SCIENTIFIC HIGHLIGHT OF THE MONTH

Petascale computing opens new vistas for quantum Monte Carlo

M.J. Gillan[1,2,3], M.D. Towler[4,5,6] and D. Alfè[1,2,3,4]

[1] *Thomas Young Centre at UCL, Gordon Street, London WC1H 0AH, U.K.*
[2] *London Centre for Nanotechnology, UCL, Gordon Street, London WC1H 0AH, U.K.*
[3] *Department of Physics and Astronomy, UCL, Gower Street, London WC1E 6BT, U.K.*
[4] *Department of Earth Sciences, UCL, Gower Street, London WC1E 6BT, U.K.*
[5] *TCM Group, Cavendish Laboratory, University of Cambridge, 19 J.J. Thomson Avenue, Cambridge CB3 0HE, U.K.*
[6] *Apuan Alps Centre for Physics, via del Collegio 22, Vallico Sotto, Fabbriche di Vallico 55020 (LU), Tuscany, Italy*

Abstract

For many kinds of problem the accuracy of quantum Monte Carlo (QMC) is much better than that of density functional theory (DFT), and its scaling with number of atoms is much more favourable than that of high-level quantum chemistry. However, the widespread use of QMC has been hindered by the fact that it is considerably more expensive than DFT. We show here that QMC is very well placed to exploit the power of petascale supercomputers that are now becoming available, and we explain how this is opening up new scientific areas to investigation with QMC. We describe how we have been able to modify the Cambridge QMC code CASINO so that it runs with almost perfect parallel efficiency on 100000 cores and more on the JaguarPF machine at Oak Ridge. We also present illustrative results showing how QMC calculations run in this way are enabling us to go beyond the limitations of DFT in three important areas: the surface formation energies of materials, the adsorption energies of molecules on surfaces, and the energetics of water systems.

1 Introduction

The past twenty years have seen an extraordinary transformation in the ability of computer simulation to mimic the material world on the atomic scale. The accuracy and realism of simulations have been raised to completely new heights, partly by huge increases in computer power and partly by the development of powerful electronic-structure techniques of various kinds, including density functional theory (both standard and hybrid) [1, 2], high-level quantum chemistry

methods such as 2nd-order Møller-Plesset and coupled-cluster methods [3, 4, 5], computational many-body theory (for example the random-phase approximation) [6, 7], and quantum Monte Carlo[8, 9]. The power of supercomputers continues to grow at a dizzying rate: the petaflop barrier (10^{15} floating point operations per second) was broken in 2008, and exaflop speeds (10^{18} flops) are projected for sometime around 2016. However, the materials modelling community faces a daunting challenge. The new increases in computer power are coming almost entirely from enormous increases in the number of processors, and the practical exploitation of this power will force a comprehensive reappraisal of how materials modelling is done. In the next few years, some modelling techniques will benefit more than others from the increase of supercomputer power. We will argue here that the techniques of quantum Monte Carlo (QMC) are particularly well placed to benefit, and we will outline some of the steps we have taken to implement the CASINO QMC code [8, 10] on very large parallel computers, including the Jaguar supercomputer at Oak Ridge National Laboratory. We will also give some illustrations of how these new implementations of QMC are already allowing us to tackle problems that were out of reach before.

One way to characterize the growth of supercomputer power is to refer to the publicly available information on the Top500 website [11], which gives the technical specifications of the 500 most powerful supercomputers in the world. The data there shows that in the fifteen years from 1993 to 2008 the aggregate computing power of all 500 supercomputers in the list has grown by a factor of 14221. This implies a doubling time of only a little more than a year. Some of this vast increase has come from the growth of clock speed of the individual processors, but in recent years most of it has come from the increasing numbers of processors. In June 1993, the average number of processors of machines in the Top 500 was 142, but in November 2008 it was 6234. Today, the JaguarPF machine (currently second in the Top 500 list) has 224256 cores. Since clock speeds have not increased significantly in the past five years, and are unlikely to increase much in the future, the push towards exaflops will come almost entirely from further increases in the core count.

This all means that the materials modelling techniques that are likely to benefit most from the availability of petascale facilities are those for which the calculations can be broken into a very large number of separate pieces that can be executed simultaneously, preferably with only simple communications between the pieces. Quantum Monte Carlo is a technique of this kind, because it relies on very extensive statistical sampling. Diffusion Monte Carlo (DMC) is at present the type of QMC most commonly used for high-precision calculations on materials, and DMC works with large numbers of 'walkers' or 'configurations', whose job is to explore the electronic configuration space of the system. These walkers are to quite a large extent independent of each other, and this means that groups of walkers can be given to individual processors, with only fairly light communications between them. This is why DMC is particularly well suited to large parallel machines, and we will describe in this article how we have been able to run such calculations efficiently on machines having tens of thousands of processors.

We believe that there are strong scientific reasons for taking seriously the capabilities of QMC on large parallel machines. Although DFT dominates atomic-scale materials modelling and will

obviously continue to be extremely important, its accuracy is sometimes insufficient. When people want to comment on failures of DFT, they often like to refer to strongly-correlated systems, where are there are clear reasons for expecting DFT to struggle. But there are many other kinds of problems where the accuracy of DFT falls far short of what is needed. An obvious example is the inclusion of van der Waals dispersion, a crucially important effect in many fields, including molecular biology, surface science and aqueous systems. Although the generalisation of DFT to include dispersion has been hugely important (see e.g. Refs. [12, 13, 14]), there are still unsolved issues. Another important example is the adsorption energy of molecules on surfaces, where DFT is sometimes unable to give predictions of useful accuracy. DFT values of surface formation energies of quite simple paradigm materials such as silicon and magnesium oxide also depend strongly on the assumed exchange-correlation functional, and there is usually no way of knowing in advance which functional to trust. The well-known problem of calculating electronic band gaps could also be mentioned. The urgent practical need to do better in these and other areas is driving current efforts to develop more accurate methods, and the phrase 'beyond DFT' has become familiar over the past few years.

There is abundant evidence that there are large classes of problems for which QMC techniques are considerably more accurate than DFT, and we shall point out some of the evidence in this article. One of the main factors that has tended to deter researchers from applying QMC to their problems is that it demands much larger computer resources than DFT. Roughly speaking, one can expect a DMC calculation of the ground-state energy of a reasonably-sized assembly of atoms in a given geometry to take about 10^4 times more CPU time than the same calculation with standard (as opposed to hybrid) DFT. Obviously, 10^4 is a large factor, but it happens to be very similar to the factor by which Top500 supercomputer power grew from 1993 to 2008. This means that the situation with QMC now is similar to the DFT situation in the early 1990s, which was when the parallelisation of DFT first started to make an impact on materials modelling. The kinds of DFT calculations that appeared ridiculously daunting then are now performed routinely on desktop machines, and will presumably be performed by smart-phone apps in the future. A similar evolution path may well be followed by QMC, and we believe that now is the time to find out what can be achieved with QMC on petascale machines.

In the next Section, we will sketch the main ideas of QMC, noting the important role played by variational Monte Carlo, but then focusing mainly on the rather standard methods of diffusion Monte Carlo, and summarising the features of the CASINO code. In Section 3 we will outline the issues that we had to address in implementing CASINO on large parallel computers, particularly the UK national supercomputer HECToR, and the Jaguar machine at ORNL. In that Section, we will present some results of our parallel scaling tests, which show the possibility of running production calculations on 100000 or more cores. Then in Section 4, we give some illustrations of the calculations we are performing, including preliminary results that suggest how hitherto intractable scientific problems can be tackled in the near future. At the end of the article, we will offer some speculations about further developments that can be expected in the next few years.

2 The quantum Monte Carlo method

The fundamental quantum-mechanical object in the QMC method - somewhat unfashionably, given the emphasis normally and quite understandably placed on reducing the number of variables in the problem - is the full many-body wave function $\Psi(\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_N)$. This is a function of $3N$ variables, a complication which compares very unfavourably with the fundamental quantity in DFT - the electron density - which depends on only three variables. We use it nonetheless simply because we know the mathematical form of the differential equation which it satisfies, namely the Schrödinger equation:

$$\hat{H}\Psi = E\Psi. \tag{1}$$

If we wish to reformulate this problem in terms of the density then we face the issue that the exact equation satisfied by the ground-state density is completely unknown. In DFT, the complicated many-body problem is effectively subsumed in the definition of the exchange-correlation functional whose correct mathematical expression is unlikely ever to be known exactly. The inevitable approximations to this quantity, from the simplest LDAs up to the best modern functionals, substantially reduce the attainable accuracy and predictability of the method. This is an excellent reason for looking further at QMC.

Two alternatives for 'solving the Schrödinger equation' using QMC methods are normally used. First, take the wave function as a given analytic form and evaluate the energy using numerical integration; if necessary change the shape of the wave function by varying the parameters which define it until the energy is minimized. This is variational Monte Carlo. A more accurate alternative is to represent the wave function using a non-analytic method (the distribution in time and configuration space of an ensemble of diffusing particles) and encourage the particles to distribute themselves according to the true ground state wave function through the use of a projection technique. When they have done so, numerically integrate by sampling the wave function as before. This is diffusion Monte Carlo.

These two techniques will now be described in turn.

2.1 Variational Monte Carlo

Variational Monte Carlo, or VMC, is a relatively straightforward stochastic numerical integration method. It is in principle capable of computing quantum-mechanical expectation values for any many-electron wave function whose value can be evaluated at arbitrary points in its configuration space. Given some trial wave function Ψ_T satisfying the appropriate boundary conditions one may, for example, simply calculate the total energy as the expectation value of the many-body Hamiltonian operator $\hat{H}$,

$$\frac{\int \Psi_T^*(\mathbf{R}, \{\alpha\}) \hat{H} \Psi_T(\mathbf{R}, \{\alpha\}) \, \mathrm{d}\mathbf{R}}{\int \Psi_T^*(\mathbf{R}, \{\alpha\}) \Psi_T(\mathbf{R}, \{\alpha\}) \, \mathrm{d}\mathbf{R}} = E(\{\alpha\}) \geq E_0 \, , \tag{2}$$

where $\mathbf{R}$ is a $3N$-dimensional vector giving the configuration coordinates $(\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_N)$ of the N particles in the system (we ignore spin for the moment). The numerical integration is performed

by sampling the configuration space of the wave function at random points. Now for a wave function with appropriate properties the usual variational theorem tells us that by evaluating this integral we obtain an energy which is an upper bound to the exact ground-state energy, that is, energies for approximate wave functions are always higher than that of the true ground state. Such wave functions generally depend on some parameters - here collectively denoted by $\{\alpha\}$ - and these parameters may thus be varied to minimize an objective function such as the energy; in so doing the 'shape' of the wave function can be optimized.

The reason for sampling the wave function at *random* points is that the error in the integral then decreases as the square root of the number M of sampling points - irrespective of the dimensionality d of the integral. One may contrast this with a standard grid method such as the trapezoidal rule where the error decreases as $\mathcal{O}(M^{-\frac{2}{d}})$. Though Monte Carlo is less efficient in one dimension, it wins for more than four and as d increases it becomes the only practical approach. For example, in a system of thirty electrons in three dimensions one must integrate over the ninety degrees of freedom of the system, and the trapezoidal rule would need $\sim 10^{67}$ function evaluations to achieve the same accuracy as the Monte Carlo method with 1000 data points. When evaluating quantum-mechanical expectation values for N-particle systems we must do $3N$-dimensional integrals and it is clear there is simply no alternative to Monte Carlo methods.

Now in practice we do not wish to sample the random points from a uniform probability distribution - as implied above - but rather to group the points in regions where the integrand is finite, and to do this in such a way as to minimize the sample variance. Such *importance sampling* requires us to generate points distributed according to some non-uniform probability distribution $p(\mathbf{R})$, following which the calculation of the mean proceeds as usual. This sampling can be accomplished using a random walk moved according to the rules of the *Metropolis algorithm* [15]. We propose random moves taken from some standard distribution (usually Gaussians of appropriate width centred on the current points), always accepting moves to points of higher probability, and occasionally rejecting moves to regions of lower probability according to a particular formula obeying a detailed balance condition. Assuming *ergodicity* - that is, any point in the configuration space can be reached in a finite number of moves - then the distribution of the moving points will converge to the desired $p(\mathbf{R})$ after some appropriate period of equilibration.

To apply this procedure to evaluate an integral like Eq. 2 we need to rewrite the expression so that the integrand looks like a probability distribution times some appropriate function to be evaluated at each point (that can later be averaged). The best probability distribution to use (in the sense of minimizing the sample variance) is $p_{best}(\mathbf{R}) = |f(\mathbf{R})| / \int |f(\mathbf{R}')| \, \mathrm{d}\mathbf{R}'$, that is, we concentrate sampling points in regions where the absolute value of the integrand is large. In general we do not know the normalization integral in the denominator, so the best one can do is to make $p(\mathbf{R})$ look as much like this as possible. It is readily seen that in the Schrödinger case case $p(\mathbf{R}) = |\Psi_T(\mathbf{R})|^2$ and for an approximate eigenstate this is a good approximation to the above ideal sampling distribution. We may therefore rewrite the expectation value of the Hamiltonian $\hat{H}$ with respect to the trial wave function Ψ_T as

$$\langle \hat{H} \rangle = \frac{\int |\Psi_T(\mathbf{R})|^2 E_L(\mathbf{R})\, \mathrm{d}\mathbf{R}}{\int |\Psi_T(\mathbf{R})|^2\, \mathrm{d}\mathbf{R}}, \tag{3}$$

where the function to be evaluated - $E_L(\mathbf{R}) = \frac{\hat{H}(\mathbf{R})\Psi_T(\mathbf{R})}{\Psi_T(\mathbf{R})}$ - is known as the *local energy*. If Ψ_T were in fact the exact ground state wave function note that, according to Schrödinger's equation, the local energy should have a constant value E_0 over the whole of configuration space. For an *approximate* wave function this is no longer the case; a plot of the local energy for 1000 Metropolis-sampled points in the configuration space of an approximate trial function for a hydrogen atom (approximate since it is expanded in a finite Gaussian basis set) might look as in Fig. 1:

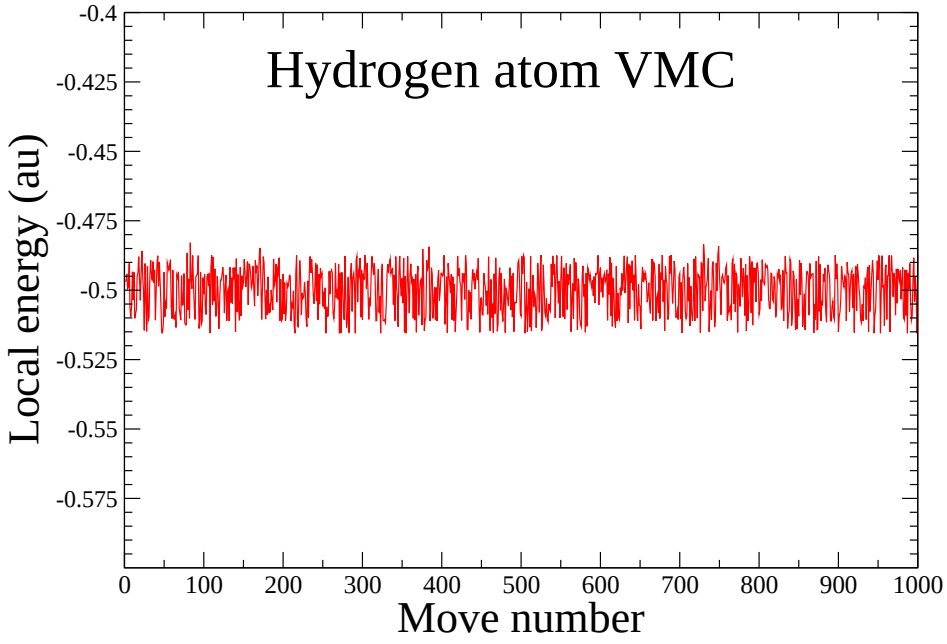

Figure 1: *Local energies of points in the random walk for a VMC run with an approximate wave function. All values are clustered around the true value of* -0.5 *Ha.*

So having used the Metropolis algorithm to generate a sequence of configurations $\mathbf{R}$ distributed according to $\Psi_T^2(\mathbf{R})$ we may then compute the desired expectation value by averaging the set of local energies:

$$\langle \hat{H} \rangle = \frac{1}{M} \sum_{i=1}^{M} E_L(\mathbf{R}_i) = \frac{1}{M} \sum_{i=1}^{M} \frac{\hat{H}\Psi_T(\mathbf{R}_i)}{\Psi_T(\mathbf{R}_i)} \ . \tag{4}$$

It should be clear from the figure that for hydrogen the energy thus obtained should (correctly) be -0.5 Ha plus or minus some small error bar. The error bar may need to be refined somewhat by particular statistical techniques to account for serial correlation of the points along the walk. Clearly expectation values other than the energy could be calculated in a similar way.

