VMC method on a molecule using a Slater-Jastrow wave function.

cific instant 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. 2). 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.

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

IV. METHODS

A. Variational Monte Carlo

The VMC module implements the Metropolis method to sample the probability density $\Psi_T^2(\mathbf{R})$. It has been described in Sec. II A to some detail–the method is more or less a direct translation. Beyond the basic algorithm, it implements correlated sampling as explained in Sec. IV E for small energy differences between very similar systems.

```
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. 2: Simple VMC code

TABLE II: 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	$\langle|E_L(\mathbf{R}) - E_{ref}|\rangle$
Lorentz	$\langle ln(1 + (E_L(\mathbf{R}) - E_{ref})^2/2)\rangle$

B.   Optimization of Wave Functions

We have implemented three different methods for optimization. All methods are capable of optimizing the first three objective functions from Table II. In principle, any objective function from this table will obtain the correct ground state with an infinitely flexible function, but may obtain different minima for incomplete wave functions and some are easier to optimize than others. The first (OPTIMIZE) is based on Umrigar et al.'s [20] variance optimization. The method minimizes the objective function on a set of fixed configurations from VMC using a conjugate gradient technique, usually not reweighting the averages as the wave function changes. Optimizing the energy using OPTIMIZE is quite expensive, because it requires many configurations to evaluate an unbiased estimate of the energy derivative.

The next two are based on Umrigar and Filippi's Newton optimization [21] method. OPTIMIZE2 also uses a fixed set of configurations, but instead of evaluating only the first derivatives of the objective function, as conjugate gradients do, it uses a low-variance estimator for the Hessian matrix and Newton's method to find the zeros of the first derivatives. OPTIMIZE2 is able to produce better wave functions with lower energies than OPTIMIZE by directly optimizing the energy even for very large systems (we have applied it for up to 320 electrons) while costing slightly more.

Finally, NEWTON_OPT uses a fixed set of configurations to calculate the same low-variance estimator for the Hessian matrix only at the single step, then evaluates the optimal length of the optimization step using VMC correlated sampling [21]. The later step enables us to decrease the number of iterations needed to converge. Further, this method is able to find the very lowest energy wave function, since the configurations are regenerated at every optimization step. However, the expense of one iteration in NEWTON_OPT is larger than for other two methods due to the additional cost associated with VMC and VMC correlated sampling.

C.   Diffusion Monte Carlo

DMC is implemented almost identically to VMC, except that the time step is typically much smaller and each walker accumulates a weight equal to $\exp(-\frac{\tau_{eff}}{2}(E_L(\mathbf{R}') + E_L(\mathbf{R}) - 2E_{ref}))$. Since we use an acceptance/rejection step, τ_{eff} is chosen somewhat smaller than τ as $\tau_{eff} = p\tau$, where p is the acceptance ratio. To control the fluctuations in the weights, we employ a constant-walker branching algorithm, which improves the parallel load balancing properties of DMC. Every few steps we choose a set of walkers that have large weights (w_1) for branching. Each one of these walkers is matched with a smaller weight walker (w_2) which is due for killing. The large weight walker is branched and the small weight walker 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 walker is branched and the large weight walker is killed, with the copies having the same weight as before. Walkers are then exchanged between nodes to keep the number of walkers 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

$$E_0 = E_{ref} - log\left(\frac{\sum w_i}{N_{conf}}\right), \qquad (9)$$

where N_{conf} is the number of sample points (configurations) in the simulation.

