Calculating derivatives within quantum Monte Carlo

Thomas Poole

Department of Physics

Imperial College London

December 2014

A thesis submitted for the degree of Doctor of Philosophy of Imperial College London and Diploma of the Imperial College

Acknowledgements

This work would not have been possible without the foresight, guidance and support of my supervisors Matthew Foulkes, Peter Haynes and James Spencer, who I would like to thank for individually and collectively making this PhD extremely enjoyable.

I would also like to thank the members and governors of the TSM CDT for providing this opportunity and creating such a pleasant environment in which to work.

Finally, I would like to thank my family, my friends and loved ones, who have either encouraged or tolerated me over the past few years.

Declaration of Originality

Except where stated otherwise, the work contained within this thesis is my own. This thesis has not been submitted in whole or in part for any other award at this or any other university.

Declaration of Copyright

The copyright of this thesis rests with the author and is made available under a Creative Commons Attribution Non-Commercial No Derivatives licence. Researchers are free to copy, distribute or transmit the thesis on the condition that they attribute it, that they do not use it for commercial purposes and that they do not alter, transform or build upon it. For any reuse or redistribution, researchers must make clear to others the licence terms of this work.

Abstract

Quantum Monte Carlo (QMC) methods are powerful, stochastic techniques for computing the properties of interacting electrons and nuclei with an accuracy comparable to the standard post-Hartree–Fock methods of quantum chemistry. Whilst the favourable scaling of QMC methods enables a quantum, many-body treatment of much larger systems, the lack of accurate and efficient total energy derivatives, required to compute atomic forces, has hindered their widespread adoption.

The work contained within this thesis provides an efficient procedure for calculating exact derivatives of QMC results. This procedure uses the programming technique of algorithmic differentiation (AD), which allows access to the derivatives of a computed function by applying chain rule differentiation to the underlying source code. However, this thesis shows that a straightforward differentiation of a stochastic function fails to capture the important contribution to the derivative from probabilistic decisions. A general approach for calculating the derivatives of a stochastic function is presented, where a similar adaptation of AD applied to the diffusion Monte Carlo (DMC) algorithm yields exact DMC atomic forces. The approach is validated by performing the largest ever DMC force calculations, which demonstrate the feasibility of treating systems containing thousands of electrons. The efficiency of AD also enables molecular dynamics simulations driven entirely

by DMC, adding new functionality to the QMC toolkit.

Another focus of this thesis is using the phenomenon of stochastic coherence to correlate DMC simulations, allowing finite difference derivatives to be obtained with a small error. Whilst this method is far easier to implement than AD, preliminary results show an instability when treating larger systems. A different approach is obtained from extrapolating this method to a finite difference step size of zero, producing algebraic expressions for a direct differentiation of the DMC algorithm.

Contents

Acronyms **12**

List of Figures **13**

1 Introduction **19**

2 Simulating atoms and molecules **26**

2.1 Introduction 26

2.2 Force field methods 28

2.3 Electronic structure methods 30

2.3.1 The Born–Oppenheimer approximation 31

2.3.2 Hartree–Fock and post-Hartree–Fock methods 33

2.3.3 Density functional theory 37

2.3.4 Quantum Monte Carlo 38

3 Monte Carlo methods **42**

3.1 Introduction 42

3.2 Integration 43

3.3 Importance sampling 45

3.4 The Metropolis algorithm 47
3.5 Langevin sampling 50
3.6 Serial correlation 52
3.7 Correlated sampling 52

4 Quantum Monte Carlo methods **54**
4.1 Introduction 54
4.2 Variational Monte Carlo 55
4.2.1 The variational principle 55
4.2.2 Evaluating the total energy 55
4.2.3 An outline of the algorithm 57
4.3 Diffusion Monte Carlo 58
4.3.1 The imaginary time Schrödinger equation 58
4.3.2 Branching 59
4.3.3 The fermion sign problem 62
4.3.4 Importance sampling 63
4.3.5 Reducing time step errors 67
4.3.6 Evaluating the total energy 69
4.3.7 Drift velocity and local energy limiting 69
4.3.8 An outline of the algorithm 71
4.4 Trial wave functions 72
4.4.1 The nodal surface 72
4.4.2 Slater–Jastrow wave functions 73
4.4.3 Updating wave functions 74
4.4.4 Optimising wave functions 76

4.5 Pseudopotentials 80
4.5.1 Introduction 80
4.5.2 Evaluating pseudopotentials 81
4.5.3 Diffusion Monte Carlo 83

5 Differentiating stochastic functions 85
5.1 Introduction 85
5.2 Stochastic functions 86
5.3 Finite differences 88
5.4 Curve fitting 90
5.5 Differentiating stochastic algorithms 94

6 Existing approaches to differentiating quantum Monte Carlo total energies 102
6.1 Introduction 102
6.2 The Hellmann–Feynman theorem 102
6.3 Correlated sampling 104
6.4 Direct differentiation 106
6.5 Outlook 107

7 Algorithmic differentiation 108
7.1 Introduction 108
7.2 The forwards mode 109
7.3 The reverse mode 114
7.4 Implementation 118

8 Algorithmic differentiation of diffusion Monte Carlo 121

8.1 Introduction . . . 121

8.2 Algorithm structure . . . 123

8.3 Stochastic branching . . . 134

8.4 Pseudopotentials . . . 147

8.5 Results . . . 148

9 Correlated sampling diffusion Monte Carlo 157

9.1 Introduction . . . 157

9.2 Current approaches . . . 158

9.3 Stochastic coherence . . . 162

9.4 Implementation . . . 167

9.5 Results . . . 175

9.6 Extrapolation to zero step size . . . 180

10 Conclusions 185

10.1 Summary . . . 185

10.2 Further work . . . 188

A The diffusion Monte Carlo code 191

A.1 Obtaining the code . . . 191

A.2 Code information . . . 192

A.3 Function definitions . . . 193

B Extended energy curve fits 196

C Stochastic coherence examples 199

Bibliography **204**

Acronyms

AD algorithmic differentiation

CC coupled cluster

CI configuration interaction

CSF configuration state function

DFT density functional theory

DMC diffusion Monte Carlo

FIFO first-in, first-out

GAMESS The General Atomic and Molecular Electronic Structure System

HF Hartree–Fock

HFT Hellmann–Feynman theorem

HPC high-performance computing

LIFO last-in, first-out

MP2 second order Møller–Plesset perturbation theory

QMC quantum Monte Carlo

RAM random-access memory

SWCT space-warp coordinate transformation

VMC variational Monte Carlo

List of Figures

3.1 Trial probability densities for the importance sampling of equation (3.8), where the ideal probability density is given by the $a = 1.0$ curve. 47

5.1 Shown in the top pane is the DMC total energy as a function of an arbitrary displacement of the carbon atom in a methane molecule. The bottom pane displays the gradient of the two functions in the top pane. 92

5.2 The top pane compares smoothed spline fits to the data from Figure 5.1 and data obtained from an independent set of DMC calculations. Shown in the bottom pane are the gradients of each fitted curve. 92

5.3 Compared to Figure 5.2 the stochastic total energy function has been sampled 4 times more frequently and with double the precision. The bottom pane compares the corresponding gradient to that of the fits from Figure 5.2. 93

5.4 The top pane shows data gathered from a larger range of displacements than the previous figures in this section, where vertical dotted lines mark the extent of the previous domain. Shown in the lower pane are the gradients of all fits performed so far. 93

5.5 The output from the simple branching program when the random number η has a value of 0.75. 96

7.1 An illustration of the accumulation of differential variables in the first two steps of the example program. 117

7.2 An outline of a basic checkpointing strategy used when performing reverse mode AD. The procedure starts at the top right corner with a conventional program execution flowing from right to left. Vertical dotted lines represent the positions of checkpoints. 120

8.1 An outline of the checkpointing strategy employed to perform AD of DMC. In contrast with Figure 7.2 all checkpoints are created on the initial forwards pass. 126

8.2 A call graph of the significant functions in the forwards pass of the DMC code. 131

8.3 A call graph of the significant functions in the reverse pass of the DMC code. 132

8.4 The effect of stochastic decisions on AD derivatives. The top pane shows the variation of the DMC total energy with the bond length of a lithium hydride molecule. The bottom pane compares the gradient of the total energy function with derivatives calculated using AD of the total energy evaluations of the top pane. 135

8.5 The derivative of the DMC total energy over a very small range. The top pane shows the same content as in Figure 8.4 but over a small range of displacements on the leftmost side of the x-axis. Here each calculation uses an identical stream of random numbers and the associated stochastic error bars have been omitted. The bottom pane compares the gradient of the total energy function with derivatives calculated using AD of the total energy evaluations of the top pane. 136

8.6 Shown in the top pane is the DMC total energy as a function of an arbitrary displacement of the carbon atom in a methane molecule. The bottom pane displays the gradient of the two fits in the top pane and the derivatives obtained by performing AD of the original DMC calculations. 149

8.7 Shown in the top pane is the DMC total energy as a function of the displacement of a single carbon atom relative to its neighbouring carbon atom in an ethane molecule. The bottom pane displays the AD derivative of the energy evaluations in the top pane. The smoothed spline fit and its gradient are obtained from the average of 30 fitting procedures to independent DMC energy evaluations extending outside the range of the plot, where the resultant stochastic error is smaller than the line width. 150

8.8 The same information as in Figure 8.7 for a propane molecule. 151

8.9 The same information as in Figure 8.7 for a butane molecule. . 151

8.10 Snapshots from an *ab initio* DMC molecular dynamics simulation of the dimerisation of borane. 154

8.11 The potential and kinetic energies of the nuclei during the molecular dynamics simulation depicted in Figure 8.10. 155

8.12 The total energy during the molecular dynamics simulation depicted in Figure 8.10. 156

9.1 The top pane displays the variation in the DMC total energy of a water molecule as a function of the oxygen-hydrogen distance during a symmetric stretch. Shown in the bottom pane are derivatives calculated using finite difference correlated sampling calculations with a step size of 0.004 bohr. . . . 176

9.2 This figure shows the separation of the correlated and reference trajectories averaged over all electrons and all walkers on each time step of a DMC simulation of a water molecule. The geometry is taken from figure 9.1 with an initial O-H separation of 1.66 bohr and a correlated displacement of 0.002 bohr. Each simulation contains 1200 walkers performing 3600000 time steps of 0.02 a.u. The inset zooms in on the start of the simulation. 177

9.3 The variation of the standard error in the correlated sampling total energy difference computed in Figure 9.2 over a limited history of local energies taken from the previous 2400 a.u. of simulation time. 178

A.1 A representation of the relative number of commits made during each hour of the week. 192

A.2 A call graph of the significant functions in the forwards pass of the DMC code (a duplicate of Figure 8.2). 193

B.1 The full extent of the average fit and gradient of 30 independent series of DMC total energies for the ethane molecule in Figure 8.7. Each series was composed of 17 total energy evaluations of the same precision as the data series shown, spaced evenly over the range of the fit. 197

B.2 The same information as in Figure B.1 for the propane molecule in Figure 8.8. 197

B.3 The same information as in Figure B.1 for the butane molecule in Figure 8.9. 198

C.1 Snapshots from a simulation of two non-interacting, frictionless balls sliding over an energy landscape. The balls are initially coincident, with a tiny difference in their starting velocities. Below the visualisation of the dynamics is a plot of the logarithmic distance between the balls. 200

C.2 Snapshots from a simulation identical to that shown in Figure C.1 but where the balls are stopped every few time steps and restarted with identical, randomly selected velocities. 201

C.3 Snapshots from a simulation identical to that shown in Figure C.2 but where the balls are initially placed in different, well separated minima in the potential. 202

C.4 Snapshots of two balls sampling a 2D function (shown inverted) using Langevin dynamics, where the same diffusive component is used for both balls. Below the visualisation of the dynamics is a plot of the logarithmic distance between the balls. 203

Chapter 1

Introduction

Computer simulations are now firmly established as a complement to physical experiments. In particular, *ab initio* simulations, which are independent of empirical information, provide a perfect test of fundamental theories, with the ability to explain results and predict physical phenomena from first principles. However, the exact simulation of quantum mechanics, which underpins a large part of fundamental physics, is only possible for the very simplest systems.[1] The solution of the Schrödinger equation requires the determination of a wave function from a highly coupled differential equation. For a collection of N charged particles, such as an atom or molecule, the wave function is a $3N$-dimensional quantity dependent upon the position of all particles. Obtaining a solution is a task which scales exponentially in difficulty with N, and will not produce any useful results; just storing a simple representation of the wave function on a real-space grid with 100 points in each dimension requires 100^{3N} floating point numbers, amounting to a memory consumption of 4 terabytes for two particles and 4 exabytes for three. The futility

of treating larger systems exactly is obvious. It is therefore necessary to use approximations to make the computation of quantum mechanical properties feasible.

One ubiquitous class of approximate computations are Monte Carlo methods, which influence a vast range of disparate fields encompassing finance,[2] epidemiology,[3] genetics,[4] metrology,[5] astrophysics,[6] materials modelling,[7,8] chemistry,[9] particle physics[10] and quantum mechanics.[11,12] The key to their success is that, by sampling equations rather than solving them exactly, it is possible to produce accurate estimates of results far faster than deterministic methods, using fewer computational resources. Even when the time required to calculate a deterministic solution scales exponentially with system size, stochastic estimates of results with a set error bar can often be obtained in just polynomial time. Additionally, the inherently parallel nature of Monte Carlo algorithms is ideally suited for modern high-performance computing (HPC) architectures, enabling an efficient exploitation of large supercomputers.

Quantum Monte Carlo (QMC) methods,[11,12] and in particular the variational Monte Carlo (VMC)[13–16] and diffusion Monte Carlo (DMC)[17–19] algorithms, are powerful *ab initio* tools which rely on the stochastic sampling of wave functions to reduce the complexity of computational quantum mechanics. Compared to the standard post-Hartree–Fock methods of quantum chemistry,[20] QMC calculations yield equivalently accurate electronic structures of atoms, molecules and solids, with a cost that scales relatively modestly with system size. The more accurate, but expensive, DMC method represents a many-electron wave function as a stochastic collection of weighted

electronic configurations. As such a representation is not restricted to an expression in terms of a basis set, the DMC method can excel in systems where other electronic structure methods typically struggle; the storage requirements of the fully interacting DMC wave function are spectacularly small and its variational freedom allows sharp or discontinuous features to be represented exactly.

Whilst stochastic sampling can be enormously beneficial, there are disadvantages to Monte Carlo methods. Results containing stochastic error bars can be difficult to validate and, in many cases, obtaining accurate derivatives of results with respect to the parameters of a simulation is not feasible. The latter problem can be a significant drawback; often knowing how the result of a simulation will respond to changes in its inputs is more valuable than the result itself. This is particularly pertinent for DMC simulations, where the stochastic representation of the wave function prevents the evaluation of the algebraic derivatives required for atomic force calculations. Existing differentiation strategies[21–32] rely on approximations to the DMC wave function and all scale very poorly with system size, limiting previous DMC force calculations to molecules containing 8 or fewer electrons. This contrasts markedly with the evaluation of total energies, where simulations involving more than 1000 electrons are possible.[33]

A lack of accurate and efficient forces has hindered a wider adoption of DMC as many important physical properties of systems are determined by the molecular geometry, which is often too difficult to determine using current methods. The aim of this thesis is to investigate new approaches to calculating total energy derivatives within DMC simulations using insights

from physics and techniques from computer science. These derivatives will increase the utility of the DMC method by providing a mechanism to perform wave function optimisation, structure relaxation and molecular dynamics simulations. The principal difficulty in this objective is designing novel strategies for calculating the derivatives of a stochastic algorithm.

Traditional numerical differentiation schemes, based on finite difference methods, struggle in the presence of noise. These techniques estimate derivatives from small differences between slightly displaced systems. For deterministic methods the selection of very small displacements can yield highly accurate derivatives, but in the presence of independent stochastic errors the resolution of the small differences becomes very difficult. One solution is to employ the technique of correlated sampling,[34] where a single, reweighted set of samples is used to evaluate multiple similar Monte Carlo estimates. In this regime the difference between two displaced systems varies smoothly, with the stochastic error tending to zero with the displacement. Correlated sampling is prevalent in VMC wave function optimisations,[35,36] but its application to DMC is more complicated.[27–30]

A different way of correlating stochastic simulations uses the phenomenon of stochastic coherence, where two similar dynamical trajectories exposed to a common noise converge to a single path.[37] This property is rather remarkable, as dynamical simulations are notoriously sensitive to their initial conditions; small changes to the parameters of a simulation typically result in chaotic behaviour with an exponential divergence in trajectories. The form of the equations governing the evolution of a DMC simulation are particularly amenable to such a treatment, and by correlating simulations in such

a manner it is possible to estimate the total energy derivatives required for force calculations using finite differences.[31] This thesis presents an extension of this approach to a more sophisticated DMC algorithm and, by extrapolating the resultant equations to a finite difference step size of zero, derives an analytic expression to calculate DMC forces directly.

An alternative route to the direct evaluation of derivatives is algorithmic differentiation (AD),[38,39] a programming technique for extracting the derivatives of a computed function that occupies the ground between the traditional numerical finite differencing methods and symbolic differentiation. This method proceeds by decomposing an arbitrarily complex function into a (potentially enormous) sequential list of basic operations. Such a decomposition is always possible as the instructions ultimately fed to a CPU can contain only elementary arithmetic. Supplied with a list of operations, AD calculates derivatives via the systematic application of chain rule differentiation. This allows AD to extract derivatives from computed functions that are too complex to permit an explicit algebraic solution. Another attraction of AD is that the 'reverse mode' of operation allows access to the derivatives of a selected program output with respect to all of its inputs, simultaneously, in a small, fixed multiple of the computational cost of evaluating the underlying function in isolation. This makes the AD method extremely efficient at differentiating calculations containing many degrees of freedom.

Whilst prevalent in the financial sector, the adoption of AD by the scientific community has been fairly limited. One application[40] of particular pertinence is the calculation of force estimators within VMC, which is the first association of AD with QMC methods. This thesis details a new, first

principles method of calculating accurate DMC atomic forces via the application of AD to the DMC algorithm.

The following chapter provides some context for QMC methods by giving a quick introduction to techniques for simulating atoms and molecules, ranging from classical interatomic potentials to the highly accurate calculations of quantum chemistry. Chapter 3 introduces Monte Carlo approaches, including derivations of the sampling techniques underlying the VMC and DMC algorithms. This is followed by a chapter dedicated to the VMC and DMC algorithms themselves, containing an outline of the theory behind both methods and some details of their implementations. Chapter 5 presents some strategies for calculating the derivative of a stochastic function, and provides the first theoretical contribution towards the differentiation schemes developed later on in this thesis. Existing strategies for calculating derivatives within QMC methods are summarised in Chapter 6, which is followed by an introduction to AD in Chapter 7. Chapter 8 details the application of AD to the DMC algorithm, which yields an accurate and efficient method of obtaining the derivatives of DMC results. Calculations of DMC forces in large systems are used to validate and benchmark the new approach, before driving the first DMC molecular dynamics simulation. The penultimate chapter presents a DMC correlated sampling method based on stochastic coherence, and describes its implementation in an established DMC code. This method shows promise for simulations of a water molecule, but the correlated sampling scheme breaks down for systems of a larger size. The chapter concludes with an extrapolation of the divergence of correlated trajectories to a finite difference step size of zero, producing an algebraic expression for the direct

calculation of the DMC total energy. Finally, Chapter 10 puts the results into context and outlines avenues for future work.

Chapter 2

Simulating atoms and molecules

2.1 Introduction

Over the past 40 years the maximum processing speed of a computer, measured in instructions per second, has increased by 11 orders of magnitude, with corresponding gains in the availability of fast random-access memory (RAM) and disk storage. This dramatic increase in computing power, accompanied by algorithmic and software developments to exploit it, means that scientific calculations requiring a supercomputer 10 years ago can now be performed on personal desktop machines. The software to perform these calculations is also far more widely available, making computer simulations of complex physical phenomena feasible without having to invest significantly in computing infrastructure. When compared to experiments, computer simulations therefore represent a relatively cost efficient way of evaluating analytical

models or testing theories. Simulations are particularly useful when physical or financial constraints render the construction of a particular experiment impossible, allowing access to extremes in temperature, pressure, length or time scales, and many more regimes.

One relevant application is the search for new compounds or materials with desirable properties. Whilst current simulation techniques are not accurate, reliable or flexible enough to have made a significant impact so far, progress in this area has been the driving force behind the establishment of the Materials Genome Initiative[41] in the United States, and related activities internationally. The common idea behind these projects is to accelerate the design and development of new materials using computational data to guide the selection of candidate substances.

Another related application where simulations have played an important role is the study of the interactions between electrons and nuclei in individual and small collections of molecules. The small length scales involved make physical manipulation and testing extremely difficult, and simulations have provided valuable insights into the nature of electronic properties and chemical bonding.

This chapter presents a brief overview of the methods commonly used by physicists and chemists to simulate atoms and molecules. Here, and throughout this thesis unless stated otherwise, equations are presented using atomic units, $m_e = e = \hbar = 4\pi\epsilon_0 = 1$, where m_e is the mass of an electron, e is the charge on an electron, $\hbar$ is the reduced Planck constant, and ϵ_0 is the permittivity of free space.

2.2 Force field methods

Atomic bonds determine a large number of the properties of condensed matter systems, and the simplest method of modelling interactions between atoms is to give them an explicit functional form. This approach assumes that a reasonable description of the local atomic coordination can be captured via the optimisation of functions describing the energetics of bond stretching, bond angles, electrostatic interactions, van der Waals interactions, and other effects. The combined action of these functions is referred to by chemists as a force field, which is typically optimised to reproduce properties such as crystal structures, lattice constants, stresses and strains, melting points, and densities of bulk systems in molecular dynamics simulations. Whilst predominantly labelled as an empirical method, the reference data for force field optimisation can be obtained from more expensive, *ab initio* quantum mechanical simulations of smaller systems, as well as experimental results.

Molecular dynamics simulations[42,43] obtain the classical time evolution of an initial configuration of atoms using accelerations determined by the atomic forces. Using a time step, which is typically chosen to be around 5% of the period of the fastest natural oscillation in the system, classical equations of motion are combined with current and previous information about the positions, velocities and accelerations of atoms to determine the state of the system a short time in the future. A number of different schemes exist for integrating the equations of motion, with various degrees of accuracy and difficulty which can be selected based upon the demands of the application. Additionally, it is possible to control thermodynamic variables such as

the temperature, pressure and volume of a system, allowing the simulation of thermodynamic ensembles. This makes the molecular dynamics method an extremely versatile tool for investigating equations of state and phase transitions by adjusting the temperature and pressure over the course of a simulation.[44,45] It is also relatively straightforward to study the time evolution of an initial disturbance to the system or model dynamic effects such as reaction rates or diffusion processes.[46,47] Other properties often obtained from molecular dynamics simulations include heat capacities, equilibrium geometries and pair distribution functions.

The drawback of the molecular dynamics method is that the calculation of many properties requires a very large number of atoms or a very long simulation time; for small systems the thermodynamic limit is not reached, making many useful formulas meaningless, and short time scales can fail to capture rare events or slower processes like phase changes. To address these issues, molecular dynamics simulations are commonly partnered with atomic forces obtained from force fields, which, compared to approaches based on quantum mechanics, can be computed many orders of magnitude faster, and allow the simulation of hundreds of millions of atoms simultaneously.

The accuracy of force fields can vary tremendously based on their form and the type of system being studied. Whilst simple pairwise interactions, such as those defined by the Lennard-Jones potential, are extremely quick to evaluate, they are also correspondingly limited in scope. More complicated interactions require additional terms to represent different features of each atom's environment, which are often highly system dependent. This has resulted in the publishing of many different force field parametrisations with

fairly limited transferability beyond their initial applications. Even within their ideal domain, the most advanced force fields cannot hope to accurately reproduce phenomena arising from electron transfer or quantum effects. More accurate simulations of atoms and molecules can only be obtained using a quantum mechanical treatment of the electrons.

2.3 Electronic structure methods

Electronic structure methods are tools which determine the properties of systems by directly solving the quantum-mechanical equations of motion for the electrons. Whilst far more expensive than force field methods, the results obtained are typically much more accurate, and the explicit inclusion of electrons allows the computation of properties including ground state geometries, reaction paths and associated barrier heights, vibrational frequencies, thermochemistry, dipole moments, polarisabilities, magnetic properties and excited states. Many of these are, in principle, attainable with polarisable force fields, but any physical effect determined by complex interactions between electrons is likely to be poorly described. The high computational cost of electronic structure methods means that only the cheapest variants are suitable for molecular dynamics simulations, and even then the largest systems sizes possible are around four orders of magnitude smaller than those treated using force fields; on a large supercomputer it might be possible to simulate 10,000 atoms over a short time scale using semi-empirical or linear scaling electronic structure methods.

2.3.1 The Born–Oppenheimer approximation

Underlying all of electronic structure physics is the many-body Schrödinger equation,

$$\begin{aligned}\hat{H} \;=\; & -\frac{1}{2}\sum_i \nabla^2_{\boldsymbol{r}_i} \;+\; \frac{1}{2}\sum_i\sum_{j\neq i}\frac{1}{|\boldsymbol{r}_i-\boldsymbol{r}_j|} \;-\; \sum_i\sum_\alpha \frac{Z_\alpha}{|\boldsymbol{r}_i-\boldsymbol{R}_\alpha|} \\ & -\frac{1}{2}\sum_\alpha \frac{1}{m_\alpha}\nabla^2_{\boldsymbol{R}_\alpha} \;+\; \frac{1}{2}\sum_\alpha\sum_{\beta\neq\alpha}\frac{Z_\alpha Z_\beta}{|\boldsymbol{R}_\alpha-\boldsymbol{R}_\beta|}\;,\end{aligned} \tag{2.1}$$

where $\{\boldsymbol{r_i}\}$ are electron positions with Roman indices, $\{\boldsymbol{R}_\alpha\}$ are nuclear positions with Greek indices and $\{Z_\alpha\}$ and $\{m_\alpha\}$ are the corresponding nuclear charges and masses respectively. Equation (2.1) can be made more tractable using the Born–Oppenheimer approximation,[48,49] which decouples the dynamics of electrons and nuclei. As the mass of an electron is approximately 10^{-3} to 10^{-5} that of a nucleus, it is only a small approximation to assume that the electrons react instantaneously to any changes in the positions of the nuclei. Writing the wave function as

$$\Psi(\bar{\boldsymbol{r}};\bar{\boldsymbol{R}})\chi(\bar{\boldsymbol{R}})\;, \tag{2.2}$$

where barred quantities represent vectors where each element is the position of a single electron or nucleus, and the electron coordinates are parametrically dependent on the nuclear geometry, it is possible to show that the terms in the Hamiltonian which couple the electronic and nuclear dynamics are small.[50] This approximation allows electronic structure methods to treat the electronic wave function in isolation, reducing the problem to that of finding

eigenstates of the combined potential contributions from the nuclei. The electronic Hamiltonian is given by

$$\hat{H} \;=\; -\frac{1}{2}\sum_i \nabla^2_{\boldsymbol{r}_i} \;+\; \frac{1}{2}\sum_i \sum_{j\neq i} \frac{1}{|\boldsymbol{r}_i - \boldsymbol{r}_j|} \;-\; \sum_i \sum_\alpha \frac{Z_\alpha}{|\boldsymbol{r}_i - \boldsymbol{R}_\alpha|}\;, \qquad (2.3)$$

which, for notational convenience, can be written as

$$\hat{H} \;=\; -\frac{1}{2}\nabla^2_{\bar{\boldsymbol{r}}} \;+\; U(\bar{\boldsymbol{r}})\;. \qquad (2.4)$$

Here $\nabla_{\bar{\boldsymbol{r}}}$ is a vector of the corresponding single electron gradient operators, and the dependence of the Schrödinger equation on the nuclear positions $\bar{\boldsymbol{R}}$ has been absorbed into the definition of $U(\bar{\boldsymbol{r}})$.

For some physical processes the Born–Oppenheimer approximation excludes important information about the interaction between the dynamics of electrons and nuclei. The instantaneous acceleration of the nuclei is obtained from the gradient of the Born–Oppenheimer energy surface, which is itself determined solely by the nuclear geometry. Therefore, any subsequent nuclear motion can only ever be conservative; there exists no mechanism to transfer energy from the nuclear dynamics to the electronic system. Another limitation of the Born–Oppenheimer approximation is that it only considers a single Born–Oppenheimer energy surface. A quantum mechanical treatment of the nuclei associates a Born–Oppenheimer energy surface with each electronic energy level. When two energy levels become close it should be possible to transition between the Born-Oppenheimer surfaces, ending up in a linear combination of two different electronic states. The forces on the nu-

clei depend on the electronic state, so the transition into a superposition of two different electronic states causes the nuclear wave packet to divide into two components, which separate as time evolves. Thankfully, a great deal of interesting and important science occurs relatively close to the ground state where first order corrections to the Born–Oppenheimer approximation are of the order 10^{-4} or 10^{-5} times the electronic kinetic energy.[50]

2.3.2 Hartree–Fock and post-Hartree–Fock methods

The Hartree–Fock (HF) ansatz is that, for systems without spin polarisation, the solution of the electronic Born–Oppenheimer Hamiltonian can be expressed as a single Slater determinant of one-electron orbitals. These orbitals are generated using the quantum mechanical variational principle which states that, for any reasonable trial wave function, Ψ_{HF}, the expectation value of the energy,

$$E_{\mathrm{HF}} = \frac{\int \Psi^*_{\mathrm{HF}}(\bar{\boldsymbol{r}})\hat{H}\Psi_{\mathrm{HF}}(\bar{\boldsymbol{r}})d\bar{\boldsymbol{r}}}{\int \Psi^*_{\mathrm{HF}}(\bar{\boldsymbol{r}}')\Psi_{\mathrm{HF}}(\bar{\boldsymbol{r}}')d\bar{\boldsymbol{r}}'} , \tag{2.5}$$

is greater than or equal to the true ground state energy. The best possible energy within this theory is therefore obtained by varying the form of the orbitals to produce the lowest value possible. For systems with spin polarisation the wave function can be expressed using a product of spin polarised determinants, but the remainder of this thesis will deal with calculations involving doubly occupied orbitals.

Whilst the antisymmetry of a Slater determinant enforces the correct exchange symmetry, individual electrons only experience the Coulomb repul-

sion due to the average positions of all the other electrons. Herein lies the main source of error in the HF method: Coulombic interactions between electrons are only treated on average, so the resultant correlations between their instantaneous positions are neglected. This error is known as the dynamical correlation energy. Another source of error, known as the static correlation energy, is indicated by the inability of a single Slater determinant to properly describe the form of the ground state wave function. Whilst generally a small fraction of the total energy, the correlation energy can play an essential role in determining the properties of a system because it can make a significant contribution to total energy *differences*.[51] Techniques for improving upon the HF wave function, known as post-Hartree–Fock methods,[20] aim to recover as much of the correlation energy as possible.

Instead of a single Slater determinant, the configuration interaction (CI) method[52] constructs a wave function out of a linear combination of determinants containing the ground state and excited occupations of the HF orbitals. These determinants are known as configurations, and the relative contribution of each is determined in a similar manner to the HF orbitals via equation (2.5) and energy minimisation. The number of configurations to include is determined by the types of excitations allowed. Common truncation levels include single excitations (CIS), single and double excitations (CISD), single, double and triple excitations (CISDT), and all possible excitations (Full CI), where the maximum excitation level is set by the number of electrons. If the basis of one-electron functions is complete, which is only ever a theoretical possibility, then the Full CI procedure is known as 'Complete CI'.

Whilst CI can produce extremely accurate results, the number of config-

urations required scales very strongly with system size. Brillouin's theorem states that single excitations cannot improve the ground state wave function, so at least double excitations are required to make any gain in the amount of ground state correlation energy recovered. The number of configurations required is dominated by the contribution of the highest excitation level, which for double excitations corresponds to a computational demand of approximately $\mathcal{O}(n^6)$ in typical use cases, where n represents the number of one-electron basis functions. Here n is normally a small multiple of number of electrons, and is determined by the size of the basis set produced by the underlying HF calculation. The inclusion of triple excitations increases the scaling to approximately $\mathcal{O}(n^8)$, and Full CI scales factorially. In the limiting case where n is far larger than the number of electrons, the computational scaling of CISD and CISDT is reduced to $\mathcal{O}(n^4)$ and $\mathcal{O}(n^5)$ respectively.

A drawback of truncated CI methods is a lack of size-extensivity. As the size of a system increases, the proportion of the correlation energy contained within a certain truncation level decreases, and the method performs less well. This can make it difficult to compare properties of different sized system, as there will not be a reliable cancellation of errors. Truncated CI also lacks size-consistency, where the energy of two atoms at a large separation is not the same as the sum of their energies when calculated individually. The combined system requires a higher truncation level to accommodate the possibility of simultaneous excitations on both atoms. Restricted HF, which operates on closed-shell systems with all orbitals doubly occupied, is also not size-consistent, as it cannot properly describe the dissociation of the combined system into open-shell fragments.

An improvement to the CI approach is the coupled cluster (CC) method,[53] which, rather than using a linear combination of configurations, expresses the wave function in terms of an exponentiated sum of excitation operators acting on the HF ground state. The main idea behind this representation is that, compared to CI at the same truncation level, the CC wave function includes contributions from higher order excitations due to the power series expansion of the exponential. Whilst having a larger prefactor, the computational scaling of truncated CC is similar to that of truncated CI, where the increased accuracy and the introduction of size-extensivity usually justifies the additional cost. Furthermore, if the underlying HF calculation is size consistent, then truncated CC calculation is also size consistent. Due to the enormous computational cost of including single, double and triple excitations (CCSDT), the effect of the triple excitations is usually incorporated perturbatively, and the resultant method, CCSD(T), is sometimes referred to as the 'gold standard' of electronic structure calculations.

Despite its accuracy the utility of CCSD(T) is severely diminished by its poor scaling with system size, which limits it to systems containing fewer than approximately 30 atoms. A less accurate, but cheaper way of recovering correlation energy is given by second order Møller–Plesset perturbation theory (MP2). Here the zeroth-order Hamiltonian is the Fock operator that generates the HF orbitals, which is augmented by the difference between the Fock operator and the two-body Coulomb operator. This explicitly correlates the positions of the electrons, and MP2 provides the second order correction to the ground state energy via perturbation theory. Whilst its accuracy is highly system dependent, this approach can produce better total energies,

geometries, vibrational frequencies and excitation energies than HF calculations,[54–57] with a more favourable scaling than CISD allowing the treatment of systems containing around 100 atoms.