2.2 Diffusion Monte Carlo

In general, commonly-used expressions for the VMC wave function turn out not to have enough variational freedom to represent the true wave function - it is simply not possible to write down an analytic formula for an arbitrarily complex many-electron wave function. Using VMC to calculate the expectation value of the Hamiltonian therefore gives an energy which is incorrect (sometimes substantially so) and thus VMC in general is not accurate enough to justify the bother. It is at this point that diffusion Monte Carlo (DMC) comes to the rescue. This technique is able to automatically correct the shape of a 'guessed' wave function (particularly when given a good guess, such as the output of a VMC optimization) so that it looks much more like the exact one before calculating the expectation value.

This is clearly a nice trick, but as one might expect, the DMC algorithm is necessarily rather more involved than that for VMC. An approachable way of understanding it is to focus on the properties of quantum-mechanical propagators. Let's say we wish to integrate the time-dependent Schrödinger equation,

$$i\hbar\frac{\partial\Psi(\mathbf{R},t)}{\partial t} = -\frac{\hbar^2}{2m}\nabla^2\Psi(\mathbf{R},t) + V(\mathbf{R},t)\Psi(\mathbf{R},t) = \hat{H}\Psi(\mathbf{R},t)\ , \tag{5}$$

where $\mathbf{R} = \{\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_N\}$, V is the potential energy operator, and $\nabla = (\nabla_1, \nabla_2, \ldots, \nabla_N)$ is the $3N$-dimensional gradient operator. Integrating this is equivalent to wanting a formula for Ψ and, to find this, we must invert this differential equation. The result is an integral equation involving the *propagator* K:

$$\Psi(\mathbf{R},t) = \int K(\mathbf{R},t;\mathbf{R}',t')\Psi(\mathbf{R}',t')\,\mathrm{d}\mathbf{R}'. \tag{6}$$

The propagator is interpreted as the probability amplitude for a particle to travel from one place to another (in this case, from $\mathbf{R}'$ to $\mathbf{R}$) in a given time $t - t'$. It is a *Green's function* for the Schrödinger equation. We see that the probability amplitude for a particle to be at $\mathbf{R}$ sometime in the future is given by the probability amplitude of it travelling there from $\mathbf{R}'$ - which is just $K(\mathbf{R},t;\mathbf{R}',t')$ - weighted by the probability amplitude of it actually starting at $\mathbf{R}'$ in the first place - which is $\Psi(\mathbf{R}',t')$ - summed over all possible starting points $\mathbf{R}'$. This is a straightforward concept.

How might we calculate the propagator? A typical way might be to use the *Feynman path-integral* method. For given start and end points $\mathbf{R}'$ and $\mathbf{R}$ one gets the overall amplitude by summing the contributions of the infinite number of *all* possible 'histories' or paths which include those points. It doesn't matter why for the moment but the amplitude contributed by a particular history is proportional to $e^{iS_{cl}/\hbar}$ where S_{cl} is the classical *action* of that history, i.e. the time integral of the classical Lagrangian $\frac{1}{2}mv^2 - V$ along the corresponding phase space path of the system. The full expression for the propagator in Feynman's method may then be written as

$$K^F(\mathbf{R},t;\mathbf{R}',t') = N\sum_{\text{all paths}}\exp\left[\frac{i}{\hbar}\int_{t'}^{t} L_{cl}(t'')\,\mathrm{d}t''\right]. \tag{7}$$

An alternative way to calculate the propagator is to use the de Broglie-Bohm pilot-wave interpretation of quantum mechanics [16], where the electrons both objectively exist and have the obvious definite trajectories derived from a straightforward analysis of the streamlines of the quantum-mechanical probability current. From this perspective we find we can achieve precisely the same result as the Feynman method by integrating the *quantum* Lagrangian $L_q(t) = \frac{1}{2}mv^2 - (V + Q)$ along precisely *one* path - the path that the electron actually follows - as opposed to linearly superposing amplitudes obtained from the *classical* Lagrangian associated with the infinite number of all possible paths. Here Q is the 'quantum potential', which is the potential energy function of the quantum force (the force that the wave field exerts on the electrons). It is easy to show the equivalent pilot-wave propagator is:

$$K^B(\mathbf{R}, t; \mathbf{R}', t') = \frac{1}{J(t)^{\frac{1}{2}}} \exp\left[\frac{i}{\hbar}\int_{t'}^{t} L_q(t'')\,\mathrm{d}t''\right] \tag{8}$$

where J is a simple Jacobian factor. This formula should be contrasted with Eq. 7. One should also note that because de Broglie-Bohm trajectories do not cross, one need not sum over all possible starting points $\mathbf{R}'$ to compute $\Psi(\mathbf{R}, t)$ - one simply uses the $\mathbf{R}'$ that the unique trajectory passes through.

What is the connection of all this with DMC? Well, in DMC an arbitrary starting wave function is evolved using a (Green's function) propagator just like the ones we have been discussing. The main difference is that the propagation occurs in *imaginary time* $\tau = it$ as opposed to real time t. For reasons that will shortly become apparent this has the effect of 'improving' the wave function, i.e. making it look more like the ground state as imaginary time passes. For technical reasons, it also turns out that the propagation has to take place in a sequence of very short hops in imaginary time, and so our evolution equation now looks like this:

$$\Psi(\mathbf{R}, \tau + \delta\tau) = \int K^{DMC}(\mathbf{R}, \mathbf{R}', \delta\tau)\Psi(\mathbf{R}', \tau)\,\mathrm{d}\mathbf{R}'. \tag{9}$$

The evolving wave function is not represented in terms of a basis set of known analytic functions but by the distribution in space and time of randomly-diffusing electron positions over an ensemble of copies of the system ('walkers' or 'configurations'). So in other words, the DMC method is a 'stochastic projector method' whose purpose is to evolve/project out the solution to the *imaginary-time Schrödinger equation* from an arbitrary starting state. We shall write this equation - which is simply what you get by taking the regular time-dependent equation and substituting τ for the time variable it - in atomic units as

$$-\frac{\partial\Psi^{DMC}(\mathbf{R}, \tau)}{\partial\tau} = -\frac{1}{2}\nabla^2\Psi(\mathbf{R}, \tau) + (V(\mathbf{R}) - E_T)\Psi(\mathbf{R}, \tau)\ . \tag{10}$$

Here the real variable τ measures the progress in imaginary time and, for purposes to be revealed presently, we have included a constant E_T - an energy offset to the zero of the potential which only affects the wave function normalization.

How then does propagating our trial function in imaginary time 'improve' it? For eigenstates, the general solution to the usual time-dependent Schrödinger equation is clearly $\phi(\mathbf{R}, t) =$

$\phi(\mathbf{R},0)\mathrm{e}^{-i(\hat{H}-E_T)t}$. By definition, we may expand an arbitrary 'guessed' $\Psi(\mathbf{R},t)$ in terms of a complete set of these eigenfunctions of the Hamiltonian $\hat{H}$:

$$\Psi(\mathbf{R},t)=\sum_{n=0}^{\infty}c_n\phi_n(\mathbf{R})\mathrm{e}^{-i(E_n-E_T)t}. \tag{11}$$

On substituting it with imaginary time τ the oscillatory time-dependence of the complex exponential phase factors becomes an exponential decay:

$$\Psi(\mathbf{R},\tau)=\sum_{n=0}^{\infty}c_n\phi_n(\mathbf{R})\mathrm{e}^{-(E_n-E_T)\tau} \tag{12}$$

Let us assume our initial guess for the wave function is not orthogonal to the ground state (i.e. $c_0\neq 0$). Then if we magically choose the constant E_T to be the ground state eigenvalue E_0 (or, in practice, keep very tight control of it through some kind of feedback procedure) then it is clear we should eventually get *imaginary-time independence* of the probability distribution, in the sense that as $\tau\to\infty$, our initial $\Psi(\mathbf{R},0)$ comes to look more and more like the stationary ground state $\phi_0(\mathbf{R})$ as the contribution of the excited-state eigenfunctions dies away:

$$\Psi(\mathbf{R},\tau)=c_0\phi_0+\sum_{n=1}^{\infty}c_n\phi_n(\mathbf{R})\mathrm{e}^{-(E_n-E_0)\tau}\ . \tag{13}$$

So now we know *why* we do this propagation, how in practice do we find an expression for the propagator K? Consider now the imaginary-time Schrödinger equation in two parts:

$$\frac{\partial\Psi(\mathbf{R},\tau)}{\partial\tau}=\frac{1}{2}\nabla^2\Psi(\mathbf{R},\tau) \tag{14}$$

$$\frac{\partial\Psi(\mathbf{R},\tau)}{\partial\tau}=-(V(\mathbf{R})-E_T)\Psi(\mathbf{R},t). \tag{15}$$

These two formulae respectively have the form of the usual diffusion equation and of a rate equation with a position-dependent rate constant. The appropriate propagator for the diffusion equation is well-known; it is a $3N$-dimensional Gaussian with variance $\delta\tau$ in each dimension. The propagator for the rate equation is also known - it gives a so-called 'branching factor' which can be interpreted as a position-dependent weight for a member of an ensemble. Multiplying the two together to get the following propagator for the imaginary-time Schrödinger equation is an approximation - the '*short time approximation*' - valid only in the limit of small $\delta\tau$ (which is why we need to do the evolution as a sequence of short hops):

$$K^{DMC}(\mathbf{R},\mathbf{R}',\delta\tau)=\frac{1}{(2\pi\delta\tau)^{\frac{3N}{2}}}\exp\left(-\frac{|\mathbf{R}-\mathbf{R}'|^2}{2\delta\tau}\right)\exp\left[-\delta\tau\left(\frac{V(\mathbf{R})+V(\mathbf{R}')-2E_T}{2}\right)\right]. \tag{16}$$

Let us then summarize with a simple example how the DMC algorithm works. If we interpret Ψ as a *probability density*, then the diffusion equation $\frac{\partial\Psi}{\partial\tau}=\frac{1}{2}\nabla^2\Psi$ represents the movement of N diffusing particles. If we turn this around we may decide to *represent* $\Psi(\mathbf{R},\tau)$ by an *ensemble* of such sets of particles. Each member of such an ensemble is a 'configuration' or 'walker'. We

interpret the full propagator $K^{DMC}(\mathbf{R}, \mathbf{R}', \delta\tau)$ as the probability of a configuration moving from $\mathbf{R}'$ to $\mathbf{R}$ in a time $\delta\tau$. The branching factor in the propagator will generally be interpreted as a stochastic survival probability for a given configuration rather than as a simple weight, as the latter is prone to numerical instabilities. This means that the configuration population becomes dynamically variable; walkers that stray into regions of high V have a good chance of being killed (removed from the calculation); in low V regions walkers have a high probability of multiplying (i.e. they create copies of themselves which then propagate independently). It is solely this branching or reweighting that 'changes the shape of the wave function' as it evolves. So, as we have seen, after a sufficiently long period of imaginary-time evolution all the excited states will decay away leaving only the ground-state wave function, at which point the propagation may be continued in order to accumulate averages of interesting observables.

As a simple example, consider Fig 3. Here a deliberately bad guess is made for the trial function: the ground-state wave function for a single electron in a harmonic potential well is assumed to be a constant in the vicinity of the well and zero everywhere else. The calculation begins with seven copies of the system or configurations in our ensemble; the electrons in this ensemble are initially randomly distributed according to the uniform probability distribution in the region where the trial function is finite. The particle distribution is then evolved in imaginary time according to the scheme developed above. The electrons are subsequently seen to become distributed according to the proper Gaussian shape of the exact ground-state wave function. It is evident from the figure that the change in shape is produced by the branching factor occasionally eliminating walkers in high V regions and duplicating them in low V regions.

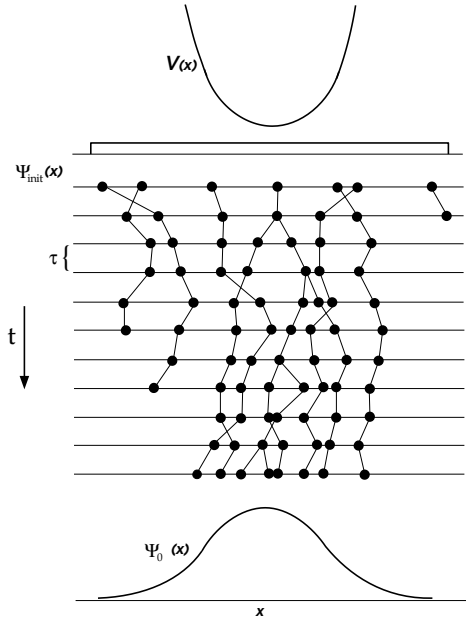

Figure 2: *Schematic illustration of the DMC algorithm for a single electron in a harmonic potential well, showing the evolution of the shape of the wave function due to propagation in imaginary time. Figure taken from Ref. [18].*

This 'pure DMC' algorithm works very well in a single-particle system with a nicely-behaved

potential, as in the example. Unfortunately it suffers from two very serious drawbacks which become evident in multi-particle systems with divergent Coulomb potentials.

The first problem arises due to our assumption that Ψ is a probability distribution - necessarily positive everywhere - even though the antisymmetric nature of multi-particle fermionic wave functions means that it must have both positive and negative parts separated by a 'nodal surface', that is, a $3N-1$-dimensional hypersurface on which it has the value zero. One might think that two separate populations of walkers with attached positive and negative weights might get round this problem (essentially the well-known 'fermion sign problem') but in practice there is a severe signal-to-noise issue. It is possible to construct formally-exact algorithms of this nature which overcome some of the worst practical problems [19] but to date all seem highly inefficient with poor system size scaling.

The second problem is less fundamental but in practice very severe. The required rate of removing or duplicating walkers diverges when the potential energy diverges (which occurs whenever two particles are coincident) due to the presence of V in the branching factor of Eq. 16. This leads to stability problems and poor statistical behaviour.