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 cutting off 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. 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.

D.   Reptation Monte Carlo

The fluctuations in the local energy part of the Green's function can cause the path in RMC to get stuck, so we cut off the effective time step in the same way as in DMC. The branching part of the Green's function is otherwise quite smooth. We use the same dynamic Green's function as we do in DMC (either a standard Metropolis rejection step or the UNR [12] algorithm), so we accept/reject based only on the branching part of the Green's function. We use the bounce algorithm [19], which improves the efficiency by allowing the path to explore the many-body phase space much more quickly.

E. Correlated Sampling

Correlated sampling is a technique where one samples two very similar systems with the same sets of samples. The variance in the difference will decrease as $Var(X - Y) = Var(X) + Var(Y) - 2Cov(X, Y)$, so for perfectly correlated sampling, the variance will be zero for the difference. In QWalk, this is handled by performing a primary walk that samples some probability distribution $P_1(X)$. Averages are obtained as usual by calculating the integral $\langle O_1 \rangle = \int P_1(X) O_1 dX$. Suppose we wish to find $\langle O_2 - O_1 \rangle$. It can be written as

$$\int P_2(X)O_2 - P_1(X)O_1 dX = \int P_1(X)\left[\frac{P_2}{P_1}O_2 - O_1\right]dX. \tag{10}$$

Since we are sampling $P_1(X)$, in the Monte Carlo averaging, this integral is evaluated by averaging the weighted difference over sample points:

$$\sum_i^N \left[\frac{w_i(X_i)O_2(X_i)}{\sum_j w_j(X_i)} - \frac{O_1(X_i)}{N}\right] \tag{11}$$

The difference in the methods is only in how they determine the weights.

VMC, DMC and RMC all support correlated sampling between arbitrary systems. In VMC, the weights are $w(X) = \frac{\Psi_2^2(X)}{\Psi_1^2(X)}$, which is an exact relationship. DMC and RMC both require some approximation to the Green's function to weight the secondary averages properly. In both, we use the approximation of Filippi and Umrigar [22], who discuss the subject in a greater detail.

V. SYSTEMS

A. Boundary Conditions

Most systems of interest are treatable either by open boundary conditions or periodic boundary conditions. Adding new boundary conditions is also quite simple. Molecules with arbitrary atoms, charge, spin state, and with finite electric field are supported. In 3D periodic systems, the calculation can be done at any real k-point, allowing k-point integrations. In many-body simulations, there is an additional finite size approximation due to the Coulomb interaction between image electrons. We correct this as $\delta E = \frac{c}{r_s}$, where the r_s is that of the homogeneous electron gas and c has been empirically fitted to 0.36 Hartrees. We have found this correction to function about as well as other attempts to correct the finite size error [23, 24]. The code has been used on systems with up to 135 atoms and 1080 electrons; the limiting factor is the amount of computer time needed to reduce the stochastic uncertainties.

B. Pseudopotentials

QWalk accepts pseudopotentials as an expansion of nonlocal angular momentum operators:

$$\hat{V}_{ECP} = V_{local}(\mathbf{R}) + \sum_{l=0}^{lmax} V_l(\mathbf{R})|l\rangle\langle l| \tag{12}$$

for arbitrary maximum angular moment. V_l is a basis function object that is typically a spline interpolation of a grid or a sum of Gaussian functions. While any pseudopotential of this form can be used, we use soft potentials in which the $\frac{Z}{r}$ divergence has been removed from the nuclei-electron interaction. These potentials have been created specifically for QMC and are available in the literature [25, 26, 27, 28], although more traditional Hartree-Fock or DFT pseudopotentials in the Troullier-Martins form work as well.

VI. FORMS OF THE WAVE FUNCTION

For chemical problems, the first-order trial function is usually written as a single Slater determinant of Hartree-Fock or Density Functional Theory orbitals multiplied by a correlation factor (known as a Jastrow factor) which is optimized in Variational Monte Carlo. Between 90% and 95% of the correlation energy is typically obtained with this trial wave function in Diffusion Monte Carlo.