The electronic structure methods described in this section are all *ab initio* techniques, which operate without any empirical input. This makes the methods very transferable, as they have not been optimised for any specific system. However, for a specific system, it can be possible to trade transferability for execution speed. Semi-empirical methods replace many of the integrals performed in the HF algorithm with parameters obtained from experiment or a higher level of theory.[58–61] The subsequent reduction in computational cost allows semi-empirical methods to simulate approximately 10,000 atoms, rather than the 1,000 or so that HF can manage.

2.3.3 Density functional theory

The density functional theory (DFT) method[62] uses the Hohenberg–Kohn[63] theorem and the Kohn-Sham[64] equations to replace the many-electron ground state with a simpler, three dimensional electron density. This reduction in complexity is achieved in two stages: the Hohenberg–Kohn theorem states that the total energy is a unique functional of the electron density, and that the density which minimises the total energy is the exact ground state density; the Kohn–Sham equations map the fully interacting electronic system onto a fictitious non-interacting system which reproduces the same ground state density. The total energy can thus be written as the sum of the kinetic energy of the non-interacting system, the classical Hartree energy of

the electron density, the interaction of the electron density with the nuclei, and a corrective exchange-correlation functional. This exchange-correlation functional must account for both the difference between the kinetic energies of the real and fictitious systems, and the difference between the real and Hartree electron-electron interaction energies.

Even the simplest forms for the exchange-correlation functional, which use parametrised exchange and correlation energies from QMC simulations of a homogeneous electron gas of the same density, yield an *ab initio* DFT algorithm which is, in most cases, both more accurate and more efficient than HF. Even higher accuracies can be obtained using empirical forms for the exchange-correlation operator, but results from more expensive CCSD calculations are generally better. Systematic schemes to improve exchange-correlation functionals are unavailable[65] and, in practice, the inherent decomposition into single-electron orbitals fails to describe many interesting classes of system, such as those involving high temperature superconductivity or van der Waals interactions. However, with the ability to treat quantum systems containing thousands of atoms, DFT simulations are easily the most widely used electronic structure tools.

2.3.4 Quantum Monte Carlo

The objective of QMC methods is to use stochastic sampling to avoid dealing directly with the complexity of the many-electron ground state of the Born–Oppenheimer Hamiltonian. Two prominent QMC techniques are VMC and DMC, which improve an initial trial wave function, Ψ_{T}, typically obtained

from the single-electron orbitals of a HF or DFT calculation, towards the exact, fully interacting, many-electron ground state.

The VMC method employs the quantum mechanical variational principle to optimise adjustable parameters of Ψ_{T}. As the optimisation proceeds, Ψ_{T} improves towards the ground state, but the ultimate quality of a VMC wave function is dependent upon the flexibility of its form.

The more accurate, but more demanding, DMC method overcomes this limitation by projecting out a stochastic representation of a fully interacting, many-electron ground state in the regions between surfaces where Ψ_{T} is zero. This approach can be thought of as a super-variational technique, producing the lowest energy wave function compatible with a given nodal surface; as the nodal surface of Ψ_{T} is improved, the DMC wave function tends towards the exact ground state wave function. Over the course of a DMC calculation the wave function is manifest as a collection of weighted electronic configurations. Unlike other approaches, this form of a wave function is not restricted to an expression in terms of a set of functions, eliminating the corresponding basis set error which can afflict other electronic structure methods.

Compared to well converged CCSD(T), DMC can produce equivalently accurate results with a far more favourable scaling with system size. To obtain a fixed stochastic error in the total energy, the computational cost of the DMC method is typically $\mathcal{O}(N^3)$ or better,[33] placing it on a par with DFT, albeit with a far larger prefactor. For a fixed stochastic error per atom or electron, the scaling is reduced to $\mathcal{O}(N)$. However, there is a component of the standard DMC algorithm which scales at least exponentially in the square root of the system size,[66] becoming observable in systems containing

hundreds of atoms or molecules with poor wave functions. Most DMC studies performed so far have been at system sizes well below this limit but, as computational power increases, the treatment of ever larger systems will have to contend with this problem.

Whilst expensive, QMC methods are inherently parallel, making them extremely well suited to modern HPC architectures. Large scale DMC calculations have been performed using 80,000 processes with near perfect parallel efficiency,[67,68] and, since then, CASINO[12] has been run on larger computers still, using 0.12 million cores of 'Jaguar' and more than 0.5 million cores of the Japanese K computer. A powerful exploitation of large computers means that systems treated using DMC can contain hundreds of atoms, roughly an order of magnitude larger than what is possible using CCSD or CCSD(T) which do not scale efficiently to many processes.

Another advantage of QMC methods is that they are able to perform calculations on periodic systems, where, despite some progress,[69–71] quantum chemistry methods struggle. For the simulation of solids or other extended systems, QMC algorithms represent the only tractable way of obtaining accurate many-electron phenomena.

Despite great success in computing total energies, a lack of reliable and efficient atomic forces means that the DMC method cannot be used to relax structures or study dynamics. Poor scaling with system size has limited previous, approximate force calculations[21–32] to molecules containing 8 or fewer electrons. This is a significant drawback, as the DMC method is otherwise uniquely well placed to study the bonding and intermolecular interactions between larger molecules where electron correlation becomes increasingly im-

portant. As technology enables the fabrication of devices and materials on ever smaller length scales, including structures like carbon nanotubes or two dimensional systems, the ability to simulate these systems accurately will be essential to their discovery and development. The remainder of this thesis focusses on methods for determining accurate derivatives of stochastic results, with the ultimate aim of incorporating the calculation of exact and efficient atomic forces within DMC calculations.

Chapter 3

Monte Carlo methods

3.1 Introduction

This chapter introduces the concepts required to understand Monte Carlo methods, with a particular focus on the techniques used in the VMC and DMC methods to estimate integrals over wave functions. The first section describes the extraordinary performance benefits of Monte Carlo approaches compared to traditional grid based integration schemes, and the second section explains how to further increase their efficiency using importance sampling transformations. This is followed by an overview of the Metropolis algorithm and Langevin sampling which underpin the statistical selection of electronic configurations in VMC and DMC simulations respectively. The chapter is concluded with a description of the correlated sampling technique, used to reduce the relative error between two distinct Monte Carlo calculations.

3.2 Integration

A simple example which illustrates the power of a Monte Carlo approach is the numerical integration of multidimensional functions. Numerical integration is typically performed by evaluating an integrand, f, on a fixed, evenly spaced grid of points. One of the simplest schemes is the trapezium rule for $N+1$ points, or N line segments, given by

$$I \;=\; \int_a^b f(x)dx \;=\; \frac{h}{2}\sum_{k=1}^{N}\left[f(x_{k+1})+f(x_k)\right]+\mathcal{O}(h^2)\;, \tag{3.1}$$

where $h=(b-a)/N$. This scheme can be extended into additional dimensions by dividing up each extra dimension in the same way. The resulting integration grid can be thought of as an outer product of the vectors of $N+1$ points on each integration axis. As in the one dimensional case the integrand is linearly interpolated between the grid points, retaining a relative error proportional to h^2.

The integral in equation (3.1) can also be evaluated by selecting the sampling points randomly,

$$I \;=\; \int_a^b f(x)dx \;=\; (b-a)\bar{f} \;=\; \frac{b-a}{N}\sum_{i=1}^{N} f(x_i)+\mathcal{O}\left(\frac{1}{\sqrt{N}}\right)\;, \tag{3.2}$$

where $\bar{f}$ denotes the mean value of f over the integration range. For an $f(x)$ with a well defined mean and variance the central limit theorem dictates that the set of all possible sums over different $\{x_i\}$ will have a Gaussian distribution. Thus, the error in an integral obtained from a Monte Carlo

sampling can be estimated using

$$\begin{aligned} \sigma_N &= V\sqrt{\frac{\langle f^2\rangle - \langle f\rangle^2}{N-1}} \\ &= V\sqrt{\frac{\left(\frac{1}{N}\sum_{i=1}^{N} f^2(x_i)\right) - \left(\frac{1}{N}\sum_{j=1}^{N} f(x_j)\right)^2}{N-1}} \,, \end{aligned} \tag{3.3}$$

where

$$V = \int_a^b dx \,. \tag{3.4}$$

Crucially, equation (3.3) is proportional to $\frac{1}{\sqrt{N}}$ and independent of the dimensionality of the integral, d.

For small d the slow convergence of the statistical error in the Monte Carlo method means that deterministic approaches can obtain an integral to a given error with far fewer evaluations of the integrand. However, when increasing the number of dimensions, the number of integrand evaluations required for the trapezium rule scales as N^d. Thus, for some d, usually around 4 or 5, the efficiency of the Monte Carlo approach dominates. Another way to think about this scaling is to consider the effort required to reduce the error in the integral by a factor of 10. For the trapezium rule h must be reduced by a factor $10^{\frac{1}{2}}$, which increases the required number of grid points in each dimension proportionally. The total computational effort therefore scales rather catastrophically with the number of dimensions as $10^{\frac{d}{2}}$. Conversely, to reduce the error in a Monte Carlo by a factor of 10 requires only 100 times as much work.

3.3 Importance sampling

When sampling points are selected uniformly across the entire integration range it is likely that considerable effort will be spent evaluating regions of integrand that contribute very little to the value of the integral. The efficiency of Monte Carlo schemes can be improved by correlating the density of sampling points with the magnitude of the integrand, a procedure called called importance sampling.

To evaluate the integral in equation (3.1) using a Monte Carlo method, the integrand is split into the product of a sampling probability distribution $\pi(x)$ and a score function $g(x) = \frac{f(x)}{\pi(x)}$,

$$I = \int g(x)\pi(x)dx \ . \tag{3.5}$$

The aim of importance sampling is to minimise the statistical error in the result, which is proportional to the standard deviation of the sampled quantity, as shown in equation (3.3). Minimising the variance of $g(x)$ over $\pi(x)$, given by

$$\sigma_g^2 = \int \left(g(x) - I\right)^2 \pi(x)dx \ , \tag{3.6}$$

gives the optimal sampling distribution,

$$\pi_{\text{opt}}(x) = \frac{|f(x)|}{\int |f(x')|dx'} \ . \tag{3.7}$$

Thus, when the importance sampling is perfect and $f(x) > 0$ everywhere, *any* evaluation of the score function yields the exact value of I and only a

$\pi(x,\ a)$	$I_{\rm MC}$	σ_I
Linear	0.9594	0.0046
a=3.2	0.9548	0.0041
a=1.8	0.9524	0.0030
a=1.4	0.9557	0.0021
a=1.2	0.9548	0.0013
a=1.0	0.9545	0

Table 3.1: The results of evaluating equation (3.8) stochastically using 10000 samples drawn from the probability distributions of equation (3.10) shown in Figure 3.1, where $I_{\rm MC}$ is the estimate of I and σ_I is the statistical error.

single sample is needed to perform the integral. On first sight this is not much of an improvement; the calculation of the denominator in equation (3.7) is just as hard as computing the original integral. However, approximations to $\pi_{\rm opt}(x)$ still increase the efficiency of the integration without introducing a bias. Figure 3.1 and Table 3.1 show how increasingly good estimates of $\pi_{\rm opt}(x)$ reduce the Monte Carlo error in a stochastic integration of a unit variance Gaussian distribution between -2 and 2. Here,

$$I \;=\; \int_{-2}^{2} \frac{1}{\sqrt{2\pi}}\ \exp\left(-\frac{x^2}{2}\right) dx \tag{3.8}$$

$$=\; \mathrm{erf}(\sqrt{2}\,) \;\simeq\; 0.9545 \tag{3.9}$$

and trial probability distributions are generated from

$$\pi(x,\ a) \;=\; \frac{\frac{1}{\sqrt{2\pi a^2}}\ \exp\left(-\frac{x^2}{2a^2}\right)}{\mathrm{erf}\left(\frac{2}{a\sqrt{2}}\right)} \tag{3.10}$$

where $\pi(x,\ 1) \;=\; \pi_{\rm opt}(x)$.

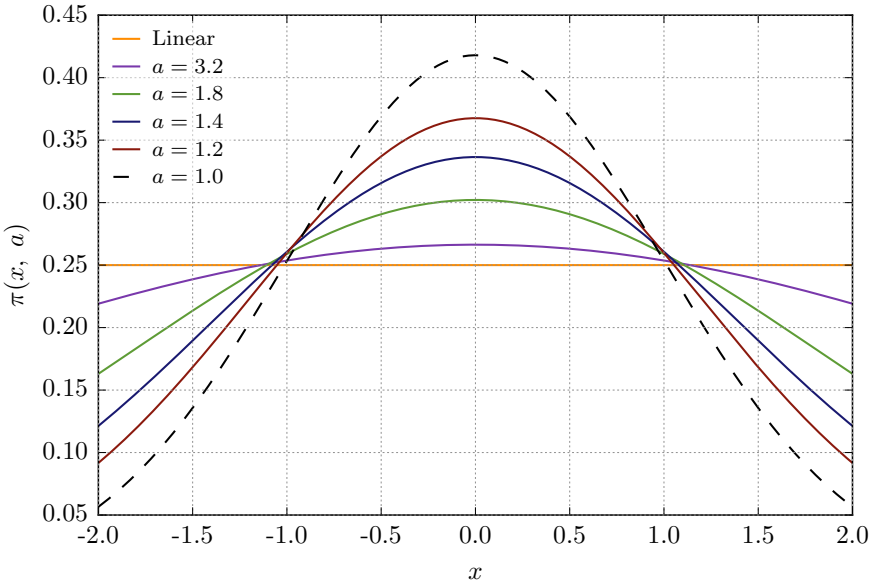


Figure 3.1: Trial probability densities for the importance sampling of equation (3.8), where the ideal probability density is given by the $a = 1.0$ curve.

3.4 The Metropolis algorithm

To make use of importance sampling it is essential be be able to generate points distributed according to an arbitrary, multidimensional probability density $\pi(\boldsymbol{x})$. The simplest and most well known method of doing this is the Metropolis algorithm.[72] Here points are sampled by a 'walker' moving through the space of $\boldsymbol{x}$ by accepting or rejecting moves based upon the change in value of $\pi(\boldsymbol{x})$ between points. To fully explore the space it is essential to perform a sufficient number of moves and ensure that moves are proposed using an ergodic scheme where any point $\boldsymbol{x}'$ can be reached from a point $\boldsymbol{x}$ in a finite number of moves. If the probability of proposing a move from $\boldsymbol{x}$ to $\boldsymbol{x}'$ is $T(\boldsymbol{x}' \leftarrow \boldsymbol{x})$, and the probability of acceptance is $P(\boldsymbol{x}' \leftarrow \boldsymbol{x})$, then,

after a large number of moves, there is an equilibrium

$$n(\boldsymbol{x})T(\boldsymbol{x}' \leftarrow \boldsymbol{x})P(\boldsymbol{x}' \leftarrow \boldsymbol{x})d\boldsymbol{x}d\boldsymbol{x}' \;=\; n(\boldsymbol{x}')T(\boldsymbol{x} \leftarrow \boldsymbol{x}')P(\boldsymbol{x} \leftarrow \boldsymbol{x}')d\boldsymbol{x}'d\boldsymbol{x} \;, \tag{3.11}$$

where $n(\boldsymbol{x})$ is the density of walkers at $\boldsymbol{x}$. This equation is known as the detailed balance condition and it provides the ratio of equilibrium walker densities at pairs of points,

$$\frac{n(\boldsymbol{x})}{n(\boldsymbol{x}')} \;=\; \frac{P(\boldsymbol{x} \leftarrow \boldsymbol{x}')T(\boldsymbol{x} \leftarrow \boldsymbol{x}')}{P(\boldsymbol{x}' \leftarrow \boldsymbol{x})T(\boldsymbol{x}' \leftarrow \boldsymbol{x})} \;. \tag{3.12}$$

The Metropolis algorithm accepts a move with probability

$$P_{\mathrm{M}}(\boldsymbol{x}' \leftarrow \boldsymbol{x}) \;=\; \min\left(1,\; \frac{T(\boldsymbol{x} \leftarrow \boldsymbol{x}')\pi(\boldsymbol{x}')}{T(\boldsymbol{x}' \leftarrow \boldsymbol{x})\pi(\boldsymbol{x})}\right) \;. \tag{3.13}$$

Considering the case where $T(\boldsymbol{x} \leftarrow \boldsymbol{x}')\pi(\boldsymbol{x}') \geq T(\boldsymbol{x}' \leftarrow \boldsymbol{x})\pi(\boldsymbol{x})$, the ratio of Metropolis acceptance probabilities for a forwards and reverse move are

$$\frac{P_{\mathrm{M}}(\boldsymbol{x} \leftarrow \boldsymbol{x}')}{P_{\mathrm{M}}(\boldsymbol{x}' \leftarrow \boldsymbol{x})} \;=\; \frac{\frac{T(\boldsymbol{x}' \leftarrow \boldsymbol{x})\pi(\boldsymbol{x})}{T(\boldsymbol{x} \leftarrow \boldsymbol{x}')\pi(\boldsymbol{x}')}}{1} \;=\; \frac{T(\boldsymbol{x}' \leftarrow \boldsymbol{x})\pi(\boldsymbol{x})}{T(\boldsymbol{x} \leftarrow \boldsymbol{x}')\pi(\boldsymbol{x}')} \;. \tag{3.14}$$

Substituting this expression into equation (3.12) yields

$$\frac{n(\boldsymbol{x})}{n(\boldsymbol{x}')} \;=\; \frac{\pi(\boldsymbol{x})}{\pi(\boldsymbol{x}')} \;, \tag{3.15}$$

which produces the desired sampling density. The same result is obtained by considering $T(\boldsymbol{x} \leftarrow \boldsymbol{x}')\pi(\boldsymbol{x}') < T(\boldsymbol{x}' \leftarrow \boldsymbol{x})\pi(\boldsymbol{x})$. A nice feature of this algorithm is that there is no requirement for $\pi(\boldsymbol{x})$ to be normalised, as it

only ever appears as a ratio.

A simple implementation of the Metropolis algorithm is outlined in Algorithm 1. To be accurate this procedure must include an equilibration period before any sampling points are recorded, as the start of the walk is biased by the choice of the initial position. It is not possible to calculate the length of the equilibration period required *a priori*, but the inspection of the average value of blocks of consecutive samples can indicate when the initial bias has been lost.

Algorithm 1 A basic Metropolis algorithm

```
set x to an arbitrary position
while collecting samples do
    propose x' from T(x' ← x)
    decide to accept move with probability P_M(x' ← x)
    if move accepted then
        set x to x'
    end if
    add x to a list of sampling points
end while
```

The efficiency of a Metropolis sampling scheme is dependent upon the magnitude of the transitions proposed by $T(\boldsymbol{x}' \leftarrow \boldsymbol{x})$. This 'step size' is commonly selected to make the average acceptance of transitions about 50%.[73] If transitions are accepted with an average probability which is either very high or very low, then the sampling space is clearly explored rather inefficiently. Whilst an average acceptance probability of 50% is midway between the extremes of these two regimes, there is no rigorous justification that it represents an optimal sampling strategy.

3.5 Langevin sampling

Langevin sampling[74] is an alternative method for selecting points from a probability distribution. Here the walkers can be thought of as particles diffusing through a carrier fluid which has a velocity $\boldsymbol{v}(\boldsymbol{x})$. The particle current density, $\boldsymbol{j}(\boldsymbol{x},\ t)$, is therefore given by

$$\boldsymbol{j}(\boldsymbol{x},\ t)\ =\ -D\nabla n(\boldsymbol{x},\ t)\ +\ \boldsymbol{v}(\boldsymbol{x})n(\boldsymbol{x},\ t)\ , \tag{3.16}$$

where D is a diffusion constant and $n(\boldsymbol{x},\ t)$ is the number density of particles. Combining this law of motion with a particle conservation law,

$$\nabla\cdot\boldsymbol{j}(\boldsymbol{x},\ t)\ +\ \frac{\partial n(\boldsymbol{x},\ t)}{\partial t}\ =\ 0\ , \tag{3.17}$$

gives the Fokker–Planck equation

$$\frac{\partial n(\boldsymbol{x},\ t)}{\partial t}\ =\ D\nabla^2 n(\boldsymbol{x},\ t)\ -\ \nabla\cdot[\,\boldsymbol{v}(\boldsymbol{x})n(\boldsymbol{x},\ t)\,]\ =\ \hat{L}n(\boldsymbol{x},\ t)\ . \tag{3.18}$$

where $\hat{L}$ is the Fokker–Planck operator. Finally, when choosing a carrier fluid velocity with a form

$$\boldsymbol{v}(\boldsymbol{x})\ =\ D\nabla\left(\ln\left[\pi(\boldsymbol{x})\right]\right)\ , \tag{3.19}$$

equation (3.18) becomes

$$\frac{\partial n(\boldsymbol{x},\ t)}{\partial t}\ =\ D\nabla^2 n(\boldsymbol{x},\ t)\ -\ D\nabla\cdot[\,\nabla\left(\ln\left[\pi(\boldsymbol{x})\right]\right)n(\boldsymbol{x},\ t)\,]\ . \tag{3.20}$$

This is now an expression that can be turned into a sampling technique. As $t \to \infty$ the particles will settle into a steady state where $\frac{\partial n}{\partial t} = 0$ and $n(\boldsymbol{x}) \propto \pi(\boldsymbol{x})$. To sample points from $\pi(\boldsymbol{x})$ it is sufficient to update the positions of a population of walkers according to Langevin drift-diffusion dynamics,

$$\Delta \boldsymbol{x} \;=\; \boldsymbol{v}(\boldsymbol{x})\Delta t \;+\; \boldsymbol{\xi}\sqrt{2D\Delta t}\;. \tag{3.21}$$

The first term is the contribution from the drift velocity in equation (3.19) and the second term is the contribution from the diffusive part of equation (3.16) where $\boldsymbol{\xi}$ is vector of independent random numbers drawn from a Gaussian distribution of unit variance. In the long time limit the walker positions sample the selected probability density $\pi(\boldsymbol{x})$.

As with Metropolis sampling, there is no requirement for $\pi(\boldsymbol{x})$ to be normalised and walkers must first be equilibrated to avoid a bias from the initial selection of positions. There is a compromise to be made over the selection of the walker step size: using a large time step allows a walker to explore space far more rapidly, but equation (3.21) is exact only in the limit $\Delta t \to 0$. A finite time step neglects the variation of $\boldsymbol{v}(\boldsymbol{x})$ between sampling points, introducing small errors into the dynamics. For accurate results it is necessary to use very small time steps, but a consequence of this is an increase of serial correlation in the data. When consecutive samples are strongly correlated it takes many time steps to gather statistically new information.

3.6 Serial correlation

Monte Carlo errors are obtained from the standard error in the mean of all collected samples, which is determined by their variance. The presence of serial correlation becomes problematic when trying to assess this error; if samples are not statistically independent then a straightforward calculation of their variance will be an underestimate. A simple approach to estimating the statistical significance of samples is the reblocking algorithm.[75] This procedure groups data into a series of contiguous blocks and the variance is calculated from the values of the block averages. As the block size increases each block average becomes increasingly independent and the calculated variance increases up to a plateau. Here the block size is sufficiently large that the serial correlation between blocks has been removed and the value of the variance at the plateau yields an estimate of the true sample variance.

3.7 Correlated sampling

It is often important to have an accurate estimate of the difference between two related Monte Carlo integrations. Two pertinent examples are during optimisation schemes, discussed briefly in Section 4.4.4, and derivative estimation, discussed in Chapter 5. The simplest expression for a difference between two integrals is given by

$$\begin{aligned} \Delta I &= I_2(\lambda_2) - I_1(\lambda_1) \\ &= \int g(\boldsymbol{x},\ \lambda_2)\pi(\boldsymbol{x},\ \lambda_2)d\boldsymbol{x} - \int g(\boldsymbol{x},\ \lambda_1)\pi(\boldsymbol{x},\ \lambda_1)d\boldsymbol{x}\ , \end{aligned} \tag{3.22}$$

where both the score function, $g(\boldsymbol{x},\ \lambda)$, and probability density, $\pi(\boldsymbol{x},\ \lambda)$, depend upon a generalised parameter λ. For stochastic integrations, the value of ΔI can be difficult to obtain, especially when the magnitude of ΔI is small. If the integrals are performed in isolation, the error in the difference is given by

$$\sigma_{\Delta I} \ = \ \sqrt{\sigma_{I_2}^2 \ + \ \sigma_{I_1}^2} \ , \tag{3.23}$$

where σ denotes the statistical error in each quantity. As $\lambda_2 \rightarrow \lambda_1$ and ΔI tends towards zero, $\sigma_{\Delta I}$ remains constant and swamps any useful information.

The technique of correlated sampling[34] addresses this issue by using the same set of sampling points to evaluate multiple integrals. Equation (3.22) can be expressed in the form

$$\Delta I \ = \ \int \left[g(\boldsymbol{x},\ \lambda_2) \frac{\pi(\boldsymbol{x},\ \lambda_2)}{\pi(\boldsymbol{x},\ \lambda_1)} \ - \ g(\boldsymbol{x},\ \lambda_1) \right] \pi(\boldsymbol{x},\ \lambda_1) d\boldsymbol{x} \ , \tag{3.24}$$

where the new score function in the square brackets tends smoothly to zero as λ_2 and λ_1 become more similar. This property is essential, as it ensures that the statistical error in ΔI tends to zero as ΔI tends to zero, alleviating the problem of equation (3.23). For an integration of N samples, ΔI is evaluated as

$$\Delta I \ = \ \frac{1}{N} \sum_{i=1}^{N} \left[\, g(\boldsymbol{x}_i,\ \lambda_2) \, w^{\mathrm{c}}(\boldsymbol{x}_i,\ \lambda_2,\ \lambda_1) \ - \ g(\boldsymbol{x}_i,\ \lambda_1) \, \right] \ , \tag{3.25}$$

where $\{\boldsymbol{x}_i\}$ are distributed according to $\pi(\boldsymbol{x},\ \lambda_1)$ and the weighting of correlated samples is given by

$$w^{\mathrm{c}}(\boldsymbol{x},\ \lambda_2,\ \lambda_1) \ = \ \frac{\pi(\boldsymbol{x},\ \lambda_2)}{\pi(\boldsymbol{x},\ \lambda_1)} \ . \tag{3.26}$$

Chapter 4

Quantum Monte Carlo methods

4.1 Introduction

The QMC family encompasses several different types of algorithm, all based on random sampling within quantum mechanics calculations. This chapter focusses on just two, the DMC and VMC methods, where both are long-established tools for performing averages over stochastically sampled wave functions. Many features are shared by both methods, and this chapter provides an outline of all the relevant theory required to produce basic implementations and perform total energy calculations.

4.2 Variational Monte Carlo

4.2.1 The variational principle

The VMC method is based upon the quantum mechanical variational principle, which states that for any physical wave function, $\Psi_{\mathrm{T}}(\bar{\boldsymbol{r}})$, the expectation value

$$E_{\mathrm{VMC}} \;=\; \frac{\int \Psi_{\mathrm{T}}^{*}(\bar{\boldsymbol{r}})\hat{H}\Psi_{\mathrm{T}}(\bar{\boldsymbol{r}})d\bar{\boldsymbol{r}}}{\int \Psi_{\mathrm{T}}^{*}(\bar{\boldsymbol{r}}')\Psi_{\mathrm{T}}(\bar{\boldsymbol{r}}')d\bar{\boldsymbol{r}}'} \;, \tag{4.1}$$

where $\hat{H}$ is the Born–Oppenheimer Hamiltonian, is greater than or equal to the exact Born–Oppenheimer ground state energy, E_0. The equality is only obtained when the wave function $\Psi_{\mathrm{T}}(\bar{\boldsymbol{r}})$ is proportional to the ground state wave function, $\Psi_0(\bar{\boldsymbol{r}})$. If the error in the normalised wave function,

$$\delta_{\Psi_{\mathrm{T}}}(\bar{\boldsymbol{r}}) \;=\; \frac{\Psi_{\mathrm{T}}(\bar{\boldsymbol{r}})}{\sqrt{\int |\Psi_{\mathrm{T}}(\bar{\boldsymbol{r}}')|^2 d\bar{\boldsymbol{r}}'}} \;-\; \Psi_0(\bar{\boldsymbol{r}}) \;, \tag{4.2}$$

is small then the error in the energy, $\delta_{E_{\mathrm{VMC}}} = E_{\mathrm{VMC}} - E_0$ is of order ${\delta_{\Psi_{\mathrm{T}}}}^2$. Thus, a fairly good estimate of the ground state wave function yields an even better estimate of the ground state energy.

4.2.2 Evaluating the total energy

To evaluate equation (4.1) efficiently using a Monte Carlo method the integrand must be partitioned into a normalised probability density and a score function. Whilst not optimal,[76,77] the most prevalent choice is to select

$\pi(\bar{\boldsymbol{r}}) \propto |\Psi_{\mathrm{T}}(\bar{\boldsymbol{r}})|^2$,

$$E_{\mathrm{VMC}} \;=\; \int E_{\mathrm{L}}(\bar{\boldsymbol{r}}) \frac{|\Psi_{\mathrm{T}}(\bar{\boldsymbol{r}})|^2}{\int |\Psi_{\mathrm{T}}(\bar{\boldsymbol{r}}')|^2 d\bar{\boldsymbol{r}}'} d\bar{\boldsymbol{r}} \;, \tag{4.3}$$

giving the 'local energy' as a score function

$$E_{\mathrm{L}}(\bar{\boldsymbol{r}}) \;=\; \frac{\hat{H}\Psi_{\mathrm{T}}(\bar{\boldsymbol{r}})}{\Psi_{\mathrm{T}}(\bar{\boldsymbol{r}})} \;. \tag{4.4}$$

Electronic configurations are sampled from $\pi(\bar{\boldsymbol{r}})$ using the Metropolis algorithm, and the VMC energy is evaluated using

$$E_{\mathrm{VMC}} \;\simeq\; \frac{1}{N} \sum_{i=1}^{N} E_{\mathrm{L}}(\bar{\boldsymbol{r}}_i) \;, \tag{4.5}$$

where N is the number of samples.

As $\Psi_{\mathrm{T}}(\bar{\boldsymbol{r}})$ tends to $\Psi_0(\bar{\boldsymbol{r}})$, each $E_{\mathrm{L}}(\bar{\boldsymbol{r}}_i)$ tends to E_0, and the stochastic error in E_{VMC} decreases towards zero. This is known as the 'zero variance principle'[78] which allows the quality of $\Psi_{\mathrm{T}}(\bar{\boldsymbol{r}})$ to be measured using the variance of $\{E_{\mathrm{L}}(\bar{\boldsymbol{r}}_i)\}$.

The variational or zero variance principles can now be employed to optimise trial wave functions. As the parameters of $\Psi_{\mathrm{T}}(\bar{\boldsymbol{r}})$ are adjusted, successive evaluations of the VMC energy using equation (4.5) assess the quality of the wave function. By minimising either the energy itself, or the variance of the local energy, $\Psi_{\mathrm{T}}(\bar{\boldsymbol{r}})$ can be improved towards the exact ground state. More details of this optimisation process, including the form of initial QMC wave functions, are discussed in Section 4.4.

4.2.3 An outline of the algorithm

Below is an outline of a VMC algorithm, consisting of a simple equilibration phase followed by the calculation of the VMC total energy.

Algorithm 2 A basic VMC algorithm

```
set r̄ to an arbitrary position
set a to 1

while average acceptance is not ∼50% do
    adjust a to make average acceptance closer to 50%
    clear store of acceptance decisions
    for 1000 steps do
        propose r̄′ from a Gaussian distribution of width a centred at r̄
        decide to accept move with probability P_M(r̄′ ← r̄)
        if move accepted then
            set r̄ to r̄′
        end if
        store acceptance decision
    end for
    calculate average acceptance from stored decisions
end while

for the desired number of VMC samples do
    propose r̄′ from a Gaussian distribution of width a centred at r̄
    decide to accept move with probability P_M(r̄′ ← r̄)
    if move accepted then
        set r̄ to r̄′
    end if
    calculate and store the local energy E_L(r̄)
end for

calculate E_VMC from the store of local energies
```

4.3 Diffusion Monte Carlo

4.3.1 The imaginary time Schrödinger equation

The DMC method is a stochastic approach for improving a trial ground state wave function using the solutions of the imaginary-time Born–Oppenheimer Schrödinger equation,

$$\left[-\frac{1}{2}\nabla^2_{\bar{\boldsymbol{r}}} \; + \; U(\bar{\boldsymbol{r}})\right] \Psi(\bar{\boldsymbol{r}},\; t) \;=\; -\frac{\partial}{\partial t}\Psi(\bar{\boldsymbol{r}},\; t) \;, \tag{4.6}$$

where the potential term, $U(\bar{\boldsymbol{r}})$, is as described by equation (2.4). Given an arbitrary starting state,

$$\Psi_{\mathrm{T}}(\bar{\boldsymbol{r}}) \;=\; \Psi(\bar{\boldsymbol{r}},\; t=0) \;=\; \sum_j c_j \Psi_j(\bar{\boldsymbol{r}}) \;, \tag{4.7}$$

expressed in the complete basis of energy eigenfunctions $\{\Psi_j(\bar{\boldsymbol{r}})\}$, its time evolution is given by

$$\Psi(\bar{\boldsymbol{r}},\; t) \;=\; \sum_j c_j e^{-E_j t} \Psi_j(\bar{\boldsymbol{r}}) \;, \tag{4.8}$$

where E_j are the corresponding eigenvalues. Assuming a non-zero ground state component, $c_0 \neq 0$, the long-time limit of the solution,

$$\Psi(\bar{\boldsymbol{r}},\; t \to \infty) \;=\; c_0 e^{-E_0 t} \Psi_0(\bar{\boldsymbol{r}}) \;, \tag{4.9}$$

projects out the exact ground state wave function.

Equation (4.6) is fairly straightforward to simulate. Ignoring the contribution from $U(\bar{\boldsymbol{r}})$ gives a diffusion equation with a diffusion constant $D = \frac{1}{2}$. In this case the wave function can be interpreted as the number density of a population of walkers diffusing through the space of $\bar{\boldsymbol{r}}$. The contribution from $U(\bar{\boldsymbol{r}})$ produces a solution defining an exponential growth or decay. To introduce this effect into a diffusion simulation, a weight, w, can be assigned to each walker which has an initial value of one and which changes according to

$$\frac{dw}{dt} \;=\; -U(\bar{\boldsymbol{r}}(t))\, w \;. \tag{4.10}$$

The wave function is now manifest as the weighted number density of the population of walkers.

4.3.2 Branching

As a simulation of equation (4.6) progresses, the exponential growth of equation (4.10) means that the weight of a single walker inevitably becomes far greater than all of the others. This makes the sampling process extremely inefficient as a lot of computational effort is spent sampling with the other walkers which only contribute negligibly to the final result. Only using a single walker to do the sampling does not help matters, as the exponential nature of weight accumulation means that a minority of points along its trajectory will contribute overwhelmingly. Another problem arises when simulating atoms or other finite systems. In the long time limit the diffusive dynamics of the walkers drives them to electronic configurations that are very dispersed over space. As the electron positions move further from

the nuclear positions, the high potential energies quickly reduce the walker weights to negligible values, and as $t \to \infty$ the probability of a walker making a significant contribution to the wave function tends to zero.