These problems may be dealt with at the cost of introducing the most important approximation in the DMC algorithm: the *fixed-node approximation* [20]. We say, in effect, that particles may not cross the nodal surface of the *trial* wave function Ψ_T, that is, there is an infinite repulsive potential barrier on the nodes. This forces the DMC wave function Ψ to be zero on that hypersurface. If the nodes of the trial function coincide with the exact ones, then such an algorithm will give the exact ground-state energy (it is of course well-known that the exact de Broglie-Bohm particle trajectories cannot pass through the nodal surface). If the trial function nodes do not coincide with the exact ones then the DMC energy will be higher than the ground-state energy (but less than or equal to the VMC energy). The variational principle thus applies.

To make such an algorithm efficient we must introduce importance sampling, and this is done in the following way. We require that the imaginary-time evolution produces the *mixed distribution* $f = \Psi_T \Psi$, rather than the pure distribution. Substituting this into the imaginary time Schrödinger equation Eq. 10 we obtain

$$-\frac{\partial f(\mathbf{R},\tau)}{\partial \tau} = -\frac{1}{2}\nabla^2 f(\mathbf{R},\tau) + \nabla \cdot [\mathbf{v}_D(\mathbf{R}) f(\mathbf{R},\tau)] + (E_L(\mathbf{R}) - E_T) f(\mathbf{R},\tau) \ , \tag{17}$$

where $\mathbf{v}_D(\mathbf{R})$ is the $3N$-dimensional *drift velocity* vector defined by

$$\mathbf{v}_D(\mathbf{R}) = \nabla \ln |\Psi_T(\mathbf{R})| = \frac{\nabla \Psi_T(\mathbf{R})}{\Psi_T(\mathbf{R})} \ , \tag{18}$$

and

$$E_L(\mathbf{R}) = \Psi_T^{-1} \left(-\frac{1}{2}\nabla^2 + V(\mathbf{R}) \right) \Psi_T \ , \tag{19}$$

is the usual local energy. The propagator from $\mathbf{R}'$ to $\mathbf{R}$ for the importance sampled algorithm now looks like this:

$$K^{DMC}(\mathbf{R},\mathbf{R}',\delta\tau) = \frac{1}{(2\pi\delta\tau)^{\frac{3N}{2}}} \exp\left[-\frac{(\mathbf{R}-\mathbf{R}'-\delta\tau\mathbf{F}(\mathbf{R}'))^2}{2\delta\tau}\right] \exp\left[-\frac{\delta\tau}{2}\left(E_{\mathrm{L}}(\mathbf{R})+E_{\mathrm{L}}(\mathbf{R}')-2E_{\mathrm{T}}\right)\right] . \tag{20}$$

Because the nodal surface of Ψ is constrained to be that of Ψ_T then their product f is positive everywhere and can now be properly interpreted as a probability distribution. The time evolution generates the distribution $f = \Psi_T\Psi$, where Ψ is now the lowest energy wave function with the same nodes as Ψ_T. This solves the first of our two problems. The second problem of the poor statistical behaviour due to the divergences in the potential energy is also solved because the term $(V(\mathbf{R}) - E_T)$ in Eq. 10 has been replaced by $(E_L(\mathbf{R}) - E_T)$ in Eq. 17 which is much smoother. Indeed, if Ψ_T was an exact eigenstate then $(E_L(\mathbf{R}) - E_T)$ would be independent of position in configuration space. Although we cannot in practice find the exact Ψ_T it is possible to eliminate the local energy divergences due to coincident particles by choosing a trial function which has the correct cusp-like behaviour at the relevant points in the configuration space [21]. Note that this is all reflected in the branching factor of the new propagator of Eq. 20.

The nodal surface partitions the configuration space into regions that we call 'nodal pockets'. The fixed-node approximation implies that we are restricted to sampling only those nodal pockets that are occupied by the initial set of walkers, and this appears to introduce some kind of ergodicity concern since at first sight it seems that we ought to sample every nodal pocket. This would be an impossible task in large systems. However, the *tiling theorem* for exact fermion ground states [22, 23] asserts that all nodal pockets are in fact equivalent and related by permutation symmetry; one need therefore only sample one of them. This theorem is intimately connected with the existence of a variational principle for the DMC ground state energy [23]. Other interesting investigations of properties of nodal surfaces have been published. [24, 25, 26]

A practical importance-sampled DMC simulation proceeds as follows. First we pick an ensemble of a few hundred walkers chosen from the distribution $|\Psi_T|^2$ using VMC and the standard Metropolis algorithm. This ensemble is then evolved according to the short-time approximation to the Green function of the importance-sampled imaginary-time Schrödinger equation (Eq. 17), which involves repeated steps of biased diffusion followed by the deletion and/or duplication of walkers. The bias in the diffusion is caused by the drift vector arising out of the importance sampling which directs the sampling towards parts of configuration space where $|\Psi_T|$ is large (i.e. it plays the role of an Einsteinian *osmotic velocity*). This drift step is always directed away from the node, and $\nabla\Psi_T$ is in fact a normal vector of the nodal hypersurface. After a period of equilibration the excited state contributions will have largely died out and the walkers start to trace out the probability distribution $f(\mathbf{R})/\int f(\mathbf{R})\,d\mathbf{R}$. We can then start to accumulate averages, in particular the DMC energy. Note that throughout this process the reference energy E_T is varied to keep the walker population under control through a specific feedback mechanism.

The DMC energy is given by

$$E_{DMC} = \frac{\int f(\mathbf{R}) E_L(\mathbf{R})\, d\mathbf{R}}{\int f(\mathbf{R})\, d\mathbf{R}} \approx \sum_i E_L(\mathbf{R}_i)\ . \tag{21}$$

This energy expression would be exact if the nodal surface of Ψ_T were exact, and the fixed-node error is second order in the error in the Ψ_T nodal surface (when a variational theorem exists [23]). The accuracy of the fixed-node approximation can be tested on small systems and normally leads to very satisfactory results. The trial wave function thus limits the final accuracy that can be obtained and it also controls the statistical efficiency of the algorithm. Like VMC, the DMC algorithm satisfies a zero-variance principle, i.e. the variance of the energy goes to zero as the trial wave function goes to an exact eigenstate. For other expectation values of operators that do not commute with the Hamiltonian then the DMC mixed estimator is biased and other techniques are required in order to sample the pure distribution [27, 28, 29].

A final point: the necessity of using the fixed-node approximation suggests that the best way of optimizing wave functions would be to do it in DMC directly. The nodal surface could then in principle be optimized to the shape which minimizes the DMC energy. The backflow technique [32] has some bearing on the problem, but the usual procedure involving optimization of the energy or variance in VMC will not usually lead to the optimal nodes in the sense that the fixed-node DMC energy is minimal. The large number of parameters - up to a few hundred - in your typical Slater-Jastrow(-backflow) wave function means that direct variation of the parameters in DMC is too expensive. Furthermore, we note that optimizing the energy in DMC is tricky for the nodal surface as the contribution of the region near the nodes to the energy is small. More exotic ways of optimizing the nodes are still being actively developed [30, 31].

2.3 The CASINO code

The CASINO quantum Monte Carlo program [8, 10] has been developed in Cambridge by Richard Needs, Mike Towler, Neil Drummond, Pablo López Ríos and their collaborators since the mid-1990s, and is freely available to the academic community. It is able to do variational and diffusion Monte Carlo calculations for finite systems such as atoms and molecules, for real materials periodic in one, two or three dimensions, and for various model systems such as homogeneous and inhomogeneous electron gases [35, 36, 37], Wigner crystals [38], and excitonic bilayers [39].

A general problem for QMC programs arises from the need to interact with other electronic structure codes for the purpose of importing trial wave functions. The basic form of many-body wave function used by CASINO is the common Slater-Jastrow type. This consists of a single Slater determinant of orbitals (or sometimes a linear combination of a small number of them) multiplied by an optimizable positive-definite Jastrow correlation function which is symmetric in the electron coordinates and depends on the inter-particle distances. The Jastrow function, through its explicit dependence on interparticle separations, allows efficient inclusion of both long and short range correlation effects. As we have observed, the final DMC answer depends only on the nodal surface of the wave function and this cannot be affected by the nodeless positive-

definite Jastrow, and in DMC it serves mainly to decrease the amount of computer time required to achieve a given statistical error bar and to improve the stability of the algorithm. The basic form of the Slater-Jastrow wave function may thus be written as:

$$\Psi(\mathbf{X}) = e^{J(\mathbf{X})} \sum_n c_n D_n(\mathbf{X}) \ , \tag{22}$$

where $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N)$ and $\mathbf{x}_i = \{\mathbf{r}_i, \sigma_i\}$ denotes the space-spin coordinates of electron i, $e^{J(\mathbf{X})}$ is the Jastrow factor, the c_n are determinant coefficients, and the $D_n(\mathbf{X})$ are Slater determinants of single-particle orbitals ϕ_j. Orbitals for realistic atomic systems are generally obtained from self-consistent DFT or Hartree-Fock calculations, and CASINO has interfaces to many codes capable of producing these, including GAUSSIAN [40], CRYSTAL [41], TURBOMOLE [42], GAMESS-US [43], PWSCF [44], ABINIT [45], CASTEP [46],and ADF [47], Of course the exact degree of support for each can vary as the programs evolve, as the developers forget that the interface exists, and as the relevant expertise moves in and out of the Cambridge group. This set of codes requires CASINO to work with orbitals expanded in a variety of basis sets, including Gaussians, Slater functions, plane-waves, and blip functions [50], in addition to orbitals tabulated on grids.

CASINO has been designed to run on essentially any hardware with minimal setup, but it is particularly effective on machines with large numbers of processors. This is the principal topic of this article to which we now turn.

3 DMC on massively parallel computers

3.1 Introduction

Quantum Monte Carlo is in general an intrinsically parallel technique, and as such is ideally placed to exploit new and future generations of massively parallel computers. This is trivially realized in the case of VMC and in the various associated techniques for carrying out VMC wave function optimization. In the pure VMC case, essentially no interprocessor communication is required during a simulation. Each processor carries out an independent random walk using a fixed wave function and a different random number sequence. The resulting energies are then averaged over the processors at the end. Assuming the equilibration time to be negligible, running for a length of time T on N_p processors generates the same amount of data as running for time N_pT on a single processor (though of course the results will only agree within statistical error bars since the random walks are different in the two cases). VMC should therefore scale to an arbitrarily large number of processors.

We have not spoken in detail about the various techniques used to optimize wave functions in VMC, nor is this necessary for our purposes. In general, one is required to minimize an objective function (usually the variance or the energy) with respect to a set of parameters in the wave function. In the basic variance minimization algorithm we run a sequence of short VMC runs to generate sets of walkers distributed according to the current wave function, and each of these is followed by an optimization (Levenberg-Marquardt non-linear least squares or

similar). The VMC stages are perfectly parallel, as described above. In the optimization stages, the set of configurations is distributed evenly between the processors. The master processor broadcasts the current set of optimizable parameters, then each processor calculates the local energy of each of its configurations and reports the energies (and weights, if required) to the master. The CPU time required to evaluate the local energies of the configuration set usually far exceeds the time spent communicating (reporting one or two numbers per configuration to the master and receiving a handful of parameter values at each iteration). In particular the time spent evaluating the local energies increases with system size, whereas the time spent on interprocessor communication is independent of system size.

For the case of energy minimization, the VMC stages are perfectly parallel as described above. The optimization stages involve various matrix algebra operations. In this scheme the walkers are divided evenly between the processors, each of which separately generates one section of the full matrices. The full matrices are then gathered on the master processor, where the matrix algebra is done. The time taken to do the matrix algebra is usually insignificant in comparison to the time taken in VMC and matrix generation. The time taken in interprocessor communication is recorded and written out during energy minimization, and is typically at maximum a few percent of the total time spent in an iteration (and often much less than one percent). Overall, energy minimization is very nearly perfectly parallel.

The problem - if there is a problem - therefore lies in the DMC algorithm, and this is largely to do with *load balancing*. DMC is parallelized in a similar way to VMC - by assigning separate walkers to different processors - but in DMC by contrast the processors are required to communicate. The branching algorithm explained in Section 2.2 leads to a dynamically variable population of walkers, that is, the population fluctuates during the run as walkers are killed or duplicated to 'change the shape of the wave function'. One of the reasons that this leads to interprocessor communication is that the population must be adjusted dynamically to some initial target via a kind of feedback mechanism as the simulation proceeds. This relies on a knowledge of the instantaneous total energy, which must be calculated and averaged over all processors after each time step. The most important problem, however, is the necessity to transfer walkers between the cores (a walker, in this sense, being the list of current electron positions for the configuration, along with various associated quantities related to the energy and wave function). These transfers are purely for efficiency; in order to maintain load balance it is important to ensure that each core has roughly the same number of walkers, since the cost of each time step is determined by the processor with the largest population. The total number of walkers communicated between processors increases with the number of cores and ends up being the single greatest cause of inefficiency for runs on massively parallel machines. A rough theoretical analysis of the expected scaling behaviour might run as follows [17].

The time t_{move} required to propagate each walker scales as N_e^{α}, where N_e is the number of particles, and α is an integer power. For typical systems, where extended orbitals represented in a localized basis are used and the CPU time is dominated by the evaluation of the orbitals, $\alpha = 2$ [18]. The use of localized orbitals can improve this to $\alpha = 1$ [33, 34]. For very large systems, or systems in which the orbitals are trivial to evaluate, the cost of updating the determinants

will start to dominate: this gives $\alpha = 3$ with extended orbitals and $\alpha = 2$ with localized orbitals. Hence the average cost of propagating all the walkers over one time step, which is approximately the same on each processor, is

$$T_{\mathrm{CPU}} \approx a \frac{N_e^{\alpha} N_{\mathrm{target}}}{N_{\mathrm{proc}}}, \tag{23}$$

where a is a constant which depends on both the system being studied and the details of the hardware, N_{target} is the target population, and N_{proc} is the number of processors.

So now the population varies on each processor. How? Let $n_{\mathrm{redist}}\tau$ be the redistribution period, that is, we redistribute the walker population after every n_{redist} time steps τ. Given the form of the branching factor (the second exponential in Eqn. 20), the population on a processor p at any given time must be increasing or decreasing exponentially, because the mean energy $E(p)$ of the walker population on that processor is unlikely to be exactly equal to the reference energy E_T in the argument of the branching factor. We assume that $E(p) - E_T$ remains roughly constant over the redistribution period. At the start of the redistribution period the population $N_w(p, 0)$ on each processor is the same. At the end of the redistribution period, the expected population on processor p is $N_w(p, n_{\mathrm{redist}}) = N_w(p, 0) \exp[-(E(p) - E_T) n_{\mathrm{redist}} \tau]$. Hence $\bar{N}_w(n_{\mathrm{redist}}) \approx \bar{N}_w(0) \exp[-(\bar{E} - E_T) n_{\mathrm{redist}} \tau] + \mathcal{O}(n_{\mathrm{redist}}^2 \tau^2)$, where the bar denotes an average over the processors, and so the average growth or decay of the population is the same as that of the entire population (which should be small, because E_T is chosen so as to ensure this).