One of the attractions 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. QWalk's modular structure makes adding new wave function forms as simple as coding one-electron updates of the function value and derivatives, and adding one line to the main code to support loading of the module. We have implemented several forms of wave functions, which the user can combine. For example, to create the Slater-Jastrow wave function, the user first asks for a multiply object, which contains two wave function objects. The user then fills in a Slater determinant object and a Jastrow object. For a Pfaffian-Jastrow wave function, the user replaces the Slater determinant input with the Pfaffian input. Obviously, it is up to the user to make sure that the total wave function is properly antisymmetric and represents the problem correctly.

A. Slater Determinant(s)

This is the standard sum of Slater determinants, written as $\Psi_T = \sum_i c_i D_i^\uparrow D_i^\downarrow$, where $D_i^{\uparrow(\downarrow)}$ is a determinant of the spin up (down) one-particle orbitals. The weights of the determinants $\{c_i\}$ are optionally optimizable within VMC.

B. Jastrow Factor

The Jastrow factor is written as e^U, where

$$
U = \sum_{iIk} c_k^{ei} a_k(r_{iI}) + \sum_{ijk} c_k^{ee} b_k(r_{ij})
$$
$$
+ \sum_{ijIklm} c_{klm}^{eei}[a_k(r_{iI})a_l(r_{jI}) + a_k(r_{jI})a_l(r_{iI})]b_m(r_{ij}),
$$
$$
(13)
$$

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 $\{c^{eei}\}$ coefficients can be made spin-dependent. For the basis functions, we satisfy the exact electron-electron cusp conditions with the function $b(r) = cp(r/rcut)/(1 + \gamma p(r/rcut))$, where $p(z) = z - z^2 + z^3/3$, 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 $b_k(r) = a_k(r) = \frac{1-zpp(r/rcut)}{1+\beta zpp(r/rcut)}$ where $zpp(x) = x^2(6 - 8x + 3x^2)$ and β 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 $rcut$. 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. (13), which sums over two electron indexes 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 eei term included is called a three-body Jastrow.

C. Pfaffian Pairing Wave Function

Pairing wave functions with a Jastrow factor for molecules were first investigated by Casula and coworkers [29], who studied the constant number of particles projection of the BCS wave function. The general Jastrow-BCS pairing wave function can be expressed as $\Psi_T = e^U \det[\mathbf{\Phi}]$, where e^U is the Jastrow factor of above and the matrix $\mathbf{\Phi}_{ij} = \phi(r_i, r_j)$ 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 implemented the Pfaffian [4] pairing wave function, which allows not only unlike-spin pairing, as the canonical projection of the BCS wave function does, but also allows like-spin pairing. The general Pfaffian pairing wave function Ψ_{PF} is written as the Pfaffian of the antisymmetric matrix

$$
\Phi_{PF} = \mathrm{pf}
\begin{bmatrix}
\boldsymbol{\xi}^{\uparrow\uparrow} & \mathbf{\Phi}^{\uparrow\downarrow} & \boldsymbol{\varphi}^{\uparrow} \\
-\mathbf{\Phi}^{\uparrow\downarrow T} & \boldsymbol{\xi}^{\downarrow\downarrow} & \boldsymbol{\varphi}^{\downarrow} \\
-\boldsymbol{\varphi}^{\uparrow T} & -\boldsymbol{\varphi}^{\downarrow T} & 0
\end{bmatrix},
\qquad (14)
$$

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 from 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

$$
\Phi^{\uparrow\downarrow(\mathbf{r_1},\mathbf{r_2})} = \sum_{kl} c_{kl}\varphi_k^{\uparrow}(\mathbf{r}_1)\varphi_l^{\downarrow}(\mathbf{r}_2) \qquad (15)
$$

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

$$
\boldsymbol{\xi}^{\uparrow\uparrow(\downarrow\downarrow)}(\mathbf{r}_1, \mathbf{r}_2) = \sum_{kl} d_{kl}^{\uparrow\uparrow(\downarrow\downarrow)}\varphi_k^{\uparrow(\downarrow)}(\mathbf{r}_1)\varphi_l^{\uparrow(\downarrow)}(\mathbf{r}_2) \qquad (16)
$$