These problems can be partly addressed via the introduction of an energy shift, E_{T}, into equation (4.6),

$$\left[-\frac{1}{2}\nabla^2_{\bar{\boldsymbol{r}}} \; + \; U(\bar{\boldsymbol{r}}) \; - \; E_{\mathrm{T}}\right] \Psi(\bar{\boldsymbol{r}},\; t) \; = \; -\frac{\partial}{\partial t}\Psi(\bar{\boldsymbol{r}},\; t) \; . \tag{4.11}$$

This leaves the diffusive dynamics unchanged, but changes the weight accumulation to

$$\frac{dw}{dt} \; = \; -[U(\bar{\boldsymbol{r}}(t)) \; - \; E_{\mathrm{T}}] \, w \; . \tag{4.12}$$

The long time solution to equation (4.11) is then given by

$$\Psi(\bar{\boldsymbol{r}},\; t \to \infty) \; = \; c_0 e^{-(E_0 \; - \; E_{\mathrm{T}})t} \Psi_0(\bar{\boldsymbol{r}}) \; , \tag{4.13}$$

which differs only in a normalisation factor. Over the course of a simulation E_{T} is adjusted to keep the average walker weight close to unity and can be used to estimate the value of E_0.

The final part of the solution is the introduction of a branching algorithm which replaces high weight walkers with multiple walkers of unit weight and removes walkers with a low weight. Each diffusive step the number of walkers spawned from a walker of weight w_{in} is given by

$$N_{\mathrm{spawn}} \; = \; w_{\mathrm{out}} \; = \; \mathtt{INT}(w_{\mathrm{in}} \; + \; \eta) \; , \tag{4.14}$$

where η is a uniform random number in the range $[0,\ 1)$, w_{out} is the resultant weight and $\texttt{INT}(x)$ truncates a floating point number x to the largest integer not greater than x. When averaged over multiple branching events this procedure preserves the average weight at all points in the simulation,

$$\begin{aligned} \langle w_{\text{out}} \rangle &= [\texttt{INT}(w_{\text{in}}) + 1][w_{\text{in}} - \texttt{INT}(w_{\text{in}})] \\ &\quad + [\texttt{INT}(w_{\text{in}})][(\texttt{INT}(w_{\text{in}}) + 1) - w_{\text{in}}] , \\ &= w_{\text{in}} . \end{aligned} \tag{4.15}$$

Aside from their weight, the spawned walkers are identical to their parent but undergo independent diffusive motion. The main consequences of this procedure are that the walker population is concentrated in regions of low $U(\bar{\boldsymbol{r}})$, and that the significance of each walker is roughly equivalent.

Whilst bringing obvious efficiency benefits, the fluctuations in the walker population need to be managed carefully to produce a useful algorithm. The exponential growth of equation (4.12) sets the spawning rate of walkers, so changes in the population can be moderated by carefully selecting E_{T} over the course of a simulation. A simple scheme[79] is

$$E_{\text{T}}(t) = E^{*}_{\text{DMC}} - \frac{1}{T_{\text{pop}}} \ln\left(\frac{W(t)}{N_0}\right) , \tag{4.16}$$

where E^{*}_{DMC} is the current best estimate of the energy of the ground state wave function, T_{pop} is a time scale over which fluctuations in the population are damped, $W(t)$ is the total weight of all walkers and N_0 is the target number of walkers in the population.

If T_{pop} is too large then the strength of the population control may be in-

sufficient to avoid a catastrophically large population explosion or extinction. However, choosing T_{pop} to be too small results in a population control bias by limiting the walkers' response to changes in $U(\bar{\boldsymbol{r}})$. In regions of favourable configuration space the natural proliferation of walkers is restricted by an artificial adjustment of E_{T}, and in regions of unfavourable space an artificial population is maintained. Both of these effects result in an overestimate of the total energy. It is relatively easy to check for population control effects and for conventional simulation populations and lengths the bias is negligible.

4.3.3 The fermion sign problem

Equation (4.9) is not particularly useful for determining electronic wave functions. The lowest energy solution of the many-electron Hamiltonian in equation (2.3) is a totally symmetric many-boson state, and any trial wave function with a non-zero symmetric component will eventually converge to this bosonic solution. This is the root of the 'fermion sign problem' in DMC where, to obtain fermionic properties, antisymmetry must be maintained in the wave function over the course of a simulation. Trial wave functions for QMC calculations are discussed in Section 4.4, but for now we assume that it is possible to generate a real, completely antisymmetric trial wave function $\Psi_{\text{T}}(\bar{\boldsymbol{r}}) = \Psi(\bar{\boldsymbol{r}},\ t = 0)$. In principle this guarantees an antisymmetric ground state, but in practice extracting this from a simulation is prohibitively difficult.

An antisymmetric wave function can be sampled by selecting initial electronic configurations from $|\Psi(\bar{\boldsymbol{r}},\ t = 0)|$ and choosing the sign of the unit

walker weights to be the same as that of the wave function at the walker positions. The walkers are then allowed to diffuse and spawn as described in Sections 4.3.1 and 4.3.2, and after a period of time their weighted number density will represent $\Psi(\bar{\boldsymbol{r}},\ t)$. However, the distribution sampled by walkers with a positive weight contains a non-zero symmetric component, and in the long-time limit the contribution from the positive walkers yields the bosonic ground state of equation (4.9). The same is true of the negative walkers, which converge to the negative bosonic solution. This presents a numerical barrier that prevents the extraction of the fermionic ground state obtained from the sum of the two distributions. As the fermionic ground state has a higher energy, E_0^{F}, than the bosonic ground state, E_0^{B}, its relative decay, $\exp^{-(E_0^{\mathrm{F}}-E_0^{\mathrm{B}})}$, means that it becomes increasingly difficult to determine. With time the positive and negative distributions become increasingly similar and the exponentially decaying fermionic solution is swamped by noise.[80]

4.3.4 Importance sampling

The DMC sign problem can be mitigated substantially with the use of an importance sampling transformation.[81,82] Taking a time independent trial wave function, $\Psi_{\mathrm{T}}(\bar{\boldsymbol{r}})$, from equation (4.7) and a solution of the imaginary-time Schrödinger equation from equation (4.8), their product,

$$f(\bar{\boldsymbol{r}},\ t)\ =\ \Psi_{\mathrm{T}}(\bar{\boldsymbol{r}})\Psi(\bar{\boldsymbol{r}},\ t)\ , \tag{4.17}$$

coupled with the identity

$$\begin{aligned}
\frac{1}{2}\nabla^2 f &= \frac{1}{2}\nabla^2(\Psi_{\mathrm{T}}\Psi) \\
&= \frac{1}{2}\Psi_{\mathrm{T}}\nabla^2\Psi \;+\; \frac{1}{2}\Psi\nabla^2\Psi_{\mathrm{T}} \;+\; \nabla\Psi\cdot\nabla\Psi_{\mathrm{T}} \\
&= \frac{1}{2}\Psi_{\mathrm{T}}\nabla^2\Psi \;-\; \frac{1}{2}\Psi\nabla^2\Psi_{\mathrm{T}} \;+\; \nabla(\Psi\nabla\Psi_{\mathrm{T}}) \\
&= \frac{1}{2}\Psi_{\mathrm{T}}\nabla^2\Psi \;-\; \frac{1}{2}f\frac{\nabla^2\Psi_{\mathrm{T}}}{\Psi_{\mathrm{T}}} \;+\; \nabla(f\frac{\nabla\Psi_{\mathrm{T}}}{\Psi_{\mathrm{T}}}) \;,
\end{aligned} \tag{4.18}$$

can be used to rewrite equation (4.11) as

$$\begin{aligned}
-\Psi_{\mathrm{T}}(\bar{\boldsymbol{r}})\frac{\partial}{\partial t}\Psi(\bar{\boldsymbol{r}},\,t) &= \Psi_{\mathrm{T}}(\bar{\boldsymbol{r}})\left[-\frac{1}{2}\nabla^2_{\bar{\boldsymbol{r}}} \;+\; U(\bar{\boldsymbol{r}}) \;-\; E_{\mathrm{T}}\right]\Psi(\bar{\boldsymbol{r}},\,t) \\
-\frac{\partial}{\partial t}f(\bar{\boldsymbol{r}},\,t) &= \left[-\frac{1}{2}\nabla^2_{\bar{\boldsymbol{r}}} \;+\; U(\bar{\boldsymbol{r}}) \;-\; E_{\mathrm{T}}\right]f(\bar{\boldsymbol{r}},\,t) \\
&\qquad -\; \frac{1}{2}f(\bar{\boldsymbol{r}},\,t)\frac{\nabla^2_{\bar{\boldsymbol{r}}}\Psi_{\mathrm{T}}(\bar{\boldsymbol{r}})}{\Psi_{\mathrm{T}}(\bar{\boldsymbol{r}})} \;+\; \nabla_{\bar{\boldsymbol{r}}}\left(f(\bar{\boldsymbol{r}},\,t)\frac{\nabla_{\bar{\boldsymbol{r}}}\Psi_{\mathrm{T}}(\bar{\boldsymbol{r}})}{\Psi_{\mathrm{T}}(\bar{\boldsymbol{r}})}\right) \\
&= \left[-\frac{1}{2}\nabla^2_{\bar{\boldsymbol{r}}} \;+\; E_{\mathrm{L}}(\bar{\boldsymbol{r}}) \;-\; E_{\mathrm{T}}\right]f(\bar{\boldsymbol{r}},\,t) \\
&\qquad +\; \nabla_{\bar{\boldsymbol{r}}}[\,f(\bar{\boldsymbol{r}},\,t)\boldsymbol{v}(\bar{\boldsymbol{r}})\,]\;,
\end{aligned} \tag{4.19}$$

where

$$\boldsymbol{v}(\bar{\boldsymbol{r}}) \;=\; \frac{\nabla_{\bar{\boldsymbol{r}}}\Psi_{\mathrm{T}}(\bar{\boldsymbol{r}})}{\Psi_{\mathrm{T}}(\bar{\boldsymbol{r}})} \;=\; \frac{1}{2}\nabla_{\bar{\boldsymbol{r}}}\left(\,\ln[\,|\Psi_{\mathrm{T}}(\bar{\boldsymbol{r}})|^2\,]\,\right)\;. \tag{4.20}$$

In the absence of $E_{\mathrm{L}}(\bar{\boldsymbol{r}})$ and E_{T} this equation reduces to the Fokker–Planck form of equation (3.20) with $D = \frac{1}{2}$. Thus, moving walkers according to the corresponding drift-diffusion dynamics of equation (3.21),

$$\Delta\bar{\boldsymbol{r}} \;=\; \boldsymbol{v}(\bar{\boldsymbol{r}})\Delta t \;+\; \boldsymbol{\xi}\sqrt{\Delta t}\;, \tag{4.21}$$

samples electronic configurations from $|\Psi_{\mathrm{T}}(\bar{\boldsymbol{r}})|^2$. To incorporate the contribution of $E_{\mathrm{L}}(\bar{\boldsymbol{r}})$ and E_{T}, the drift-diffusing walkers gain a weight of

$$\frac{dw}{dt} \;=\; -[E_{\mathrm{L}}(\bar{\boldsymbol{r}}(t)) - E_{\mathrm{T}}]\, w \;. \tag{4.22}$$

It is these weights which differentiate between what is calculated in a VMC and an importance sampled DMC simulation; if all walkers had the same fixed weight, the DMC algorithm reduces to a way of sampling $|\Psi_{\mathrm{T}}(\bar{\boldsymbol{r}})|^2$ which uses Langevin dynamics instead of the Metropolis algorithm.

By replacing $U(\bar{\boldsymbol{r}})$ with $E_{\mathrm{L}}(\bar{\boldsymbol{r}})$ the importance sampling transformation has an enormous effect on the variability of the sampling procedure. The strong spatial variation and Coulomb singularities of $U(\bar{\boldsymbol{r}})$ can lead to catastrophic changes in the weights of the walkers, with the corresponding fluctuations in the walker population either consuming all of the available memory or causing a population extinction. In contrast, if $\Psi_{\mathrm{T}}(\bar{\boldsymbol{r}})$ is a fairly good approximation to the ground state wave function then, for the majority of electronic configurations, $E_{\mathrm{L}}(\bar{\boldsymbol{r}})$ is smooth and close to the ground state energy. As the population control becomes far less onerous in this regime, E_{T} also becomes a better estimate of E_0 and fluctuations in the walker weights are diminished.

In the long-time limit the importance sampled DMC algorithm does not converge to the bosonic ground state of the imaginary-time Schrödinger equation. When $\Psi_{\mathrm{T}}(\bar{\boldsymbol{r}})$ has both negative and positive regions, the drift velocity of equation (4.20) keeps walkers away from the nodal surface of $\Psi_{\mathrm{T}}(\bar{\boldsymbol{r}})$, which defines all points where $\Psi_{\mathrm{T}}(\bar{\boldsymbol{r}}) = 0$. This partitioning of space gives rise to

the fixed-node approximation[83,84] where, for an idealised DMC simulation with a time step of zero, walkers are prevented from crossing from one nodal pocket to another. Rather than sampling $\Psi_{\mathrm{T}}(\bar{\boldsymbol{r}})\Psi_0^{\mathrm{B}}(\bar{\boldsymbol{r}})$, where Ψ_0^{B} is the bosonic ground state, the importance sampled fixed-node DMC algorithm yields a distribution given by

$$f(\bar{\boldsymbol{r}},\ t \to \infty) \ = \ \Psi_{\mathrm{T}}(\bar{\boldsymbol{r}})\Psi_0^{\mathrm{FN}}(\bar{\boldsymbol{r}}) \ , \tag{4.23}$$

where $\Psi_0^{\mathrm{FN}}(\bar{\boldsymbol{r}})$ is known as the fixed-node ground state. As the product $\Psi_{\mathrm{T}}(\bar{\boldsymbol{r}})\Psi_0^{\mathrm{FN}}(\bar{\boldsymbol{r}})$ is non-negative it is not necessary to use both positive and negative walkers, avoiding the difficulties described in Section 4.3.3.

In a fixed-node DMC calculation the walker distribution within the nodal pockets tends to $\Psi_0^{\alpha}(\bar{\boldsymbol{r}})$, the lowest energy node-less wave function satisfying zero boundary conditions on the enclosing nodal surface. Outside a nodal pocket $\Psi_0^{\alpha}(\bar{\boldsymbol{r}})$ is zero, and inside it satisfies the equation

$$\hat{H}\Psi_0^{\alpha}(\bar{\boldsymbol{r}}) \ = \ E_0^{\alpha}\Psi_0^{\alpha}(\bar{\boldsymbol{r}}) \ + \ \delta^{\alpha} \ , \tag{4.24}$$

where the delta functions arise from the kinetic energy operator and the discontinuous gradient of $\Psi_0^{\alpha}(\bar{\boldsymbol{r}})$ as the nodal surface is crossed, and E_0^{α} is the energy of the nodal pocket. It can be proven[85] that E_0^{α} is always greater than or equal to the exact fermionic ground state energy E_0, making the DMC algorithm obey a variational principle: as the nodal surface of $\Psi_{\mathrm{T}}(\bar{\boldsymbol{r}})$ is improved towards the nodal surface of the fermionic ground state $\Psi_0^{\mathrm{F}}(\bar{\boldsymbol{r}})$, the DMC energy decreases towards E_0. If the nodal surface of $\Psi_{\mathrm{T}}(\bar{\boldsymbol{r}})$ is

exactly the same as the nodal surface of $\Psi_0^{\mathrm{F}}(\bar{\boldsymbol{r}})$, then $\Psi_0^{\mathrm{FN}}(\bar{\boldsymbol{r}}) = \Psi_0^{\mathrm{F}}(\bar{\boldsymbol{r}})$ and the exact fermionic ground state of the imaginary time Schrödinger equation is recovered.

4.3.5 Reducing time step errors

As described in Section 3.5 the Langevin dynamics of equation (4.21) are exact only in the limit of $\Delta t \to 0$. For finite time steps it is possible that the diffusive component will propose walker moves that cross the nodal surface. This is a violation of the fixed-node approximation and such moves are usually rejected. To reduce other time step errors it is possible to incorporate an additional Metropolis rejection step into the dynamics.[85] Taking the Metropolis rejection criterion from equation (3.13), each drift-diffusion move is accepted with the probability

$$P_{\mathrm{M}}(\bar{\boldsymbol{r}}' \leftarrow \bar{\boldsymbol{r}}) \;=\; \min\left(1,\; \frac{G_{\mathrm{L}}(\bar{\boldsymbol{r}} \leftarrow \bar{\boldsymbol{r}}',\; \Delta t)|\Psi_{\mathrm{T}}(\bar{\boldsymbol{r}}')|^2}{G_{\mathrm{L}}(\bar{\boldsymbol{r}}' \leftarrow \bar{\boldsymbol{r}},\; \Delta t)|\Psi_{\mathrm{T}}(\bar{\boldsymbol{r}})|^2}\right) \;, \tag{4.25}$$

where $G_{\mathrm{L}}(\bar{\boldsymbol{r}}' \leftarrow \bar{\boldsymbol{r}},\; \Delta t)$ is the Green's function describing the probability that a walker will transition from $\bar{\boldsymbol{r}}$ to $\bar{\boldsymbol{r}}'$ when moving under Langevin dynamics. A short-time approximation for $G_{\mathrm{L}}(\bar{\boldsymbol{r}}' \leftarrow \bar{\boldsymbol{r}},\; \Delta t)$ that is exact when $\Delta t \to 0$ is given by

$$G_{\mathrm{L}}(\bar{\boldsymbol{r}}' \leftarrow \bar{\boldsymbol{r}},\; \Delta t) \;=\; \frac{1}{(2\pi\Delta t)^{\frac{3N_e}{2}}} \exp\left(-\frac{[\bar{\boldsymbol{r}}' - \bar{\boldsymbol{r}} - \boldsymbol{v}(\bar{\boldsymbol{r}})\Delta t]^2}{2\Delta t}\right) \;, \tag{4.26}$$

where N_e is the number of electrons. The introduction of this extra rejection step means that, in the absence of walker weights, $|\Psi_{\mathrm{T}}(\bar{\boldsymbol{r}})|^2$ is always sampled

exactly, despite the finite time step of the Langevin dynamics. This preserves exact results in the limit of perfect importance sampling when $\Psi_{\mathrm{T}}(\bar{\boldsymbol{r}})$ is the ground state wave function. In this regime each evaluation of the local energy in equation (4.4) produces the exact fermionic ground state energy, irrespective of $\bar{\boldsymbol{r}}$. Thus, once the reference energy E_{T} has settled down, the weights of all walkers, determined by equation (4.12), are exactly unity. In practical simulations $\Psi_{\mathrm{T}}(\bar{\boldsymbol{r}})$ is never perfect, but it is usually a good enough approximation to the real ground state that the rejection step reduces time step errors significantly. As $\Delta t \to 0$, the Green's function becomes increasingly accurate and the Metropolis rejection probability tends to zero.

The time step error can be further reduced by using a symmetric branching factor, replacing equation (4.22) with

$$G_w(\bar{\boldsymbol{r}}' \leftarrow \bar{\boldsymbol{r}},\ \Delta t) \ = \ \exp\left(-\frac{1}{2}[E_{\mathrm{L}}(\bar{\boldsymbol{r}}') \ + \ E_{\mathrm{L}}(\bar{\boldsymbol{r}}) \ - \ 2E_{\mathrm{T}}]\Delta t\right) \ , \qquad (4.27)$$

where a walker's weight is multiplied by $G_w(\bar{\boldsymbol{r}}' \leftarrow \bar{\boldsymbol{r}},\ \Delta t)$ when performing a move from $\bar{\boldsymbol{r}}$ to $\bar{\boldsymbol{r}}'$. Using equations (4.26) and (4.27) the evolution of the DMC wave function can be expressed as the integral equation

$$\Psi(\bar{\boldsymbol{r}}',\ t+\Delta t) \ = \ \int G_{\mathrm{DMC}}(\bar{\boldsymbol{r}}' \leftarrow \bar{\boldsymbol{r}},\ \Delta t)\Psi(\bar{\boldsymbol{r}},\ t)d\bar{\boldsymbol{r}} \qquad (4.28)$$

where

$$G_{\mathrm{DMC}}(\bar{\boldsymbol{r}}' \leftarrow \bar{\boldsymbol{r}},\ \Delta t) \ = \ G_L(\bar{\boldsymbol{r}}' \leftarrow \bar{\boldsymbol{r}},\ \Delta t)G_w(\bar{\boldsymbol{r}}' \leftarrow \bar{\boldsymbol{r}},\ \Delta t) \ . \qquad (4.29)$$

4.3.6 Evaluating the total energy

The fixed-node DMC energy, E_{DMC}, can be evaluated from the long-time limit of the importance sampled wave function of equation (4.23),

$$\begin{aligned} E_{\mathrm{DMC}} &= E_0^{\mathrm{FN}} \\ &= \frac{\langle \Psi_{\mathrm{T}} | \hat{H} | \Psi_0^{\mathrm{FN}} \rangle}{\langle \Psi_{\mathrm{T}} | \Psi_0^{\mathrm{FN}} \rangle} \\ &= \frac{\langle \frac{\hat{H}\Psi_{\mathrm{T}}}{\Psi_{\mathrm{T}}} \Psi_{\mathrm{T}} | \Psi_0^{\mathrm{FN}} \rangle}{\langle \Psi_{\mathrm{T}} | \Psi_0^{\mathrm{FN}} \rangle} \\ &= \frac{\int E_{\mathrm{L}}(\bar{\boldsymbol{r}}) \Psi_{\mathrm{T}}(\bar{\boldsymbol{r}}) \Psi_0^{\mathrm{FN}}(\bar{\boldsymbol{r}}) d\bar{\boldsymbol{r}}}{\int \Psi_{\mathrm{T}}(\bar{\boldsymbol{r}}) \Psi_0^{\mathrm{FN}}(\bar{\boldsymbol{r}}) d\bar{\boldsymbol{r}}} \\ &= \frac{\int E_{\mathrm{L}}(\bar{\boldsymbol{r}}) f(\bar{\boldsymbol{r}},\ t \to \infty) d\bar{\boldsymbol{r}}}{\int f(\bar{\boldsymbol{r}},\ t \to \infty) d\bar{\boldsymbol{r}}} \\ &\simeq \frac{\sum_{i=1}^{N} E_{\mathrm{L}}(\bar{\boldsymbol{r}}_i)\, w_i}{\sum_{j=1}^{N} w_j}\ , \end{aligned} \tag{4.30}$$

where E_0^{FN} is the ground state energy of the fixed-node ground state wave function $\Psi_0^{\mathrm{FN}}(\bar{\boldsymbol{r}})$, N is the total number of samples from all walkers, $\{\bar{\boldsymbol{r}}_i\}$ are the electronic configurations visited over the course of the Langevin dynamics, and $\{w_i\}$ are the weights of the walkers at each associated configuration.

4.3.7 Drift velocity and local energy limiting

A significant source of error in DMC calculations comes from sampling electronic configurations near the nodal surface. Here both the drift velocity and local energy diverge, causing large time step errors and increasing the

variance of energy estimates respectively. To reduce these undesirable effects it is necessary to limit the magnitude of both quantities. A smoothly limited drift velocity[79] can be obtained from

$$\boldsymbol{v}_{\mathrm{lim}}(\bar{\boldsymbol{r}}) \;=\; \frac{\sqrt{1 \;+\; 2a|\boldsymbol{v}(\bar{\boldsymbol{r}})|^2\Delta t} \;-\; 1}{a|\boldsymbol{v}(\bar{\boldsymbol{r}})|^2\Delta t}\,\boldsymbol{v}(\bar{\boldsymbol{r}}) \;, \tag{4.31}$$

where a is a parameter of order unity in atomic units which determines the strength of the limiting. The local energy can then be limited using

$$E_{\mathrm{L,lim}}(\bar{\boldsymbol{r}}) \;=\; E^{*}_{\mathrm{DMC}} \;-\; [E^{*}_{\mathrm{DMC}} \;-\; E_{\mathrm{L}}(\bar{\boldsymbol{r}})]\frac{|\boldsymbol{v}_{\mathrm{lim}}(\bar{\boldsymbol{r}})|}{|\boldsymbol{v}(\bar{\boldsymbol{r}})|} \;, \tag{4.32}$$

where E^{*}_{DMC} represents the current best estimate of the DMC energy, as used in equation (4.16). A number of different limiting schemes exist,[86,87] but these relatively smooth forms are the most common and are usually found to work well.[11]

4.3.8 An outline of the algorithm

Below is an outline of a simple DMC algorithm which performs N_{eq} equilibration time steps and calculates E_{DMC} using N_{E} time steps.

Algorithm 3 A basic DMC algorithm

```
set N_0 to the target number of walkers
distribute N_0 walkers randomly
set the weight of all walkers to 1

set N_eq to the number of equilibration steps
set N_E to the number of calculation steps
for N_eq + N_E time steps do

    for each walker do
        calculate the drift velocity [eqn (4.20)]
        limit the drift velocity [eqn (4.31)]
        propose a configuration update [eqn (4.21)]
        accept or reject the update [eqn (4.25)]
        calculate the local energy [eqn (4.4)]
        limit the local energy [eqn (4.32)]
        update the walker weight [eqn (4.27)]
    end for

    if time step is equilibration then
        calculate E_DMC^eq [eqn (4.30)] using w and E_L from equilibration steps
        adjust the reference energy using E_DMC^eq [eqn (4.16)]
    else
        calculate E_DMC [eqn (4.30)] using w and E_L from calculation steps
        adjust the reference energy using E_DMC [eqn (4.16)]
    end if

    branch walkers [eqn (4.14)]

end for

return E_DMC
```

4.4 Trial wave functions

4.4.1 The nodal surface

The VMC and DMC methods are variational approaches; as an optimisable trial wave function is improved, the results of subsequent total energy calculations converge towards the ground state energy. In the case of VMC, the lowest, most accurate energy achievable is dependent on the flexibility of the trial wave function during the optimisation process. The DMC algorithm can be thought of as a super-variational approach, producing the lowest total energy compatible with the nodal surface supplied by the trial wave function. Improvements to the estimate of the nodal surface give more accurate DMC results.

Despite their importance in determining the topology of a wave function, only a few properties of nodal surfaces are known. For non-degenerate ground states the tiling theorem[88] states that there is only one distinct type of positive or negative nodal pocket, and thus the partitioning of space creates equivalent cells related via permutation symmetry. A DMC simulation therefore only requires that single nodal pocket is populated with walkers to determine the fixed-node ground state energy. The nodal structures of simple many body systems, He and H_3, have been characterised by fixing all but one particle and creating 3D cross sections through the full $3N$ dimensional space,[89–91] where it is found that the ground states have smooth, simple forms with the excited states of He displaying a higher symmetry. Wave functions obtained from HF calculations of Li to C atoms contain just 4 nodal regions

formed by the intersection of two nodal surfaces,[92] but many-body correlations relax this to the minimal number of 2 in the fully interacting ground state.[93]

4.4.2 Slater–Jastrow wave functions

Trial wave functions in QMC calculations are typically of the Slater–Jastrow form[94] consisting of an antisymmetric Slater determinant component multiplied by a node-less Jastrow factor.[95] The Jastrow factor is symmetric in the interchange of identical particles and introduces correlation into the wave function through an explicit dependence on the particle separations. Although the final DMC energy is independent of the form of the Jastrow factor, the improvement to the trial wave function reduces the statistical error bars in tandem with the time-step and population-control biases. The antisymmetric component is usually comprised of a combination of one or more Slater determinants of single-particle orbitals. Single-determinant Slater–Jastrow wave functions often perform well for solids, retrieving around 90% of the correlation energy for bulk pseudo-silicon,[95] but less well for molecules. Calculations performed using both restricted and unrestricted HF, and generalized valence bond[96] wave functions suggest that whilst there is a systematic improvement in the overall wave function, there is little improvement to the nodal surface.[97] Multiple determinant Slater-Jastrow wave functions can perform better, but often only a few configuration state functions (CSFs) contribute significantly to the improvement of the nodal structure over the single-determinant case; long expansions of CSFs are expensive to evaluate

and can include terms that spoil the nodal structure by introducing an artefact that is not cancelled by the subset of CSFs in the trial function.[98] Thus, when constructing a multiple determinant wave function it can be better to include just a few determinants built from orbitals with different angular momenta.

A unique feature of QMC methods is that they are unconstrained by a particular form of wave function. Compact wave functions consisting of an anti-symmetrised product of two-electron pairing functions ('geminals') and triplet-pairing Pfaffians have both been successfully employed in the study of atoms and molecules,[99–102] typically resulting in a better nodal structure than that obtained from HF calculations. Another method of improving upon a single-particle Slater determinant is to introduce backflow correlation parameters[101,103,104] which allow the orbitals to depend upon the positions of all the other electrons and, crucially, alter the nodes of the wave function.

4.4.3 Updating wave functions

The most computationally demanding tasks in QMC algorithms involve the evaluation of the trial wave function or its derivatives in order to calculate local energies and determine the sampling of electronic configurations. A Slater–Jastrow trial wave function can be expressed as,

$$\Psi_{\mathrm{T}}(\bar{\boldsymbol{r}}) \;=\; e^{J(\bar{\boldsymbol{r}})}|\boldsymbol{D}^{\uparrow}(\bar{\boldsymbol{r}}^{\uparrow})||\boldsymbol{D}^{\downarrow}(\bar{\boldsymbol{r}}^{\downarrow})|\;, \tag{4.33}$$

where $J(\bar{\boldsymbol{r}})$ is the Jastrow factor and the Slater determinants, $|\boldsymbol{D}^{\uparrow}(\bar{\boldsymbol{r}}^{\uparrow})|$ and $|\boldsymbol{D}^{\downarrow}(\bar{\boldsymbol{r}}^{\downarrow})|$, are obtained from Slater matrices of the form

$$D_{ij}^{\uparrow} \;=\; \psi_i^{\uparrow}(\boldsymbol{r}_j^{\uparrow}) \;, \tag{4.34}$$

where $\psi_i^{\uparrow}$ is the ith single-electron spin $\uparrow$ orbital and $\boldsymbol{r}_j^{\uparrow}$ is the position of the jth spin $\uparrow$ electron. Both the VMC and DMC methods can be made more efficient by replacing single, multidimensional configuration updates with the proposal of multiple one-electron moves which are accepted or rejected individually. Whilst this incurs some computational overhead, as multiple acceptance probabilities are required, one-electron updates allow much larger moves to be accepted, reducing the serial correlation between configurations.[73] The acceptance probability of a single spin up electron moving from $\bar{\boldsymbol{r}}_i^{\,\mathrm{old}}$ to $\bar{\boldsymbol{r}}_i^{\,\mathrm{new}}$ is determined from

$$\frac{\Psi_{\mathrm{T}}(\bar{\boldsymbol{r}}^{\,\mathrm{new}})}{\Psi_{\mathrm{T}}(\bar{\boldsymbol{r}}^{\,\mathrm{old}})} \;=\; e^{J(\bar{\boldsymbol{r}}^{\,\mathrm{new}})-J(\bar{\boldsymbol{r}}^{\,\mathrm{old}})}\frac{|\boldsymbol{D}^{\uparrow}(\bar{\boldsymbol{r}}^{\,\mathrm{new}})|}{|\boldsymbol{D}^{\uparrow}(\bar{\boldsymbol{r}}^{\,\mathrm{old}})|} \;. \tag{4.35}$$

For a system of N electrons the difference in the Jastrow factors $J(\bar{\boldsymbol{r}}^{\,\mathrm{new}}) - J(\bar{\boldsymbol{r}}^{\,\mathrm{old}})$ can be evaluated in a time proportional to N,[11] but evaluating the ratio of determinants,

$$\frac{|\boldsymbol{D}^{\uparrow}(\bar{\boldsymbol{r}}^{\,\mathrm{new}})|}{|\boldsymbol{D}^{\uparrow}(\bar{\boldsymbol{r}}^{\,\mathrm{old}})|} \;=\; \frac{|\boldsymbol{D}^{\uparrow}(\boldsymbol{r}_1,\,...,\,\boldsymbol{r}_i^{\,\mathrm{new}},\,...,\,\boldsymbol{r}_{N^{\uparrow}})|}{|\boldsymbol{D}^{\uparrow}(\boldsymbol{r}_1,\,...,\,\boldsymbol{r}_i^{\,\mathrm{old}},\,...,\,\boldsymbol{r}_{N^{\uparrow}})|} \;, \tag{4.36}$$

is less straightforward. The computational cost of constructing these determinants in isolation scales as $\mathcal{O}(N^{\uparrow\,3})$. However, as the determinants only differ in a single electron coordinate, the Sherman–Morrison formula[105,106]

can be used to evaluate equation (4.36) more efficiently,[107]

$$\frac{|\boldsymbol{D}^{\uparrow}(\boldsymbol{r}_1, ..., \boldsymbol{r}_i^{\text{new}}, ..., \boldsymbol{r}_{N^{\uparrow}})|}{|\boldsymbol{D}^{\uparrow}(\boldsymbol{r}_1, ..., \boldsymbol{r}_i^{\text{old}}, ..., \boldsymbol{r}_{N^{\uparrow}})|} = \sum_j \psi_i(\boldsymbol{r}_i^{\text{new}}) \bar{D}_{ji}^{\uparrow\,\text{old}} , \tag{4.37}$$

where $\bar{D}_{ji}$ is the inverse transpose of D_{ji}, given by

$$\bar{D}_{ji} = \frac{\text{cof}(D_{ji})}{\det(D)} . \tag{4.38}$$

Constructing the initial $\bar{D}_{ji}$ remains as computationally expensive as a regular determinant, but the evaluation of equation (4.35) is now possible in a time proportional to $N^{\uparrow}$, and the $\bar{D}_{ji}$ matrices can be updated after each single-electron move with $\mathcal{O}(N^{\uparrow\,2})$ operations.

4.4.4 Optimising wave functions

Before performing a comparatively expensive DMC calculation, the wave function can be optimised in VMC. This process is typically applied to the variational parameters in the Jastrow factor with the aim of increasing the efficiency of the subsequent DMC simulation. The variance of the DMC local energy is usually proportional to the difference between the ground-state energy and the VMC energy,[108,109] so lower VMC energies correspond to fewer DMC time steps required to reach a specific error bar. Changes to the Jastrow factor leave the nodal surface intact; only the optimisation of parameters defining the antisymmetric part of the wave function has the capacity to improve the mean value of DMC energy. Such parameters include the coefficients of the determinants in a multi-determinant wave function,

the parameters in backflow functions, or parameters defining the orbitals. Although in practice lower VMC energies usually lead to lower DMC energies, it is unclear how improvements in the energy relate to improvements in the nodal structure.