What is the optimal redistribution period? Recall t_{move} is the cost of propagating a single walker over one time step. Let t_{trans} be the cost of transferring a single walker between processors. Let q be the processor with the largest number of walkers, i.e., the one with the lowest energy $E(q) \equiv \min\{E(p)\}$. Both the cost of propagating walkers and the cost of transferring them are determined by processor q. The expected number of walkers on processor q at the end of the redistribution period (i.e., after n_{redist} time steps) is $\max\{N_w(p, n_{\mathrm{redist}})\} \approx \bar{N}_w(n_{\mathrm{redist}}) + c n_{\mathrm{redist}} + \mathcal{O}(n_{\mathrm{redist}}^2)$, where $c = \bar{N}_w(1)(\bar{E} - \min\{E(p)\})\tau$. Here $\langle \bar{N}_w(0) \rangle = N_{\mathrm{target}}/N_{\mathrm{proc}}$ and $\langle c \rangle$ is a positive constant. At the end of the redistribution period, $c n_{\mathrm{redist}}$ walkers are to be transferred from processor q. Hence the average cost of transferring walkers per time step is $t_{\mathrm{trans}} \langle c \rangle$, which is independent of n_{redist}.

The average cost per time step of waiting for the processor q with the greatest number of walkers to finish propagating all its excess walkers is

$$\frac{t_{\mathrm{move}} \langle c \rangle \left[0 + 1 + \ldots + (n_{\mathrm{redist}} - 1)\right]}{n_{\mathrm{redist}}} = \frac{t_{\mathrm{move}} \langle c \rangle (n_{\mathrm{redist}} - 1)}{2}. \tag{24}$$

So the total average cost per time step in DMC is

$$T = \frac{t_{\mathrm{move}} N_{\mathrm{target}}}{N_{\mathrm{proc}}} + \frac{t_{\mathrm{move}} \langle c \rangle (n_{\mathrm{redist}} - 1)}{2} + t_{\mathrm{trans}} \langle c \rangle. \tag{25}$$

Clearly the redistribution period should be chosen to be as small as possible to minimize T. Numerical tests confirm that increasing the redistribution period only acts to slow down calculations. One should therefore choose $n_{\mathrm{redist}} = 1$, i.e., redistribution should take place after every time step. We assume this to be the case henceforth.

What, then, is the cost of load balancing? Let $\sigma_{E(p)}$ be the standard deviation of the set of processor energies. We assume that $\langle E(p)\rangle - \langle \min\{E(p)\}\rangle \propto \sigma_{E(p)} \propto \sqrt{N_e N_{\rm proc}/N_{\rm target}}$. Hence $\langle c\rangle \propto \sqrt{N_e N_{\rm target}/N_{\rm proc}}$. The cost $t_{\rm trans}$ of transferring a single walker is proportional to the system size N_e. Hence the cost of load balancing is

$$T_{\rm comm} \approx b\sqrt{\frac{N_{\rm target}N_e^3}{N_{\rm proc}}}, \tag{26}$$

where the constant b depends on the system being studied, the wave-function quality and the computer architecture. Note that good trial wave functions will lead to smaller population fluctuations and therefore less time spent load-balancing.

Clearly one would like to have $T_{\rm CPU} \gg T_{\rm comm}$, as the DMC algorithm would in theory be perfectly parallel in this limit. The ratio of the cost of load balancing to the cost of propagating the walkers is

$$\frac{T_{\rm comm}}{T_{\rm CPU}} = \frac{b}{a}\left(\frac{N_{\rm proc}}{N_{\rm target}}\right)^{1/2} N_e^{3/2-\alpha}. \tag{27}$$

It is immediately clear that by increasing the number of walkers per processor $N_{\rm target}/N_{\rm proc}$ the fraction of time spent on interprocessor communication can be made arbitrarily small. In practice the number of walkers per processor is limited by the available memory, and by the fact that carrying out DMC equilibration takes longer if more walkers are used. Increasing the number of walkers does not affect the efficiency of DMC statistics accumulation, so, assuming that equilibration remains a small fraction of the total CPU time, the walker population should be made as large as memory constraints will allow.

For $\alpha > 3/2$ (which is always the case except in the regime where the cost of evaluating localized orbitals dominates), the fraction of time spent on interprocessor communication falls off with system size. Hence processor-scaling tests on small systems may significantly underestimate the maximum usable number of processors for larger problems.

So that's the theory; how does this work in practice? In analyzing scaling behaviour one normally distinguishes between 'strong scaling' - where we ask how the time to solution for a fixed system size varies with the number of processors - and 'weak scaling' where we ask how the time to a solution varies for a fixed system size per processor (i.e. if we double the number of processors). Perfect weak scaling is thus a constant time to solution, independent of processor count.

What is the appropriate definition of 'system size' in this context? One would think that QMC is different from DFT, of course, since if we double what we normally consider to be the size of the system (the number of electrons in the molecule, or whatever) then we must double the number of samples of the wave function in order to get the same error bar. So our criterion for the system size, is something like 'the number of samples of the wave function configuration space required to get a fixed error bar'. In all the scaling calculations reported here, we report the time taken to sample the wave function N times, where N is the number of walkers times the number of moves. N is constant for all core counts, therefore we are looking at the strong scaling.

Two ways of doing this have been considered. The first way is to consider both a fixed total target population of walkers and a fixed number of moves, neither of which varies with the number of processors. For a code that scaled ideally one would then expect the time taken to halve if we double the number of processors since each individual processor will have half the number of walkers to deal with. This is usually not the best way to exploit QMC on a parallel machine, but it serves to illustrate several important points.

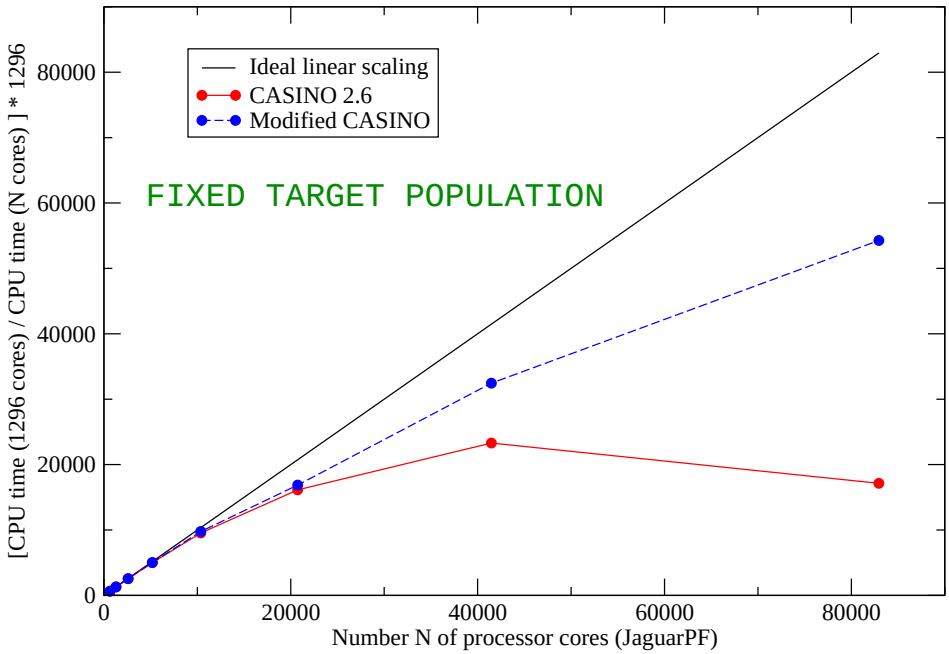

Figure 3: *Scaled CPU time required by various numbers of cores of the JaguarPF machine to carry out one ten-move DMC statistics accumulation block for a water molecule adsorbed on a two-dimensionally periodic graphene sheet containing fifty carbon atoms per cell, using both the September 2010 version of CASINO 2.6 (solid-red line) and our newly modified version of CASINO (dotted blue line). For comparative purposes 'ideal linear scaling' is shown by the solid black line. Fixed target of 486000 for total walker population. Note that fixing the total target population can introduce considerable inefficiency at higher core counts and this graph should not be looked on as representing the general scaling behaviour of the CASINO program. This inefficiency can generally be decreased by increasing the number of walkers per core.*

In Fig. 3 we display timing data of this nature obtained with the Cambridge QMC program CASINO [8, 10] in a typical DMC simulation for a system of one water molecule adsorbed on a graphene sheet represented by a 2D periodic cell containing fifty carbon atoms. The initial number of walkers and the subsequent target population is fixed at 486000. The graph shows the parallel performance of CASINO on the JaguarPF machine (a Cray XT5 machine with 224256

cores, at Oak Ridge National Laboratory, U.S.A. currently - that is, Jan 2011 - listed in second position on the Top 500 supercomputer list). Because of the nature of the multi-core processors on this machine the core count must be a multiple of twelve. We therefore start with 684 cores, and progressively double it seven times. The CPU time taken on 648, 1296, 2592, 5184, 10368, 20736, 41472 and 82944 cores to do one block of ten DMC statistics accumulation moves for 486000 configurations was, respectively, 5787, 2867, 1452, 742, 389, 230, 159, 216 seconds. In plotting the data we have rescaled it (by dividing by the time taken for the 1296-core case and multiplying by 1296) in order to display the deviation from linear scaling. Focussing on the red solid line for CASINO 2.6 in the diagram, one can see that that pretty good scaling is obtained up to around 20000 cores, but beyond that the performance starts to fall away, and the code is actually slower the more processors are used beyond around 50000 cores.

Is it possible to improve this behaviour? One of us (MDT) - having finally been allowed near a computer where he can routinely use more than 500 processors - has recently investigated this problem [49]. It turns out to be possible to substantially improve the performance, and even to effectively eliminate the cost of walker redistribution completely. The improved performance was obtained via the following strategies:

(1) The use of asynchronous, non-blocking communications. CASINO uses the standard Message Passing Interface (MPI) [48] to handle interprocessor communication. Using this software to send a message from one processor to another, one might typically call blocking `MPI_SEND` and `MPI_RECV` routines on a pair of communicating processors. All other work will halt until the transfer completes. However, one may also use *non-blocking* MPI calls, which allow processors to continue doing computations while communication with another processor is still pending. Another advantage is that if used correctly, some internal MPI buffers may be bypassed with a dramatic increase in the communication bandwidth. On calling the non-blocking `MPI_ISEND` routine, for example, the function will return immediately, usually before the data has finished being sent.

Bearing this in mind, the CASINO DMC algorithm has now been modified to do something like the following:

```
MOVE 1

- Move all currently existing walkers forward by one time step

- Compute the multiplicities for each walker (the number of copies of each
 config to continue in the next move).

- Looking at the current populations of walkers on each processor, and at the
 current multiplicities, decide which walkers to send between which pairs of
 processors, and how many copies of each are to be created when they reach
 their destination.
```

```
- Sending processors initiate the sends using non-blocking MPI_ISENDs; receiving
 processors initiate the receives using non-blocking MPI_IRECVs. All continue
 without waiting for the operations to complete.

- Perform on-site branching (kill or duplicate walkers which require it on any
 given processor).

MOVE 2 AND SUBSEQUENT MOVES

- Move all currently existing walkers on a given processor by one time step (not
 including  walkers which may have been sent to this processor at the end of the
 previous move).

- Check that the non-blocking sends and receives have completed (they will
 almost certainly have done so) using MPI_WAITALL. When they have, duplicate
 newly-arrived walkers according to their multiplicities and move by one time
 step.

- Compute the multiplicities for each moved walker.

- Continue as before
```

It was also found necessary to:

(2) Parallelize the procedure for deciding which walkers to send between which pairs of processors, as follows.

Having received a report of the current population of walkers on each core, the master computes a set of instructions for the walker transfers. The algorithm aims to produce the fewest possible number of transfers by carefully matching the requirements of receiving processors (those with a population less than the target) and the availability and multiplicity of surplus walkers on sending processors (those with a population greater than the target). For example, if processor A had a deficit of five walkers and processor B had a surplus of one walker with a multiplicity of four, then both target populations could be satisfied by the transfer of one walker (which would duplicate itself four times on arriving at the destination processor). This is quite clearly more efficient than having five separate processors transfer one walker each to processor A, or by processor C sending five separate walkers each with a multiplicity of one to processor A.

Unfortunately doing this process carefully and exactly (working out on the master a set of optimally-efficient instructions for each processor, and sending the instructions from the master to the slaves) scales linearly with the number of processors, and eventually the cost of working out the most efficient transfers becomes more expensive than doing the transfers themselves. This is the sort of thing that is easily missed in formal analyses, but for very large numbers of

processors this was the rate limiting step in CASINO. It is quite easy to fix, for example, by only considering transfers in small 'redistribution groups' of around 500 cores which are large enough for a full set of 'good matches' to be made. Having done this for our test case the time taken to create and broadcast the whole set of transfer instructions for the ten-move block was only a second or two, independent of the number of processors, rather than a few hundred seconds as was the case on 82944 processors of the JaguarPF machine.

Taken together, these improvements (together with some other more minor refinements) have effectively removed the cost of redistributing and branching in CASINO, as shown by the timings in Table 1. For the largest number of cores studied, the new redistribution algorithm was over 270 times faster. The improvements to the scaled timing data are also shown in Fig. 3 as the dashed blue line (in terms of raw data, the CPU time required for a ten move block of DMC statistics accumulation moves on 648, 1296, 2592, 5184, 10368, 20736, 41472 and 82944 cores was, respectively, 5767, 2883, 1462, 743, 382, 221, 115, and 68 seconds).

Number of cores	Time, CASINO 2.6 (s.)	Time, Modified CASINO (s.)
648	1.00	1.05
1296	3.61	1.27
2592	7.02	1.52
5184	18.80	3.06
10368	37.19	3.79
20736	75.32	1.32
41472	138.96	3.62
82944	283.77	1.04

Table 1: *CPU time taken to carry out operations associated with redistribution of walkers between processors in the original version of CASINO and in the newly-modified version, during one ten-move DMC block for a water molecule adsorbed on a 2d graphene sheet (these numbers include ten moves of DMC equilibration, and should be roughly halved to compare with the times quoted in the text).*

So while this represents a great improvement for large core counts (the total CPU time required for the calculation on 82944 cores was over three times faster than before) the scaling is still not linear with the number of processors. Why? We shall use an alternative way of doing the scaling calculations to illustrate. Previously we used a fixed target number of walkers and a fixed number of DMC moves for all the calculations. Every time the core count was doubled, the number of configs per processor was halved, and by 82944 cores there are only five or so walkers per processor (down from 750 per processor in the 684-core case). Each processor was able to move these five walkers so quickly that the time taken for various minor tasks normally considered unimportant became significant in determining the scaling. The rate limiting step in the 82944-core case turned out to be the summing of the energies and associated quantities over all the processors using an `MPI_REDUCE` operation, in preparation for computing averages over the nodes, an operation which is difficult to make any more efficient than it already is. Interestingly, the cost of walker transfers using the new algorithm was negligible by comparison.

It is vital therefore to ensure that each processor has enough to do during each move of the entire ensemble of configurations, and a much better way of utilizing a massively parallel machine (if it turns out to be possible) is to consider a large-enough fixed target number of walkers per processor, rather than a fixed total target population. If we start with N walkers we can, at least in principle, decrease the error bar on the answer by the same amount either by doubling the number of walkers to $2N$ or by moving the N walkers for twice as many moves. Only in the latter case do we double the amount of interprocessor communication required. We have therefore redone the calculations using a fixed target of 100 walkers per processor. Every time we double the processor count the total number of walkers doubles and, in order that we maintain the 'system size' - defined earlier as the total number of sampling configurations of the configuration space to get a fixed error bar - we halve the number of moves (one cannot of course do this indefinitely!). In such a case doubling the number of processors should halve the required CPU time as before. Note that all we are really doing here is changing how many samples we do on each core between each global communication; we are exploiting the freedom that we are allowed in choosing the number of moves and walkers to make sure that there is enough work for the processors to do during a move. We are, in effect, improving the ratio in Eq. 27 in the way suggested. So, the resulting graph is shown in Fig. 4, and we see that now the processors have enough to do, the scaling is essentially linear with our new, updated algorithm. Our modifications have also significantly improved the behaviour of CASINO relative to the current 2.6 version. For the 82944-core case, the modified version was more than 30% faster and the total cost of redistributing walkers (including the DMC equilibration) went down from 412 seconds to 1 second, demonstrating that the overhead from this process has now been essentially eliminated. Clearly, these results imply that it will be possible in the future to use CASINO on machines with numbers of cores well in excess of 100000.