under the constraint that $d_{kl}^{\uparrow\uparrow(\downarrow\downarrow)} = -d_{lk}^{\uparrow\uparrow(\downarrow\downarrow)}$. The sum extends over the space of occupied and virtual orbitals. All pairing functions as well as unpaired orbitals are fully optimizable within VMC method. The extensions of Pfaffian pairing wave function to linear combinations of Pfaffians with one or many sets of different pairing functions are also fully implemented. For more information about the performance and implementation of the Pfaffian wave function, see Refs. [4, 30].

D. Backflow Correlated Wave Function

Another way to systematically improve the nodal structure of trial wave function is through the introduction of backflow transformation [31, 32, 33, 34, 35, 36, 37, 38, 39]. Given a trial wave function of form $\Psi_T(\mathbf{R}) = \Psi_A(\mathbf{R}) \times \exp[U(\mathbf{R})]$, 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 the 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 QWalk implementation of the backflow transformation into Slater and Pfaffian wave functions closely follows the approach in Refs. [36, 39]. The quasi-coordinate of ith electron at position $\mathbf{r}_i$ is given as

$$
\mathbf{x}_i = \mathbf{r}_i + \boldsymbol{\xi}_i(\mathbf{R})
$$
$$
= \mathbf{r}_i + \boldsymbol{\xi}_i^{en}(\mathbf{R}) + \boldsymbol{\xi}_i^{ee}(\mathbf{R}) + \boldsymbol{\xi}_i^{een}(\mathbf{R}), \qquad (17)
$$

where $\boldsymbol{\xi}_i$ 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

$$\boldsymbol{\xi}_i^{en}(\mathbf{R}) = \sum_I \left[\sum_k c_k^{ei} a_k(r_{iI}) \right] \mathbf{r}_{iI} \tag{18}$$

$$\boldsymbol{\xi}_i^{ee}(\mathbf{R}) = \sum_{j \neq i} \left[\sum_k c_k^{ee} b_k(r_{ij}) \right] \mathbf{r}_{ij} \tag{19}$$

$$\boldsymbol{\xi}_i^{een}(\mathbf{R}) =$$
$$\sum_{I, j \neq i} \left[\sum_{klm} c_{klm}^{eei} \left[a_{k,r_{iI}} a_{l,r_{jI}} + a_{k,r_{jI}} a_{l,r_{iI}} \right] b_{m,r_{ij}} \right] \mathbf{r}_{ij}$$
$$+ \left[\sum_{klm} d_{klm}^{eei} \left[a_{k,r_{iI}} a_{l,r_{jI}} + a_{k,r_{jI}} a_{l,r_{iI}} \right] b_{m,r_{ij}} \right] \mathbf{r}_{iI}, \tag{20}$$

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 our familiar one, two and two three-body Jastrow terms from Eq. (13). 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. [30].

VII.   ONE-PARTICLE ORBITAL EVALUATION

We provide two major ways of evaluating the one-particle orbitals, the most expensive part of the QMC calculation. For a single electron, this is the problem of finding $\vec{m} = M_{orb}\vec{b}$, where $\vec{m}$ is a vector of the values of each orbital, M_{orb} is the orbital coefficient matrix, and $\vec{b}$ is the vector of basis functions. The first (CUTOFF_MO) is a linear scaling technique, which, for localized orbitals and large enough systems, will take $O(N)$ time to evaluate all orbitals for all electrons. For each basis function, it creates a list of orbitals for which the coefficient is above a cutoff. This is done at the beginning of the calculation. Then, given an electron position, it loops over only the basis functions within range of the electron, and then only the orbitals contributed to by the basis function. These are both $O(1)$ cost for large enough systems, so all the orbitals for each electron is evaluated in $O(1)$ time, giving $O(N)$ scaling.

The second method (BLAS_MO) is slightly simpler. While it scales in principle as $O(N^2)$, it can be faster than CUTOFF_MO in medium-sized systems and certain types of computers that have very fast BLAS routines, such as Itaniums. Given an electron position, it loops through the basis functions within range of the electron,
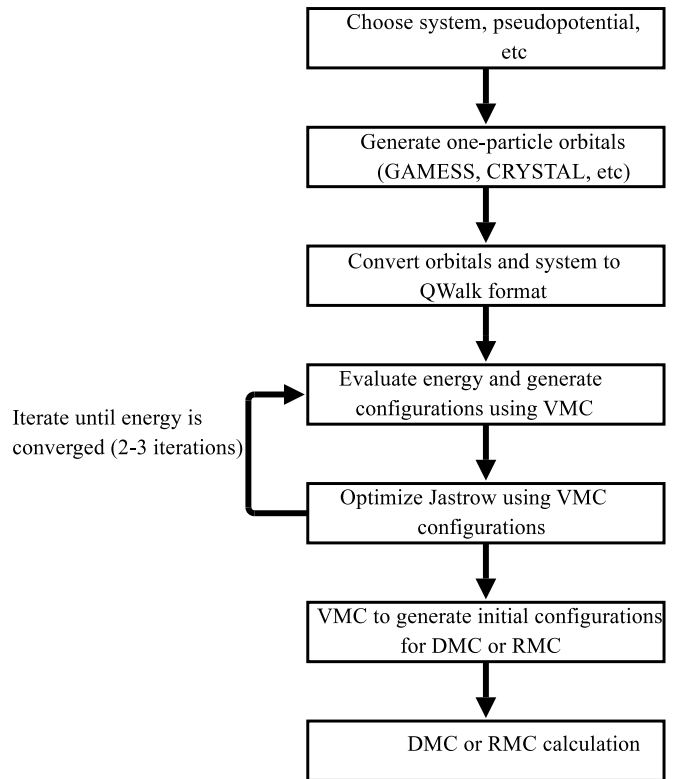

FIG. 3: Flow of a QMC calculation

and adds to each molecular orbital the coefficient times the value of that basis function using fast BLAS routines.

VIII.   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. 3. The first two steps are to choose the system and use a one-particle 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 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 obtain some correlation energy by adding the Jastrow factor (Fig. 5). The converter has already created a null Jastrow wave function, so we request a Slater-Jastrow wave function. The first wave function is the Slater determinant that we used before, and the second is the Jastrow created by the converter. We re-