The two commonly used VMC optimisation methods are variance and energy minimisation. Variance minimisation[35] is possible due to the zero variance principle described in Section 4.2.2, which states that the statistical fluctuations in the energy estimate tend to zero as the wave function tends to the ground state. An advantage of variance minimisation schemes is the use of correlated sampling[35] where suitably reweighted electronic configurations generated from a single run are used to test many different parameter values. The benefits of this are twofold: it is not required to regenerate configurations for each run, making the process very efficient, and the correlation between the two simulations reduces the statistical noise in energy differences between them. The drawback of the reweighting scheme is that the required weights can vary rapidly as the parameters change, especially for large systems, leading to instabilities in optimization procedures.[110] This problem may be surmounted via the use of unreweighted variance minimisation[36] which has long been the preferred method for reliably generating very high quality trial wave functions.

Although initially unpopular due to the relative inefficiency of a basic implementation, energy minimisation[111–113] has a number of key advantages: whilst variance minimisation is relatively effective at optimising the parameters of the Jastrow factor, it performs less well when attempting to optimise the antisymmetric part of the wave function.[112] Trial wave functions can-

not, in general, exactly represent the ground state and thus the minima in energy and variance do not coincide; energy minimisation should be able to access lower VMC energies than variance minimisation. Finally, better estimates of expectation values other than the energy have been obtained from energy-minimised wave functions.[114,115]

The optimisation of DMC wave functions in VMC is limited by the sampling method. The Metropolis algorithm samples the wave function with a frequency proportional to the probability density, resulting in optimisation schemes primarily focussed on improving the wave function in regions far from the nodes. A direct optimisation in DMC should display improved convergence properties. Previous work[116–120] has shown some success in improving DMC energies by analysing local properties of the nodes. As DMC walkers are forbidden from crossing the nodal surface, a calculation where all the walkers are placed on one side of the boundary is completely decoupled from one with all the walkers on the other side. If the nodal surface is correct, the energy obtained from each calculation should be equal. In addition, the imposition of the fixed-node approximation creates discontinuities in the wave function derivative; the fixed-node approximation can be thought of as an infinite potential placed on the nodes of the trial wave function, separating the bosonic solutions for each nodal pocket. As the derivative of the ground state eigenfunction must be continuous across a node, the magnitude of the discontinuity can be used as a local measure of the quality of the nodal surface, with the discontinuity vanishing only when the nodal surface is correct. This information can be accessed by comparing the local density of DMC walkers attempting node crossings in both directions by

constructing 'flux histograms' from DMC calculations on either side of the node.[116] The matching of DMC energies and derivatives at the node has been used as the basis of a fitness function to guide the evolution of a nodal surface using a genetic algorithm.[117] Whilst this method has enabled the computation of the rovibrational states of CO–^{4}He$_N$ ($N \leq 10$) complexes,[121] the application to more general systems is not straightforward. In order to implement the method, a physically motivated parametrisation of the nodal surface is needed, which is a quantity that is inaccessible for more complex systems. The same obstacle hinders similar work where the accuracy of the nodal surface is assessed locally based on the distances between the nodal surfaces of ψ_T, $\hat{H}\psi_T$, and $\hat{T}\psi_T$, where ψ_T is the trial wave function, $\hat{H}$ is the Hamiltonian and $\hat{T}$ is the kinetic energy operator, which coincide for the exact wave function.[118]

Another DMC optimisation scheme based upon nodal discontinuities in the wave function derivative is self-healing DMC.[119,120] It is shown that, by locally smoothing discontinuities in the DMC wave function, a new trial wave function can be obtained with an improved nodal structure. Thus, the nodal surface can be systematically improved towards the ground state eigenfunction by repeatedly convolving the walker density obtained from a DMC simulation with a smoothing function that generates the next iteration of the wave function. For wave functions expanded in a complete basis, it is shown that the final accuracy is limited only by the statistical nature of the DMC algorithm. This approach has been successfully applied to the oxygen atom, and the N_2 and C_{20} molecules,[122] giving energies with an accuracy equivalent to energy minimisation in VMC. Whilst there is a much larger

prefactor in the computational cost of self-healing DMC compared to VMC energy minimisation, it exhibits a scaling linearly proportional to, rather than quadratic in, the number of optimised degrees of freedom, making it suitable for systems of fairly large size or complexity.

4.5 Pseudopotentials

4.5.1 Introduction

The computational cost of a QMC calculation scales rather drastically with the atomic number Z. As Z increases, the nuclear potential and the wave function vary over much shorter length scales, necessitating a much higher sampling density to get a good representation. In tandem, fluctuations in the local energy increase in the vicinity of a nucleus as both the kinetic energy and potential energy are large. The use of a more accurate trial function can reduce this effect a little, but in practice improved estimates of Ψ_{T} become much harder to obtain. Estimates of the scaling of the computational demand range from around $Z^{5.5}$ to $Z^{6.5}$,[108,123] prohibiting the use of QMC methods on heavier atoms.

However, most interesting interatomic properties are determined almost exclusively by the valence electrons. It is therefore a relatively small approximation to use pseudopotentials, where the effects of the innermost shells of electrons are absorbed into the nuclear potential. By replacing a bare Coulomb potential with a smoother, finite function, the variation of the wave function and fluctuations in the local energy are both reduced. It is

also possible to include relativistic effects, which become more important as the nuclear charge increases. The use of pseudopotentials in other electronic structure calculations is widespread, and many calculations would be too computationally demanding to perform in their absence. Pseudopotentials give good estimates of total energy differences between systems and, given their utility, the resultant errors are usually small enough to be acceptable.

4.5.2 Evaluating pseudopotentials

Pseudopotentials used in QMC methods are generally non-local, containing corrections dependent upon angular momentum. It is conventional to divide pseudopotentials into a local part, $V_{\mathrm{loc}}^{\mathrm{pp}}(r)$, common to all electrons, and a correction, $V_{\mathrm{nl},l}^{\mathrm{pp}}(r)$, for electrons with an angular momentum of l. The electron-ion terms in the Hamiltonian are thus obtained from

$$\sum_i V_{\mathrm{loc}}^{\mathrm{pp}}(r_i) \; + \; \sum_i \hat{V}_{\mathrm{nl},i}^{\mathrm{pp}} \tag{4.39}$$

where $\hat{V}_{\mathrm{nl},i}^{\mathrm{pp}}$ is a non-local operator which acts on a function $g(\boldsymbol{r})$ as

$$\hat{V}_{\mathrm{nl},i}^{\mathrm{pp}}\, g(\boldsymbol{r}_i) \; = \; \sum_l V_{\mathrm{nl},l}^{\mathrm{pp}}(r_i) \sum_{m=-l}^{l} Y_l^m(\Omega_i) \int Y_l^{m*}(\Omega_i') g(\boldsymbol{r}_i') d\Omega_i' \; . \tag{4.40}$$

Here Y_l^m are spherical harmonics, and the integrals select out the different angular momentum components of the function $g(\boldsymbol{r})$, ensuring each component (s, p, d, ...) experiences its own potential $V_{\mathrm{nl},l}^{\mathrm{pp}}(r_i)$. This expression can be simplified by choosing the z-axis to be along $\boldsymbol{r}$ and using the spherical

harmonic addition theorem to give

$$\hat{V}^{\mathrm{pp}}_{\mathrm{nl},\,i}\, g(\boldsymbol{r}_i) \;=\; \sum_l V^{\mathrm{pp}}_{\mathrm{nl},\,l}(r_i)\, \frac{2l+1}{4\pi} \int P_l[\cos(\theta'_i)] g(\boldsymbol{r}'_i) d\Omega'_i \;, \tag{4.41}$$

where P_l is a Legendre polynomial.

The contribution to the local energy from the non-local component of a pseudopotential is given by

$$\begin{aligned} V_{\mathrm{nl}} \;&=\; \frac{\hat{V}_{\mathrm{nl}}\Psi_{\mathrm{T}}}{\Psi_{\mathrm{T}}} \\ &=\; \sum_i \frac{\hat{V}^{\mathrm{pp}}_{\mathrm{nl},\,i}\Psi_{\mathrm{T}}}{\Psi_{\mathrm{T}}} \\ &=\; \sum_i V_{\mathrm{nl},\,i} \;. \end{aligned} \tag{4.42}$$

Considering, without loss of generality, a single atom at the origin yields

$$V_{\mathrm{nl},\,i} \;=\; \sum_l V^{\mathrm{pp}}_{\mathrm{nl},\,l}(r_i)\, \frac{2l+1}{4\pi} \int P_l[\cos(\theta'_i)] \frac{\Psi_{\mathrm{T}}(\boldsymbol{r}_1,\, ...,\, \boldsymbol{r}'_i,\, ...,\, \boldsymbol{r}_N)}{\Psi_{\mathrm{T}}(\boldsymbol{r}_1,\, ...,\, \boldsymbol{r}_i,\, ...,\, \boldsymbol{r}_N)} d\Omega'_{\boldsymbol{r}'_i} \;, \tag{4.43}$$

where the angular integration is over a sphere centred on the origin which passes through the position of electron i. During QMC simulations this integration is performed numerically on a grid using a quadrature rule where, to avoid bias, each evaluation of the integral uses a randomly selected grid orientation.[124]

4.5.3 Diffusion Monte Carlo

Non-local pseudopotentials complicate the exponential growth term in DMC. Equation (4.19) becomes

$$-\frac{\partial}{\partial t}f(\bar{\boldsymbol{r}},\ t)\ =\ \left[-\frac{1}{2}\nabla^2_{\bar{\boldsymbol{r}}}\ +\ E_{\mathrm{L}}(\bar{\boldsymbol{r}})-E_{\mathrm{T}}\right]f(\bar{\boldsymbol{r}},\ t)\ +\ \nabla_{\bar{\boldsymbol{r}}}\cdot[\ \boldsymbol{v}(\bar{\boldsymbol{r}})f(\bar{\boldsymbol{r}},\ t)\ ] + \left\{\frac{\hat{V}_{\mathrm{nl}}\Psi(\bar{\boldsymbol{r}},\ t)}{\Psi(\bar{\boldsymbol{r}},\ t)}\ -\ \frac{\hat{V}_{\mathrm{nl}}\Psi_{\mathrm{T}}(\bar{\boldsymbol{r}})}{\Psi_{\mathrm{T}}(\bar{\boldsymbol{r}})}\right\}f(\bar{\boldsymbol{r}},\ t) \tag{4.44}$$

where the addition of the terms contained in the curly braces can change the sign of the walker weights as they accumulate. With time the sign of the weights becomes random which is a difficulty analogous to the fermion sign problem discussed in Section 4.3.3. The simplest solution is to use the 'localisation approximation' and ignore the terms in the curly braces. In this case the weight accumulation is only affected via the additional pseudopotential terms in the local energy. A consequence of this procedure is that the DMC energy is no longer variational, as the lowest eigenvalue of the approximate Hamiltonian, E_0^{pp}, does not give a strict upper bound for E_0. However, as Ψ_{T} tends to Ψ_0^{FN}, E_0^{pp} tends to E_0^{FN} quadratically[124] and the resultant error is small.

It is possible to restore the DMC variational principle by introducing T-moves[125] into the walker dynamics, which are dependent upon the short ranged, non-local component of a pseudopotential. In the near vicinity of a nucleus, T-moves allow walkers to transition between the points of the selected pseudopotential integration grid. The relative probability of a tran-

sition is roughly proportional to the product of the ratios of the trial wave function at the proposed and original points, $\frac{\Psi_{\mathrm{T}}(\bar{\boldsymbol{r}}')}{\Psi_{\mathrm{T}}(\bar{\boldsymbol{r}})}$, and the values of the non-local potential V_{nl}. This procedure also acts to stabilise DMC calculations as the T-moves keep walkers away from regions close to the nodal surface of $\Psi_{\mathrm{T}}(\bar{\boldsymbol{r}})$ where, in addition to the drift velocity and local energy, terms in equation (4.43) can cause large fluctuations in the walker weights.

Chapter 5

Differentiating stochastic functions

5.1 Introduction

This chapter presents the theoretical foundations of the differentiation schemes developed in this thesis. The first section introduces some notation to help clarify what is meant by a 'stochastic function', and distinguish between results obtained from stochastic functions of infinite length and expectation values of results from functions of a finite length. Such a distinction becomes important later on during the analysis of a straightforward differentiation of a stochastic algorithm. The following section provides a mathematical illustration of why finite difference differentiation methods perform poorly in the presence of stochastic error, and shows why, for Monte Carlo results, the error in an ideal finite difference calculation converges very slowly. Section 5.4 contains the development of a curve fitting method specifically targeted at

reproducing high accuracy gradients of stochastic functions. This method is essential for validating high precision stochastic derivatives calculated using other techniques, where the robust resolution of small differences in derivatives is required to assess accuracy. The final section focusses on the differentiation of a stochastic algorithm directly, showing why a straightforward approach yields derivatives which are incompatible with expectation values of stochastic results. Here a new differentiation strategy is derived which produces exact derivatives of algorithms containing probabilistic decisions. This approach is applicable to any kind of stochastic function, and underpins the differentiation of the DMC algorithm presented in Chapter 8.

5.2 Stochastic functions

Computers are purely deterministic machines. Stochastic or Monte Carlo algorithms actually refer to functions which evolve mechanically when supplied with a given set of random numbers, $\{\eta\}$. These numbers are typically obtained from a deterministic pseudorandom number generator where the initial seed value forms part of the input to a program. It is possible to collect random numbers which are completely unpredictable using physical sources of natural entropy, like cosmic background radiation or radioactive decay, but the rate at which these are generated is usually too slow to use in stochastic algorithms.

The exact result of an N decision stochastic function $f(x)$ can be described using

$$y_N(x) \;=\; f(x,\; \{\eta\}_N)\;, \tag{5.1}$$

which, in Monte Carlo methods, is an approximation to the $N \to \infty$ limit where the stochastic error vanishes and the result becomes independent of $\{\eta\}$,

$$\begin{aligned} y_{N\to\infty}(x) &= y_N(x,\ \{\eta\}_N) \ +\ \mathcal{O}(\frac{1}{\sqrt{N}}) \\ &= f(x,\ \{\eta\}_{N\to\infty})\ . \end{aligned} \tag{5.2}$$

To obtain the exact result $y_{N\to\infty}$ the computer program must run for an infinite amount of time over $\{\eta\}_{N\to\infty}$. However, computations can only be ever be performed to machine precision and this provides an effective upper bound on the computational effort required. In principle it is possible to reach this limit, but, as the stochastic errors of Monte Carlo algorithms decay in proportion to $\frac{1}{\sqrt{N}}$, even spectacularly long runs cannot hope to get close to the exact answer. This causes significant problems when evaluating the derivatives of stochastic quantities, where the correct derivative of a Monte Carlo result is given by

$$\frac{dy_{N\to\infty}(x)}{dx} \ =\ \frac{df(x,\ \{\eta\}_{N\to\infty})}{dx}\ , \tag{5.3}$$

obtained from taking the derivative of an infinitely long calculation. Given that it is only ever possible to perform finite calculations, the evaluation of the derivatives of stochastic functions must contend with stochastic error.

5.3 Finite differences

The finite difference method provides estimates of derivatives from

$$\frac{df(x)}{dx} \simeq \frac{\Delta f}{\Delta x} = \frac{f(x+\Delta x) - f(x)}{\Delta x} , \tag{5.4}$$

where Δx is the finite difference step size. In principle, for non-stochastic functions, the exact derivative of $f(x)$ is recovered as the step size tends to zero. However, in practice, very small values of Δx lead to the subtraction of two very similar values in the numerator of equation (5.4). As numbers stored in a computer have a limited precision, the subtraction of two similar values can cause catastrophic cancellation, where the much smaller result has far fewer significant figures than the larger numbers. Selecting a step size that is too large results in a truncation error, where the curvature of $f(x)$ over the range of Δx contributes significantly to the estimated derivative. Taking the Taylor expansion of $f(x + \Delta x)$,

$$f(x+\Delta x) = f(x) + \frac{df}{dx}\Delta x + \frac{1}{2}\frac{d^2 f}{dx^2}(\Delta x)^2 + \mathcal{O}((\Delta x)^3) , \tag{5.5}$$

and substituting it into equation (5.4), gives

$$\frac{\Delta f}{\Delta x} = \frac{df}{dx} + \frac{1}{2}\frac{d^2 f}{dx^2}\Delta x + \mathcal{O}((\Delta x)^2) , \tag{5.6}$$

where, for very small Δx, the extra terms in the expansion of $\frac{\Delta f}{\Delta x}$ are insignificant and the approximation to the derivative is very accurate. There is clearly an optimal step size which minimises the sum of the two sources of

error, but it cannot be calculated *a priori* without knowing the value of $\frac{d^2f}{dx^2}$, which is itself generally much harder to obtain than $\frac{df}{dx}$. However, for the vast majority of non-stochastic functions, the finite difference method provides a near automatic way of extracting usefully accurate derivatives from complicated computations.

When applied to stochastic functions the finite difference method performs far less well. Here two completely separate evaluations of $f(x,\ \{\eta\}_N)$ are used to obtain an estimate of a derivative,

$$\frac{\Delta f}{\Delta x} \;=\; \frac{[f(x+\Delta x)\pm\sigma]-[f(x)\pm\sigma]}{\Delta x}\ , \tag{5.7}$$

where σ represents the Monte Carlo error in an evaluation of $f(x)$. As these errors are independent, the resultant error in the difference of two evaluations of $f(x)$ is given by adding the errors in quadrature,

$$\frac{\Delta f}{\Delta x} \;=\; \frac{f(x+\Delta x)-f(x)}{\Delta x} \;\pm\; \frac{\sqrt{2}\sigma}{\Delta x}\ . \tag{5.8}$$

Using the Taylor expansion of equation (5.5) gives

$$\frac{\Delta f}{\Delta x} \;\approx\; \frac{df}{dx} \;+\; \left[\frac{1}{2}\frac{d^2f}{dx^2}\Delta x \;\pm\; \frac{\sqrt{2}\sigma}{\Delta x}\right]\ , \tag{5.9}$$

which is the standard result for the error in a finite difference, with an extra component. Immediately this illustrates the difficulty in taking a finite difference in the presence of stochastic noise; a very large step size is required to reduce the Monte Carlo error to an acceptable level, which inevitably produces a large truncation error. Selecting a step size than minimises the

quantity in the square brackets, which again requires knowledge of $\frac{d^2f}{dx^2}$, yields

$$\frac{\Delta f}{\Delta x} = \frac{df}{dx} \pm \sim 2^{\frac{3}{4}} \sqrt{\sigma \frac{d^2 f}{dx^2}} \, . \tag{5.10}$$

This expression shows that even optimal finite differences are far harder to evaluate than the underlying function in isolation; the error in a finite difference scales as the square root of the Monte Carlo error. Monte Carlo errors are proportional to $N_{\mathrm{s}}^{-\frac{1}{2}}$, where N_{s} is the number of samples, so the error in a finite difference is therefore proportional to $N_{\mathrm{s}}^{-\frac{1}{4}}$, and converges far more slowly. The same is true for a centred finite difference, where a similar analysis yields an error proportional to $N_{\mathrm{s}}^{-\frac{1}{3}}$.

5.4 Curve fitting

An alternative, and more computationally expensive, method of determining derivatives is to obtain an accurate representation of a stochastic function by fitting a curve to data. If the form of the underlying function is known then it is relatively straightforward to fit this to the corresponding data by minimising appropriately weighted residuals, where the weights are determined by the relative size of the error bars. If the form of the underlying function is not known then the data must be interpolated.

The interpolation of noisy data inevitably involves a trade off between minimising residuals and minimising the curvature of the interpolated function. Figure 5.1 shows the value and gradient of a cubic spline which interpolates exactly through all of a set of noisy data points using a procedure

based on the DIERCKX[126] routines. Whilst this may be an appropriate strategy for data without noise, the natural scatter of data containing error bars suggests that this yields a poor representation of the stochastic function. A more sensible fit can be achieved by allowing the interpolated function to have non-zero residuals and reducing the number of points that the function must pass through, which are known as knots. Smoothed spline fitting[126] distributes new knots evenly across the range of the data and progressively increases their number until the smoothing condition,

$$\sum_{i}^{N} \left(\frac{y[i] - f_{\text{smooth}}(x[i])}{\sigma[i]} \right)^2 \leq N \ , \tag{5.11}$$

holds, where i is the index of a datum with value $(x[i],\ y[i])$ and y-error $\sigma[i]$, $f_{\text{smooth}}(x)$ is the smoothed spline fit, and N is the number of data. This yields a function that is tolerably close to the data whilst keeping curvature to a minimum, as illustrated by Figure 5.1.

Whilst smoothed spline fitting works well for interpolating values, getting an accurate gradient is more difficult. Figure 5.2 shows that the gradient of fits to independent, equivalently precise stochastic data can vary significantly, despite having very similar fluctuations in value. In particular the data at the extremal values of the x range has a strong influence on the gradient as the fitting procedure is free to introduce curvature to get closer to these points. One method of increasing the accuracy of the fit is to increase the sampling density and precision of the data. This approach was used to produce Figure 5.3 where it is seen that there is an approximate agreement in the gradient of the fits in the central region of the x range. However, gradients obtained

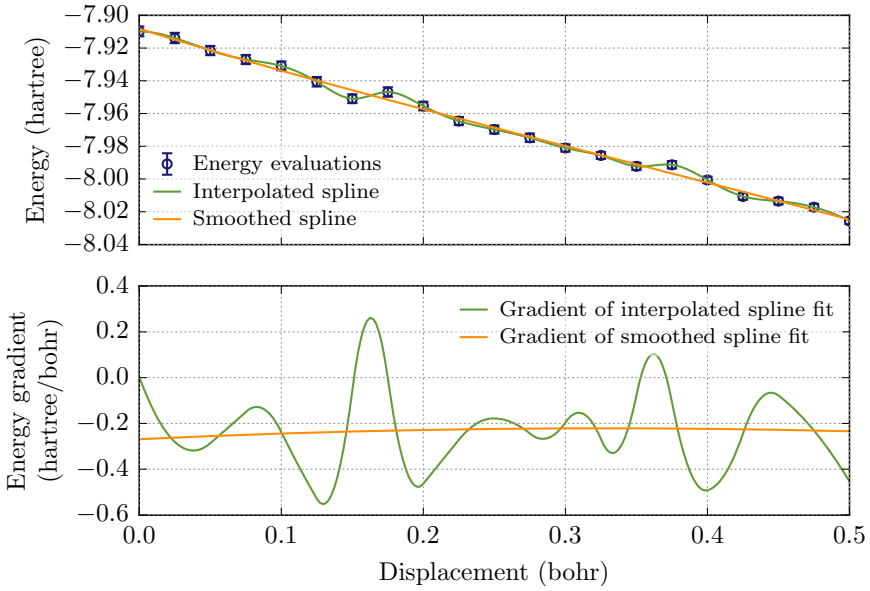


Figure 5.1: Shown in the top pane is the DMC total energy as a function of an arbitrary displacement of the carbon atom in a methane molecule. The bottom pane displays the gradient of the two functions in the top pane.

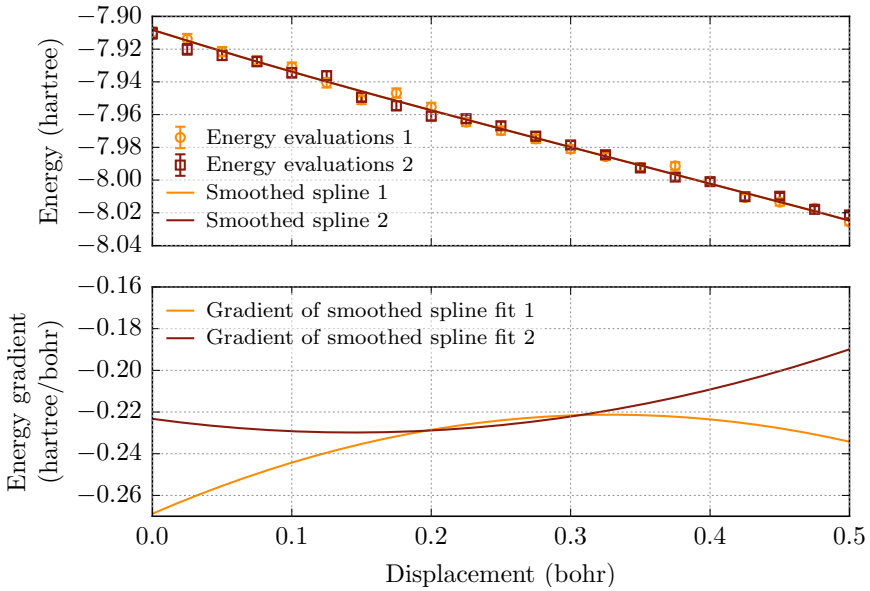


Figure 5.2: The top pane compares smoothed spline fits to the data from Figure 5.1 and data obtained from an independent set of DMC calculations. Shown in the bottom pane are the gradients of each fitted curve.

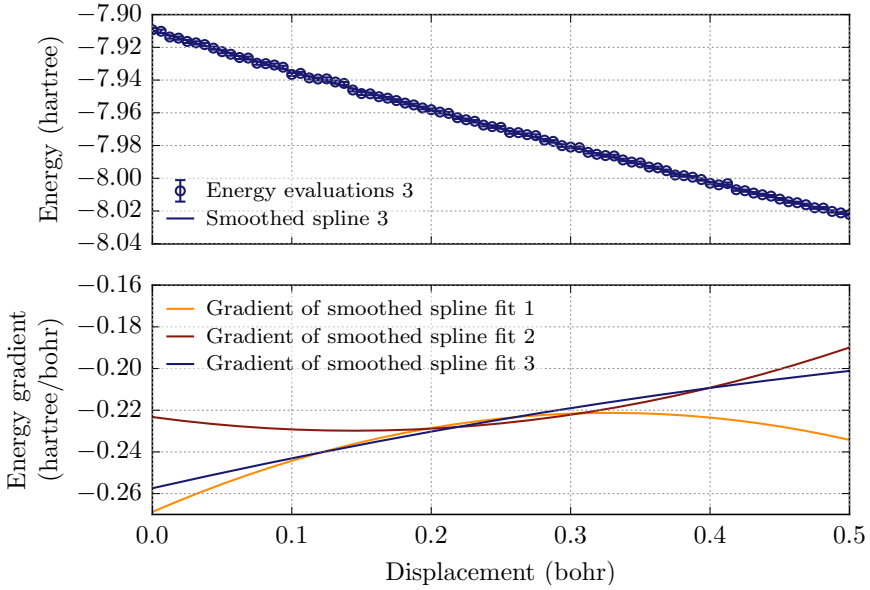


Figure 5.3: Compared to Figure 5.2 the stochastic total energy function has been sampled 4 times more frequently and with double the precision. The bottom pane compares the corresponding gradient to that of the fits from Figure 5.2.

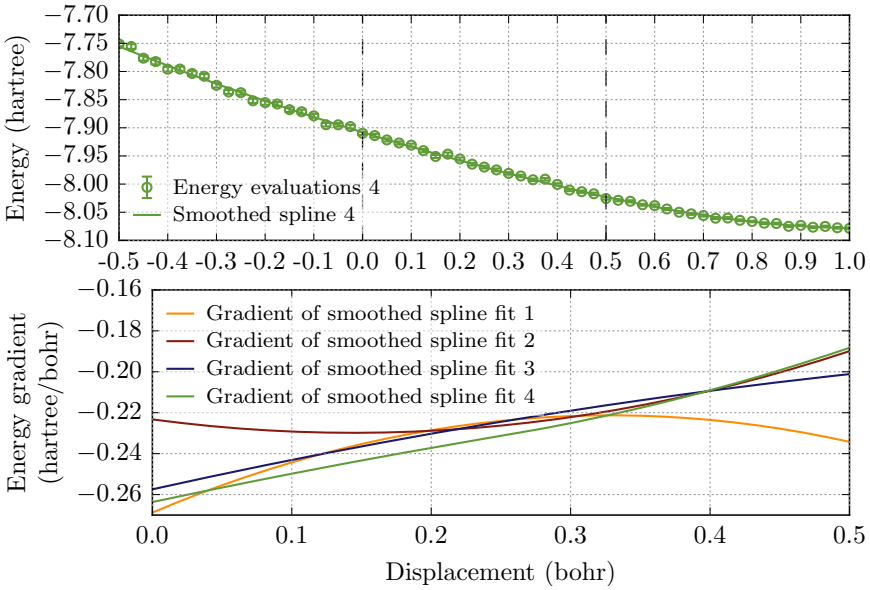


Figure 5.4: The top pane shows data gathered from a larger range of displacements than the previous figures in this section, where vertical dotted lines mark the extent of the previous domain. Shown in the lower pane are the gradients of all fits performed so far.

from such a procedure should still be treated with caution as they remain susceptible to edge effects. A more reliable technique is to sample the function over an extended range that better captures its curvature, as shown in Figure 5.4. Here the gradient in the range of interest has been iteratively improved by extending the range of the data and increasing its precision until the resultant effect on the gradient is sufficiently small. This provides a relatively robust method for determining gradients of stochastic functions but, as the required range and precision is not known *a priori*, the computational effort involved is very large.

5.5 Differentiating stochastic algorithms

A natural approach to differentiating a stochastic quantity is to differentiate the algorithm that generates it,

$$f'(x) \;=\; \frac{df(x)}{dx} \;, \tag{5.12}$$

and evaluate $f'(x)$ using $\{\eta\}_N$,

$$y'_N(x,\; \{\eta\}_N) \;=\; f'(x,\; \{\eta\}_N) \;. \tag{5.13}$$

Unfortunately, a straightforward differentiation of a stochastic algorithm produces derivatives which are incompatible with results obtained from an infinite calculation,

$$y'_{N\to\infty}(x) \;\neq\; \frac{dy_{N\to\infty}(x)}{dx} \;. \tag{5.14}$$

The root of the problem is that the processes of differentiating and letting N tend to infinity do not commute, so the expectation value of the derivative of a finite length algorithm gives the wrong answer,

$$\left\langle \frac{dy(x)}{dx} \right\rangle \neq \frac{d\langle y(x)\rangle}{dx} . \tag{5.15}$$

A simple example illustrates this point. Here the body of $f(x)$ will take the form

```
eta = rand()
if (p(x) > eta) {
  y = f1(x);
} else {
  y = f2(x);
}
```

where

$$p(x) = \frac{\sin(x)+1}{2} \tag{5.16}$$

$$f_1(x) = x^2 \tag{5.17}$$

$$f_2(x) = 5\cos(x) \tag{5.18}$$

and `rand()` returns a uniform random number in the interval $[0,\ 1)$. This function stochastically selects either $f_1(x)$ or $f_2(x)$ to evaluate, with probabilities of $p(x)$ or $[1-p(x)]$ respectively. The $\{\eta\}_{N\to\infty}$ limit is obtained from the expected value of y,

$$\langle y(x)\rangle = [\,p(x)\,]f_1(x) + [\,1-p(x)\,]f_2(x) , \tag{5.19}$$

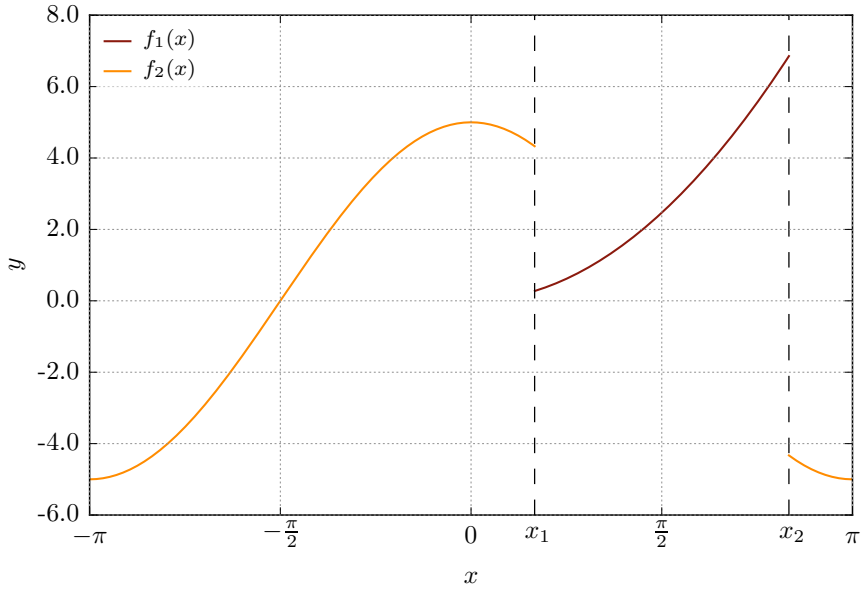


Figure 5.5: The output from the simple branching program when the random number η has a value of 0.75.

giving the derivative as

$$\frac{d\langle y(x)\rangle}{dx} = [\,p(x)\,]\frac{df_1(x)}{dx} + [\,1{-}p(x)\,]\frac{df_2(x)}{dx} + \frac{dp(x)}{dx}[f_1(x) - f_2(x)]\,. \quad (5.20)$$

The stochastic algorithm $f(x)$ can be differentiated by inspecting the effect of an infinitesimal change in x,

$$\frac{df(x)}{dx} = \lim_{\Delta x \to 0}\left\{\frac{f(x+\Delta x) - f(x)}{\Delta x}\right\}\,, \quad (5.21)$$

and using the same random number η for the evaluation of $f(x+\Delta x)$ and $f(x)$. Crucially, the decision made by the `if` statement is identical in both evaluations; an infinitesimal change in x cannot cause a jump to the other

logical branch. Considering the equality $p(x) = \eta$, the solutions are given by

$$\begin{aligned} x_1 &= -\arcsin(1-2\eta) + 2\pi n \\ x_2 &= \arcsin(1-2\eta) + \pi + 2\pi n \ , \end{aligned} \tag{5.22}$$

and between these points either $f_1(x)$ or $f_2(x)$ is evaluated. For $\eta = 0.75$ the function $f(x)$ between $-\pi$ and π is shown in Figure 5.5, where $x_1 = \frac{\pi}{6}$ and $x_2 = \frac{5\pi}{6}$. When x is between x_1 and x_2, the function returns $f_1(x)$, otherwise it returns $f_2(x)$. The variation of $p(x + \Delta x)$ places x_1 and x_2 at slightly different locations, but as $\Delta x \to 0$, x and $x + \Delta x$ always sample the same region of the piecewise function. Thus, for any value of η, equation (5.21) yields either

$$\frac{df_1(x)}{dx} = 2x \tag{5.23}$$

or

$$\frac{df_2(x)}{dx} = -5\sin(x) \ . \tag{5.24}$$

The differentiated function is then given by

```
eta = rand()
if (p(x) > eta) {
  dydx = d1(x);
} else {
  dydx = d2(x);
}
```

where

$$d_1(x) \;=\; \frac{df_1(x)}{dx} \tag{5.25}$$

$$d_2(x) \;=\; \frac{df_2(x)}{dx}\;. \tag{5.26}$$

Taking the $\{\eta\}_{N\to\infty}$ limit by looking at the expected value of $\frac{dy(x)}{dx}$ produces

$$\left\langle \frac{dy(x)}{dx} \right\rangle \;=\; [\,p(x)\,]\frac{df_1(x)}{dx} \;+\; [\,1-p(x)\,]\frac{df_2(x)}{dx} \tag{5.27}$$

$$\neq\; \frac{d\langle y(x)\rangle}{dx}\;, \tag{5.28}$$

which misses the terms involving the derivative of the probability shown in equation (5.20). These missing terms arise from an infinitesimal change in a probability causing an infinitesimal number of stochastic decisions to give a different result. Given that it is only ever possible to work with finite length stochastic algorithms, this kind of information is impossible to access directly. This can also be seen by expressing the expectation value of the algorithm using integrals over the distribution of random numbers,

$$\langle y(x)\rangle \;=\; f_1(x)\int_0^{p(x)} d\eta \;+\; f_2(x)\int_{p(x)}^1 d\eta\;, \tag{5.29}$$

where, for a finite sampling of η, infinitesimal changes to the limits of the integration from Δx have no effect on the result.