3.2 Other considerations for large parallel QMC computations

The moral of the previous Section appears to be to use as many walkers as we can on each processor in order to maximize the efficiency on a parallel machine. This leads us to other considerations since, as we have already stated, the number of walkers per processor will be constrained by the available memory. Furthermore, the memory architecture of modern multi-core processors can be somewhat complicated and it is important to ensure that CASINO can take full advantage of this. How can the memory best be utilized on a massively parallel machine?

Moving the walkers requires repeated evaluation of the wave function $\Psi(\mathbf{R}_1 \dots \mathbf{R}_N)$, that is, the values of the single particle orbitals ϕ_j, various determinants of these orbitals, and the many-particle Jastrow factor. The information required to evaluate these must be held in memory. The largest block of data to store is generally the orbital coefficients (the coefficients in the linear expansion over basis functions used to construct the orbitals) and, as we have seen, there is a choice of various different basis set to represent the orbitals in CASINO including Gaussians, Slater functions, blips (B-splines), and plane waves.

Plane waves are generally obsolete in QMC; although CASINO supports them for historical

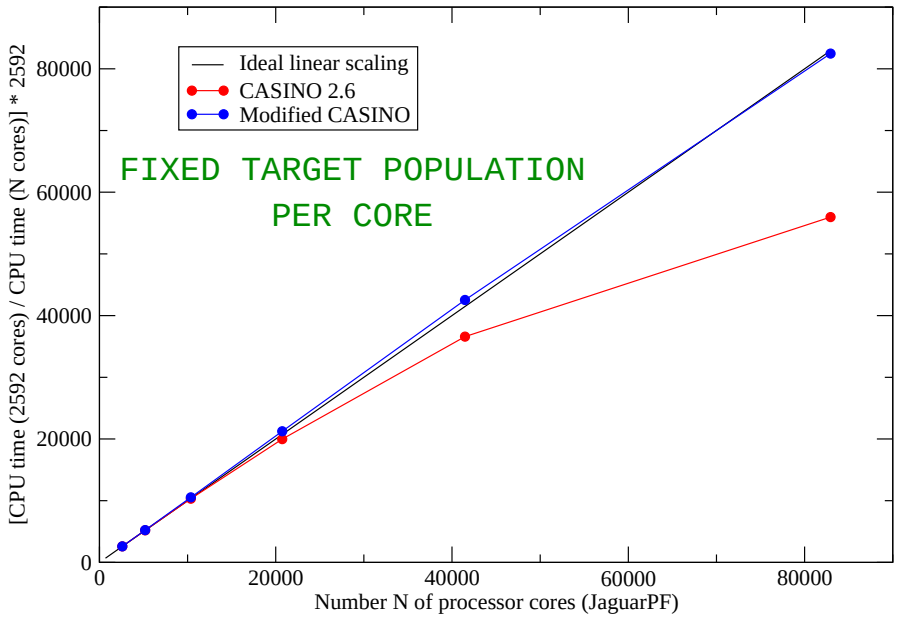

Figure 4: *Scaled CPU time required by various numbers of cores of the JaguarPF machine to carry out one ten-move DMC statistics accumulation block for a water molecule adsorbed on a two-dimensionally-periodic graphene sheet containing fifty carbon atoms per cell, using both the September 2010 version of CASINO 2.6 (solid-red line) and our newly modified version of CASINO (dotted blue line). For comparative purposes 'ideal linear scaling' is shown by the solid black line. Fixed target of 100 per core for total walker population.*

reasons, their use is not recommended since the number of delocalized plane waves to evaluate at a point is proportional to system size and this multiplies an extra factor of N into the scaling of CPU time with the number of particles. Physicists who refuse to use anything else in DFT calculations - of which there are many - need not worry, since the output of plane-wave DFT codes such as CASTEP, PWSCF and ABINIT can be easily transformed into a basis of localized functions which CASINO also supports.

In QMC, therefore, we prefer to use strictly-localized basis functions since then only a fixed number of them are non-zero at any randomly-chosen point, no matter what the size of the system. We may choose atom-centred functions such as the Gaussians and Slater functions widely-used in quantum chemistry, or functions localized on a three-dimensional grid which are unaware of the existence of atoms. Blips, which are 3rd-order polynomials strictly localized in particular boxes surrounding each grid point, are an example of the latter type [50]. In terms of memory use Gaussians and Slaters provide a very compact representation (Slaters slightly more so) since the total number of basis functions is relatively small. They are, however, somewhat less efficient to evaluate than blips. This is because they require the evaluation of exponential

functions as well as polynomials, and because at any random point where an electron may end up it is not known in advance which functions are non-zero there, and one ends up having to dynamically screen them.

Blips, like plane waves, have the advantage of being systematically improvable, universal, and unbiased. They are in fact closely related to plane waves: each single particle orbital ϕ_j expressed as a linear combination of plane waves (that is, as $\phi_j = \sum_{\mathbf{G}} c_{i\mathbf{G}} \exp(i\mathbf{G} \cdot \mathbf{r})$ where $\mathbf{G}$ is a wave vector and $c_{i\mathbf{G}}$ are complex orbital coefficients) can also be approximately expressed as a linear combination of blip basis functions, that is, as $\phi_j \sim \sum_n a_{in} b_n$, where b_n is the blip function sitting on grid point n. The grid spacing is closely related to the plane-wave cutoff, and the set of coefficients a_{in} can be obtained from the set of coefficients $c_{i\mathbf{G}}$ using Fast Fourier Transform routines [50].

For most purposes, blips are probably our preferred basis set in QMC. The only problem is that there are generally a hell of a lot of them, and the orbitals coefficients a_{in} can require a great deal of memory to store. The number of these coefficients is in fact proportional to the square of the number of electrons in the system, with a pre-factor that depends on the fineness of the grid (fine grids are required in particular for hard pseudopotentials). For large enough systems the memory needed to store the coefficients can be of the order of several Gb, which often exceeds the memory available on a single core. This problem has recently been alleviated by allowing CASINO to exploit the architecture of modern multi-core processors which share a common memory on a node (currently JaguarPF has twelve cores per node, and the UK national supercomputer HECToR twenty-four cores per node, with sixteen and thirty-two Gb of memory per node respectively). This is done by having only one of the cores on the node allocate the required memory to store the wave functions, while allowing all cores access to the shared memory area. This modification is now allowing simulations of systems more than one order of magnitude bigger than previously possible.

In our scaling experiments we have not concerned ourselves with aspects of real practical calculations, such as whether the error bar on the result is small enough. Furthermore, in repeatedly doubling the number of walkers we have blithely and repeatedly halved the number of moves without considering whether the number of moves left is great enough that we can perform a proper statistical analysis of the sampled data (a procedure called 'reblocking' [51] is normally applied to the sampled data in order to compute an accurate statistical error bar on the DMC energy). Under certain circumstances (say, if we fix in advance a required error bar and demand at least a given number of moves are performed) then our ability to use the maximum number of walkers allowed by the available memory will be reduced.

Consider that for a pure-MPI QMC calculation with N_{proc} processors, the total CPU time t is roughly given by $t \approx N_{\mathrm{target}} N_{\mathrm{move}} t_{\mathrm{move}} / N_{\mathrm{proc}}$, where N_{move} is the number of moves, N_{target} is the target walker population and t_{move} is the average time to move one walker at each step. On very large machines one can easily be in a situation where N_{proc} is great than the required N_{move} which means that there will be processors with no associated walkers at all; they are therefore forced to be idle and this is a waste of resources. An additional refinement is suggested for such cases where it is *unnecessary* to use as many walkers per processor as possible, namely the

adding of a second level of parallelisation, capable of splitting each walker over more than one core, over and above the MPI parallelism. In CASINO this has been realized using OpenMP directives; OpenMP [53] is an implementation of multithreading, a method of parallelization whereby the master 'thread' (a series of instructions executed consecutively) forks a specified number of slave threads and a task is divided among them. The threads then run concurrently, with the runtime environment allocating threads to different processors. This second level of parallelism becomes useful when $N_{\rm proc} > N_{\rm target}$. Running multiple threads on multiple cores allows keeping $N_{\rm target}$ small, effectively reducing $t_{\rm move}$ in the cost formula above.

The general strategy of this implementation is to use OpenMP parallelism for the loops whose trip counts scale with the number of electrons or atoms. In the QMC algorithm the basic logical units that need to be parallelized are functions like orbital evaluation, Jastrow factor evaluation, inverse Slater matrix updating, potential energy evaluation, and electron-electron and electron-nucleus distance evaluation. In practice this is done by defining subgroups or 'pools' of small numbers of cores. Parallelisation over walkers is maintained over pools, but inside each pool the work to be performed by each walker is parallelised by splitting the number of orbitals over the pools (this of course helps to address the memory problem in general). Then, each core in the pool will only evaluate, for example, the value of a subset of orbitals. When this is done, all the cores within the pool communicate to construct the Slater determinants, which are evaluated again in parallel using the cores in the pool. This turns out to be efficient only if the walkers are split among a small number of cores (typically two, and not more than four).

A good example of when this might be important is when calculating the total energy of solids. In QMC one cannot reduce this problem to the primitive cell as in DFT calculations, and one must in general try to eliminate finite-size effects by extrapolating to the infinite system size limit; this is done through calculations on supercells formed by periodically repeating a number of primitive cells. In these types of calculations one is obviously interested in the energy per primitive cell, and therefore the many primitive cells that build the supercells used in the simulations all contribute to reduce the statistical errors on the energy per primitive cell. A consequence of this is that with large supercells the number of necessary walkers is reduced, and therefore parallelisation over walkers becomes less efficient.

We discuss finally an additional bottleneck related to *DMC equilibration*. Having first computed an initial set of walkers distributed according to the chosen VMC trial function, we must propagate the walkers for a certain amount of imaginary time until the ensemble becomes distributed according to the ground-state DMC wave function, that is, it becomes 'equilibrated'. Once this is the case the DMC statistics accumulation phase begins and we begin to average data to compute the final answer and its error bar. As we have already mentioned, the total number of walkers per processor is limited not only by the available memory, but also by the fact that carrying out DMC equilibration takes longer if more walkers are used. An interesting idea suggested by Neil Drummond [52] and recently implemented by him in CASINO shows us how to greatly alleviate this problem. Having decided on a target walker population $N_{\rm target}$ for the statistics accumulation phase one can initially generate and equilibrate a much smaller number of walkers than the target. This can be done very rapidly and once equilibration is achieved, rather

than beginning the usual immediate accumulation of statistics, one does a quick 'configuration generation DMC run' with this small population. That is to say, widely-spaced configurations distributed according to the equilibrated ground state DMC wave function are saved for use as the initial population in the subsequent big DMC statistics accumulation run. There is then relatively little correlation between the walkers in the initial large population. This would of course not be the case if the target population were simply increased to N_{target} after a small population equilibration; large numbers of the initial walkers would be highly correlated. Preliminary results indicate that the answer is the same to within error bars but the time taken for DMC equilibration can be reduced by an order of magnitude or more.

3.3 Conclusions about QMC scaling on massively parallel machines

Which of our two scaling graphs produced by our modified CASINO (Fig. 3 or Fig. 4) most accurately represents what will happen in real-world simulations? In answering this, we have to remind ourselves what million-core machines are actually for.

Note first that because QMC is a sampling technique, then for any given system, there is a maximum number of processors you can exploit if you insist that your answer has no less than some required error bar and that it has a minimum number of moves (so that we can perform a reblocking statistical analysis of the result and its error bar).

For example, let's say that your system requires 1000000 random samples of the wave function configuration space to get the required error bar ϵ. Let's say we need at least 1000 sampling moves to accurately reblock the results. And let's say we have a 1000 processor computer. In that case only one walker per node is required to get the error bar ϵ (even though the available memory may be able to accommodate many more than this).

Let's say we now buy a 2000 processor machine. How do we exploit it to speedup the calculation? We can't decrease the number of moves, since then we can't reblock. It is wasteful to just run the calculation anyway, since then the error bar will become smaller than we require. We can split each walker over two nodes, and use OpenMP to halve the time taken to propagate the walkers, but let's say we find that OpenMP doesn't really work very well over more than two processors.

How then do we exploit a 4000 processor machine? Answer - we can't. The computer is simply too big for the problem if you don't need the error bar to be any smaller.

Now it is possible to imply that our first graph in Fig. 3 (which is not linear scaling with the number of processors, even for the modified CASINO) is more representative of real calculations than our second graph in Fig. 4 (which is linear scaling for the modified version), but a more fundamental observation is that Fig. 3 is just running into the limitations of the method. In the example above, one could not talk about the scaling of the problem to 100000 cores, in the same way as it would be silly to use a 100000 core machine to do a Hartree-Fock calculation of a hydrogen atom, but it doesn't mean we can't talk about the theoretical scaling of CASINO on that many processors for a general system, and in general it seems to be the case that, following

our modifications, CASINO is now linear scaling with the number of processors providing the problem is large enough to give the processors enough work to do. This should normally be easy enough to arrange, and if you find yourself unable to do this, then you don't need a computer that big. One may conclude that massively parallel machines are now increasingly capable of performing highly accurate QMC simulations of the properties of materials that are of the greatest interest scientifically and technologically. In the next Section, we therefore turn to some examples of how these techniques may be applied in practice.

4 Examples of current QMC work

In choosing examples of current QMC work to illustrate the importance of petascale computing, we naturally concentrate on work that we know best, in other words the work that we are involved in. The illustrations that follow are all based on calculations that are being performed on machines such as HECToR and JaguarPF, and we choose these illustrations because we believe them to be scientifically important, and because they are known to be difficult for DFT. We start with the surface formation energy of materials, focusing particularly on ionic and semi-ionic materials. We then turn to the problem of predicting the adsorption energies of molecules on surfaces, with particular reference to water on the surface of ionic materials and on graphene. Our final example concerns water and ice, where the difficulties encountered by DFT are notorious.

4.1 Surface energy of materials

When a macroscopic sample of a material is separated into two pieces, new surfaces are created. The work done in creating the new surfaces divided by their area is the surface formation energy, usually denoted by σ. For a crystal, σ depends on the orientation of the surface. Surface formation energies are important in fields as far apart as geology and fracture mechanics. They are particularly important in nanoscience, because in particles a few nm across a significant fraction of the atoms are at or near the surface. One of the consequences is that the crystal structures of nano-particles are sometimes not those of the bulk material, because it may be advantageous to adopt a less stable bulk structure if the energies of the surfaces are lowered [54]. For nano-particles supported on substrates, the equilibrium form of the nano-particles is often determined by the balance between and surface and interfacial energies.

DFT has quite serious problems in predicting σ, and it has been known for a long time that predicted σ values can depend strongly on the exchange-correlation functional [55]. For example, as a broad generalisation, it seems that GGA approximations tend to give σ values that are about 30 % less than LDA values [56]. Even more surprising is that LDA values sometimes seem to agree better with experiment than their GGA counterparts [57]. This is unexpected, because common sense suggests that the formation of a surface involves the breaking of bonds, and GGA is usually much better than LDA for bond formation energies. The problem is that σ values are not easy to measure experimentally, and those that have been measured are sometimes subject

to large errors. In this unsatisfactory situation, there is an obvious need for accurate computed benchmarks for σ, which can be used to assess both DFT predictions and experimental values.