```
#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. 3.

```
include sysfile

trialfunc {
 #a meta wave function that
 #multiplies two wave functions
  multiply
  wf1 { include slaterfile }

   #Jastrow correlation factor (created by converter)
  wf2 { include jastrowfile }
}

method {  OPTIMIZE
   nconfig 500
   iterations 30

   #read in the configurations from
   #the previous VMC run
   readconfig configfile
}
```

FIG. 5: Example input file for optimization of variational parameters. This is the fifth box in Fig. 3.

quest optimization using a fixed set of walkers 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 is the same as the last, except that we include the output wave function from the optimization in the trial-func section. Also in the example, we perform a DMC calculation immediately after the VMC calculation. Its input is nearly identical to that already discussed.

```
include sysfile

#load the wavefunction file generated by OPTIMIZE
trialfunc { include optfile.wfout }

#Same as above
method {  VMC
   nconfig 500
   timestep 1.0
   nstep 10
   nblock 16
   readconfig configfile
   storeconfig configfile
}

#perform DMC

method { DMC
   nconfig 500
   timestep 0.02     #smaller timestep
   nstep 50          #more steps per block
   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. 3.

IX. OTHER UTILITIES

A. Conversion of One-particle Orbitals

Currently, QWalk can import and use the orbitals from GAMESS [40] (gaussian basis on molecules), CRYSTAL [41] (gaussian basis for extended systems), SIESTA [42], and GP [43] (plane waves for extended systems). The GP interface is not currently available for distribution due to licensing issues. More interfaces are planned, and are quite easy to add.

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 k-th orbital $\Phi_k(\vec{r}) = \sum_{\vec{G}} c_{k\vec{G}} e_{\vec{G}}(\vec{r})$, and wish to find the LCAO equiv-
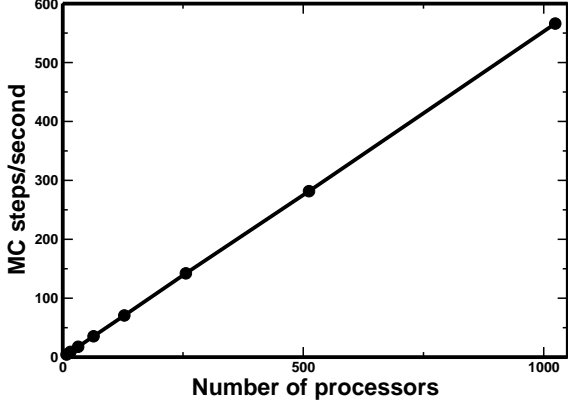
FIG. 7: Scaling of QWalk code over processors in Monte Carlo steps per second. The system is a 2x2x2 cell of BaTiO$_3$ with 320 electrons and one walker per node, on the San Diego Supercomputing Center DataStar machine. This is VMC; DMC is very much the same, because of the constant walker algorithm. This is close to a worst-case scenario for QMC, since the run was only approximately 40 seconds long.

alent $\Phi_k^{LCAO}(\vec{r}) = \sum_j a_{kj}\phi_j(\vec{r})$, 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.

X. CONCLUSION

QWalk is a step forward in creating 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, multi-core computers, etc. 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 electronion 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.

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(Array1 <doublevar> & r,
              Array1 <doublevar> & vals);
  void getLap(Array1 <doublevar> & r,
              Array2 <doublevar> & vals);
private:
  //put local variables here
  double alpha;
  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(Array1 <doublevar> & r,
                       Array1 <doublevar> & vals) {//getLap is omitted for space reasons
  //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));
}
```

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_funtion can be used. All the modules follow this basic procedure, just with different functions.