A solution is to engineer an explicit differential dependence of y on the probabilities. Consider the execution of the example program at $x+\Delta x$. Ideally the clauses of the `if` statement would be selected with probabilities

based on $p(x + \Delta x)$, but on any particular evaluation of $f(x)$ the decisions are constrained to use $p(x)$. This can be corrected by weighting results by a factor

$$\omega \;=\; \frac{P(x+\Delta x)}{P(x)}\;, \tag{5.30}$$

where $P(x)$ is the probability of selecting that result. For $f(x)$ the weights have the form

$$\omega \;=\; \begin{cases} \dfrac{p(x+\Delta x)}{p(x)} & \text{for the } \texttt{if} \text{ clause} \\ \dfrac{1-p(x+\Delta x)}{1-p(x)} & \text{for the } \texttt{else} \text{ clause} \end{cases}\;. \tag{5.31}$$

As Δx tends to zero, ω tends to unity, leaving the values produced by $f(x)$ intact. However, a finite contribution to the derivative remains. Applying equation (5.21) to $\omega f_1(x)$ or $\omega f_2(x)$,

$$\frac{d\,[\,\omega f_i(x)\,]}{dx} \;=\; \lim_{\Delta x \to 0} \left\{ \frac{\frac{P(x+\Delta x)}{P(x)} f_i(x+\Delta x) - f_i(x)}{\Delta x} \right\}\;, \tag{5.32}$$

and using Taylor expansions of $P(x + \Delta x)$ and $f_i(x + \Delta x)$ from equation (5.5), gives

$$\frac{d\,[\,\omega f_i(x)\,]}{dx} \;=\; \frac{df_i(x)}{dx} \;+\; \frac{1}{P(x)}\frac{dP(x)}{dx} f_i(x)\;. \tag{5.33}$$

These weights are incorporated into the differentiated program by modifying

equations (5.25) and (5.26),

$$d_1(x) \;=\; \frac{df_1(x)}{dx} \;+\; \frac{1}{p(x)}\frac{dp(x)}{dx}f_1(x) \tag{5.34}$$

$$d_2(x) \;=\; \frac{df_2(x)}{dx} \;-\; \frac{1}{1-p(x)}\frac{dp(x)}{dx}f_2(x)\;, \tag{5.35}$$

and the differentiated algorithm calculates a new quantity $\left.\frac{dy(x)}{dx}\right|_\omega$. Inspecting its expectation value,

$$\begin{aligned}\left\langle \left.\frac{dy(x)}{dx}\right|_\omega \right\rangle \;=\; [\,p(x)\,]\frac{df_1(x)}{dx} \;+\; [\,1-p(x)\,]\frac{df_2(x)}{dx} \\ +\; \frac{dp(x)}{dx}[f_1(x) \;-\; f_2(x)]\end{aligned} \tag{5.36}$$

shows that the differential weighting produces the correct derivative in the $\{\eta\}_{N\to\infty}$ limit,

$$\left\langle \left.\frac{dy(x)}{dx}\right|_\omega \right\rangle \;=\; \frac{d\langle y(x)\rangle}{dx}\;. \tag{5.37}$$

This approach can be generalised to accommodate algorithms containing multiple branching events. Here, each possible result of an N decision program, $r(x)$, can be uniquely enumerated by assigning labels to each of the possible `if` statement branches. In this case the $P(x)$ in equation (5.30) is replaced by a product of all the probabilities of the decisions made on a particular evaluation of a stochastic function,

$$\omega \;=\; \frac{\prod_{i_{\text{chosen}}} P_i(x+\Delta x)}{\prod_{i_{\text{chosen}}} P_i(x)}\;. \tag{5.38}$$

Differentiating a weighted result of the algorithm, $\omega r_{\{i_{\text{chosen}}\}}$, using the same

approach as in equation (5.32), yields

$$\begin{aligned} \frac{d\left[\,\omega r_{\{i_{\text{chosen}}\}}\,\right]}{dx} &= \frac{dr_{\{i_{\text{chosen}}\}}}{dx} + \frac{r_{\{i_{\text{chosen}}\}}}{\prod_{i_{\text{chosen}}} P_i} \frac{d\left[\,\prod_{i_{\text{chosen}}} P_i\,\right]}{dx} \\ &= \frac{dr_{\{i_{\text{chosen}}\}}}{dx} + r_{\{i_{\text{chosen}}\}} \sum_{i_{\text{chosen}}} \frac{1}{P_i} \frac{dP_i}{dx} \ . \end{aligned} \tag{5.39}$$

Chapter 6

Existing approaches to differentiating quantum Monte Carlo total energies

6.1 Introduction

Despite great success in computing total energies, and a number of recent advances in the area, the calculation of total energy derivatives within QMC calculations remains relatively difficult. This chapter gives a brief overview of existing differentiation strategies.

6.2 The Hellmann–Feynman theorem

Total energy derivatives are typically used in atomic force calculations which, for the majority of electronic structure methods, proceed via the Hellmann–

Feynman theorem (HFT).[127,128] This theorem states that, when the wave function is an energy eigenstate, the gradient of the energy is the expectation value of the gradient of the Hamiltonian. In QMC methods the wave function is not, in general, an eigenstate of the Hamiltonian and the resultant forces are given by the expectation value of the gradient of the Hamiltonian plus additional Pulay-like terms[21,22,129] that involve gradients of the probability distribution. The fixed-node DMC algorithm generates expectation values for operators that commute with the Hamiltonian using the 'mixed' probability distribution $\Psi_{\mathrm{T}}(\bar{\boldsymbol{r}})\Psi_0^{\mathrm{FN}}(\bar{\boldsymbol{r}})$ from equation (4.23). As Ψ_0^{FN} is only sampled stochastically, rather than explicitly constructed, its derivative cannot be calculated in a straightforward manner. The common approach is to approximate the Pulay corrections involving Ψ_0^{FN} by substituting Ψ_{T} in its place.[130,131] It is also possible to generate the 'pure' DMC probability distribution, $\Psi_0^{\mathrm{FN}}\Psi_0^{\mathrm{FN}}$, using either the future walking[132] or reptation Monte Carlo[133] methods. As Ψ_0^{FN} is a better representation of an energy eigenfunction than Ψ_{T}, the Pulay terms are smaller and approximations to the derivatives of Ψ_0^{FN} are less important.

An additional complication in the evaluation of forces using QMC methods is the infinite variance of the HFT estimator when doing all-electron calculations with bare Coulomb potentials. One solution is to use an improved or renormalised force estimator obtained through the addition of a term to the HFT force that has a mean value of zero but greatly suppresses the statistical fluctuations.[78,134–136] Another approach is to selectively remove the small part of the electron density that gives rise to the infinite variance.[137] This procedure can also be applied to the bare Coulomb potential by absorb-

ing the chemically inert core electrons into pseudopotentials, as described in Section 4.5. However, the calculation of forces using non-local pseudopotentials requires the evaluation of the gradient of the non-local pseudopotential operator,[23,24] necessitating an additional localisation approximation where Ψ_0^{FN} is substituted for Ψ_{T} in a manner similar to the HFT Pulay corrections of the mixed distribution.

6.3 Correlated sampling

Alternatively, atomic forces can be determined from the energy differences between slightly displaced nuclear configurations using a finite difference approach. Whilst other electronic structure calculations can rely on a smoothly varying energy for small changes in nuclear position, the statistical noise inherent in QMC methods make the resolution of small energy differences difficult. Although it is not possible to eliminate random errors entirely, the use of correlated sampling, which employs suitably reweighted electronic configurations generated from a single reference QMC run, reduces the error in the relative energies of two related systems, with the error vanishing in the limit of the two systems becoming identical. This approach has long been established in VMC but the branching, drift-diffusion dynamics of DMC walkers means that the application of correlated sampling in DMC is less straightforward. Initial attempts at a DMC correlated sampling algorithm used the original wave function to guide walkers in a displaced 'differential' system,[25,26] leading to instabilities when differential walkers were free to drift-diffuse into exposed nodal regions. This effect can be mitigated, but only

entirely removed in certain cases, via more complex schemes which utilise the space-warp coordinate transformation (SWCT).[27–30] Here the paths of the walkers generated from the original wave function experience a smooth semi-rigid distortion when close to atomic centres, commensurate with the displacements of the nuclei. The differential wave function is obtained from a corresponding shift in its nuclear positions, and the weights obtained by the differential walkers are explicitly correlated to the originals through a scheme where they differ only over a limited history of the most recent generations.

A different method of correlating DMC dynamics relies on the convergence of similar stochastic trajectories sharing a common noise. Previous work[37] has shown that when the influence of common noise exceeds a certain frequency, initially distinct trajectories are driven together and remain close. By imposing the same, purely random, diffusive component of the walker dynamics when sampling two similar systems, the sampling points become correlated and the relative error is reduced.[31] The application of this method to branching, fixed-node DMC is presented in Chapter 9.

A drawback common to all correlated sampling approaches is that each correlated calculation generates only a single derivative. Thus, the computational effort required to determine all force components in a system scales in proportion to the number of nuclei. In most cases a correlated calculation will be comparable in difficulty to a regular total energy evaluation, resulting in a total cost of $\sim\ 3N_n + 1$ for all force components where N_n is the number of nuclei. This poor scaling makes correlated approaches ill suited to the treatment of larger systems.

6.4 Direct differentiation

Some progress has been made towards the differentiation of the DMC total energy by focussing of the DMC algorithm itself. Early work[138] parametrised the effect of a small perturbation to the Hamiltonian, and incorporated this information into the weights obtained by walkers in the perturbed system. Differentiating the expression for the perturbed DMC energy with respect to the chosen parameter, and evaluating the result at a parameter value of zero, gives quantities that can be averaged over the course of the calculation to calculate derivatives of the unperturbed energy. This procedure was coined 'Hellmann–Feynman sampling' (due to the similarity of the HFT) by later work[139] that employed a comparable method to evaluate expectation values of operators that do not commute with the Hamiltonian. However, the primary drawback of this approach is that it neglects the effect of a perturbation on the walker dynamics and thus excludes energy derivatives with respect to parameters that affect the nodal surface. A generalisation of the method,[32] relying on appropriately renormalised estimators, includes additional information from the derivative of the Langevin Green's function of equation (4.26). The variation of the nodal surface is still not captured completely, but the errors in the approximate forces on a BH molecule are shown to be small.

6.5 Outlook

All existing methods of calculating DMC forces scale badly with system size. As the complexity of the system increases, the errors in the forces become much harder to converge compared to the total energy, limiting previous applications to non-model systems containing 8 or fewer electrons. The correlated or direct DMC force methods mentioned in this section omit the branching of walkers, reducing the stability of a DMC calculation, and face a computational bottleneck when confronted with many degrees of freedom.

In VMC it is possible to differentiate the total energy analytically using correlated sampling. The most efficient strategy for computing VMC forces[40] combines a differential version of the SWCT with AD, allowing the efficient evaluation of a force estimator. This approach was able to treat a system containing 256 hydrogen atoms,[140] but its extension to DMC is not straightforward.

The remaining chapters focus on two new methods for calculating total energy derivatives from DMC using the techniques of AD and correlated sampling.

Chapter 7

Algorithmic differentiation

7.1 Introduction

Finite difference methods have been the *de facto* numerical differentiation tool since the earliest iterations of modern computers. Their great appeal is in their simplicity; a derivative can be estimated from just two evaluations of the same function, so no extra programming effort is required to determine the gradients of calculated quantities. However, the finite difference approach is not without its drawbacks. Section 5.3 details the inevitable compromise in accuracy when selecting a finite difference step size and explains the poor performance in the face of stochastic noise. There is also a rather high computational cost incurred when evaluating many derivatives of a single quantity; each component of a calculated gradient requires an additional execution of the underlying function. This makes the finite difference scheme ill suited to algorithms with many interesting inputs.

The AD method[38,39] is a relatively new, numerical approach to calculat-

ing derivatives that is unencumbered by the deficiencies of finite differences. Using this technique it is possible to calculate every single derivative of a computed result, simultaneously, to full numerical precision, with a total computational cost comparable to obtaining the result itself. This chapter provides an overview of the operation of AD. Contained in the first two sections is a description of its two different forms and an outline of the mathematics underpinning its remarkable efficiency presented using a novel matrix formulation. The final section supplies information required for a practical implementation of AD and details the procedures used by automatic AD tools.

7.2 The forwards mode

The simplest and most accessible form of AD is called the 'forwards mode', and its basic operation can be illustrated using an outline of a simple computer program,

$$
\begin{aligned}
&[x_1,\ x_2,\ x_3]^{\mathrm{T}} \\
&\qquad \Downarrow \qquad z \ = \ f^1(x_2,\ x_3) \\
&[x_1,\ x_2,\ z]^{\mathrm{T}} \\
&\qquad \Downarrow \qquad y_2 \ = \ f^2(x_2, z) \\
&\quad [x_1,\ y_2]^{\mathrm{T}} \\
&\qquad \Downarrow \qquad y_1 \ = \ f^3(x_1) \\
&\quad [y_1,\ y_2]^{\mathrm{T}}
\end{aligned}
$$

(Program Outline 1)

where the left hand side describes a short function that takes an input vector $[x_1,\ x_2,\ x_3]^{\mathrm{T}}$ and, via the evaluation of the expressions f^k, generates the output $[y_1,\ y_2]^{\mathrm{T}}$. The terms contained within the square brackets represent the *state* of the algorithm, consisting of the set of *active* variables that have some subsequent influence on the calculation of the function output. Each step of the program, k, comprises a *state transformation*, where a new active variable is computed, changing the state vector from $V^{k-1} = [v_1^{k-1},\ v_2^{k-1}, \cdots,\ v_n^{k-1}]^{\mathrm{T}}$ to $V^k = [v_1^k,\ v_2^k, \cdots,\ v_m^k]^{\mathrm{T}}$.

The application of AD proceeds by constructing the differential expressions of each of the variable assignments in the original program, which, in themselves, constitute an algorithm that takes a differential input vector $[dx_1,\ dx_2,\ dx_3]^{\mathrm{T}}$ and calculates $[dy_1,\ dy_2]^{\mathrm{T}}$.

$[x_1,\ x_2,\ x_3]^{\mathrm{T}}$	$[dx_1,\ dx_2,\ dx_3]^{\mathrm{T}}$
$\Downarrow \quad z = f^1(x_2,\ x_3)$	$\Downarrow \quad dz = \left.\frac{\partial f^1}{\partial x_2}\right\|_{x_2,\,x_3} dx_2 + \left.\frac{\partial f^1}{\partial x_3}\right\|_{x_2,\,x_3} dx_3$
$[x_1,\ x_2,\ z]^{\mathrm{T}}$	$[dx_1,\ dx_2,\ dz]^{\mathrm{T}}$
$\Downarrow \quad y_2 = f^2(x_2, z)$	$\Downarrow \quad dy_2 = \left.\frac{\partial f^2}{\partial x_2}\right\|_{x_2,\,z} dx_2 + \left.\frac{\partial f^2}{\partial z}\right\|_{x_2,\,z} dz$
$[x_1,\ y_2]^{\mathrm{T}}$	$[dx_1,\ dy_2]^{\mathrm{T}}$
$\Downarrow \quad y_1 = f^3(x_1)$	$\Downarrow \quad dy_1 = \left.\frac{\partial f^3}{\partial x_1}\right\|_{x_1} dx_1$
$[y_1,\ y_2]^{\mathrm{T}}$	$[dy_1,\ dy_2]^{\mathrm{T}}$,

(Program Outline 2)

As in the case of the original program, the evaluation of each differential

expression transforms the differential state vector from

$$dV^{k-1} = [dv_1^{k-1},\ dv_2^{k-1}, \cdots,\ dv_n^{k-1}]^{\mathrm{T}} \tag{7.1}$$

to

$$dV^{k} = [dv_1^{k},\ dv_2^{k}, \cdots,\ dv_m^{k}]^{\mathrm{T}}\ . \tag{7.2}$$

At each step of the differential program, a newly computed differential variable is assigned the value

$$dv_{\mathrm{new}}^{k} \ = \ \sum_{j=1}^{n} \left.\frac{\partial f^k}{\partial v_j^{k-1}}\right|_{V^{k-1}} dv_j^{k-1}\ , \tag{7.3}$$

with the persisting members of dV^k remaining unchanged. This transformation can be put into a matrix form, $dV^k = M^k dV^{k-1}$, where M^k contains ones down the diagonal to preserve the differentials of active variables, a single row is the Jacobian of the scalar function f^k used to calculate the new differential variable, and the remainder is zero. For the step $(k-1) \to k$, where the number of active variables decreases from n to m, this transformation can be written as

$$\begin{bmatrix} dv_1^k \\ dv_2^k \\ \vdots \\ dv_{m-1}^k \\ dv_{\mathrm{new}}^k \end{bmatrix} = \begin{pmatrix} 1 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ 0 & 1 & & \vdots & \vdots & & \vdots \\ \vdots & & \ddots & \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & 1 & 0 & \cdots & 0 \\ \frac{\partial f^k}{\partial v_1^{k-1}} & \frac{\partial f^k}{\partial v_2^{k-1}} & \cdots & \frac{\partial f^k}{\partial v_{m-1}^{k-1}} & \frac{\partial f^k}{\partial v_m^{k-1}} & \cdots & \frac{\partial f^k}{\partial v_n^{k-1}} \end{pmatrix} \begin{bmatrix} dv_1^{k-1} \\ dv_2^{k-1} \\ \vdots \\ dv_{m-1}^{k-1} \\ dv_m^{k-1} \\ \vdots \\ dv_n^{k-1} \end{bmatrix}\ . \tag{7.4}$$

The operation of the differential expressions can now be constructed as a

series of matrix vector products,

$$dV^k \;=\; M^k(M^{k-1}\dots(M^2(M^1dV^0))), \tag{7.5}$$

where, for the expressions in Program Outline 2, these matrices have the form

$$\begin{bmatrix} dy_1 \\ dy_2 \end{bmatrix} = \begin{pmatrix} \frac{\partial f^3}{\partial x_1}\Big|_{x_1} & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & \frac{\partial f^2}{\partial x_2}\Big|_{x_2,\,z} & \frac{\partial f^2}{\partial z}\Big|_{x_2,\,z} \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & \frac{\partial f^1}{\partial x_2}\Big|_{x_2,\,x_3} & \frac{\partial f^1}{\partial x_3}\Big|_{x_2,\,x_3} \end{pmatrix} \begin{bmatrix} dx_1 \\ dx_2 \\ dx_3 \end{bmatrix} . \tag{7.6}$$

Upon completion of an N step program, the combined action of all the matrices produces the Jacobian, J, of the program as a whole,

$$dV^N \;=\; JdV^0 \;=\; \begin{bmatrix} dv_1^N \\ dv_2^N \\ \vdots \\ dv_m^N \end{bmatrix} \;=\; \begin{pmatrix} \frac{\partial v_1^N}{\partial v_1^0} & \frac{\partial v_1^N}{\partial v_2^0} & \cdots & \cdots & \frac{\partial v_1^N}{\partial v_n^0} \\ \frac{\partial v_2^N}{\partial v_1^0} & \frac{\partial v_2^N}{\partial v_2^0} & \cdots & \cdots & \frac{\partial v_2^N}{\partial v_n^0} \\ \vdots & \vdots & & & \vdots \\ \frac{\partial v_m^N}{\partial v_1^0} & \frac{\partial v_m^N}{\partial v_2^0} & \cdots & \cdots & \frac{\partial v_m^N}{\partial v_n^0} \end{pmatrix} \begin{bmatrix} dv_1^0 \\ dv_2^0 \\ \vdots \\ \vdots \\ dv_n^0 \end{bmatrix} . \tag{7.7}$$

To obtain derivatives from the differential program, the AD expressions must be seeded with a differential input vector which selects the input variable of interest. By selecting an input of the form $dv_i^0 = \delta_{ij}$, where δ is the Kronecker delta, each differential variable dq calculated during the execution of the program, where q is any variable in the original program, takes the numerical value of $\frac{dq}{dv_j^0}$. The overall action of the differential seed is to select out the jth column of the Jacobian in equation (7.7), returning the derivatives

of all the output variables with respect to the chosen input.

As the expressions in the differential program may depend upon the variables calculated in the original program, both must be executed concurrently. Some bounds may be placed on the complexity and memory requirements of propagating the forwards mode AD derivatives alongside the function evaluation.[141,142] By breaking the derivative code down into its simplest possible operations and counting the number of extra memory fetches and stores, additions and subtractions, multiplications, and non-linear operations required, it can be shown that the computational demand will not exceed 2.5 times that of evaluating the original function alone. The memory requirement is similarly bounded at twice that of the original function evaluation, as each calculated variable cannot have more than one corresponding differential counterpart. For differentiating single program outputs this makes the cost of the most demanding implementation of forwards mode AD comparable to that of calculating derivatives obtained from a finite difference approach. The primary advantage of forwards mode AD over finite differencing is that the user is not required to select a step size *a priori* and the derivatives obtained from AD will retain the same level of numerical accuracy as the evaluation of the underlying function.

Forwards mode AD is particularly useful when there are a large number of program outputs to be differentiated; in a single execution of the AD algorithm all output variables are simultaneously differentiated with respect to a single input. Given that most scientific codes produce just a few outputs from many inputs, the efficiency of forwards mode AD suffers the same computational bottleneck as finite differencing methods.

7.3 The reverse mode

Whilst a suitably seeded execution of the forwards mode AD algorithm returns a column of the Jacobian matrix, the reverse mode enables the extraction of a row via the *pre*-multiplication of a reverse differential seed vector bV^N,

$$bV^{N\mathrm{T}}J \;=\; \begin{bmatrix} bv_1^N & bv_2^N & \cdots & bv_m^N \end{bmatrix} \begin{pmatrix} \frac{\partial v_1^N}{\partial v_1^0} & \frac{\partial v_1^N}{\partial v_2^0} & \cdots & \frac{\partial v_1^N}{\partial v_n^0} \\ \frac{\partial v_2^N}{\partial v_1^0} & \frac{\partial v_2^N}{\partial v_2^0} & \cdots & \frac{\partial v_2^N}{\partial v_n^0} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial v_m^N}{\partial v_1^0} & \frac{\partial v_m^N}{\partial v_2^0} & \cdots & \frac{\partial v_m^N}{\partial v_n^0} \end{pmatrix} \;=\; \begin{bmatrix} bv_1^0 & bv_2^0 & \cdots & bv_n^0 \end{bmatrix} \;=\; bV^{0\mathrm{T}} \;, \tag{7.8}$$

producing the reverse differential output bV^0. In a similar approach to the forwards case, a reverse differential seed vector of $bV_i^N = \delta_{ij}$ returns in bV^0 all the derivatives of the jth output with respect to all of the inputs. To perform this pre-multiplication, the series of matrix transformations that constitute the Jacobian must be traversed backwards via the reverse differential state vectors bV^k,

$$(((bV^{N\mathrm{T}}M^N)M^{N-1})\ldots M^{k+1}) \;=\; bV^{k\mathrm{T}} \;, \tag{7.9}$$

from bV^N to bV^0. Whilst the content of each of the matrices M^k is invariant between the forwards and reverse modes,

$$\begin{bmatrix} by_1 & by_2 \end{bmatrix} \begin{pmatrix} \left.\frac{\partial f^3}{\partial x_1}\right|_{x_1} & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & \left.\frac{\partial f^2}{\partial x_2}\right|_{x_2,\,z} & \left.\frac{\partial f^2}{\partial z}\right|_{x_2,\,z} \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & \left.\frac{\partial f^1}{\partial x_2}\right|_{x_2,\,x_3} & \left.\frac{\partial f^1}{\partial x_3}\right|_{x_2,\,x_3} \end{pmatrix} \tag{7.10}$$
$$= \begin{bmatrix} bx_1 & bx_2 & bx_3 \end{bmatrix} \;,$$

the differential states visited during an execution of the reverse AD code, bV^k, contain retrograde differential information when compared to the forwards case of dV^k. Rather than accumulating the changes in the numerical values taken by variables as a result of a change in input, the reverse states accumulate the sensitivity of an output to the values taken by each variable. On completion of the reverse AD algorithm, each member, bq, of the reverse differential state takes the numerical value of $\frac{dv_j^N}{dq}$, the differential of the selected output with respect to the variable q. Intermediate b-variables are less clearly defined than the d-variables of the forwards pass. Forwards differential variables propagate a differential change in an input alongside the execution of the original program. Thus, each variable dq takes the numerical value of $\frac{dq}{dv_j^0}$ on its assignment. However, if many functions in the original program have a dependence upon the same variable, each of these functions will produce a contribution to bq in a reverse pass of the AD code. Only when all contributions have been summed does a variable bq take the numerical value of $\frac{dv_j^N}{dq}$.

Program Outline 3 displays the forwards and reverse AD expressions for

the example program,

$$
\begin{array}{ccccc}
 & & [dx_1,\ dx_2,\ dx_3]^{\mathrm{T}} & & [bx_1,\ bx_2,\ bx_3] \\
dz = \left.\frac{\partial f^1}{\partial x_2}\right|_{x_2,\,x_3} dx_2 + \left.\frac{\partial f^1}{\partial x_3}\right|_{x_2,\,x_3} dx_3 & & \Downarrow & M^1 & \Uparrow & & bx_2 = bx_2 + bz \left.\frac{\partial f^1}{\partial x_2}\right|_{x_2,\,x_3} \\
 & & & & & & bx_3 = bz \left.\frac{\partial f^1}{\partial x_3}\right|_{x_2,\,x_3} \\
 & & [dx_1,\ dx_2,\ dz]^{\mathrm{T}} & & [bx_1,\ bx_2,\ bz] & & \\
dy_2 = \left.\frac{\partial f^2}{\partial x_2}\right|_{x_2,\,z} dx_2 + \left.\frac{\partial f^2}{\partial z}\right|_{x_2,\,z} dz & & \Downarrow & M^2 & \Uparrow & & bx_2 = by_2 \left.\frac{\partial f^2}{\partial x_2}\right|_{x_2,\,z} \\
 & & & & & & bz = by_2 \left.\frac{\partial f^2}{\partial z}\right|_{x_2,\,z} \\
 & & [dx_1,\ dy_2]^{\mathrm{T}} & & [bx_1,\ by_2] & & \\
dy_1 = \left.\frac{\partial f^3}{\partial x_1}\right|_{x_1} dx_1 & & \Downarrow & M^3 & \Uparrow & & bx_1 = by_1 \left.\frac{\partial f^3}{\partial x_1}\right|_{x_1} \\
 & & [dy_1,\ dy_2]^{\mathrm{T}} & & [by_1,\ by_2] & &
\end{array}
$$

(Program Outline 3)

where it can be seen that the reverse operation of the M^k matrices produces differential expressions of a different form. In the forwards case variable assignments are generally many-to-one; a value taken by a single forwards differential variable is dependent on the values of a number of others. In the reverse case the opposite is true, where in each step the assignment of multiple differential variables are dependent upon only a single value, resulting in a larger number of simpler expressions. The accumulation of reverse differential quantities can be seen in the calculation of bx_2 in the example program, where both f^1 and f^2 are dependent on x_2. A representation of differential accumulation in the forwards and reverse modes is shown in Fig. 7.1.

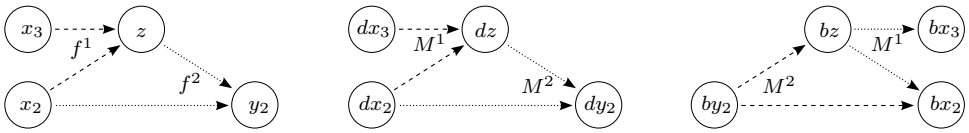


Figure 7.1: An illustration of the accumulation of differential variables in the first two steps of the example program.

Although the implementation of reverse mode AD is more involved than that of the forwards mode, similar bounds can be placed on its computational complexity.[141,142] Using the same methodology it can be shown that the cost of evaluating a selected function output and all of its derivatives will not exceed four times the cost of evaluating the original function alone, a result known as the 'cheap gradient principle'. The memory requirements are more complicated than the forwards case. When executing the reverse AD algorithm, the evaluation of the differential expressions constituting the elements of the M^k matrices requires the values of the variables in V^k. In the forwards mode these values are available at the time of execution as the differential code is executed alongside the original. In the reverse case, the values of all the variables at each step of the original program must be saved and subsequently restored during the reverse pass. If the original function is encoded with no variable overwriting, then the memory requirement of calculating the function value and gradient is twice that required when evaluating the original function in isolation. With memory overwriting permitted, the requirements are less clearly defined. All of the unique values taken by each variable must be stored, resulting in a memory demand that is proportional to the run time required to evaluate the underlying function. For very complex functions the memory demands of the basic reverse mode

can be prohibitively large, as the cheap gradient principle is valid only in the presence of sufficient, fast RAM.

7.4 Implementation

The application of AD can proceed mechanically, and there are a number of tools which automate the process.[143–145] These tools come in two varieties, those which function via source code transformations and those which use operator overloading.

A source code transformation is performed in a similar way to how the example program in the previous section is displayed. Presented with the source code defining the function to be differentiated, a source code transforming AD tool can parse the contents and automatically generate the corresponding differential expressions. These expressions are provided as a complementary section of source code containing the differential algorithm which, when compiled alongside the original source code, allows a programmer access to the derivative evaluating routines.

The operator overloading approach modifies basic arithmetic procedures so that, in addition to calculating values, they also evaluate differential quantities. Programs treated in such a manner are typically 10% to 20% as efficient as those produced by a source code transformation as it is much more difficult for compilers to optimise the overloaded operations; native instructions for handling operations on floating point numbers are replaced with slower procedures that handle structures of floating point numbers. However, if the original source code contains a lot of abstract, high level structure, such

as C++ templating, then a source code transformation may not be possible.

A straightforward implementation of reverse mode AD consists of two components, a forward sweep which executes the original code and stores all the values taken by all the variables, and a reverse sweep which uses these values to compute derivatives. As previously mentioned, the memory demand of expensive computations using many intermediate variables can exceed the availability of fast memory. In this regime 'checkpointing' strategies[146,147] can be employed, involving the recomputation of variables and the partitioning of the reverse pass into stages. When a stage of the program is checkpointed it is executed without the storage of the intermediate variable values. On performing the reverse sweep, when a checkpointed section of the program is encountered, the checkpointed stage is executed again, this time including the storage of intermediate variable values. Once this section is complete the reverse sweep can continue until it encounters the next checkpoint. A simple checkpointing process is illustrated in Figure 7.2.

This procedure provides an easy way of trading memory demand for execution time, but obtaining reasonable performance is difficult; currently, no automatic AD tool is capable of optimal checkpointing and most require programmer intervention to get a reasonably efficient implementation. Whilst the additional computational burden of checkpointing can be severe, the computational complexity remains independent of the number of input parameters, retaining the primary advantage of using reverse mode AD in systems with many degrees of freedom.

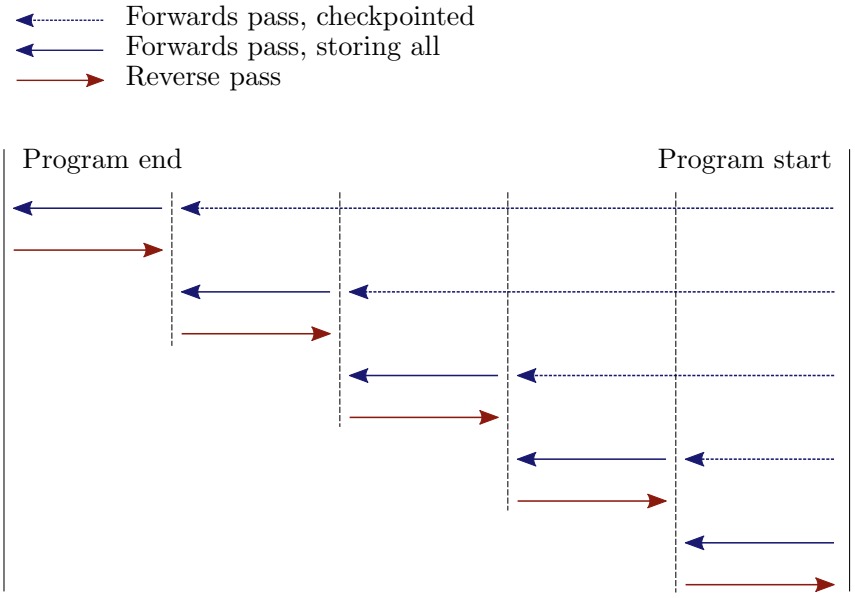


Figure 7.2: An outline of a basic checkpointing strategy used when performing reverse mode AD. The procedure starts at the top right corner with a conventional program execution flowing from right to left. Vertical dotted lines represent the positions of checkpoints.

Chapter 8

Algorithmic differentiation of diffusion Monte Carlo

8.1 Introduction

The DMC algorithm is an attractive target for the AD method. Existing approaches to extracting derivatives within DMC simulations all rely upon approximations to the DMC wave function and scale linearly in computational cost with the number of derivatives required. Both limitations are ameliorated with an application of reverse mode AD, which allows remarkably efficient access to exact derivatives of an algorithm's result with respect to every single input parameter. In the case of DMC, implementing AD presents two principal challenges. The first is a result of the relative expense of DMC calculations, which typically consume many hours of parallel execution on supercomputers to generate total energy estimates. This puts the DMC algorithm firmly in the regime where reverse mode AD checkpointing

strategies are unavoidable. The second, more difficult, challenge is dealing with stochasticity.