In our first attempt to compute the surface formation energy of a real material using DMC, we studied the MgO(001) surface [58]. This is one of the few ionic materials for which σ has been measured experimentally, though there are considerable differences between different measurements. For what it is worth, the experiments tend to support the LDA value of 1.24 J m^{-2}, rather than the PBE value of 0.87 J m^{-2}. Five years ago, we were able to show the feasibility of computing σ to good accuracy in slab geometry, using slabs of different thicknesses and attempting to extrapolate to the limit of infinite thickness. The DMC value of σ of 1.19 J $^{-2}$ supported the correctness of LDA, but at that time there were still uncertainties about pseudopotential errors. Since that time, the ability to run calculations using large numbers of cores on machines such as the HECToR and JaguarPF supercomputers has made the DMC calculation of surface energies much easier, and it is already clear that such calculations will become fairly routine in the near future.

Very recently, a technique has been developed that in some cases allows the cohesive energy, equilibrium structure and elastic properties of perfect crystals and the surface formation energies of crystals to be calculated using wave function-based quantum chemistry at the CCSD(T) level [59, 60]. This so-called 'hierarchical method' offers for the first time the possibility of testing high-level quantum chemistry and DMC against each other for surface formation energies. Using CASINO on HECToR and JaguarPF, we have recently made comparisons of this kind for the LiH and LiF crystals, both of which have the rock-salt structure and are well suited to the 'hierarchical' quantum chemistry approach. In the case of LiH, we have been able to perform all the DMC calculations both with pseudopotentials and at the all-electron level [60]. With all-electron calculations, the only remaining error is fixed-node error (and, of course, system-size errors – but these can be systematically eliminated).

Before discussing the calculations of σ, it is worth commenting on the outstanding agreement with experiment given by all-electron DMC for the properties of the LiH crystal [60]. The equilibrium lattice parameter a_0 agrees to within 10^{-3} Å (both experiment and DMC give 4.061 Å at $T = 0$ K, once corrections for zero-point effects are made). The cohesive energy agrees with the experimental value to within 20 meV per formula unit, which is comparable with the experimental uncertainty (once again, zero-point corrections are crucial). The hierarchical quantum chemistry approach gives similar accuracy – possibly even slightly better for the cohesive energy [59]. Needless to say, there are no adjustable parameters in either method. This gives us every reason to expect excellent values of the surface formation energy.

The DMC slab calculations on LiH went up to slab thicknesses of 6 ionic layers, with 18 ions per layer in the repeating cell (the total number of ions per repeating cell thus went up to 108), and these slab thicknesses are more than enough to give convergence of σ to within ~ 1 %. We reproduce in Table 2 our final σ values from both DMC and quantum chemistry, compared with the values from various DFT approaches [60]. The agreement between all-electron DMC and high-level quantum chemistry is extremely close, as it should be, and the σ values appear to provide robust benchmarks against which to judge DFT. Just as we found for MgO, LDA

Method	σ / J m^{-2}
DMC pseudo	0.373(3)
DMC all-elec	0.44(1)
Quantum chem (frozen core)	0.402
Quantum chem (with core)	0.434
DFT(LDA)	0.466
DFT(PBE)	0.337
DFT(rPBE)	0.272

Table 2: *Calculated formation energy of the LiH (001) surface with both pseudopotential and all-electron DMC, with hierarchical quantum chemistry with and without core correlation, and with DFT approximations. For details, see Ref. [60].*

performs rather well, the PBE value is about 30 % lower and is considerably less good, while the revised PBE value (rPBE) is ~ 40 % below the correct value.

We have recently completed similar calculations on σ for LiF (001) [61], and once again we find values that support LDA, with PBE as usual underestimating by ~ 30 %. There now appears to be no reason why DMC calculations of this kind cannot be run for a range of other materials, including semiconductors and metals. This would be valuable, because it would allow a much more systematic appraisal of DFT approximations for surface formation energies than has been possible hitherto.

4.2 Molecular adsorption on surfaces

The adsorption energies of molecules on surfaces are important for innumerable reasons. A good example is the process of gas sensing, in which a trace concentration of a substance (for example a pollutant) in the atmosphere is in thermal equilibrium with molecules of the substance adsorbed on the surface of a material (for example tin oxide), the change of whose electrical properties is used to monitor the atmospheric concentration. One of the main quantities determining the concentration of absorbed molecules is the adsorption energy: the binding energy of the molecule to the surface. Adsorption energies are important in many other fields, including catalysis, corrosion, gas purification, chemical reactions in interstellar space, and atmospheric processes. But in spite of their widespread importance, our quantitative knowledge of adsorption energies remains rather poor. One of the major obstacles to progress is that DFT has rather poor predictive power for the energetics of adsorption.

A famous example of DFT failing to give the right answer is the adsorption of CO on the Pt(111) surface, where it is known experimentally that the most stable adsorption site is the atop sites but most DFT approximations predict the hollow site to be most stable [62]. An equally severe problem is that DFT values of molecular adsorption on transition metals can be seriously in error, sometimes by as much as 0.5 eV [63]. But the difficulties are not confined to transition metals. For some problems, van der Waals dispersion can play a large role in binding

the molecule to the surface, and here the well known difficulties of DFT in accurately treating dispersion can cause major problems. The adsorption of water on graphene is a famous example of this kind, but non-local electronic correlation appears to be important for many other systems, including molecular adsorbates on metal surfaces [64]. Even for apparently simple systems such as the adsorption of water on oxide surfaces, where one might imagine electrostatics to be the dominant mechanism, DFT values of adsorption energies can easily shift by a factor of two when the exchange-correlation functional is changed [65].

In some cases, good experimental values of adsorption energies may be available, but often the experiments are too difficult, or too difficult to interpret, perhaps because of surface defects or because of interactions between the adsorbate molecules. There is clearly an urgent need for reliable and accurate benchmark values of adsorption energies for selected systems, just as for surface formation energies. There has recently been important progress in the use of wave-function-based correlated quantum chemistry techniques for the calculation of adsorption energies on extended surfaces [66, 67]. But quantum Monte Carlo certainly offers another way of getting these benchmarks [68], and we have recently started investigating how to do this.

An important paradigm example for benchmark purposes is the adsorption of H_2O on the LiH(001) surface. So far as we know, there is no experimental data on the adsorption energy in this case, and indeed it seems unlikely that that there will be any, because water reacts very exothermically with LiH to form the hydroxide. Nevertheless, the system is an important test-bed for the development of techniques, because of its simplicity (there are only four electrons per formula unit in LiH, and the crystal has the very simple rock-salt structure), and because there is every reason to expect that very accurate calculations of the energetics of H_2O adsorption can be achieved using wave-function-based correlated quantum chemistry via the incremental approach [66, 67].

The H_2O/LiH system is also a fascinating target for calculations, because DFT predictions of the binding energy depend very strongly on the exchange-correlation functional. To illustrate this, we show in Fig. 5 the binding-energy curves computed with a number of different functionals, including the widely used PBE functional, as well as the BLYP and revPBE (revised PBE) functionals. None of these functionals accounts for van der Waals dispersion, so to get an indication of the likely effect of this, we also include some results obtained with long-range correlation included using the scheme of Grimme [69], which has come into quite common use in the past few years. Following the usual practice, we refer to the schemes where dispersion is added to PBE, BLYP and revPBE as PBE-D, BLYP-D and revPBE-D. The binding energy curves were computed by holding the geometry of the H_2O molecule rigid and moving it along the surface normal, with the water-O nearly above a surface Li ion. The spread of DFT predictions is surprisingly large, with the calculated adsorption energies at the minimum ranging from 0.1 to 0.3 eV. This illustrates our comment about the poor predictive power of DFT for some adsorption problems.

With CASINO running on HECToR and Jaguar, we have found it possible to compute the corresponding binding-energy curve with DMC with very small statistical error bars of less than 10 meV and with thick enough slabs and large enough repeating surface units cells to reduce

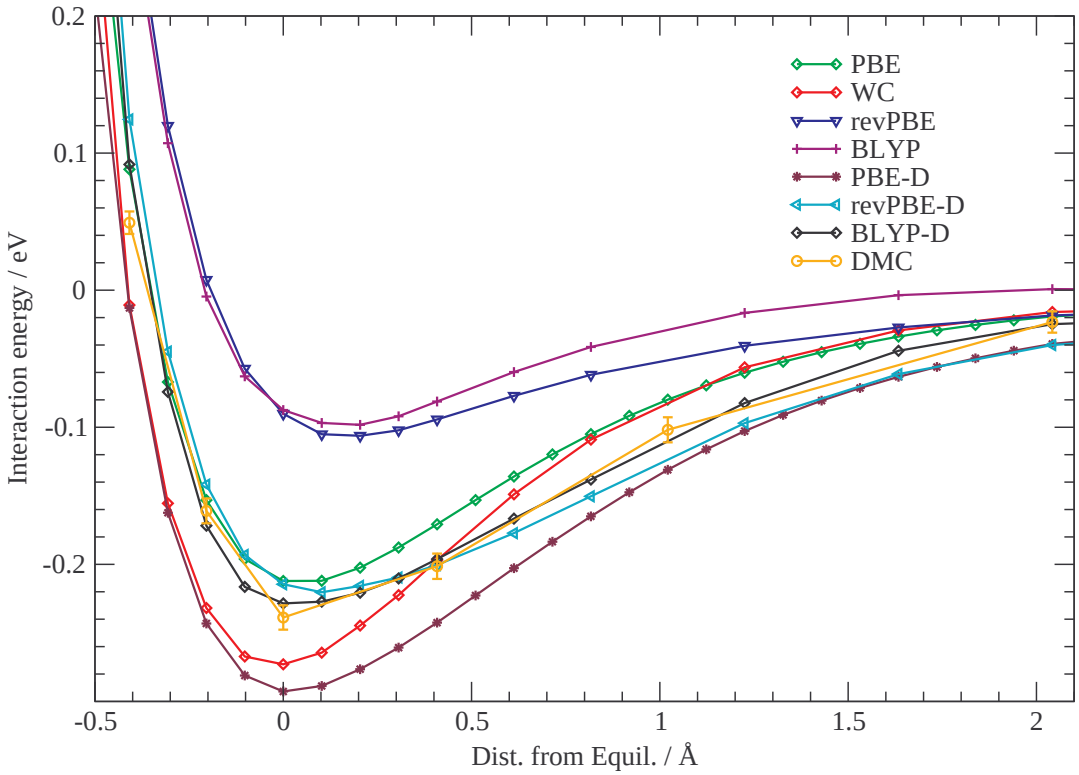

Figure 5: *Binding-energy of the H_2O molecule with the LiH (001) surface. Preliminary DMC results (yellow symbols with statistical error bars) are compared with DFT predictions using a variety of exchange-correlation functionals. The H_2O molecule has exactly the same (fixed) geometry and the same (fixed) orientation in all the calculations, and only the molecule-surface distance is changed.*

the size errors to a similar size. Technical details of the calculations, together with final results will be reported in a journal publication, but we compare our preliminary results with the DFT binding-energy curves in Fig. 5. The comparisons indicate that the PBE approximation is quite accurate, so long as no attempt is made to include dispersion, but the agreement becomes rather poor once dispersion is included. Fortuitously or otherwise, approximations such as revPBE that are seriously in error without dispersion are greatly improved with it. These comments assume, of course, that pseudopotential and fixed-node errors in DMC are negligible. Fortunately, as we shall report elsewhere, we have been able to demonstrate with high-level quantum chemistry calculations based on the incremental scheme that these DMC errors probably amount to no more than $\sim$ 10 meV over the range shown in the figure, except perhaps at very short distances [70]. This implies that the accuracy of DMC for the adsorption energy in this system exceeds chemical accuracy by a very large factor.

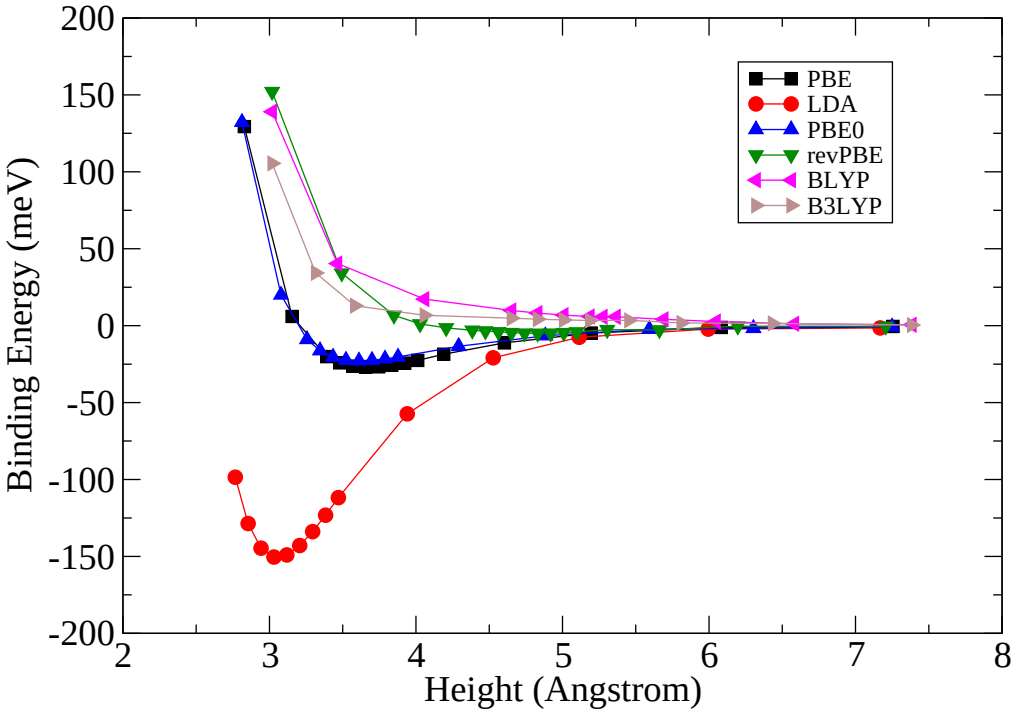

Figure 6: *Shows binding-energy curve for H_2O molecule on graphene calculated with various DFT approximations.*

Can adsorption energies for other systems be calculated beyond chemical accuracy using DMC? For water on the surfaces of other ionic materials, including MgO, our tests suggest that the answer to this question is 'yes', in the sense that DMC statistical errors and system size errors can be reduced well below the threshold of chemical accuracy, and our comparisons with high-level quantum chemistry on appropriate clusters indicate that pseudopotential and fixed-node errors also fall below this threshold. Large-scale calculations are now underway, and we hope to present results very soon.

Our work on the water-graphene interaction is motivated by the huge contemporary activity on carbon systems. The extraordinary variety of carbon structures, from graphite to buckyballs to single- or multi-walled nanotubes to graphene has inspired a vast range of scientific studies, as well as ideas for practical applications. The award of the 2010 Nobel Prize in Physics to Geim and Novoselov for their work on graphene recognises the importance of the field. The interaction of water and other molecules with these structures is often vital. Remarkably, it has been shown that small changes in the H_2O-carbon interaction can make all the difference to whether a nanotube fills with water or not [71]. Recent work shows that a small variation in the strength of the water-carbon bond leads graphite surfaces to switch between hydrophobic and hydrophilic behaviour [72]. Technological applications where the H_2O-carbon interaction is crucial include device biosensing and nanofiltration.