---

[1] W. M. C. Foulkes, L. Mitas, R. J. Needs, and G. Rajagopal, Rev Mod Phys **73**, 1 (2001).

[2] J. Grossman, J Chem. Phys. **117**, 1434 (2002).

[3] J. Grossman and L. Mitas, Phys. Rev. Lett. **94**, 056403 (2005).

[4] M. Bajdich, L. Mitas, G. Drobny, L. Wagner, and K. Schmidt, Phys. Rev. Lett **96**, 130201 (2006).

[5] M. Bajdich, L. Mitas, G. Drobny, and L. Wagner, Phys. Rev. B **72**, 075131 (2005).

[6] P. Vagner, M. Mosko, R. Nemeth, L. Wagner, and L. Mitas, Physica E p. 350 (2006).

[7] L. K. Wagner and L. Mitas, The Journal of Chemical Physics **126**, 034105 (pages 5) (2007), URL http://link.aip.org/link/?JCP/126/034105/1.

[8] M. Bajdich, L. Mitas, L. Wagner, and K. Schmidt, submitted to Phys. Rev. B pp. arxiv:cond–mat/0610850 (2006).

[9] N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, and E. Teller, J. Chem. Phys. **21**, 1087 (1953).

[10] W. Hastings, Biometrika **57**, 97 (1970).

[11] D. Bressanini, G. Morosi, S. Tarasco, and A. Mira, The Journal of Chemical Physics **121**, 3446 (2004), URL http://link.aip.org/link/?JCP/121/3446/1.

[12] C. Umrigar, M. Nightingale, and K. Runge, J. Chem. Phys. **99**, 2865 (1993).

[13] J. B. Anderson, The Journal of Chemical Physics **63**, 1499 (1975), URL http://link.aip.org/link/?JCP/63/1499/1.

[14] J. B. Anderson, The Journal of Chemical Physics **65**, 4121 (1976), URL http://link.aip.org/link/?JCP/65/4121/1.

[15] J. W. Moskowitz, K. E. Schmidt, M. A. Lee, and M. H. Kalos, The Journal of Chemical Physics **77**, 349 (1982), URL http://link.aip.org/link/?JCP/77/349/1.

[16] P. J. Reynolds, D. M. Ceperley, B. J. Alder, and W. A. Lester Jr., The Journal of Chemical Physics **77**, 5593 (1982), URL http://link.aip.org/link/?JCP/77/5593/1.

[17] D. Ceperley and B. Adler, Phys. Rev. Lett. **45**, 566 (1980).

[18] S. Baroni and S. Moroni, Phys. Rev. Lett. **82**, 4745 (1999).

[19] C. Pierleoni and D. M. Ceperley, ChemPhysChem **6**, 1872 (2005).

[20] C. Umrigar, K. Wilson, and J. Wilkins, Phys. Rev. Lett. **60**, 1719 (1988).

[21] C. J. Umrigar and C. Filippi, Phys Rev Lett **94**, 150201 (2005).

[22] C. Filippi and C. J. Umrigar, Phys Rev B **61**, R16291 (2000).

[23] S. Chiesa, D. M. Ceperley, R. M. Martin, and M. Holzmann, Physical Review Letters **97**, 076404 (pages 4) (2006), URL http://link.aps.org/abstract/PRL/v97/e076404.

[24] A. J. Williamson, G. Rajagopal, R. J. Needs, L. M. Fraser, W. M. C. Foulkes, Y. Wang, and M.-Y. Chou, Physical Review B (Condensed Matter) **55**, R4851 (1997), URL http://link.aps.org/abstract/PRB/v55/pR4851.