This chapter outlines the application of AD to a DMC algorithm. The first section presents the structure of a bespoke DMC code, which is optimised for an implementation of AD, before detailing the form of the corresponding AD expressions themselves. Here, the main idea is that, without a very careful consideration of which variables to store or recompute, the application of AD to DMC would not be possible. The manual implementation of reverse mode AD allows very fine grain control of the accumulation of derivatives and the associated memory management, making the application of AD feasible. Section 8.3 contains a strategy for calculating accurate DMC total energy derivatives in the presence of a stochastic execution path. As indicated by Section 5.2, the straightforward application of AD to the DMC algorithm produces derivatives which do not account for probabilistic decisions made. This can be corrected by modifying the branching of walkers such that differential information is propagated through integer-based branching events. The new form for the AD equations is shown to produce derivatives which are exact, and the final section presents a selection of results which verify the accuracy and utility of derivatives calculated using this method. A series of benchmark simulations of the ethane, propane and butane molecule demonstrate the efficiency and scalability of DMC force calculations, and results are displayed from a molecular dynamics simulation driven entirely by DMC forces, which captures the dimerisation of two borane molecules.

8.2 Algorithm structure

In principle it is possible to apply AD to an existing code using automated tools. However, the efficiency of the resultant AD implementation is highly dependent upon the structure of the original algorithm. Ideally the code to be differentiated should contain only functions that are atomic, where the result of each function depends upon its explicit inputs only. In this case, supplied with a particular set of inputs, each function can be executed and differentiated in isolation, irrespective of the state of the remainder of the computer program. Codes containing non-atomic functions, where the result depends upon transient information stored in a global scope, typically yield far poorer AD representations. When performing the reverse sweep of reverse mode AD, the differentiation of a non-atomic function requires version control of the global variables used; to successfully navigate the reverse AD expressions, the global scope must be restored to exactly the same state as it was when the original, undifferentiated function was evaluated. The logic required to handle restoring global variables is not particularly complex, but the computational and storage overheads can be large, especially in the presence of multiple non-atomic functions. Programs containing multiple concurrent threads suffer especially badly, as updates to global variables must be synchronised with the operations of all threads.

For the DMC algorithm this presents a significant problem as DMC calculations are already very expensive. Furthermore, the computational cost of a DMC calculation scales approximately as $\mathcal{O}(N^3)$, where N is the number of electrons in the system, and to use DMC in systems where no other electronic

structure method is comparable requires handling hundreds of electrons. The memory requirement of a straightforward application of AD is roughly proportional to the run time of the original program, so any significant DMC simulations will incur an extremely large memory cost. This will inevitably exceed the RAM available to the computer running the DMC calculation, and an appropriate RAM conservation strategy must be selected. The simplest solution is to move the useful content saved in memory to the hard disk when the available RAM is exhausted. Then, when using the saved information in the reverse sweep of AD, the contents of the RAM can be restored when required. Whilst relatively easy to implement on a desktop computer, this approach is more difficult to apply on supercomputers where the available RAM is partitioned between different subsets of processors and disk access can be prohibitively slow. Another solution, employed by automatic AD tools, is to use checkpointing strategies, as discussed in Section 7.4. These work well when the number of checkpoints required is relatively small, but, assuming comparable speeds for the forwards and reverse sweeps, the overall additional computational cost is roughly proportional to $\frac{N_c}{2}T$, where N_c is the number of checkpoints and T is the computational cost of the original program. For larger calculations the memory demands incurred by the DMC method can be many thousands of times greater than the RAM available, requiring a similar number of checkpoints and making this approach rather inefficient.

The inherent difficulty in applying an automatic AD tool to expensive calculations, combined with a pervasive lack of atomicity in existing DMC programs, motivates the design of a new DMC implementation tailored for

the subsequent implementation of AD from the outset. This is realised in a bespoke Fortran code written under the paradigm of functional programming, where atomicity of all functions is ensured, allowing automatic AD tools to differentiate sections of the code individually without consequence. A manual implementation of the AD expressions in the main loop over DMC time steps permits the implementation of a unique checkpointing strategy, mitigating the computational overhead of a basic method. The overriding principle is to store the minimal amount of information required to completely recompute all quantities on a given time step. This provides the capacity to effectively checkpoint the forwards sweep in addition to the reverse sweep, avoiding the excessive recomputation of the first stages of a calculation and keeping the overall memory requirement low. Figure 8.1 contrasts the operation of this new approach with the straightforward implementation depicted in Figure 7.2.

The recalculation of all quantities on a given time step requires the current best estimate of the DMC total energy, the weights and configurations of all walkers, and the random numbers used to evaluate the non-local component of any pseudopotentials. In the absence of a pseudorandom number generator which is able to save and restore its state quickly, or better yet, efficiently reverse the stream of numbers produced, all of these quantities aside from the best estimate of the total energy must be stored. Tracing the entire program in reverse also requires additional information about the ancestry of the walkers, the history of fluctuations in the walker population and the computational load balancing performed on each time step, where processors with a high population of walkers at the end of a time step redistribute

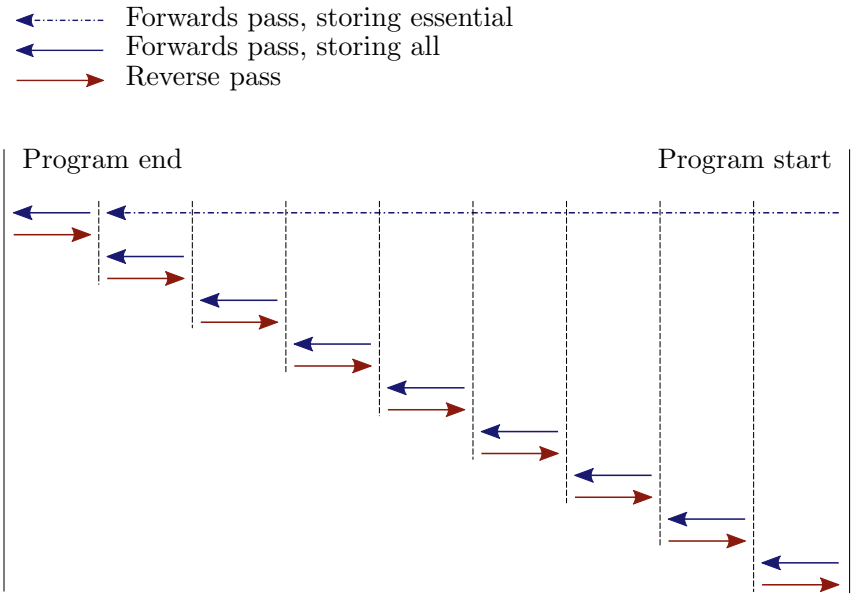


Figure 8.1: An outline of the checkpointing strategy employed to perform AD of DMC. In contrast with Figure 7.2 all checkpoints are created on the initial forwards pass.

excess walkers among processors with a lower population. Furthermore, the proposition and acceptance of single-electron configuration updates must be recorded to successfully navigate the dynamics in reverse. The total storage requirements are summarised below:

Per time step

- The current best estimate at the DMC total energy
- The number of walkers
- The difference between the number of walkers and ideal number of walkers for each processor

  Per walker

  * The initial weight

* The previous weight
* The parent

  Per electron:

  · The position
  · The random numbers used to evaluate the non-local component of any pseudopotentials
  · If the move was accepted
  · If a T-move was accepted
  · The T-move position, if a T-move was accepted
  · The previous position, if the walker was on a different processor for the previous time step

The storage of the data is handled by last-in, first-out (LIFO) stacks for each quantity listed above. Allocating the capacity of each store is complicated by fluctuations in walker numbers, configuration transfers and the number of T-moves performed; the required stack size must include redundancy when setting up a calculation. The size of the extra storage space required is dependent on the quality of the trial wave function and is thus different for every DMC calculation. Relative to a perfect simulation, where the population of walkers on each processor is a fixed constant, 0% of configurations are transferred and 0% of T-moves are made, the stack sizes for calculations in this thesis are based on a population size of 120% of the expected value, with 1% of configurations making T-moves and 1% of walkers being transferred. There is also a small, fixed minimum stack size in place to protect smaller scale calculations. If the combined stack size is greater

than the available memory then synchronised transfers of data from RAM to disk must be triggered periodically. On the reverse pass the stores will empty at the same time by construction and the stacks will be restored from data saved on disk.

An entire calculation would ideally be performed without exceeding the capacity of the RAM. The total memory cost scales as the product of the number of time steps, the number of walkers and the number of electrons. Fortunately this cost can be mitigated by features of the DMC algorithm. The first extenuation comes from the inherent parallelism of Monte Carlo methods. If a large number of walkers is required, there is no significant performance penalty associated with distributing the memory requirements over multiple processors. In HPC clusters it is common that each node of a supercomputer is provisioned with independent banks of RAM, so a high degree of parallelisation allows access to very large amounts of fast memory. The second RAM optimisation strategy is to partition a longer DMC calculation into multiple, shorter DMC runs which are then averaged together. This approach incurs extra computational effort to equilibrate each of the shorter runs individually, but effectively removes the scaling of memory demand with the number of time steps. As the simulation length decreases the fraction of computational time spent equilibrating walkers increases. However, DMC equilibration is mercifully quick, commonly taking on the order of one unit of atomic time, so very short simulations can still be performed without equilibration dominating the calculation.

An additional benefit of performing multiple independent DMC simulations is that it produces an estimate of the error in the derivatives calculated

using AD. The variance of the weighted local energies provides a measure of the error in the total energy, but, as the AD expressions do not produce an estimator for a derivative on each time step, a similar analysis is impossible. Each DMC calculation produces a single number for each derivative of interest, and estimates of their variances can only be obtained from the standard error in the mean of repeated evaluations using different streams of random numbers.

Section 4.3 describes the features of the DMC algorithm to be differentiated. Slater–Jastrow trial wave functions are generated using the The General Atomic and Molecular Electronic Structure System (GAMESS)[148,149] software package to obtain a Slater determinant of single particle orbitals expressed in a nuclei-centred Gaussian basis, and the CASINO package to optimise a Jastrow factor of the form

$$\begin{aligned} J(\bar{\boldsymbol{r}},\ \bar{\boldsymbol{R}}) \ = \ & \sum_{i=1}^{N_e-1} \sum_{j=i+1}^{N_e} u(r_{ij}) \ + \ \sum_{i=1}^{N_e} \sum_{I=1}^{N_n} \chi_I(r_{iI}) \\ & + \ \sum_{i=1}^{N_e-1} \sum_{j=i+1}^{N_e} \sum_{I=1}^{N_n} f_I(r_{ij},\ r_{iI},\ r_{jI}) \ , \end{aligned} \tag{8.1}$$

where N_e is the number of electrons, N_n is the number of nuclei, $r_{ij} = |\boldsymbol{r}_i - \boldsymbol{r}_j|$, and $r_{iI} = |\boldsymbol{r}_i - \boldsymbol{R}_I|$. The electron-electron u terms, and the nuclei-centred, electron-nucleus χ_I and electron-electron-nucleus f_I terms, are defined by the CASINO parametrisation[95] and a set of optimisable coefficients. For finite simulations the addition of a Jastrow factor reduces the variance of the estimate of a DMC total energy but leaves its mean value intact. Thus, variations in its form contribute nothing towards total energy derivatives and

wave function optimisation can be performed outside the scope of the code subject to AD. The remaining inputs to the DMC algorithm are the positions of the nuclei, $\bar{\boldsymbol{R}}$, and a set of Gaussian basis function coefficients describing the single particle orbitals, $\{\alpha_j\}$, which are themselves dependent upon $\bar{\boldsymbol{R}}$. Applying AD to the DMC algorithm yields the derivative of the DMC total energy, E_{DMC}, with respect to each of its inputs, but the calculation of nuclear forces also requires the derivatives of $\{\alpha_j\}$ with respect to $\bar{\boldsymbol{R}}$. The expression for the force acting on nucleus i is given by

$$\boldsymbol{F}_i = -\frac{dE_{\mathrm{DMC}}}{d\boldsymbol{R}_i} = -\left(\left[\frac{\partial E_{\mathrm{DMC}}}{\partial \boldsymbol{R}_i}\right]_{\mathrm{AD}} + \left[\frac{\partial E_{\mathrm{DMC}}}{\partial \alpha_j}\right]_{\mathrm{AD}} \frac{d\alpha_j}{d\boldsymbol{R}_i}\right) , \qquad (8.2)$$

where the square brackets indicate quantities obtained from AD. Ideally the matrix $\frac{d\alpha_j}{d\boldsymbol{R}_i}$ would be extracted from the wave function generating routine, but all AD force calculations presented in this thesis use numerical finite difference derivatives of GAMESS calculations.

By writing a DMC code in a functional style it is possible to trace the evaluation of functions back to the inputs of the program. Figure 8.2 shows a call graph which includes the most important functions in the DMC algorithm, where the terminal functions incorporate data from the orbital coefficients, Jastrow parameters, and/or the nuclear positions. This representation gives an indication of how changes in the inputs propagate through the structure of the code to produce a change in the output. The operation of the reverse AD code can also be revealed with a call graph, as shown in Figure 8.3. For

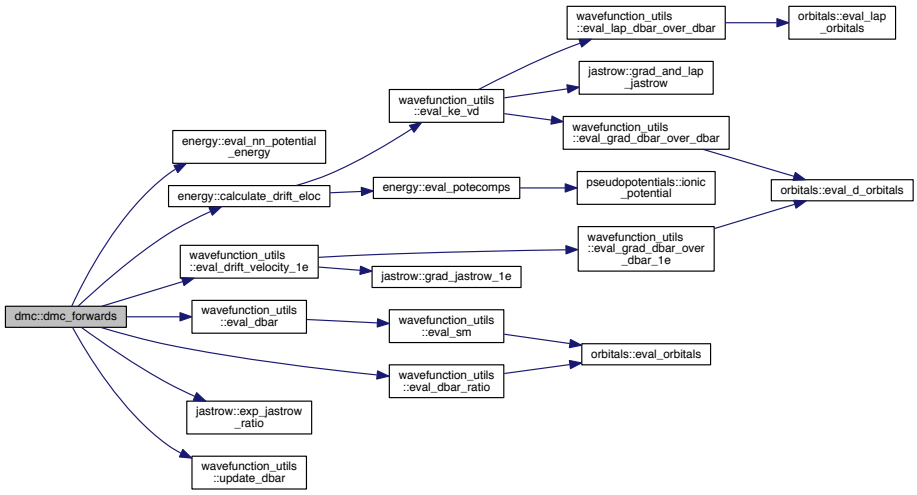


Figure 8.2: A call graph of the significant functions in the forwards pass of the DMC code.

each function or subroutine of the form

$$\texttt{calc_something}(\texttt{x1},\ \texttt{x2},\ \texttt{y}) \tag{8.3}$$

where `x1` and `x2` are inputs, and `y` is an output, there is a corresponding reverse AD routine

$$\texttt{calc_something_b}(\texttt{x1},\ \texttt{x1_b},\ \texttt{x2},\ \texttt{x2_b},\ \texttt{y_b}) \tag{8.4}$$

where `y_b` is the reverse differential seed and `x1_b` and `x2_b` contain the contributions to the derivative of the overall program output with respect to `x1` and `x2` from the evaluation of `calc_something(x1, x2, y)`. As in Figure 8.2 it is possible to trace the accumulation of the derivatives of all input variables through the network of function calls. The DMC algorithm is navigated

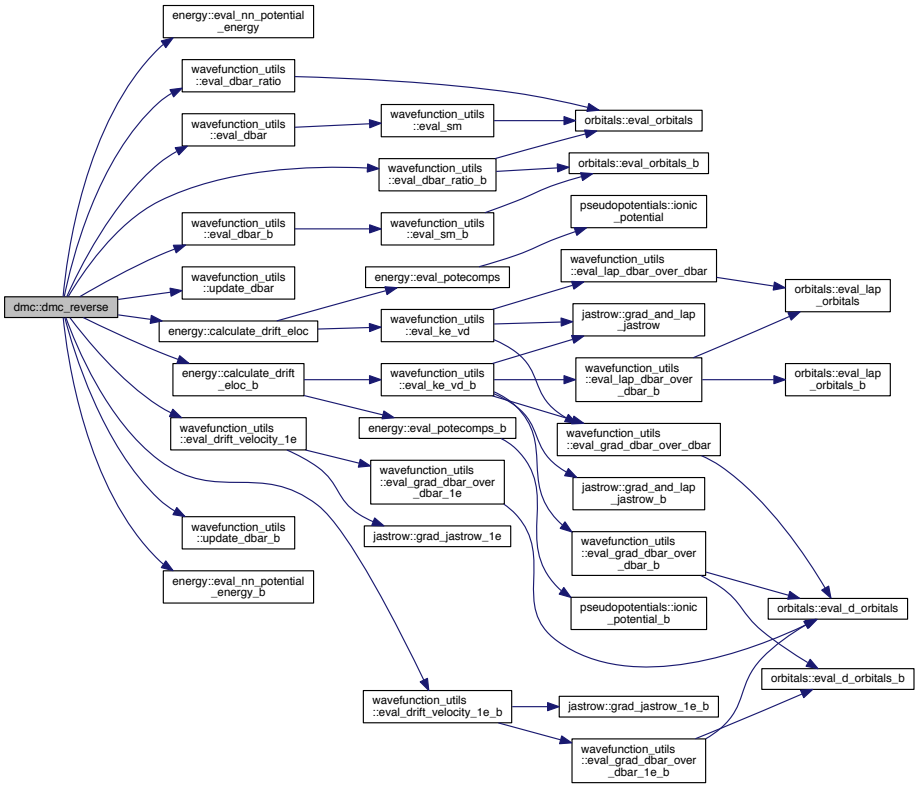


Figure 8.3: A call graph of the significant functions in the reverse pass of the DMC code.

backwards via the sequential evaluation of the reverse AD functions in the opposite order to the original functions. Figure 8.3 also visualises the recomputation of variable values, where each reverse AD function is accompanied by an evaluation of the original function to generate the required data. A brief description of each of the functions is contained in Appendix A.

The Tapenade[143] automatic AD tool is used to differentiate sections of the DMC code when accumulating derivatives on the reverse pass. These include the evaluation of the Gaussian orbitals, the Jastrow factor and the

pseudopotentials, where Tapenade manages the small amount of memory required for forwards and reverse sweeps of these functions in isolation. The obtained derivatives are then inserted into the sequence of AD equations spanning the main loop over time steps, where the memory is managed by the bespoke LIFO stores.

To ensure that the DMC algorithm is compatible with Jastrow factors optimised using CASINO, the two codes share routines to evaluate Gaussian wave function orbitals, Jastrow factors and pseudopotentials. This also grants access to the CASINO pseudopotential library and a utility for converting wave functions from a GAMESS format into a representation more suitable for DMC calculations. Molecular dynamics simulations are then performed using DMC forces via a Python script that automates the generation and conversion of GAMESS wave functions, the execution and AD of the DMC algorithm, and the application of equation (8.2). With fixed Jastrow factor coefficients, this apparatus provides a routine for evaluating DMC forces at arbitrary nuclear geometries which can be fed into a Verlet[150] algorithm to integrate the subsequent equation of motion. Performing molecular dynamics with Monte Carlo errors in the forces is equivalent to the presence of temperature, where the magnitude of the uncertainty in the force estimates can be absorbed into the temperature of a Langevin thermostat.[151] For the relatively short dynamical simulations performed in this thesis the errors in the forces are sufficiently small that the influence of Monte Carlo noise has little effect on the trajectories of the atoms. A measure of the influence of the perceived temperature can be obtained from inspecting the conservation of energy over the course of a simulation.

8.3 Stochastic branching

The result of applying AD to the DMC algorithm as described in Section 8.2 is shown in Figures 8.4 and 8.5, which display the total energy and its gradient when stretching a lithium hydride molecule. Here, each DMC simulation is performed using one-electron orbitals obtained from all-electron HF calculations with an aug-cc-pVDZ Gaussian basis set, and the total energies are averages of 768 independent DMC calculations employing 400 walkers and 400 time steps of 0.0025 a.u. The AD results are obtained from an average of the AD derivatives of each of the original total energy evaluations and error bars represent the standard error in the means.

In Figure 8.4 a 'naive' differentiation is one that fails to account for the stochastic decisions made during a DMC calculation. On a *macroscopic* scale there is a systematic discrepancy between the desired and calculated derivative. However, Figure 8.5 shows that on a *microscopic* scale the gradient calculated using naive AD is correct to numerical precision. There is, therefore, an important contribution to the macroscopic gradient from the discontinuous jumps on the micro scale. Each of these discontinuities is the result of a single different yes-no decision made inside the DMC code as a response to the change in bond length. As the length of a DMC simulation increases, the statistical error in the DMC total energy decreases and the discontinuities reduce in size, but the corresponding increase in the number of decisions made also brings the discontinuities closer together. The overall contribution of the discontinuities to the macroscopic gradient remains finite.

There are four places random numbers are used in the DMC algorithm:

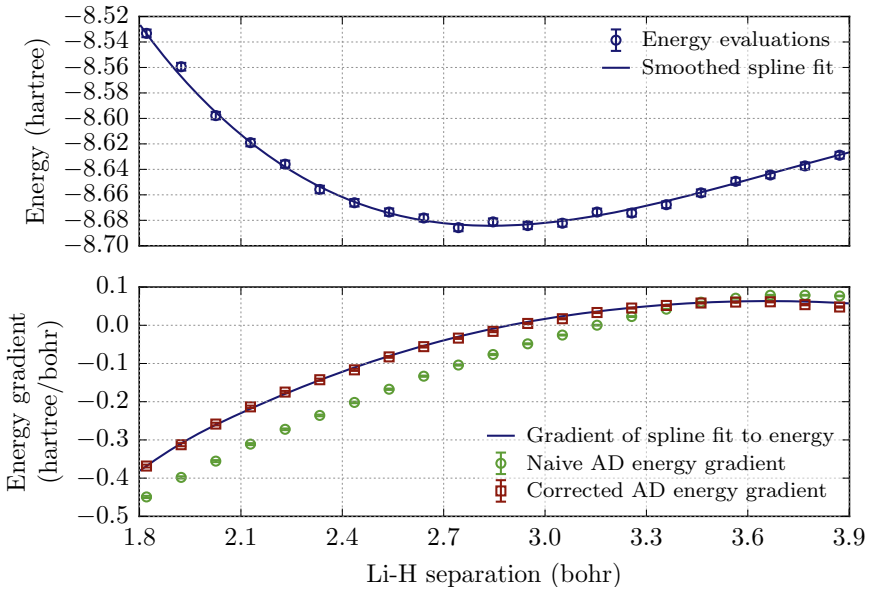


Figure 8.4: The effect of stochastic decisions on AD derivatives. The top pane shows the variation of the DMC total energy with the bond length of a lithium hydride molecule. The bottom pane compares the gradient of the total energy function with derivatives calculated using AD of the total energy evaluations of the top pane.

stochasticity is required to generate the diffusive component of the Langevin dynamics, to select the orientation of integration grids to evaluate pseudopotentials, to discretise the weights of walkers before branching events, and to accept or reject configuration updates using the Metropolis algorithm. The simulation of diffusion is system agnostic; any DMC calculation with a specific time step can propose walker positions using exactly the same set of vectors. Therefore, from a finite difference perspective, simulations at λ and $\lambda + d\lambda$ can share diffusive moves without consequence, where λ is generalised DMC parameter which can represent a trial wave function coefficient, a nuclear position or any other parameter of the simulation. The same is true for the random orientation of the integration grids used when evaluating pseu-

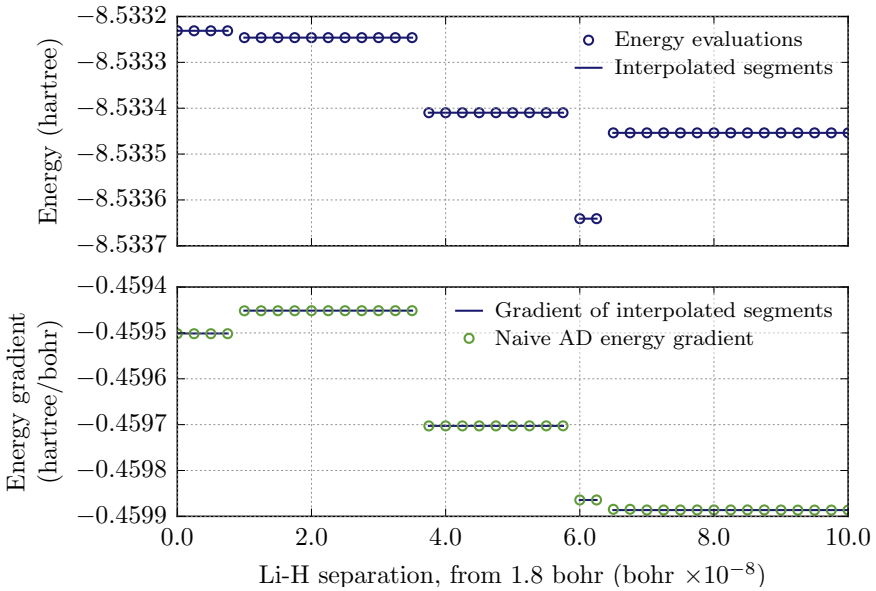


Figure 8.5: The derivative of the DMC total energy over a very small range. The top pane shows the same content as in Figure 8.4 but over a small range of displacements on the leftmost side of the x-axis. Here each calculation uses an identical stream of random numbers and the associated stochastic error bars have been omitted. The bottom pane compares the gradient of the total energy function with derivatives calculated using AD of the total energy evaluations of the top pane.

dopotentials. However, the acceptance of configuration updates, governed by

$$P_{\mathrm{M}}(\bar{\boldsymbol{r}}' \leftarrow \bar{\boldsymbol{r}}) \;=\; \min\left(1,\; \frac{G_{\mathrm{L}}(\bar{\boldsymbol{r}} \leftarrow \bar{\boldsymbol{r}}',\; \Delta t)|\Psi_{\mathrm{T}}(\bar{\boldsymbol{r}}')|^2}{G_{\mathrm{L}}(\bar{\boldsymbol{r}}' \leftarrow \bar{\boldsymbol{r}},\; \Delta t)|\Psi_{\mathrm{T}}(\bar{\boldsymbol{r}})|^2}\right) \tag{8.5}$$

from equation (4.25), or the discretisation of walker weights,

$$N_{\mathrm{spawn}} \;=\; W \;=\; \mathtt{INT}(w \;+\; \eta)\;, \tag{8.6}$$

from equation (4.14), is more troublesome. Similarly to the algorithm described in section 5.5, the output of both operations is a piecewise, discontinuous function where calculations at λ and $\lambda + d\lambda$ are constrained by the

AD equations to use the same decisions, producing incorrect derivatives.

The Metropolis rejection is not an essential component of the DMC algorithm. Its two main functions are to reduce the time step error and add some extra stability by providing a means, other than the drift velocity, of keeping walkers away from regions near the nodes of the trial wave function. Disabling this feature and using a smaller time step will generate equivalent results, albeit with an increased computational cost. However, in most cases the time step error varies slowly with any simulation parameter other than Δt. Thus, simulations at λ and $\lambda + d\lambda$ can expect to have near identical time step errors and equation (8.5) has little effect on a derivative. All AD derivatives presented in this chapter are derived from total energy evaluations using Metropolis rejection, but do not differentiate the rejection process.

The discretisation of walker weights is a process required to produce an integer number of offspring at the current electronic configuration. In principle a DMC calculation can be performed without branching walkers, but in practice the exponential rate of weight accumulation necessitates branching to stabilise fluctuations in the walker weights and increase the efficiency of the sampling. Walker branching is essential for a DMC algorithm with useful performance, but, in contrast to the Metropolis rejection, ignoring its contribution to derivatives produces very biased results.

Without branching the weights of walkers are accumulated multiplicatively via

$$G_w(\bar{\boldsymbol{r}}' \leftarrow \bar{\boldsymbol{r}},\ \Delta t) \ = \ \exp\left(-\frac{1}{2}[E_{\mathrm{L}}(\bar{\boldsymbol{r}}') \ + \ E_{\mathrm{L}}(\bar{\boldsymbol{r}}) \ - \ 2E_{\mathrm{T}}]\Delta t\right) \ , \qquad (8.7)$$

from equation (4.27). Over three time steps where a walker with unit weight and an initial configuration of $\bar{\boldsymbol{r}}_0$ visits configurations $\bar{\boldsymbol{r}}_1$, $\bar{\boldsymbol{r}}_2$ and $\bar{\boldsymbol{r}}_3$ consecutively, the final weight is given by

$$\begin{aligned} w_3' &= G_w(\bar{\boldsymbol{r}}_3 \leftarrow \bar{\boldsymbol{r}}_2,\ \Delta t) G_w(\bar{\boldsymbol{r}}_2 \leftarrow \bar{\boldsymbol{r}}_1,\ \Delta t) G_w(\bar{\boldsymbol{r}}_1 \leftarrow \bar{\boldsymbol{r}}_0,\ \Delta t) && (8.8) \\ &= G_{w_3} G_{w_2} G_{w_1} && (8.9) \\ &= G_{w_3} w_2' \ . && (8.10) \end{aligned}$$

Differentiating w_3' with respect to λ yields

$$\frac{dw_3'}{d\lambda} = \frac{dG_{w_3}}{d\lambda} w_2' + G_{w_3} \frac{dw_2'}{d\lambda} \ , \tag{8.11}$$

which can be iterated up to an arbitrary number of time steps,

$$\frac{dw_i'}{d\lambda} = \frac{dG_{w_i}}{d\lambda} w_{i-1}' + G_{w_i} \frac{dw_{i-1}'}{d\lambda} \ . \tag{8.12}$$

Reintroducing the branching of walkers makes the accumulation of weights discontinuous. At the beginning of each time step the weight of each walker is discretised according to (8.6), which converts each weight into an integer by truncating the floating-point sum of the weight and a unit magnitude uniform random number. This integer normally determines how many walkers of unit weight to spawn, but the remainder of this section will assume, without loss of generality, that no spawning takes place. In this case the population of walkers is fixed where each has a non-negative integer weight

at the start of each time step. The equivalent of equation (8.9) now reads

$$w_3 \;=\; G_{w_3} W_2 \;, \tag{8.13}$$

where W_i indicates an integer weight obtained from applying equation (8.6) to w_i, and weights without primes indicate values obtained in the presence of branching. Differentiating w_3 in the same manner as w_3' gives

$$\frac{dw_3}{d\lambda} \;=\; \frac{dG_{w_3}}{d\lambda} W_2 \;, \tag{8.14}$$

and, furthermore, the derivative of a weight after an arbitrary number of time steps is given by

$$\frac{dw_i}{d\lambda} \;=\; \frac{dG_{w_i}}{d\lambda} W_{i-1} \;. \tag{8.15}$$

Whilst mathematically correct, this expression lacks information about how the current weight is causally linked to the history of previous weights, as the derivative of the integer truncation operation is zero for all w, aside from delta functions, δ_w, when $w + \eta$ takes an integer value n,

$$\frac{dW_i}{dw_i} \;=\; \delta_w((w+\eta) \;-\; n) \;, \tag{8.16}$$

which are never sampled. This is the source of the discrepancy in the derivatives shown in Figures 8.4 and 8.5. The AD expressions evaluate equation (8.15) correctly and produce exact derivatives of an execution of the DMC algorithm for a given set of random numbers. However, the real quantity of interest is the derivative of DMC calculations averaged over a continuum of

random numbers.

Section 5.5 derives an adaptation to the calculation of derivatives that incorporates a probabilistic execution path,

$$\begin{aligned} \frac{d\left[\,\omega W_i\,\right]}{dx} &= \frac{dW_i}{dx} + W_i \sum_i \frac{1}{P_i}\frac{dP_i}{dx} \\ &= W_i \sum_i \frac{1}{P_i}\frac{dP_i}{dx} \,. \end{aligned} \tag{8.17}$$

from equation (5.39), but the resultant expression has large fluctuations in value. Here each P_i is the probability of each discretisation event in the walker's history, and if a simulation runs for any significant amount of time it becomes inevitable that a decision will be made with a very small P. As P is in the denominator of the second term, the value of the probabilistic correction becomes very large and, with far greater consequence, feeds in to derivatives of subsequent Ws. Thus, the variance of the derivatives calculated using this method increases with the number of time steps performed, ruling out an application to highly accurate DMC calculations.

It is, however, possible to formulate the calculation of derivatives that are independent of the discretisation process. Equation (8.13) can be expanded using a multiplicative representation of equation (8.6),

$$W_i = \frac{W_i}{w_i} w_i \,, \tag{8.18}$$

giving

$$w_3 = G_{w_3}\frac{W_2}{w_2}G_{w_2}\frac{W_1}{w_1}G_{w_1} \,, \tag{8.19}$$

which, again, can be extended iteratively,

$$w_i = G_{w_i}\left(\prod_{j=1}^{i-1}\frac{W_j}{w_j}G_{w_j}\right) \tag{8.20}$$

$$= \left(\prod_{j=1}^{i-1}\frac{W_j}{w_j}\right)\left(\prod_{k=1}^{i}G_{w_k}\right) \tag{8.21}$$

$$= \left(\prod_{j=1}^{i-1}\frac{W_j}{w_j}\right)w_i' \ . \tag{8.22}$$

As the number of branching events increases, the expected value of the weight in the presence of branching tends to the value in the absence of branching. Thus, when averaged over multiple branching events for a specific walker, each ratio $\frac{W_i}{w_i}$ of discretised weights over original weights in equation (8.22) has unit value, and

$$\langle w_i \rangle = \left\langle \prod_{j=1}^{i-1}\frac{W_j}{w_j} \right\rangle w_i' = w_i' \ . \tag{8.23}$$

Equation (8.16) can be redefined to propagate differential information from w_i to W_i by introducing a differential change in the integer weight of the form

$$dW_i = \frac{W_i}{w_i}\,dw_i \ , \tag{8.24}$$

which is commensurate with the fractional change in the value of the weight. This is no longer the value of dW_i determined by equation (8.16), but the following equations show that it ensures that the average of dW_i over branching decisions reproduces derivatives which are identical to those in the absence

of branching. With a differential dependence on w_i,

$$\frac{dW_i}{dw_i} = \frac{W_i}{w_i} , \tag{8.25}$$

the differentiation of each W_i with respect to λ now produces

$$\begin{aligned}
\frac{dW_i}{d\lambda} &= \frac{dW_i}{dw_i}\frac{dw_i}{d\lambda} && (8.26)\\
&= \frac{W_i}{w_i}\frac{dw_i}{d\lambda} && (8.27)\\
&= \frac{W_i}{G_{w_i}W_{i-1}}\frac{d(G_{w_i}W_{i-1})}{d\lambda} && (8.28)\\
&= \frac{W_i}{G_{w_i}W_{i-1}}\left(\frac{dG_{w_i}}{d\lambda}W_{i-1} + G_{w_i}\frac{dW_{i-1}}{d\lambda}\right) && (8.29)\\
&= W_i\sum_{j=1}^{i}\frac{1}{G_{w_j}}\frac{dG_{w_j}}{d\lambda} && (8.30)\\
&= \frac{W_i}{w_i'}\frac{dw_i'}{d\lambda} . && (8.31)
\end{aligned}$$

Thus, the derivative of each integer weight corresponds to an appropriately scaled derivative of the weight in the absence of branching. When averaging over multiple branching decisions it is possible to extract derivatives that are independent of any specific sequence of discretisation events,

$$\begin{aligned}
\left\langle\frac{dW_i}{d\lambda}\right\rangle &= \left\langle\frac{W_i}{w_i'}\frac{dw_i'}{d\lambda}\right\rangle\\
&= \left\langle\frac{W_i}{w_i'}\right\rangle\frac{dw_i'}{d\lambda}\\
&= \frac{dw_i'}{d\lambda} ,
\end{aligned} \tag{8.32}$$

where $\frac{dw_i'}{d\lambda}$ can be pulled out of the average as w_i' is the weight obtained in

the absence of any branching. The introduction of equation (8.24) into the AD expressions for the derivative of the DMC total energy yields the correct macroscopic derivative, as shown in Figure 8.4.