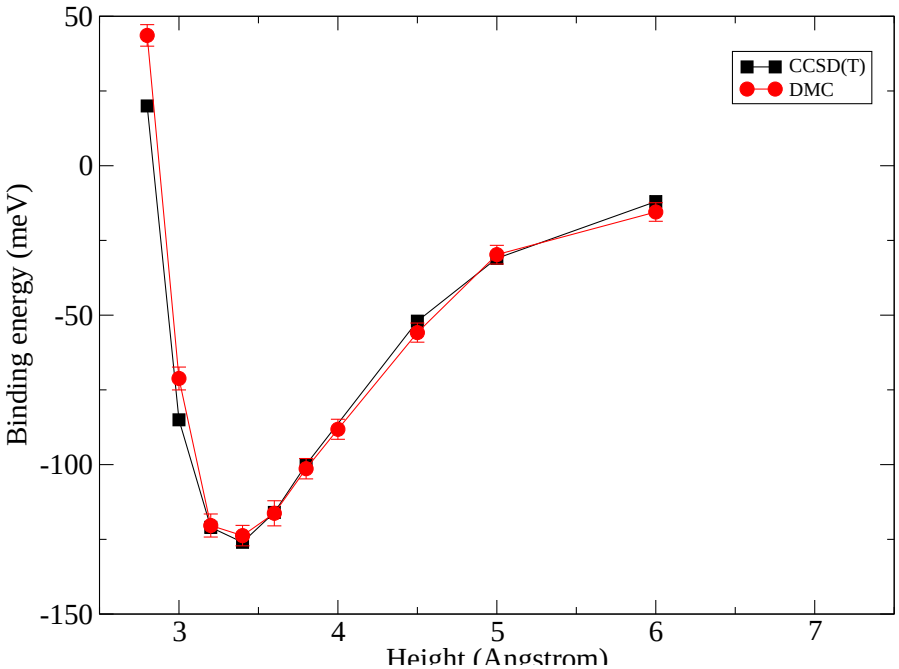

Figure 7: *Shows binding-energy curve for H_2O molecule interacting with the benzene molecule: comparison of DMC with CCSD(T).*

For all these reasons, an accurate quantitative understanding of the water-graphite (or water-graphene) interaction has been sought for many years. Unfortunately, this is a problem where DFT calculations have so far proved virtually useless, because an accurate description of van der Waals dispersion is absolutely essential. To illustrate the problem, we show in Fig. 6 a comparison of the H_2O-graphene binding-energy curve (H_2O geometry fixed, H_2O-graphene distance only is varied) obtained with different DFT functionals. Most of the standard functionals, including 'expensive' hybrid functionals such as B3LYP wrongly give almost no binding (the 'correct' binding energy is believed to be somewhere between 50 and 150 meV). Up to now, the only theoretical approach that has any credibility at all has been high-level quantum chemistry [73], but the difficulty here is that it has not so far been possible to perform calculations with a sufficiently good description of electron correlation in periodic boundary conditions. For want of anything better, the approach taken has been to start with the H_2O-benzene interaction, and then to go to the interaction of H_2O with coronene and larger acenes, and to attempt to extrapolate to the graphene limit [74]. For the interaction of H_2O with large acenes, the direct application of CCSD(T) is not possible, but there are good reasons for believing that the SAPT approximation (Symmetry-Adapted Perturbation Theory) [75] gives an accuracy that is close to that of CCSD(T), so that in practice the calculations have been based on SAPT.

In this challenging situation, the ability to establish accurate benchmarks for the H_2O interaction would be enormously important. But does DMC have the required accuracy, and can it be applied with small enough statistical errors to large enough periodic systems of H_2O-graphene?

To test the accuracy of DMC, one of us in collaboration with other groups has recently compared

DMC with high-level quantum chemistry for the interaction of the water molecule with the benzene molecule [76]. This interaction is relatively weak, and it was crucially important on the quantum chemistry side to employ the 'gold-standard' CCSD(T) technique extrapolated very close to the basis-set limit. For the DMC calculations, the system is small enough that statistical errors can be reduced to the level of ~ 3 meV with fairly modest computational resources. The comparison of the binding-energy curves is shown in Fig. 7, the calculations being done with the geometry of the H_2O molecule and the orientation of the molecule held fixed, with only the H_2O-benzene separation being varied. The agreement of the two curves is outstandingly good, with discrepancies being no more than ~ 3 meV except at the smallest separations, and this suggests that the DMC description of van der Waals dispersion and of the weak electrostatic interaction of the water multipole moments with the small quadrupole moments of the carbon atoms is being very accurately described. It also suggests that if the DMC calculations can be scaled up to treat the H_2O molecule interacting with a graphene sheet in periodic boundary conditions, then the accuracy of the computed adsorption energy should be very good, provided the DMC statistical errors can be sufficiently reduced.

We have tested the feasibility of DMC calculations on the periodic H_2O-graphene system using JaguarPF. We find that we can simulate the system by representing the graphene sheet with a 5x5 supercell (50 atoms), which should be large enough to give small finite size errors (k-point errors on the binding energy are less than 1 meV with this cell size within DFT). The simulations can be carried out on JaguarPF on more than 80000 cores, and each single point calculation with a statistical error of ~ 10 meV has a cost of around 200000 core hours.

4.3 Water systems

Water in its different fluid and solid forms is an outstandingly important substance, because it is crucial in so many different fields, ranging from biology to the earth sciences, through chemical engineering, surface science and environmental sciences and medicine. It is also been the source of many controversies, some of which continue to this day [77], and has certainly proved a major challenge to the capabilities of DFT. Because it is so important, attempts to develop empirical intermolecular potentials for water go back nearly eighty years [78], and the creation of new model potentials remains a thriving industry [79]. Nevertheless, the need for descriptions based on electronic structure theory was recognised many years ago, starting with the efforts of Clementi and others in the 1970s [80]. A major landmark was the publication in the early 1990s of the first simulations of liquid water based on DFT [81]. At that time, it seemed possible that quite good accuracy might be achieved using fairly simple exchange-correlation functionals, but it has since became apparent that the early optimism was somewhat misplaced. It is now clear that the predicted properties of liquid water depends strongly on the assumed functional and there is still no general consensus about how the properties of liquid water and the various crystal structures of ice can best be described by DFT [82].

There is, however, one standard electronic structure technique that is generally agreed to provide more than enough accuracy for a fully realistic description of water in all its phases. This is the

CCSD(T) technique (coupled cluster with single, double and perturbative triple excitations), often regarded as the 'gold standard' for the accurate description of the interactions between closed-shell molecules. Benchmark calculations on the energetics of the water dimer, essentially at the basis-set limit [83], have been compared with calculations beyond the CCSD(T) level, and the evidence suggests that the intermolecular interaction energies are correct to within ~ 1 meV. It is also well established that CCSD(T) gives a very accurate account of long-range dispersion for water and other similar molecular systems. There is just one major problem: the computational effort needed scales as N^7, where N is the number of water molecules. This ferocious scaling has so far prevented the application of CCSD(T) with well converged basis sets to bulk water systems. (The feasibility of directly applying the 2nd-order Møller-Plesset approximation to bulk water in periodic boundary conditions was reported recently [84]. It is also interesting to note that CCSD(T) calculations on clusters of up to ~ 20 water molecules have recently been reported using the NWChem code on large parallel computers [85]. These developments suggest that the future use of CCSD(T) for extended water systems may not be completely fanciful.)

In the current unsatisfactory situation, there is a strong need for techniques that can deliver the required accuracy, that can be directly used in periodic boundary conditions, and that do not scale too harshly with N. Quantum Monte Carlo clearly satisfies the second and third conditions, so that, provided its accuracy is good enough, direct QMC simulations on water and ice are likely to be very useful. It was shown several years ago that DMC gives excellent values of the binding energy of the H_2O dimer [86]. A particularly interesting test of DMC is provided by the competing structures of the H_2O hexamer. What makes this interesting is that the hexamer occupies a transitional position between smaller H_2O clusters, whose most stable structures have cyclic geometries, and larger clusters, which are more highly coordinated. The hexamer has stable structures of both kinds, and the energy differences between them are very small (on the order of 5 meV/H_2O), so that the ability to separate them is a stringent test of any technique. The consensus from the best high-level quantum chemistry calculations is that the so-called 'prism' structure is most stable [87], but all DFT approximation that omit dispersion predict the wrong energy ordering for the competing structures [88]. The predictions of DMC, however, agree closely with high-level quantum chemistry, and predict accurate values of the energy differences [88].

Recently, we have made our own systematic tests of DMC against quantum chemistry benchmarks for a wide range of geometries of H_2O clusters from the dimer to the hexamer, some of the geometry sets being obtained by extracting configurations from an MD simulation of bulk liquid water generated using the flexible AMOEBA interaction model. These tests support the high accuracy of DMC. As an example, we show in Fig. 8 the comparison for a thermal sample of 50 geometries of the H_2O trimer. The rms fluctuation of the difference between DMC and benchmark total energies is 130 μHartree (about 3.5 meV). To put this in context, we show the analogous comparison for the DFT(PBE) total energy; the very much larger rms deviation is not too surprising, given the well known rather poor performance of PBE for liquid water.

The evidence that DMC can deliver much better accuracy and reliability than DFT for water

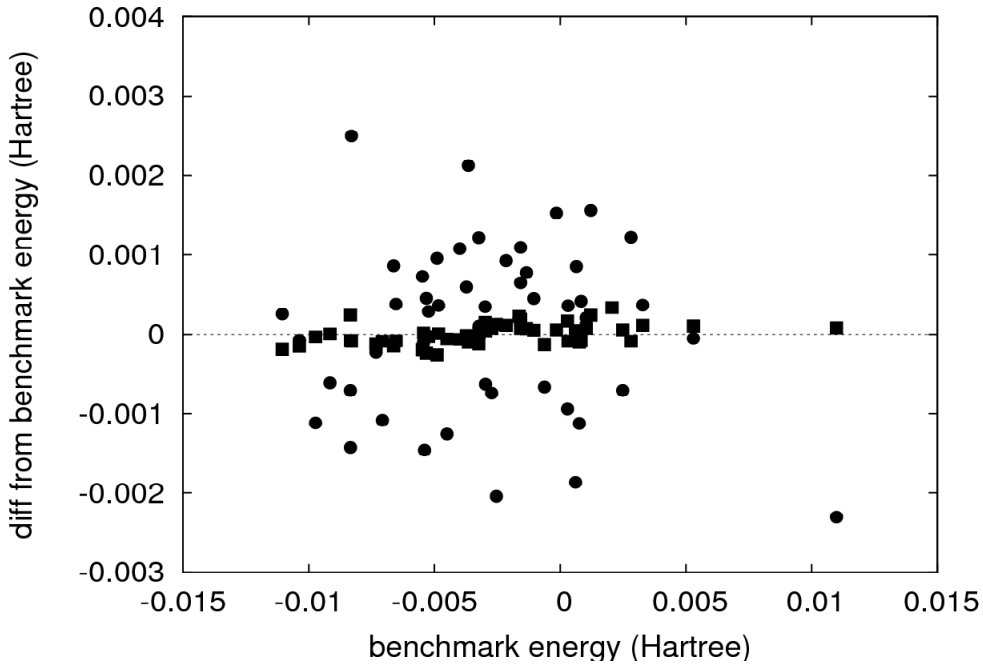

Figure 8: *Comparison of DMC total energies (filled squares) with accurate quantum chemistry benchmarks at CCSD(T) level for a sample of 50 geometries of the H_2O trimer drawn from an MD simulation of liquid water. Horizontal axis shows CCSD(T) energy, vertical axis shows deviation of DMC energy from CCSD(T) energy. Filled circles show the same comparison for DFT(PBE).*

systems is thus strong. But what is the most useful way of applying DMC to improve the atomic-scale understanding of water? Perhaps the most straightforward application is to the energetics of ice structures, since useful things can be done using only static calculations, without the need to compute forces. Our preliminary tests on various ice structures with CASINO on the HECToR and Jaguar machines show that it is possible to achieve very good statistical accuracy (statistical errors of ~ 5 meV per water molecule or better) on the periodically repeated systems of sixty-four to ninety-six molecules or more that are needed to reduce size errors to an acceptable level. We expect to report detailed DMC calculations on the relative energetics of both perfect and defective ice structures in the near future.

A further, but more challenging possibility for the future is the DMC simulation of liquid water itself. The major difficulty here is in the calculation of the forces that would be needed for MD simulation. Recent important progress in the calculation of DMC forces [89] demonstrates that this will eventually be feasible. However, DMC calculations on bulk liquid water in periodic boundary conditions have already been reported, without the calculation of DMC forces. Grossman and Mitas [90] showed several years ago how to compute the DMC energy of bulk water on an MD trajectory generated using conventional DFT. The key idea is that the efficiency of the DMC calculations is improved by exploiting the quasi-adiabatic evolution of the DMC ground state as the nuclear coordinates change with time. This approach does not allow the direct computation of the thermal equilibrium structure and dynamics of the liquid associated with

the DMC ground state energy. Nevertheless, it yields benchmark energetics that could be used to calibrate and 'tune' DFT functionals, in much the same way as DFT molecular dynamics simulations have often been used in the past to test and 'tune' empirical interaction potentials. We plan to explore this possibility with CASINO and the large parallel computers to which we have access.

5 Future prospects

We have tried to show how the availability of petascale computers having tens or hundreds of thousands of cores is opening up completely new possibilities for the techniques of quantum Monte Carlo. The basic reason for this is that diffusion Monte Carlo, operating typically with thousands of semi-independent walkers, lends itself very readily to massive parallelism. We have demonstrated this idea in action by showing how the CASINO code implemented on machines such as the UK national HECToR supercomputer and the US Jaguar supercomputer gives very good parallel scaling up to at least 100000 cores. We have argued that these developments are scientifically important, because for important types of problems QMC gives much greater accuracy than standard DFT methods, so that some of the well known deficiencies of DFT can be overcome. We have illustrated some of the new science that is now becoming possible using examples from three areas: the surface formation energies of materials, the adsorption energies of molecules on surfaces, and aqueous systems, all of these being areas where DFT struggles to deliver trustworthy results. We have also explained our belief that in these and other areas there is a pressing need for accurate benchmarks which can be used to test and calibrate DFT approximations, and we have shown how DMC can deliver these benchmarks in practice.

Until fairly recently, QMC has sometimes been regarded as a bit of a minority interest, because it demands computational resources that are typically 10^4 times those needed by DFT. However, we believe that the arrival of petascale computing, and the prospect of exascale computing in the next five to ten years will change all this. One has to remember that when the Car-Parrinello paper was published over twenty-five years ago, it was sometimes commented that the computational requirements of DFT were completely prohibitive and that the technique would never be widely used for practical problems. Obviously, the critics have been proved wrong. Given that supercomputer power has increased by at least a factor of 10000 over the past fifteen years, it takes little imagination to see that QMC may follow the same kind of evolution. However, a shift of thinking is needed. Today, it is quite common to run atomic-scale materials simulations needing a few thousand core-hours on cheap clusters. But with calculations running on, let's say, 100000 cores, individual jobs consuming several million core-hours become feasible. In fact, from our experience on Jaguar, we know of research groups (not our own) that have run jobs of this size.