[25] R. Barnett, Z. Sun, and W. Lester, J. Chem. Phys. **114**, 7790 (2001).

[26] Y. Lee, P. Kent, M. Towler, R. Needs, and G. Rajagopal, Phys Rev B and private communication **62**, 13347 (2000).

[27] J. Trail and R. Needs, J. Chem. Phys. **122**, 174109 (2005).

[28] M. Burkatzki, C. Filippi, and M. Dolg, The Journal of Chemical Physics **126**, 234105 (pages 8) (2007), URL http://link.aip.org/link/?JCP/126/234105/1.

[29] M. Casula, C. Attaccalite, and S. Sorella, J.CHEM.PHYS. **121**, 7110 (2004), URL http://www.citebase.org/cgi-bin/citations?id=oai:arXiv.org:cond-mat/0409644.

[30] M. Bajdich, Ph.D. thesis, North Carolina State University (2007), URL http://altair.physics.ncsu.edu/bajdich/Phd_thesis_m_bajdich.pdf.

[31] R. P. Feynman and M. Cohen, Phys. Rev. **102**, 1189 (1956).

[32] K. E. Schmidt, M. A. Lee, M. H. Kalos, and G. V. Chester, Phys. Rev. Lett. **47**, 807 (1981).

[33] R. M. Panoff and J. Carlson, Phys. Rev. Lett. **62**, 1130 (1989).

[34] J. W. Moskowitz and K. E. Schmidt, The Journal of Chemical Physics **97**, 3382 (1992), URL http://link.aip.org/link/?JCP/97/3382/1.

[35] Y. Kwon, D. M. Ceperley, and R. M. Martin, Phys. Rev. B **48**, 12037 (1993).

[36] Y. Kwon, D. M. Ceperley, and R. M. Martin, Phys. Rev. B **50**, 1684 (1994).

[37] Y. Kwon, D. M. Ceperley, and R. M. Martin, Phys. Rev. B **53**, 7376 (1996).

[38] N. D. Drummond, P. L. Rios, A. Ma, J. R. Trail, G. G. Spink, M. D. Towler, and R. J. Needs, The Journal of Chemical Physics **124**, 224104 (pages 6) (2006), URL http://link.aip.org/link/?JCP/124/224104/1.

[39] P. L. Rios, A. Ma, N. D. Drummond, M. D. Towler, and R. J. Needs, Physical Review E (Statistical, Nonlinear, and Soft Matter Physics) **74**, 066701 (pages 15) (2006), URL http://link.aps.org/abstract/PRE/v74/e066701.

[40] M. W. Schmidt, K. K. Baldridge, J. A. Boatz, S. T. Elbert, M. S. Gordon, J. H. Jensen, S. Koseki, N. Mat-

sunaga, K. A. Nguyen, S. J. Su, et al., J. Comput Chem **14**, 1347 (1993).

[41] V. Saunders, R. Dovesi, C. Roetti, R. Orlando, C. Zicovich-Wilson, N. Harrison, K. Doll, B. Civalleri, I. Bush, P. DArco, et al. (2003).

[42] J. D. G. A. G. J. J. P. O. José M Soler, Emilio Artacho and D. Sánchez-Portal, Journal of Physics: Condensed Matter **14**, 2745 (2002), URL http://stacks.iop.org/0953-8984/14/2745.

[43] F. Gygi (2005).

[44] L. Mitas, E. Shirley, and D. Ceperley, J. Chem Phys **95**, 3467 (1991).

[45] P. Reynolds, D. Ceperley, B. Alder, and W. Lester, Jr., J. Chem. Phys. **77**, 5593 (1982).

[46] R. L. Honeycutt, Phys. Rev. A **45**, 600 (1992).

[47] S. Manten and A. Lüchow, J. Chem. Phys. **119**, 1307 (2003).

[48] Y. Kwon, D. M. Ceperley, and R. M. Martin, Phys. Rev. B **58**, 6800 (1998).

[49] M. Holzmann, D. M. Ceperley, C. Pierleoni, and K. Esler, Physical Review E (Statistical, Nonlinear, and Soft Matter Physics) **68**, 046707 (pages 15) (2003), URL http://link.aps.org/abstract/PRE/v68/e046707.