Equation (8.24) can be derived using a procedure similar to that of Section 5.2 by considering two DMC calculations at λ and $\lambda + \Delta\lambda$ which are constrained to use identical branching decisions and the same, fixed energy shift. Whilst problematic in practice, since the total number of walkers in a simulation with a fixed energy shift eventually reduces to zero or grows beyond the available memory, this simplifies the analysis and ensures that the future evolution of any given walker depends only on its current position and weight.

The branching decisions are determined using equation (8.6), which takes an input weight of w_{in} and produces an output of either $w^{+}_{\text{out}} = \texttt{INT}(w_{\text{in}}) + 1$ or $w^{-}_{\text{out}} = \texttt{INT}(w_{\text{in}})$ with respective probabilities P and $(1 - P)$ chosen such that the expected weight is conserved across the branching decision,

$$\langle w_{\text{out}} \rangle \;=\; P w^{+}_{\text{out}} \;+\; (1 - P) w^{-}_{\text{out}} \;=\; w_{\text{in}} \;. \tag{8.33}$$

At time step i of an N step DMC calculation, the expected contribution to the overall expectation value of the total energy from a walker at $\bar{\boldsymbol{r}}$ with weight w_i is given by

$$C_{1,\ldots,i} \;+\; w_i \langle c_{i+1,\ldots,N}(\bar{\boldsymbol{r}}_i) \rangle \;, \tag{8.34}$$

where $C_{1,\ldots,i}$ is the contribution to the energy estimator that has been made

by the walker so far, and $\langle c_{i+1,\ldots,N}(\bar{\boldsymbol{r}}_i)\rangle$ is the expected contribution that a walker of *unit weight* at position $\bar{\boldsymbol{r}}_i$ at time step i will make to the energy estimator during the rest of the simulation. Hence, when averaged over decisions, the expected future contribution of the walker to the energy expectation value is also conserved,

$$\begin{aligned}\langle w_{\text{out}}\rangle\langle c_{i+1,\ldots,N}(\bar{\boldsymbol{r}}_i)\rangle &= P w^{+}_{\text{out}}\langle c_{i+1,\ldots,N}(\bar{\boldsymbol{r}}_i)\rangle \\ &\quad + (1-P) w^{-}_{\text{out}}\langle c_{i+1,\ldots,N}(\bar{\boldsymbol{r}}_i)\rangle \\ &= w_{\text{in}}\langle c_{i+1,\ldots,N}(\bar{\boldsymbol{r}}_i)\rangle ,\end{aligned} \tag{8.35}$$

where the first expectation value on the right hand side is over decisions and the second is over walks continuing from $\bar{\boldsymbol{r}}_i$. This conservation of the expected contribution ensures that the expected total energy is unaffected by the branching decisions.

In the calculation at $\lambda+\Delta\lambda$, branching decisions are made with the same probabilities, P, as in the calculation at λ. The aim is to devise a branching scheme which still produces the correct expectation value,

$$\begin{aligned}\langle w^{\Delta\lambda}_{\text{out}}\rangle\langle c^{\Delta\lambda}_{i+1,\ldots,N}(\bar{\boldsymbol{r}}^{\Delta\lambda}_i)\rangle &= P w^{\Delta\lambda+}_{\text{out}}\langle c^{\Delta\lambda}_{i+1,\ldots,N}(\bar{\boldsymbol{r}}^{\Delta\lambda}_i)\rangle \\ &\quad + (1-P) w^{\Delta\lambda-}_{\text{out}}\langle c^{\Delta\lambda}_{i+1,\ldots,N}(\bar{\boldsymbol{r}}^{\Delta\lambda}_i)\rangle \\ &= w^{\Delta\lambda}_{\text{in}}\langle c^{\Delta\lambda}_{i+1,\ldots,N}(\bar{\boldsymbol{r}}^{\Delta\lambda}_i)\rangle ,\end{aligned} \tag{8.36}$$

where superscripts of $\Delta\lambda$ indicate quantities obtained in the calculation at $\lambda+\Delta\lambda$. Fortunately, just as in the calculation at λ, the expected contribution per unit weight, $\langle c^{\Delta\lambda}_{i+1,\ldots,N}(\bar{\boldsymbol{r}}^{\Delta\lambda}_i)\rangle$, is unaffected by the outcome of the decision.

The requirement of equation (8.36) can thus be simplified to

$$\langle w_{\text{out}}^{\Delta\lambda} \rangle \;=\; P w_{\text{out}}^{\Delta\lambda+} \;+\; (1-P) w_{\text{out}}^{\Delta\lambda-} \;=\; w_{\text{in}}^{\Delta\lambda} \;, \tag{8.37}$$

which is sufficient to guarantee that the calculation at $\lambda+\Delta\lambda$ gives the correct energy expectation value, even though the 'wrong' branching probabilities have been used.

Equation (8.37) links the two output weights $w_{\text{out}}^{\Delta\lambda\pm}$ but does not suffice to define them. Their form will be a function of the input weight, $w_{\text{in}}^{\Delta\lambda}$, and any of the variables available from the calculation at λ,

$$w_{\text{out}}^{\Delta\lambda\pm} \;=\; w_{\text{out}}^{\Delta\lambda\pm}(w_{\text{in}}^{\Delta\lambda},\; w_{\text{in}},\; w_{\text{out}}^{\pm}) \;, \tag{8.38}$$

where w_{in} and $w_{\text{out}}^{\pm}$ are constant for all different values of $\Delta\lambda$. Thus, the propagation of weight derivatives is given by

$$dw_{\text{out}}^{\Delta\lambda\pm} \;=\; \frac{\partial w_{\text{out}}^{\Delta\lambda\pm}(w_{\text{in}}^{\Delta\lambda},\; w_{\text{in}},\; w_{\text{out}}^{\pm})}{\partial w_{\text{in}}^{\Delta\lambda}} dw_{\text{in}}^{\Delta\lambda} \;. \tag{8.39}$$

The functions $w_{\text{out}}^{\Delta\lambda\pm}(w_{\text{in}}^{\Delta\lambda},\; w_{\text{in}},\; w_{\text{out}}^{\pm})$ are subject to a number of additional constraints:

1. The output weights $w_{\text{out}}^{\Delta\lambda\pm}$ must both be positive. If $w_{\text{out}}^{\Delta\lambda-}$ is determined from a choice of $w_{\text{out}}^{\Delta\lambda+}$ via equation (8.37), then $w_{\text{out}}^{\Delta\lambda+}$ must satisfy $P w_{\text{out}}^{\Delta\lambda+} \leq w_{\text{in}}^{\Delta\lambda}$. Similarly, if $w_{\text{out}}^{\Delta\lambda-}$ is chosen then $(1-P) w_{\text{out}}^{\Delta\lambda-} \leq w_{\text{in}}^{\Delta\lambda}$ is required.

2. As $\Delta\lambda \to 0$ the calculations at λ and $\lambda + \Delta\lambda$ must become identical

smoothly to obtain a sensible derivative, requiring $w_{\text{out}}^{\Delta\lambda+} \to w_{\text{out}}^{+}$ and $w_{\text{out}}^{\Delta\lambda-} \to w_{\text{out}}^{-}$.

3. Whenever w_{out}^{-} is zero, and the corresponding walker is removed from the calculation, $w_{\text{out}}^{\Delta\lambda-}$ must also be zero. If this condition is not met then the calculation at $\lambda+\Delta\lambda$, which is constrained to the use the same number of walkers as the calculation at λ, misses the future contribution from walkers that are removed from the calculation despite having a finite weight.

4. To avoid large fluctuations in the derivatives accumulated using equation (8.39), the dependence of $w_{\text{out}}^{\Delta\lambda\pm}$ on $w_{\text{in}}^{\Delta\lambda}$ must be smooth.

The simplest solution to equation (8.37) that meets all of these conditions is

$$w_{\text{out}}^{\Delta\lambda+} = \frac{w_{\text{out}}^{+}}{w_{\text{in}}} w_{\text{in}}^{\Delta\lambda} \qquad w_{\text{out}}^{\Delta\lambda-} = \frac{w_{\text{out}}^{-}}{w_{\text{in}}} w_{\text{in}}^{\Delta\lambda} , \tag{8.40}$$

$$dw_{\text{out}}^{\Delta\lambda+} = \frac{w_{\text{out}}^{+}}{w_{\text{in}}} dw_{\text{in}}^{\Delta\lambda} \qquad dw_{\text{out}}^{\Delta\lambda-} = \frac{w_{\text{out}}^{-}}{w_{\text{in}}} dw_{\text{in}}^{\Delta\lambda} , \tag{8.41}$$

giving

$$\begin{aligned} \langle w_{\text{out}}^{\Delta\lambda} \rangle &= P\frac{w_{\text{out}}^{+}}{w_{\text{in}}} w_{\text{in}}^{\Delta\lambda} + (1-P)\frac{w_{\text{out}}^{-}}{w_{\text{in}}} w_{\text{in}}^{\Delta\lambda} \\ &= (Pw_{\text{out}}^{+} + (1-P)w_{\text{out}}^{-})\frac{w_{\text{in}}^{\Delta\lambda}}{w_{\text{in}}} \\ &= w_{\text{in}}\frac{w_{\text{in}}^{\Delta\lambda}}{w_{\text{in}}} \\ &= w_{\text{in}}^{\Delta\lambda} , \end{aligned} \tag{8.42}$$

where the penultimate step uses equation (8.33). Equations (8.40) and (8.41)

thus provide a continuously differentiable branching decision which, when evaluated at $\Delta\lambda = 0$, leaves the values of the weights unchanged,

$$w^+_{\text{out}} = \frac{w^+_{\text{out}}}{w_{\text{in}}} w_{\text{in}} \qquad w^-_{\text{out}} = \frac{w^-_{\text{out}}}{w_{\text{in}}} w_{\text{in}} \,, \tag{8.43}$$

but contributes to the derivative via

$$dw^+_{\text{out}} = \frac{w^+_{\text{out}}}{w_{\text{in}}} dw_{\text{in}} \qquad dw^-_{\text{out}} = \frac{w^-_{\text{out}}}{w_{\text{in}}} dw_{\text{in}} \,, \tag{8.44}$$

producing equation (8.24).

8.4 Pseudopotentials

It is well known that using bare Coulomb potentials to describe electron-nucleus interactions produces DMC force estimators with a variance that is infinite.[78,134–136] This is a problem that is also manifest in all-electron forces calculated using AD; as the simulation time increases, the resultant AD derivatives contain an ever greater stochastic error. In the limit of an infinite simulation there is the unusual situation that the variation of the DMC total energy becomes completely smooth with respect to changes in a nuclear position, but the calculation of its derivative becomes completely undefined. Forces can be calculated with a finite error via the introduction of pseudopotentials, which remove the divergent points in the nuclear potential responsible for the infinite variance.[23,136]

8.5 Results

Figure 8.6 compares derivatives obtained from performing AD on DMC total energy evaluations and the gradient produced by smoothed spline curve fitting, with the high accuracy fit taken from the figures displayed in Section 5.4. Unless the underlying functional form of the energy curve is known, which is true only in the most trivial systems, an interpolation scheme is required to estimate the variation of the energy. Here it is shown that the application of AD provides a far more efficient route to accurate determinations of total energy derivatives; whilst a very large number of DMC calculations are needed to generate a fit with a reliable gradient, a single execution of the AD code yields the correct result. Furthermore, this observation holds for DMC calculations of arbitrary precision; improved curve fits can be obtained from more precise total energy evaluations, but, as the errors in the AD derivatives and the total energies are both proportional to $N^{-\frac{1}{2}}$, where N is the number of samples, the accuracy of the corresponding AD derivatives is also improved. Derivatives obtained from AD also compare favourably to those calculated using a finite difference method. Equation (5.10) shows that the ideal error in a stochastic finite difference derivative scales as $N^{-\frac{1}{4}}$, so as the accuracy of the DMC total energy increases, AD derivatives perform much better.

For DMC force calculations to be useful they must produce results which surpass the accuracy of those obtained from less computationally expensive DFT simulations. A high quality DFT calculation of an *sp*-bonded system typically contains an error of 0.01 bohr in the equilibrium bond lengths[21,152]

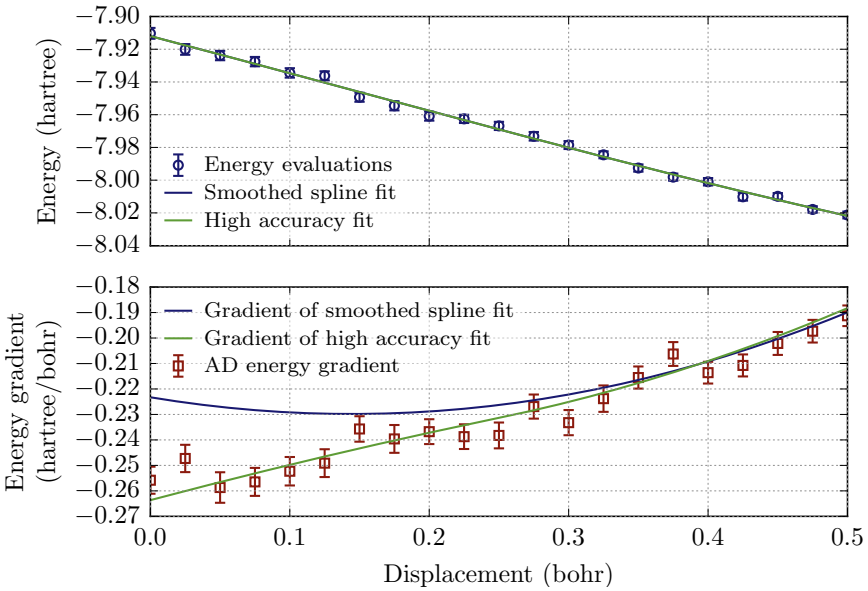


Figure 8.6: Shown in the top pane is the DMC total energy as a function of an arbitrary displacement of the carbon atom in a methane molecule. The bottom pane displays the gradient of the two fits in the top pane and the derivatives obtained by performing AD of the original DMC calculations.

which stems from the use of an approximate exchange-correlation functional. Errors in bond lengths can be converted into errors in the atomic forces via the stiffness of the bonds, obtained from experimental vibrational spectra. This approach yields an error in the carbon-carbon forces from DFT calculations of 0.003 hartree/bohr for the ethane, propane and butane molecules. Hence to demonstrate any improvement upon DFT results, the stochastic error in the DMC forces must be converged beyond this value.

Table 8.1 contains the summarised results of benchmark DMC force calculations shown in Figures 8.7, 8.8 and 8.9. All trial wave functions are composed of HF orbitals expressed in an optimised aug-cc-pVDZ-CDF Gaussian basis set[153] generated using Dirac-Fock QMC pseudopotentials.[154,155] The DMC calculations performed with these pseudopotentials, including those

Species	$\langle \sigma_F \rangle \left(\frac{\text{hartree}}{\text{bohr}}\right)$	$\left\langle \frac{a_0\ \sigma_F}{\sigma_E} \right\rangle$	$\left\langle \frac{T_{E+F}}{T_E} \right\rangle$
C_2H_6	0.0008	2.2	10.2
C_3H_8	0.0008	2.0	10.2
C_4H_{10}	0.0010	1.9	10.4

Table 8.1: Benchmark DMC force computations. The angled brackets indicate quantities averaged over 11 representative calculations, where σ_F is the error in the restoring force of a displaced carbon atom, σ_E is the error in the total energy, a_0 is the Bohr radius, T_{E+F} is the execution time required to evaluate the total energy and all three force components for every atom in the molecule, and T_E is the execution time for a total energy evaluation in isolation. Data for this Table are drawn from the results shown in Figures 8.7, 8.8 and 8.9.

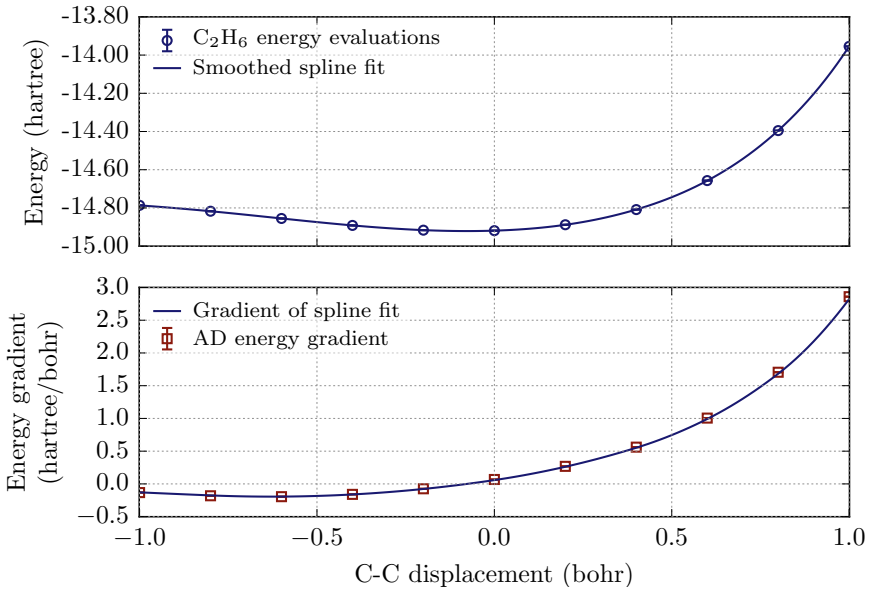


Figure 8.7: Shown in the top pane is the DMC total energy as a function of the displacement of a single carbon atom relative to its neighbouring carbon atom in an ethane molecule. The bottom pane displays the AD derivative of the energy evaluations in the top pane. The smoothed spline fit and its gradient are obtained from the average of 30 fitting procedures to independent DMC energy evaluations extending outside the range of the plot, where the resultant stochastic error is smaller than the line width.

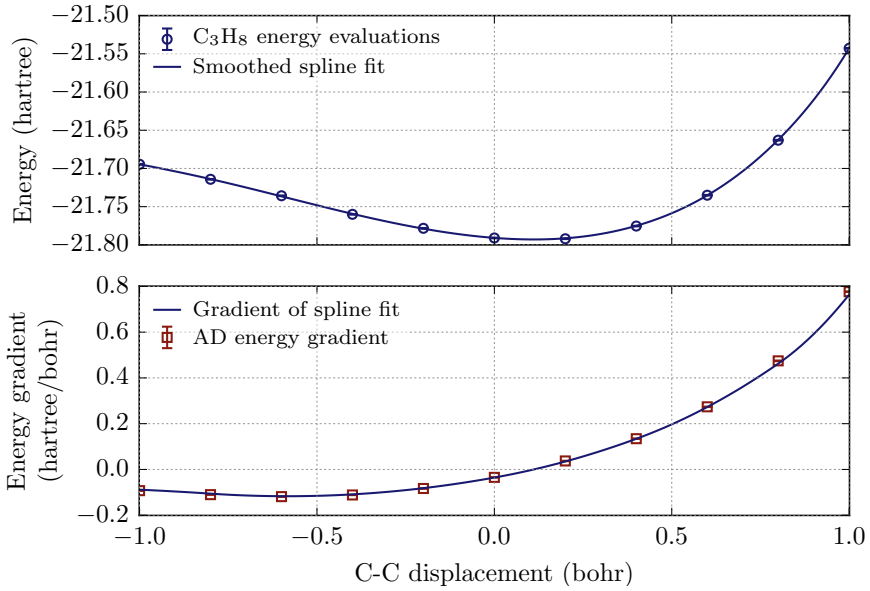


Figure 8.8: The same information as in Figure 8.7 for a propane molecule.

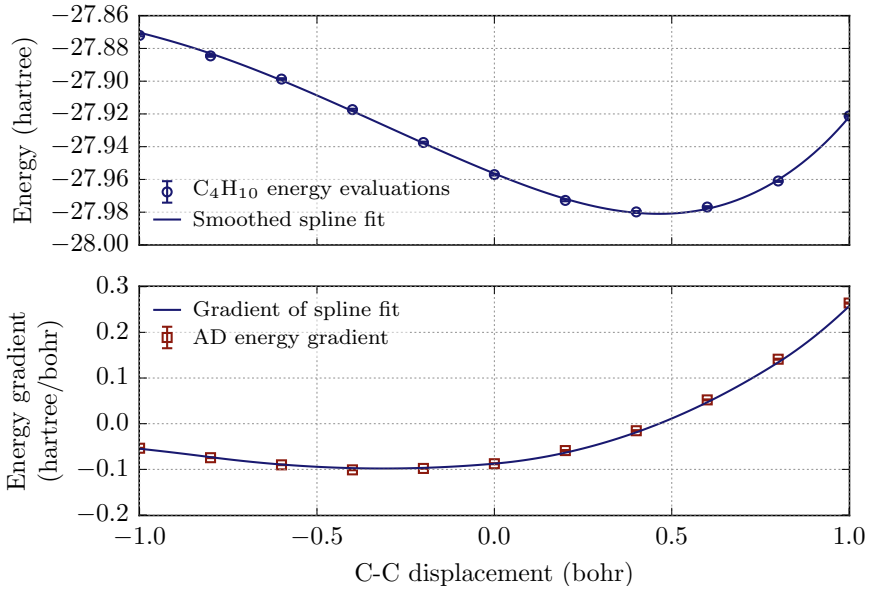


Figure 8.9: The same information as in Figure 8.7 for a butane molecule.

presented in later sections of this thesis, require the addition of T-moves into the walker dynamics in order to stabilise fluctuations in the walker populations. Each total energy evaluation and associated AD force calculation is an average of 12288 independently differentiated DMC simulations employing 60 walkers and 100 time steps of 0.01 a.u., where the errors are obtained from the standard error in the means. The second column of Table 8.1 shows the average DMC error in the restoring force of a carbon atom in each alkane as it is displaced over a range of $\pm$ 1 bohr from near its equilibrium position, representing a set of calculations typical of the relaxation of molecules and solids from a DFT geometry. These calculations are used to evaluate how the difficulty of computing AD forces scales with system size. The important metric is the relative convergence of errors in the forces compared to errors in the total energy. Displayed in atomic units, where the Bohr radius has a numerical value of one, the third column of Table 8.1 shows that the relative difficulty of converging the forces remains approximately constant as the number of electrons increases. Another important consideration is the computational complexity of the AD calculations, where the fourth column of Table 8.1 shows that the time spent computing all force components in each system, simultaneously, using reverse mode AD, is a small fixed multiple of the time spent computing a total energy, for all system sizes.

The smoothed spline fits and gradients in Figures 8.7, 8.8 and 8.9 are obtained from multiple data series extending either side of the plots. Each curve represents the average of 30 fitting procedures, where each fit was derived from independent DMC energy evaluations of the same precision as shown in the figures. The extent of the data is selected such that increasing

the range has no significant effect on the shape of the curves in the region of interest, and the stochastic error in the fits is smaller than the line width. Appendix B displays the extended fits.

The DMC calculations supporting Table 8.1 are deliberately small scale to make them as memory efficient as possible. When using a low number of walkers population control biases can become significant, but, for the DMC simulations presented here, using a larger population had little effect on the results. The individual butane derivative calculations, which involve 26 electrons, 60 walkers and 100 time steps, required 21 MB of RAM. If a larger number walkers were required then the excellent parallel scaling of the DMC algorithm allows a distribution of the memory consumption with very little effect on the run time performance. Keeping the average walker population and number of time steps fixed it would be possible to simulate 1250 electrons using less than 1 GB of memory, and 7500 electrons if the calculation were parallelised using 10 walkers per processor. However, performing multiple, small scale DMC calculations requires more computational resources to be spent on equilibrating each simulation. For the AD expressions to be accurate there must be an unbroken sequence of instructions for producing the outputs of a program from its inputs. Therefore, in contrast to how it is possible to partition a conventional DMC simulation in time, AD derivative calculations cannot begin using pre-equilibrated electronic configurations. Doing so discards important information about the relatively strong dependence of the initial configurations on the nuclear positions and the trial wave function, which would normally be accumulated in the differential quantities during the equilibration process.

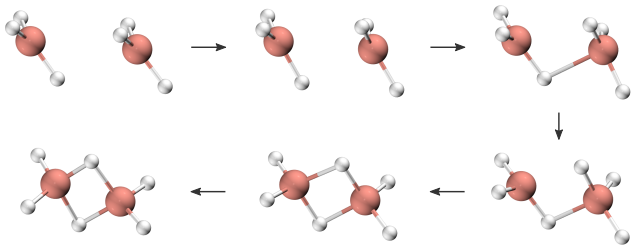


Figure 8.10: Snapshots from an *ab initio* DMC molecular dynamics simulation of the dimerisation of borane.

The columns of Table 8.1 coupled with the RAM conservation strategy indicate that this method of calculating forces can treat large systems efficiently, with an accuracy beyond the capacity of DFT. However, these force calculations make use of one spatial dimension, when a single execution of reverse mode AD yields three dimensional force components for all nuclei in the system. An application which better demonstrates the strength of DMC forces extracted using AD is a molecular dynamics simulation.

Figure 8.10 displays snapshots from a DMC molecular dynamics simulation which can be viewed in full on our website at `http://www.cmth.ph.ic.ac.uk/people/t.poole/research.html`. This simulation captures the dimerisation of borane, where two borane molecules react by forming two unusual 3-centre 2-electron bridges known as banana bonds. The dynamics of the borane molecules are determined using Verlet integration of atomic forces from DMC and the video shows 500 molecular dynamics time steps of 0.5 fs. Each time step, single-electron orbitals are generated using B3LYP[156–159] DFT with the same basis set and pseudopotential type as the simulations of the alkanes in Table 8.1. Atomic forces are then obtained from an average of 48 subsequent DMC force calculations, each employing

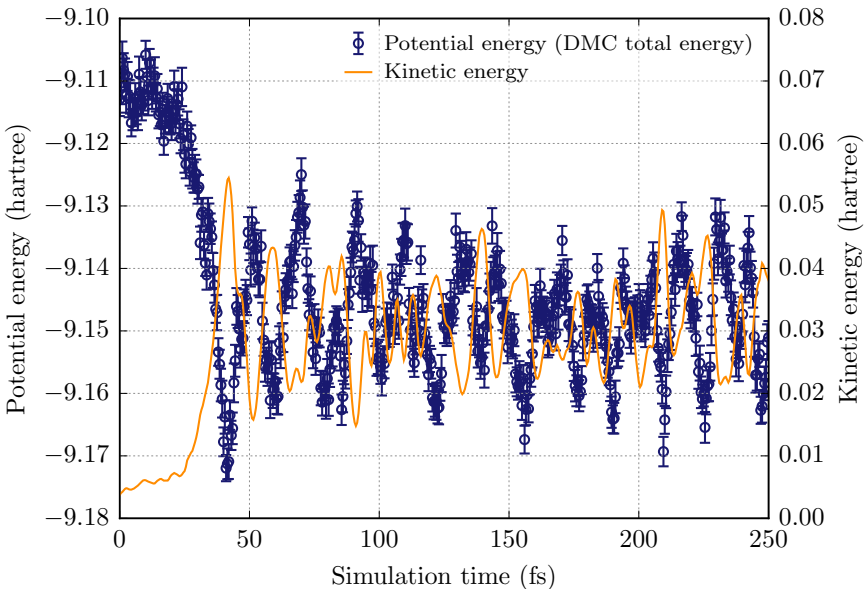


Figure 8.11: The potential and kinetic energies of the nuclei during the molecular dynamics simulation depicted in Figure 8.10.

60 walkers performing 200 time steps of 0.01 a.u. Figures 8.11 and 8.12 display the kinetic, potential and total energies in the microcanonical system. The DMC energy represents a stochastic estimate of the Born–Oppenheimer energy surface, with its gradient determining the forces on the nuclei. As previously discussed, performing molecular dynamics with forces containing stochastic noise is similar to the introduction of temperature. This noise can be formally absorbed into a Langevin thermostat,[151] but Figure 8.12 shows that, in this particular case, the stochastic fluctuations in the forces over the course of the simulation are small enough to capture the formation of the banana bonds between the two borane molecules.

Existing DMC force methods require the individual computation of each force component in the system. If these forces are used for molecular dynamics simulations the computational demand has an additional scaling with the

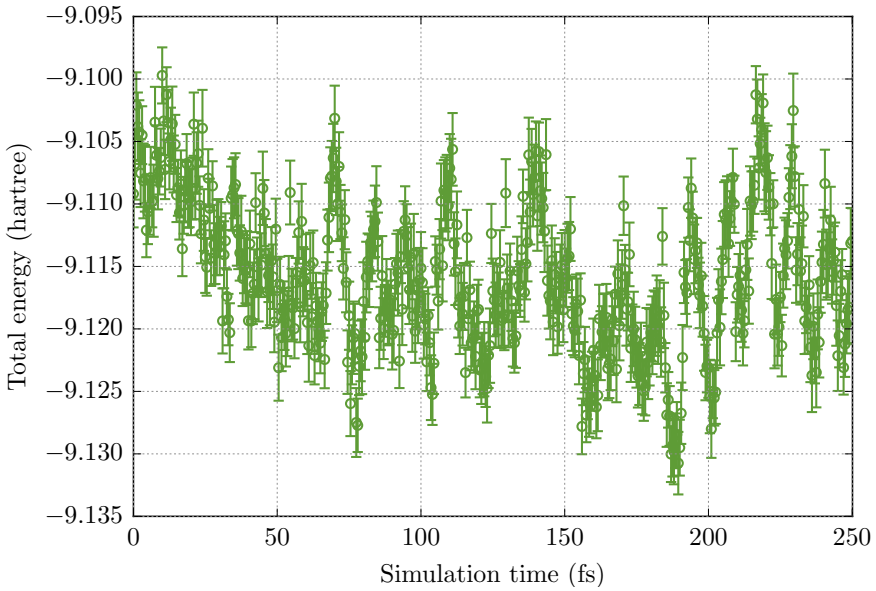


Figure 8.12: The total energy during the molecular dynamics simulation depicted in Figure 8.10.

number of atoms, making the treatment of larger systems increasingly difficult. Forces obtained from AD avoid this computational bottleneck as *all* force components are obtained simultaneously in a small multiple of the cost of a standard DMC calculation. The molecular dynamics simulation of borane is a demonstration of the efficiency of AD forces, where the average time to compute all the forces on each time step is 10.9 times that of computing a DMC total energy.

Chapter 9

Correlated sampling diffusion Monte Carlo

9.1 Introduction

The primary drawback of the AD method is that it it can be relatively difficult to apply to pre-existing programs, especially when, as is common with scientific codes, the original code has grown organically from its initial specification (if there ever was one!). Furthermore, if the implementation of AD is not purely automatic then critical sections of the code become far more difficult to maintain; any changes to the original program must be duplicated exactly in differential form to ensure that derivatives are calculated correctly. For QMC methods an alternative, far less burdensome, approach to calculating derivatives is via the technique of correlated sampling.

This chapter provides a correlated sampling method based on the phenomenon of stochastic coherence. The first section outlines conventional cor-

related sampling in VMC and its typical implementation in DMC. Section 9.3 describes how dynamical systems subjected to a common source of noise become correlated, and contains results from model systems demonstrating this effect. A mathematical treatment of the drift-diffusion equations governing the dynamics of DMC walkers shows how trajectories coupled via their diffusive components coalesce over time. This is used in the following section, which details the implementation of a correlated sampling scheme using stochastic coherence in an established DMC code. The penultimate section displays a selection of results obtained from using this method to determine the energy gradient of the symmetric stretch of a water molecule. Whilst initially promising, the correlated sampling scheme breaks down when applied to larger systems. This motivates an extrapolation of the method to a step size of zero, and the final section derives an algebraic expression for the direct calculation of the DMC total energy.