It is still not completely straightforward to obtain petascale resources. However, the INCITE Program, which gives access to Jaguar and other 'leadership-class' resources in the US has welcomed applications from international groups for several years now. The European DEISA programme (`www.deisa.eu`) is a good source of computational resources in Europe. It is de-

signed to make use of a Europe wide distributed super-computing environment, and can grant access to resources of the order of millions of core hours. Indeed, part of our work of water on graphene benefited from a DEISA grant. The rapidly evolving situation in China, currently home to the fastest supercomputer in the world, is also well worth watching.

Given the developments that are underway, we believe it is now very timely for more research groups to consider becoming involved in the QMC enterprise. But what is the right way to do this? Here are our personal suggestions about how to think about this:

- Start small and work upwards: Clearly, one should gain experience first with small problems (for example, problems involving small molecules), which can easily be run on modest local resources. Work up from there to more ambitious problems that need large machines.
- Use standard codes: A large development effort has gone into CASINO and other QMC codes, and it makes sense to work with a code that already has a substantial publication record.
- Collaborate: Even more than DFT, there is much that you need to know about QMC before trying calculations. No QMC code can (yet) be treated as a black box, and it is wise to learn from experienced practitioners. To this end, one of us (MDT, with Pablo López Ríos and Neil Drummond) has for the last six years been running the annual 'QMC and the CASINO program' summer school at his monastery in Tuscany, Italy, to which the interested reader is cordially invited [91].
- Choose your problem well: Just like DFT, QMC cannot solve all the world's problems, and it is important to play to the strengths of the techniques and to be aware of their weaknesses (fixed-node error and pseudopotential non-locality are potential weaknesses of DMC in its present form).

Our hope for the future is that more researchers will discover for themselves the possibilities offered by QMC on machines ranging all the way from local clusters to the national and international petascale services now becoming available.

Acknowledgements

MDT would like to thank the Royal Society for the award of a research fellowship. We would like to acknowledge the underlying contributions to this article of N.D. Drummond, P. López Ríos, R.J. Needs, W.M.C. Foulkes, N. Nemec and L. Anton. Some of the practical calculations that we have presented were performed as part of collaborative projects with colleagues; we acknowledge particularly the contributions of A. Michaelides, F.R. Manby, B. Paulus, S.J. Binnie and G. Kresse. This research used resources of the Oak Ridge Leadership Computing Facility, located in the National Center for Computational Sciences at Oak Ridge National Laboratory,

which is supported by the Office of Science of the Department of Energy under Contract DE-AC05-00OR22725. We thank the DEISA Consortium (www.deisa.eu), funded through the EU FP7 project RI-222919, for support within the DEISA Extreme Computing Initiative

References

[1] R.G. Parr and W. Yang, *Density functional theory of atoms and molecules*, (Oxford University Press, 1989).

[2] R.M. Martin, *Electronic structure: basic theory and practical methods* (Cambridge University Press, 2004).

[3] C.J. Cramer, *Essentials of computational chemistry* (Wiley, 2002).

[4] C. Pisani, M. Busso, G. Capecchi, S. Casassa, R. Dovesi, L. Maschio, C. Zicovich-Wilson and M. Schütz, J. Chem. Phys. **122**, 094113 (2005).

[5] A. Erba, S. Casassa, R. Dovesi, L. Maschio and C. Pisani, J. Chem. Phys. **130**, 074505 (2009).

[6] G.D. Mahan, *Many-particle physics*, 2nd ed. (Plenum Press, New York, 1990)

[7] L. Schimka, J. Harl, A. Stroppa, A. Grüneis, M. Marsman, M. Mittendorfer and G. Kresse, Nature Mater. **9**, 741 (2010).

[8] R.J. Needs, M.D. Towler, N.D. Drummond and P. López Ríos, J. Phys.: Cond. Mat. **22**, 023201 (2010).

[9] *Quantum Monte Carlo, or, how to solve the many-particle Schrödinger equation whilst retaining favourable scaling with system size*, M.D. Towler, in '*Computational Methods for Large Systems*' (Wiley, 2011).

[10] CASINO website : www.tcm.phy.cam.ac.uk/∼mdt26/casino.html

[11] Top 500 Supercomputers website: www.top500.org

[12] M. Dion, H. Rydberg, E. Schröder, D.C. Langreth and B.I. Lundqvist, Phys. Rev. Lett. **92**, 246401 (2004).

[13] A. Tkatchenko and M. Scheffler, Phys. Rev. Lett. **102**, 073005 (2009).

[14] J. Klimeš, D.R. Bowler and A. Michaelides, J. Phys. Cond. Matt. **22**, 022201 (2010).

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

[16] M.D. Towler, *Pilot waves, Bohmian metaphysics and the foundations of quantum mechanics*, Graduate lecture course available at www.tcm.phy.cam.ac.uk/∼mdt26/pilot_waves.html (2009).

[17] CASINO 2.6 User manual, R.J. Needs, M.D. Towler, N.D. Drummond and P. López Ríos (University of Cambridge, Cambridge, 2010).

[18] W.M.C. Foulkes, L. Mitas, R.J. Needs and G. Rajagopal, *Rev. Mod. Phys.* **73**, 33 (2001).

[19] M.H. Kalos, L. Colletti and F. Pederiva, *J. Low Temp. Phys.* **138**, 747 (2005).

[20] J.B. Anderson, *J. Chem. Phys.* **63**, 1499 (1975); *Ibid.* **65**, 4121 (1976).

[21] T. Kato, *Commun. Pure Appl. Math.* **10**, 151 (1957).

[22] D.M. Ceperley, *J. Stat. Phys.* **63**, 1237 (1991).

[23] W.M.C. Foulkes, R.Q. Hood and R.J. Needs, *Phys. Rev. B* **60**, 4558 (1999).

[24] W. Glauser, W. Brown, W. Lester, D. Bressanini and B. Hammond, *J. Chem. Phys* **97**, 9200 (1992).

[25] D. Bressanini and P.J. Reynolds, *Phys. Rev. Lett.* **95**, 110201 (2005).

[26] M. Bajdich, L. Mitas, G. Drobný and L.K. Wagner, *Phys. Rev. B* **60**, 4558 (1999).

[27] S.K. Liu, M.H. Kalos and G.V. Chester, *Phys. Rev. A* **10**, 303 (1974).

[28] R.N. Barnett, P.J. Reynolds and W.A. Lester Jr., *J. Comput. Phys.* **96**, 258 (1991).

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

[30] A. Lüchow, R. Petz and T.C. Scott, *J. Chem. Phys.* **126**, 144110 (2007).

[31] F.A. Reboredo, R.Q. Hood and P.R.C. Kent, *Phys. Rev. B* **79**, 195117 (2009).

[32] P. López Ríos, A. Ma, N.D. Drummond, M.D. Towler and R.J. Needs, Phys. Rev. E **74**, 066701 (2006).

[33] A.J. Williamson, R.Q. Hood and J.C. Grossman, Phys. Rev. Lett. **87**, 246406 (2001).

[34] D. Alfè and M.J. Gillan, J. Phys.: Cond. Matt. **16**, L305 (2004).

[35] N.D. Drummond and R.J. Needs, Phys. Rev. B **79**, 085414 (2009).

[36] N.D. Drummond and R.J. Needs, Phys. Rev. Lett. **102**, 126402 (2009).

[37] M. Nekovee, W.M.C. Foulkes and R.J. Needs Phys. Rev. Lett. **87**, 036401 (2001).

[38] N.D. Drummond, Z. Radnai, J.R. Trail, M.D. Towler and R.J. Needs, Phys. Rev. B **69**, 085116 (2004).

[39] R.M. Lee, N.D. Drummond and R.J. Needs, Phys. Rev. B **79**, 125308 (2009).

[40] Gaussian 03, M.J. Frisch *et al.* Gaussian, Inc., Pittsburgh PA (2003) and `www.gaussian.com`.

[41] R. Dovesi, V.R. Saunders, C. Roetti, R. Orlando, C.M. Zicovich-Wilson, F. Pascale, B. Civalleri, K. Doll, N.M. Harrison, I.J. Bush, P. D'Arco and M. Llunell, CRYSTAL09 User's Manual, University of Torino, Torino (2009) and `www.crystal.unito.it`.

[42] TURBOMOLE V6.2 2010, a development of University of Karlsruhe and Forschungszentrum Karlsruhe GmbH, 1989-2007, TURBOMOLE GmbH, since 2007; available from `www.turbomole.com`.

[43] 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 and J.A. Montgomery, J. Comput. Chem., **14**, 1347 (1993).

[44] P. Giannozzi *et al.*, J. Phys.: Cond. Mat. **21**, 395502 (2009).
(`www.quantum-espresso.org`).

[45] X. Gonze *et al.*, Comp. Phys. Commun. **180**, 2582 (2009) and `www.abinit.org`.

[46] S.J. Clark, M.D. Segall, C.J. Pickard, P.J. Hasnip, M.J. Probert, K. Refson and M.C. Payne, Zeitschrift für Kristallographie **220**, 567 (2005).

[47] Amsterdam Density Functional (ADF) software: `www.scm.com`

[48] MPI Forum. Message Passing Interface (MPI) Forum Home Page. `http://www.mpi-forum.org/` (Dec. 2009)

[49] M.D. Towler, unpublished (2011).

[50] D. Alfè and M.J. Gillan, Phys. Rev. B **70**, 161101 (2004).

[51] H. Flyvbjerg and H.G. Petersen, J. Chem. Phys. **91**, 461 (1989).

[52] N.D. Drummond, unpublished (2010).

[53] OpenMP website: `openmp.org/wp/`

[54] J.M. McHale, A. Auroux, A.J. Perrotta and A. Navrotsky, Science **277**, 788 (1997).

[55] J. Goniakowski, J.M. Holender, L.N. Kantorovich, M.J. Gillan and J.A. White, Phys. Rev. B **53**, 957 (1996).

[56] D. Yu and M. Scheffler, Phys. Rev. B **70**, 155417 (2004); J.L.F. Da Silva, C. Stampfl and M. Scheffler, Surf. Sci. **600**, 703 (2006).

[57] A.E. Mattsson and D.R. Jennison, Surf. Sci. **520**, L611 (2002).

[58] D. Alfè and M.J. Gillan, J. Phys. Cond. Matt. **18**, L435 (2006).

[59] F.R. Manby, D. Alfè and M.J. Gillan, Phys. Chem. Chem. Phys. **8**, 5178 (2006); S.J. Nolan, M.J. Gillan, D. Alfè, N.L. Allan and F.R. Manby, Phys. Rev. B **80**, 165109 (2009).

[60] S.J. Binnie, S.J. Nolan, N.D. Drummond, D. Alfè, N.L. Allan, F.R. Manby and M.J. Gillan, Phys. Rev. B **82**, 165431 (2010).

[61] S.J. Binnie, S.J. Nolan, N.L. Allan, F.R. Manby and M.J. Gillan, in preparation.

[62] P.J. Feibelman, B. Hammer, J.K. Norskov, F. Wagner, M. Scheffler, R. Stumpf, R. Watwe and J. Dumesic, J. Phys. Chem. B **105**, 4018 (2001).

[63] M. Gajdoš, A. Eichler and J. Hafner, J. Phys. Condens. Matter, **16**, 1141 (2004).

[64] J. Carrasco, B. Santra, J. Klimeš and A. Michaelides, Phys. Rev. Lett. **106**, 026101 (2011).

[65] D. Alfè and M.J. Gillan, J. Phys. Cond. Matt., **18**, L451 (2006).

[66] B. Paulus, Phys. Rep. **428**, 1 (2006).

[67] B. Paulus and K. Rosciszewski, Int. J. Quantum Chem. **109**, 3055 (2009); C. Müller, B. Paulus and K. Hermansson, Surf. Sci. **603**, 2619 (2009).

[68] M. Pozzo and D. Alfè, Phys. Rev. B **78**, 245313 (2008).

[69] S. Grimme, J. Comput. Chem. **25**, 1463 (2004).

[70] S.J. Binnie, B. Paulus, D. Alfè and M.J. Gillan, in preparation.

[71] G. Hummer, J.C. Rasaiah and J.P. Noworyta, Nature **414**, 188 (2001).

[72] T. Werder, J.H. Walther, R.L. Jaffe, T. Halicioglu, F. Noca and P. Koumoutsakos, Nano Lett. **1**, 697 (2001).

[73] D. Feller and K.D. Jordan, J. Phys. Chem. A **104**, 9971 (2000); G.R. Jenness and K.D. Jordan, J. Phys. Chem. C **113**, 10242 (2009).

[74] G.R. Jenness, O. Karalti and K.D. Jordan, Phys. Chem. Chem. Phys. **12**, 6375 (2010).

[75] A.J. Misquitta and K. Szalewicz, Chem. Phys. Lett. **357**, 301 (2002); A. Hesselmann and G. Jansen, Chem. Phys. Lett. **357**, 464 (2002); A.J. Misquitta and K. Szalewicz, J. Chem. Phys. **122**, 214109 (2005); A.J. Misquitta, R. Podeszwa, B. Jeziorski and K. Szalewicz, J. Chem. Phys. **123**, 214103 (2005).

[76] J. Ma, D. Alfè, A. Michaelides and E. Wang, J. Chem. Phys. **130**, 154303 (2009).

[77] P. Wernet, D. Nordlund, U. Bergmann, M. Cavalleri, M. Odelius, H. Ogasawara, L.A. Naslund, T.K. Hirsch, L. Ojamae, P. Glatzel, L.G.M. Pettersson and A. Nilsson, Science **304**, 995 (2004).

[78] J.D. Bernal and R.H. Fowler, J. Chem. Phys. **1**, 515 (1933).

[79] Some of the most commonly used interaction models for water are reviewed in P. Ren and J.W. Ponder, J. Phys. Chem. B **107**, 5933 (2003).

[80] G.C. Lie, E. Clementi and M. Yoshimine, J. Chem. Phys. **64**, 2314 (1976).

[81] K. Laasonen, M. Sprik, M. Parrinello and R. Car, J. Chem. Phys. **99**, 9080 (1993).

[82] An overview of the recent literature reporting attempts to treat liquid water with DFT can be found in B. Santra, A. Michaelides and M. Scheffler, J. Chem. Phys. **131**, 124509 (2009).

[83] G.S. Tschumper, M.L. Leininger, B.C. Hoffman, E.F. Valeev, H. F. Schaefer III and M. Quack, J. Chem. Phys. **116**, 690 (2002).

[84] D.P. O'Neill, N.L. Allan and F.R. Manby, in *Accurate condensed-phase quantum chemistry*, ed. F.R. Manby (Taylor and Francis, Boca Raton, 2011).

[85] S. Yoo, E. Aprà, X.C. Zeng and S.S. Xantheas, J. Phys. Chem. Lett. **1**, 3122 (2010).

[86] I.G. Gurtubay and R.J. Needs, J. Chem. Phys. **127**, 124306 (2007).

[87] R.M. Olson, J.L. Bentz, R.A. Kendall, M.W. Schmidt and M.S. Gordon, J. Chem. Theory Comput. **3**, 1312 (2007).

[88] B. Santra, A. Michaelides, M. Fuchs, A. Tkatchenko, C. Filippi and M. Scheffler, J. Chem. Phys. **129**, 194111 (2008).

[89] A. Badinski, J.R. Trail and R.J. Needs, J. Chem. Phys **129**, 224101 (2008); A. Badinski, P.D. Haynes, J.R. Trail and R.J. Needs, J. Phys.: Condens. Matter **22**, 074202 (2010).

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

[91] The Apuan Alps Centre for Physics @ The Towler Institute (`www.vallico.net/tti/tti.html`).