9.2 Current approaches

In VMC the application of correlated sampling is straightforward. A single reference system, defined by λ, generates a shared collection of electronic configurations distributed according to $|\Psi_{\mathrm{T}}(\bar{\boldsymbol{r}},\ \lambda)|^2$, which are used to evaluate total energies in displaced systems at $\lambda + \Delta\lambda$. Equations (3.25) and

(3.26) give the form of a correlated sampling VMC energy difference,

$$\Delta E_{\mathrm{VMC}}^{\Delta\lambda} = \frac{1}{N}\sum_{i=1}^{N}\left[E_{\mathrm{L}}(\bar{\boldsymbol{r}}_i,\ \lambda+\Delta\lambda)\frac{|\Psi_{\mathrm{T}}(\bar{\boldsymbol{r}}_i,\ \lambda+\Delta\lambda)|^2}{|\Psi_{\mathrm{T}}(\bar{\boldsymbol{r}}_i,\ \lambda)|^2} - E_{\mathrm{L}}(\bar{\boldsymbol{r}}_i,\ \lambda)\right] \tag{9.1}$$

$$= \frac{1}{N}\sum_{i=1}^{N}\left[E_{\mathrm{L}}(\bar{\boldsymbol{r}}_i,\ \lambda+\Delta\lambda)W_{\mathrm{VMC}}(\bar{\boldsymbol{r}}_i) - E_{\mathrm{L}}(\bar{\boldsymbol{r}}_i,\ \lambda)\right]\ , \tag{9.2}$$

where $\{\bar{\boldsymbol{r}}_i\}$ are drawn from $|\Psi_{\mathrm{T}}(\bar{\boldsymbol{r}},\ \lambda)|^2$ and $W_{\mathrm{VMC}}(\bar{\boldsymbol{r}}_i)$ is the weighting applied to the configurations. As $\Delta\lambda$ tends to zero the statistical fluctuations in $\Delta E_{\mathrm{VMC}}^{\Delta\lambda}$ also tend to zero, and total energy derivatives can be estimated with a finite statistical error using a finite difference approach,

$$\frac{dE_{\mathrm{VMC}}}{d\lambda} = \lim_{\Delta\lambda\to 0}\left\{\frac{\Delta E_{\mathrm{VMC}}^{\Delta\lambda}}{\Delta\lambda}\right\}\ . \tag{9.3}$$

The application of correlated sampling in DMC faces two problems which are absent in VMC. For the displaced system, the Metropolis rejection of walker moves in equation (4.25) is determined by probabilities appropriate for the original system. An additional weighting can be applied on each configuration update which corrects for the imposed Metropolis rejection,

$$w_{\mathrm{accept}} = \frac{P_{\mathrm{M}}^{\Delta\lambda}}{P_{\mathrm{M}}} \tag{9.4}$$

$$w_{\mathrm{reject}} = \frac{1-P_{\mathrm{M}}^{\Delta\lambda}}{1-P_{\mathrm{M}}}\ , \tag{9.5}$$

where P_{M} is the Metropolis rejection probability, but even for small $\Delta\lambda$ these weights fluctuate wildly over the course of simulation, making such an

approach impractical.[28] Worse, this approach breaks down completely in situations where the original calculation rejected a move, imposing a rejection in the correlated system, but $P_{\mathrm{M}}^{\Delta\lambda} = 1$. However, the Metropolis rejection becomes far less important when calculating total energy *differences*; simulations differing by only a small $\Delta\lambda$ will have almost identical time step errors, and the removal of the Metropolis rejection only increases statistical fluctuations modestly.

The other, more serious, problem is that the nodal surface of the trial wave function depends on λ for many interesting total energy derivatives such as atomic forces. Thus, finite values of $\Delta\lambda$ can place the originally sampled set of electronic configurations in different nodal pockets in the displaced systems. In principle this is allowed, as it is possible to perform DMC simulations where a small number of walkers cross nodes, but changes to the nodal surface without a corresponding change in the dynamics of the walkers can result in many electronic configurations becoming close to nodal regions. Here the local energy and drift velocity of equations (4.4) and (4.20) diverge, causing catastrophic variations in the weights of the walkers. When performing a standard DMC simulation, the importance sampling performed via the drift velocity keeps walkers away from problematic regions where $\Psi_{\mathrm{T}}(\bar{\boldsymbol{r}})$ is small.

One way of mitigating this effect is using the SWCT (space-warp coordinate transformation).[27–30] This method provides a way of generating walker dynamics in a displaced system which distort smoothly with the size of the displacement. Electron positions, $\boldsymbol{r}$, are updated so that they experience a

near rigid shift with any nearby nucleus at $\boldsymbol{R}$,

$$\boldsymbol{r}_i^{\Delta\lambda} \;=\; \boldsymbol{r}_i \;+\; \sum_{\alpha=1}^{N_n} [\, \boldsymbol{R}_\alpha^{\Delta\lambda} - \boldsymbol{R}_\alpha \,]\, w_\alpha(\boldsymbol{r}_i) \;, \tag{9.6}$$

where

$$w_\alpha(\boldsymbol{r}_i) \;=\; \frac{F(|\boldsymbol{r}_i - \boldsymbol{R}_\alpha|)}{\sum_{\beta=1}^{N_n} F(|\boldsymbol{r}_i - \boldsymbol{R}_\beta|)} \tag{9.7}$$

and

$$\sum_{\alpha=1}^{N_n} w_\alpha(\boldsymbol{r}_i) \;=\; 1 \;. \tag{9.8}$$

The function $F(r)$ is any sufficiently rapidly decaying function, commonly chosen to be r^{-4}. Such a transformation acts to keep the new paths of the walkers in the displaced system away from destabilising regions of $\Psi_{\mathrm{T}}(\bar{\boldsymbol{r}},\ \lambda + \Delta\lambda)$ near the nuclear positions. Whilst the reduction in noise is significant, the solution is not perfect; in the displaced system the weights of walkers must be accumulated over only a limited history of time steps to keep statistical fluctuations finite.

The remainder of this chapter presents an alternative method of correlating the dynamics of DMC walkers using the phenomenon of stochastic coherence. Our website, `http://www.cmth.ph.ic.ac.uk/people/t.poole/research.html`, has a number of videos supporting this chapter, and snapshots from these are displayed in Appendix C.

9.3 Stochastic coherence

Dynamical systems are well renowned for being extremely sensitive to their initial conditions. The double pendulum[160] is a classic example of how the behaviour of one of the simplest possible physical systems can exhibit completely different behaviour for very similar starting configurations. In more complex systems, involving a larger number of degrees of freedom, initially similar trajectories are almost guaranteed to diverge exponentially quickly. One quantity that characterises this instability is the Lyapunov exponent, κ, defined by

$$|\delta \boldsymbol{Z}(t)| \;\approx\; e^{\kappa t}|\delta \boldsymbol{Z}(t=0)| \;, \tag{9.9}$$

where $\delta \boldsymbol{Z}(t=0)$ is an initial infinitesimal separation between two trajectories. The first video on our website provides an example of this phenomenon, where two non-interacting, frictionless balls slide over a complex energy landscape. Initially the balls are coincident, with a tiny difference in their starting velocities. Displayed beneath the dynamics is a synchronised plot of the separation of the two balls, which shows an approximately exponential divergence in their positions. This is the key signature of a system with a positive Lyapunov exponent.

It would be reasonable to expect that adding a source of random noise to a dynamical system would increase the Lyapunov exponent. For the general case the addition of noise does indeed make the system more susceptible to divergence, but when all trajectories are subjected to identical random kicks then it is possible to observe the inverse. If the stochastic kicks from

a common source of noise exceed a certain frequency, and the trajectories occupy the same convex region of potential, initially distinct trajectories converge exponentially quickly in time.[37] In this case, irrespective of the initial conditions, the final trajectories become identical and are dependent upon the stream of random numbers only.

This stochastic coherence effect is demonstrated in the second video on our website. Here the potential landscape and initial conditions are identical to that of the first video, but every few time steps the balls are stopped and restarted with identical, randomly selected velocities. This regular restarting of the dynamics acts as a damping force which coerces the balls into regions near the minima of the potential. Whilst sharing a convex region of potential, the accelerations of the two balls act to bring the trajectories closer; both the balls will accelerate towards a local minimum in the potential, with the ball furthest from the minimum experiencing a larger acceleration. For a sufficiently short time step, the next restart of the dynamics will begin with a smaller separation between the balls.

The third video on the page illustrates the robustness of this procedure. For this simulation the balls are initially placed in distinct, well separated minima in the potential, before being subjected to the same sequence of restarts with random velocities. These random kicks provide a means for both balls to explore the potential landscape and, despite their large initial separation, the trajectories coalesce rapidly once both balls occupy the same convex region of potential.

This approach is particularly well suited to the importance sampled Langevin dynamics of equation (4.21), which generates configuration updates of the

form

$$\bar{\boldsymbol{r}}(t+\Delta t) \;=\; \bar{\boldsymbol{r}}(t) \;+\; \boldsymbol{v}(\bar{\boldsymbol{r}}(t))\Delta t \;+\; \boldsymbol{\xi}\sqrt{\Delta t}\;, \tag{9.10}$$

where $\boldsymbol{v}(\bar{\boldsymbol{r}}(t))$ is the drift velocity,

$$\boldsymbol{v}(\bar{\boldsymbol{r}}(t)) \;=\; \frac{\nabla_{\bar{\boldsymbol{r}}}\,\Psi_{\mathrm{T}}(\bar{\boldsymbol{r}}(t))}{\Psi_{\mathrm{T}}(\bar{\boldsymbol{r}}(t))}\;, \tag{9.11}$$

Δt is the DMC time step, and $\boldsymbol{\xi}$ is the diffusive component consisting of independent random electron displacements drawn from a Gaussian distribution of unit variance. Thus, each time step, each DMC walker experiences a purely random translation in configuration space which is dependent on Δt only, and exactly the same diffusive moves can be used in different DMC calculations to correlate the trajectories of the walkers without introducing a bias. Unlike the Newtonian dynamics of the previous systems, the current time step in Langevin dynamics is independent of the velocities of previous time steps, making it similar in form to Newtonian dynamics that are restarted stochastically at every time step.

The remainder of this section presents a new formulation for the divergence of correlated walkers in the DMC algorithm. Compared to equation (9.10), which governs the motion of a walker initially at $\bar{\boldsymbol{r}}(t)$, a displaced walker at $\bar{\boldsymbol{r}}(t) + \Delta\bar{\boldsymbol{r}}(t)$ experiences a configuration update of

$$\bar{\boldsymbol{r}}'(t+\Delta t) \;=\; \bar{\boldsymbol{r}}(t) \;+\; \Delta\bar{\boldsymbol{r}}(t) \;+\; \boldsymbol{v}(\bar{\boldsymbol{r}}(t)+\Delta\bar{\boldsymbol{r}}(t))\Delta t \;+\; \boldsymbol{\xi}\sqrt{\Delta t}\;. \tag{9.12}$$

Correlating the two walks by choosing identical random diffusive vectors $\boldsymbol{\xi}$

gives an expression for the divergence of the walkers with time,

$$\Delta\bar{\boldsymbol{r}}(t+\Delta t) \;=\; \bar{\boldsymbol{r}}'(t+\Delta t) \;-\; \bar{\boldsymbol{r}}(t+\Delta t) \tag{9.13}$$

$$\;=\; \Delta\bar{\boldsymbol{r}}(t) \;+\; [\,\boldsymbol{v}(\bar{\boldsymbol{r}}(t)+\Delta\bar{\boldsymbol{r}}(t)) \;-\; \boldsymbol{v}(\bar{\boldsymbol{r}}(t))\,]\Delta t\;. \tag{9.14}$$

Taking the $\Delta t \to 0$ limit to produce a time derivative, as done in equation (5.21), yields the rate of change of the divergence

$$\frac{d\Delta\bar{\boldsymbol{r}}(t)}{dt} \;=\; \boldsymbol{v}(\bar{\boldsymbol{r}}(t)+\Delta\bar{\boldsymbol{r}}(t)) \;-\; \boldsymbol{v}(\bar{\boldsymbol{r}}(t))\;, \tag{9.15}$$

which, using the Taylor expansion shown in equation (5.5) for small $\Delta\bar{\boldsymbol{r}}$, and adopting the Einstein summation convention, yields

$$\frac{d\Delta\bar{\boldsymbol{r}}_i(t)}{dt} \;=\; [\,\partial_j \boldsymbol{v}_i(\bar{\boldsymbol{r}}(t))\,]\,\Delta\bar{\boldsymbol{r}}_j(t)\;, \tag{9.16}$$

where i and j are indices running over all $3N_e$ components, with N_e the number of electrons. A more useful quantity is the rate of change of the magnitude squared of $\Delta\bar{\boldsymbol{r}}_i(t)$,

$$\frac{d[\Delta\bar{\boldsymbol{r}}_i(t)]^2}{dt} \;=\; 2\Delta\bar{\boldsymbol{r}}_i(t)\frac{d\Delta\bar{\boldsymbol{r}}_i(t)}{dt}\;, \tag{9.17}$$

which, using the definition of the drift velocity from equation (9.11) can be expressed as

$$\frac{d[\Delta\bar{\boldsymbol{r}}_i(t)]^2}{dt} \;=\; 2\Delta\bar{\boldsymbol{r}}_i\left[\partial_j \frac{\partial_i \Psi_{\mathrm{T}}(\bar{\boldsymbol{r}})}{\Psi_{\mathrm{T}}(\bar{\boldsymbol{r}})}\right]\Delta\bar{\boldsymbol{r}}_j\;. \tag{9.18}$$

If the trial wave function is a good approximation to the fixed-node ground

state, then the Langevin dynamics will be sampling configurations roughly distributed according to $|\Psi_{\rm T}(\bar{\boldsymbol{r}})|^2$. The effect of the drift velocity acting on a separation $\Delta\bar{\boldsymbol{r}}_j$ stems from the matrix $\partial_j \frac{\partial_i \Psi_{\rm T}(\bar{\boldsymbol{r}})}{\Psi_{\rm T}(\bar{\boldsymbol{r}})}$, which can be averaged over the distribution of configurations,

$$\int \Psi_{\rm T}^2(\bar{\boldsymbol{r}}) \partial_j \frac{\partial_i \Psi_{\rm T}(\bar{\boldsymbol{r}})}{\Psi_{\rm T}(\bar{\boldsymbol{r}})} d\bar{\boldsymbol{r}} \ . \tag{9.19}$$

Integration by parts over a nodal pocket of $\Psi_{\rm T}(\bar{\boldsymbol{r}})$ gives

$$\int \Psi_{\rm T}^2(\bar{\boldsymbol{r}}) \partial_j \frac{\partial_i \Psi_{\rm T}(\bar{\boldsymbol{r}})}{\Psi_{\rm T}(\bar{\boldsymbol{r}})} d\bar{\boldsymbol{r}} \ = \ -2 \int [\partial_i \Psi_{\rm T}(\bar{\boldsymbol{r}})][\partial_j \Psi_{\rm T}(\bar{\boldsymbol{r}})] d\bar{\boldsymbol{r}} \ , \tag{9.20}$$

which, when applied to an arbitrary, fixed separation $\Delta\bar{\boldsymbol{r}}_j$, produces

$$-2\Delta\bar{\boldsymbol{r}}_i(t) \int [\partial_i \Psi_{\rm T}(\bar{\boldsymbol{r}})][\partial_j \Psi_{\rm T}(\bar{\boldsymbol{r}})] d\bar{\boldsymbol{r}} \ \Delta\bar{\boldsymbol{r}}_j(t) \ = \ -2 \int [\Delta\bar{\boldsymbol{r}}_i(t) \partial_i \Psi_{\rm T}(\bar{\boldsymbol{r}})]^2 d\bar{\boldsymbol{r}} \ . \tag{9.21}$$

Equation (9.21) is strictly negative and shows that, on average, correlated walkers with a separation of $\Delta\bar{\boldsymbol{r}}$ will be driven together by the Langevin dynamics. In the previous examples of Newtonian dynamics, the trajectories coalesce via acceleration when both share a convex region of potential. For the Langevin dynamics of the DMC walkers, the trajectories are brought together in the same way by the drift velocity when sampling concave regions of $|\Psi_{\rm T}(\bar{\boldsymbol{r}})|^2$. Given that a stochastic representation of $|\Psi_{\rm T}(\bar{\boldsymbol{r}})|^2$ inevitably involves more configurations near the concave peaks of the wave function, it is intuitive that the average effect is to converge the sampling in the two correlated simulations. The fourth video on our website illustrates this process using Langevin dynamics to sample an arbitrary function, shown inverted

for easier comparison with the other videos.

9.4 Implementation

This phenomenon can be turned into a correlated sampling scheme by imposing the same diffusive configuration updates in otherwise distinct DMC calculations at λ and $\lambda + \Delta\lambda$. The appeal of this approach is that, unlike other methods, the trajectories of correlated walkers are free to evolve according to the drift velocity from a trial wave function appropriate for the displaced system, $\Psi_{\mathrm{T}}(\bar{\boldsymbol{r}},\ \lambda\ +\ \Delta\lambda)$. As the configurations sampled in each correlated calculation are not fixed, there is no requirement for a weighting scheme analogous to that of equation (9.2). This is an important advantage as the exact DMC correlated sampling weighting involves the ratio of the modulus squared of inaccessible DMC wave functions. Furthermore, even good approximations to the DMC wave functions can produce wildly varying weights if a change in λ has a significant effect on the nodal surface.

To keep correlated walkers synchronised with those in the original simulation requires imposing a common set of Metropolis move rejections. These acceptances introduce a bias in the correlated system and must be corrected via an additional weighting of the walkers. The reweighting scheme required is most easily explained as a generalisation of the scheme used in VMC correlated sampling.

The expression for a VMC correlated sampling total energy difference in equation (9.2) can be obtained from a weighting applied on each configuration update. During a VMC simulation the steady state distribution of walkers

emerges from the detailed balance condition in equation (3.11),

$$n(\bar{\boldsymbol{r}})T(\bar{\boldsymbol{r}}' \leftarrow \bar{\boldsymbol{r}})P(\bar{\boldsymbol{r}}' \leftarrow \bar{\boldsymbol{r}})d\bar{\boldsymbol{r}}d\bar{\boldsymbol{r}}' \;=\; n(\bar{\boldsymbol{r}}')T(\bar{\boldsymbol{r}} \leftarrow \bar{\boldsymbol{r}}')P(\bar{\boldsymbol{r}} \leftarrow \bar{\boldsymbol{r}}')d\bar{\boldsymbol{r}}'d\bar{\boldsymbol{r}} \;, \quad (9.22)$$

where the probability of proposing a move from $\bar{\boldsymbol{r}}$ to $\bar{\boldsymbol{r}}'$ is $T(\bar{\boldsymbol{r}}' \leftarrow \bar{\boldsymbol{r}})$, the probability of acceptance is $P(\bar{\boldsymbol{r}}' \leftarrow \bar{\boldsymbol{r}})$, and $n(\bar{\boldsymbol{r}})$ is the density of walkers at $\bar{\boldsymbol{r}}$. The equilibrium distribution is then given by

$$\frac{n(\bar{\boldsymbol{r}})}{n(\bar{\boldsymbol{r}}')} \;=\; \frac{P(\bar{\boldsymbol{r}} \leftarrow \bar{\boldsymbol{r}}')T(\bar{\boldsymbol{r}} \leftarrow \bar{\boldsymbol{r}}')}{P(\bar{\boldsymbol{r}}' \leftarrow \bar{\boldsymbol{r}})T(\bar{\boldsymbol{r}}' \leftarrow \bar{\boldsymbol{r}})} \;, \quad (9.23)$$

and the imposition of the Metropolis rejection criteria from equation (3.13),

$$P_{\mathrm{M}}(\bar{\boldsymbol{r}}' \leftarrow \bar{\boldsymbol{r}}) \;=\; \min\left(1,\; \frac{T(\bar{\boldsymbol{r}} \leftarrow \bar{\boldsymbol{r}}')\pi(\bar{\boldsymbol{r}}')}{T(\bar{\boldsymbol{r}}' \leftarrow \bar{\boldsymbol{r}})\pi(\bar{\boldsymbol{r}})}\right) \;, \quad (9.24)$$

yields an equilibrium walker density that is proportional to the probability density of the sampled wave function,

$$\frac{n(\boldsymbol{r})}{n(\boldsymbol{r}')} \;=\; \frac{|\Psi_{\mathrm{T}}(\boldsymbol{r})|^2}{|\Psi_{\mathrm{T}}(\boldsymbol{r}')|^2} \;. \quad (9.25)$$

When performing correlated sampling in VMC, the paths taken by walkers sampling the reference wave function, $\Psi_{\mathrm{T}}(\bar{\boldsymbol{r}})$, are used to evaluate a similar wave function, $\Psi_{\mathrm{T}}^{\Delta\lambda}(\bar{\boldsymbol{r}})$. The condition for detailed balance in the new system is given by

$$\begin{aligned} &n^{\Delta\lambda}(\bar{\boldsymbol{r}})T^{\Delta\lambda}(\bar{\boldsymbol{r}}' \leftarrow \bar{\boldsymbol{r}})P^{\Delta\lambda}(\bar{\boldsymbol{r}}' \leftarrow \bar{\boldsymbol{r}})d\bar{\boldsymbol{r}}'d\bar{\boldsymbol{r}} \\ &\quad= \; n^{\Delta\lambda}(\bar{\boldsymbol{r}}')T^{\Delta\lambda}(\bar{\boldsymbol{r}} \leftarrow \bar{\boldsymbol{r}}')P^{\Delta\lambda}(\bar{\boldsymbol{r}} \leftarrow \bar{\boldsymbol{r}}')d\bar{\boldsymbol{r}}d\bar{\boldsymbol{r}}' \;, \end{aligned} \quad (9.26)$$

and the inclusion of walker paths from the reference system requires a weighting of the transition rates to preserve detailed balance,

$$\begin{aligned} & n^{\Delta\lambda}(\bar{\boldsymbol{r}}) w_{\mathrm{VMC}}(\bar{\boldsymbol{r}}' \leftarrow \bar{\boldsymbol{r}}) T(\bar{\boldsymbol{r}}' \leftarrow \bar{\boldsymbol{r}}) P(\bar{\boldsymbol{r}}' \leftarrow \bar{\boldsymbol{r}}) d\bar{\boldsymbol{r}}' d\bar{\boldsymbol{r}} \\ & \quad = \; n^{\Delta\lambda}(\bar{\boldsymbol{r}}') w_{\mathrm{VMC}}(\bar{\boldsymbol{r}} \leftarrow \bar{\boldsymbol{r}}') T(\bar{\boldsymbol{r}} \leftarrow \bar{\boldsymbol{r}}') P(\bar{\boldsymbol{r}} \leftarrow \bar{\boldsymbol{r}}') d\bar{\boldsymbol{r}} d\bar{\boldsymbol{r}}' \; . \end{aligned} \tag{9.27}$$

As the walkers in the correlated system must accept moves according to the Metropolis rejection criteria of those in the reference system, the steady state solution is given by

$$\frac{n^{\Delta\lambda}(\bar{\boldsymbol{r}})}{n^{\Delta\lambda}(\bar{\boldsymbol{r}}')} \;=\; \frac{w_{\mathrm{VMC}}(\bar{\boldsymbol{r}} \leftarrow \bar{\boldsymbol{r}}')}{w_{\mathrm{VMC}}(\bar{\boldsymbol{r}}' \leftarrow \bar{\boldsymbol{r}})} \frac{|\Psi_{\mathrm{T}}(\bar{\boldsymbol{r}})|^2}{|\Psi_{\mathrm{T}}(\bar{\boldsymbol{r}}')|^2} \;=\; \frac{|\Psi_{\mathrm{T}}^{\Delta\lambda}(\bar{\boldsymbol{r}})|^2}{|\Psi_{\mathrm{T}}^{\Delta\lambda}(\bar{\boldsymbol{r}}')|^2} \; , \tag{9.28}$$

which, adopting the notation

$$\Psi_{\mathrm{T\,ratio}}(\bar{\boldsymbol{r}}' \leftarrow \bar{\boldsymbol{r}}) \;=\; \Psi_{\mathrm{T}}(\bar{\boldsymbol{r}}')/\Psi_{\mathrm{T}}(\bar{\boldsymbol{r}}) \; , \tag{9.29}$$

returns a weighting of

$$\frac{w_{\mathrm{VMC}}(\bar{\boldsymbol{r}}' \leftarrow \bar{\boldsymbol{r}})}{w_{\mathrm{VMC}}(\bar{\boldsymbol{r}} \leftarrow \bar{\boldsymbol{r}}')} \;=\; \frac{|\Psi_{\mathrm{T\,ratio}}^{\Delta\lambda}(\bar{\boldsymbol{r}}' \leftarrow \bar{\boldsymbol{r}})|^2}{|\Psi_{\mathrm{T\,ratio}}(\bar{\boldsymbol{r}}' \leftarrow \bar{\boldsymbol{r}})|^2} \; . \tag{9.30}$$

This weighting is simple to incorporate into a correlated sampling VMC algorithm: initially, each walker has a unit weight which is updated on a transition from $\bar{\boldsymbol{r}}$ to $\bar{\boldsymbol{r}}'$ (determined by the dynamics of the reference system) via

$$w_{\mathrm{VMC}}(\bar{\boldsymbol{r}}') \;=\; \frac{|\Psi_{\mathrm{T\,ratio}}^{\Delta\lambda}(\bar{\boldsymbol{r}}' \leftarrow \bar{\boldsymbol{r}})|^2}{|\Psi_{\mathrm{T\,ratio}}(\bar{\boldsymbol{r}}' \leftarrow \bar{\boldsymbol{r}})|^2} \, w_{\mathrm{VMC}}(\bar{\boldsymbol{r}}) \; . \tag{9.31}$$

Equation (9.31) produces a weighting that is equivalent to that of conven-

tional correlated sampling schemes shown in equation (9.2),

$$\frac{w_{\mathrm{VMC}}(\bar{\boldsymbol{r}}')}{w_{\mathrm{VMC}}(\bar{\boldsymbol{r}})} = \frac{|\Psi^{\Delta\lambda}_{\mathrm{T\,ratio}}(\bar{\boldsymbol{r}}' \leftarrow \bar{\boldsymbol{r}})|^2}{|\Psi_{\mathrm{T\,ratio}}(\bar{\boldsymbol{r}}' \leftarrow \bar{\boldsymbol{r}})|^2} \tag{9.32}$$

$$= \frac{W_{\mathrm{VMC}}(\bar{\boldsymbol{r}}')}{W_{\mathrm{VMC}}(\bar{\boldsymbol{r}})} \ . \tag{9.33}$$

A similar, though more involved, approach can be applied to the DMC algorithm. Here walkers in both the new and reference systems are free to propose moves based on the correct wave function for each system, with the acceptance of a move from $\bar{\boldsymbol{r}}$ to $\bar{\boldsymbol{r}}'$ in the new system determined by the acceptance of a corresponding move from $\bar{\boldsymbol{r}}''$ to $\bar{\boldsymbol{r}}'''$ in the reference. With a one-to-one mapping of paths in the reference and correlated systems the detailed balance condition is given by

$$\begin{aligned} & n^{\Delta\lambda}(\bar{\boldsymbol{r}}) w_{\mathrm{DMC}}(\bar{\boldsymbol{r}}' \leftarrow \bar{\boldsymbol{r}}) T^{\Delta\lambda}(\bar{\boldsymbol{r}}' \leftarrow \bar{\boldsymbol{r}}) P(\bar{\boldsymbol{r}}''' \leftarrow \bar{\boldsymbol{r}}'') d\bar{\boldsymbol{r}}' d\bar{\boldsymbol{r}} \\ & \quad = n^{\Delta\lambda}(\bar{\boldsymbol{r}}') w_{\mathrm{DMC}}(\bar{\boldsymbol{r}} \leftarrow \bar{\boldsymbol{r}}') T^{\Delta\lambda}(\bar{\boldsymbol{r}} \leftarrow \bar{\boldsymbol{r}}') P(\bar{\boldsymbol{r}}'' \leftarrow \bar{\boldsymbol{r}}''') d\bar{\boldsymbol{r}} d\bar{\boldsymbol{r}}' \end{aligned} \tag{9.34}$$

and the application of the Metropolis rejection criteria from the reference system yields

$$\begin{aligned} \frac{n^{\Delta\lambda}(\bar{\boldsymbol{r}})}{n^{\Delta\lambda}(\bar{\boldsymbol{r}}')} &= \frac{w_{\mathrm{DMC}}(\bar{\boldsymbol{r}} \leftarrow \bar{\boldsymbol{r}}')}{w_{\mathrm{DMC}}(\bar{\boldsymbol{r}}' \leftarrow \bar{\boldsymbol{r}})} \frac{T^{\Delta\lambda}(\bar{\boldsymbol{r}} \leftarrow \bar{\boldsymbol{r}}')}{T^{\Delta\lambda}(\bar{\boldsymbol{r}}' \leftarrow \bar{\boldsymbol{r}})} \frac{T(\bar{\boldsymbol{r}}''' \leftarrow \bar{\boldsymbol{r}}'')}{T(\bar{\boldsymbol{r}}'' \leftarrow \bar{\boldsymbol{r}}''')} \frac{|\Psi_{\mathrm{T}}(\bar{\boldsymbol{r}}'')|^2}{|\Psi_{\mathrm{T}}(\bar{\boldsymbol{r}}''')|^2} \\ &= \frac{|\Psi^{\Delta\lambda}_{\mathrm{T}}(\bar{\boldsymbol{r}})|^2}{|\Psi^{\Delta\lambda}_{\mathrm{T}}(\bar{\boldsymbol{r}}')|^2} \ . \end{aligned} \tag{9.35}$$

The required weightings of the transitions in the new system are thus given by

$$\frac{w_{\mathrm{DMC}}(\bar{\boldsymbol{r}}' \leftarrow \bar{\boldsymbol{r}})}{w_{\mathrm{DMC}}(\bar{\boldsymbol{r}} \leftarrow \bar{\boldsymbol{r}}')} = \frac{P^{\Delta\lambda}_{\mathrm{M}}(\bar{\boldsymbol{r}}' \leftarrow \bar{\boldsymbol{r}})}{P_{\mathrm{M}}(\bar{\boldsymbol{r}}''' \leftarrow \bar{\boldsymbol{r}}'')} \ , \tag{9.36}$$

which, in a similar way to equation (9.31), can be applied using

$$w_{\text{accept}} = \frac{P_{\text{M}}^{\Delta\lambda}(\bar{\boldsymbol{r}}' \leftarrow \bar{\boldsymbol{r}})}{P_{\text{M}}\ (\bar{\boldsymbol{r}}''' \leftarrow \bar{\boldsymbol{r}}'')} \tag{9.37}$$

$$w_{\text{reject}} = 1\ , \tag{9.38}$$

on each configuration update decision. This procedure produces a more stable weighting scheme than that of equations (9.4) and (9.5), and one which is immune from zeros in the numerator or denominator.

Despite this improvement, the inclusion of these new weights still results in an algorithm that is too noisy to be useful. One solution is to limit the fractional change in the weight introduced by equation (9.37) in a similar way to the drift velocity in equation (4.31). However, whilst this approach improves the performance of the correlated sampling scheme, the simulations presented in this chapter are performed without Metropolis rejection in the reference calculations.

Keeping the correlated and reference calculations synchronised also requires identical walker branching decisions in the two simulations. Correlated walkers entering the branching algorithm of equation (4.14),

$$N_{\text{spawn}} = w_{\text{out}} = \texttt{INT}(w_{\text{in}} + \eta)\ , \tag{9.39}$$

have a weight of $w_{\text{in}} + \Delta w_{\text{in}}$, where w_{in} is the weight of the reference walker and Δw_{in} is the accumulated weight difference between the correlated and reference systems. This can be handled as in Section 8.3 by introducing a fractional change of the weight in the correlated system commensurate with

the fractional change of the weight in the reference system,

$$w_{\text{out}}^{\Delta\lambda} \;=\; \frac{w_{\text{out}}}{w_{\text{in}}}\, w_{\text{in}}^{\Delta\lambda} \;, \tag{9.40}$$

where the output weight $w_{\text{out}}^{\Delta\lambda}$ is split evenly among the $N_{\text{spawn}} = w_{\text{out}}$ walkers such that each spawned walker has a weight of

$$\begin{aligned} w_{\text{spawn}}^{\Delta\lambda} \;&=\; \frac{\frac{w_{\text{out}}}{w_{\text{in}}}(w_{\text{in}} \;+\; \Delta w_{\text{in}})}{N_{\text{spawn}}} \\ &= 1 \;+\; \frac{\Delta w_{\text{in}}}{w_{\text{in}}} \;. \end{aligned} \tag{9.41}$$

Averaging the effect of this splitting over multiple branching events,

$$\begin{aligned} \langle w_{\text{out}}^{\Delta\lambda} \rangle \;&=\; [w_{\text{in}} \;-\; \texttt{INT}(w_{\text{in}})]\frac{\texttt{INT}(w_{\text{in}}) \;+\; 1}{w_{\text{in}}}[w_{\text{in}} \;+\; \Delta w_{\text{in}}] \\ &\qquad +\; [1 \;-\; (w_{\text{in}} \;+\; \texttt{INT}(w_{\text{in}}))]\frac{\texttt{INT}(w_{\text{in}})}{w_{\text{in}}}[w_{\text{in}} \;+\; \Delta w_{\text{in}}] \\ &=\; w_{\text{in}} \;+\; \Delta w_{\text{in}} \;, \end{aligned} \tag{9.42}$$

shows that this procedure preserves the expected value of the correlated weight at each point in the simulation.

The simplest possible route to a working correlated sampling algorithm is to store the state of the pseudorandom number generator at the start of the reference calculation and the result of each subsequent stochastic decision made over the course of a simulation. A correlated calculation can then be performed by setting the pseudorandom number generator to the same initial state as the reference calculation and selecting the results of any stochastic decisions using the store of data. Random numbers are still generated for the

predetermined decisions to ensure parity between the two simulations, where identical random numbers in each simulation mean that the diffusive configuration updates and the orientation of the pseudopotential integration grids can be reproduced exactly in the correlated system. As integration grids are only required to evaluate the non-local components of any pseudopotentials, which have a fixed cut-off distance from the nucleus, it is possible to arrive at situations where grids are constructed in only one of the correlated or reference calculations; if the vector $\Delta\boldsymbol{r}$, giving the separation of an electron in the original calculation and the corresponding electron in the correlated calculation, crosses the surface of a sphere at the non-local cut-off distance from a pseudopotential nucleus, then only a single calculation requires non-local components. For well designed pseudopotentials the effect on the total energy should be small, but the random number generators lose their synchronisation. The fix with the smallest storage overhead is to generate random numbers for integration grids irrespective of electron-nucleus distances in both the reference and correlated simulations.

A similar problem is encountered when a configuration update which is accepted in the reference calculation forces a walker in the correlated system to cross the nodal surface of $\Psi_{\mathrm{T}}^{\Delta\lambda}(\bar{\boldsymbol{r}})$. Whilst this clearly violates the fixed-node approximation, allowing a small number of walkers to transfer to different nodal pockets should not have a significant effect on the total energy. The more damaging consequence, in terms of a correlated sampling algorithm, is that the divergent drift velocity in regions where the wave function is small could act to separate reference and correlated walkers. This effect can be mitigated by limiting either $\Delta\boldsymbol{r}$ or the *difference* in the drift

velocities in the same manner as the drift velocity itself. Such an approach ensures that the effect of the limiting vanishes as the correlated and reference calculations become identical, and provides a mechanism where walkers that have crossed the nodal surface are biased towards returning to their original nodal pocket. The expression for the limiting from equation (4.31) is given by

$$\boldsymbol{x}_{\mathrm{lim}}(\bar{\boldsymbol{r}}) \;=\; \frac{\sqrt{1 \;+\; 2a|\boldsymbol{x}(\bar{\boldsymbol{r}})|^2\Delta t} \;-\; 1}{a|\boldsymbol{x}(\bar{\boldsymbol{r}})|^2\Delta t}\,\boldsymbol{x}(\bar{\boldsymbol{r}}) \;, \tag{9.43}$$

where $\boldsymbol{x}(\bar{\boldsymbol{r}})$ is the limited quantity. Compared to drift velocity limiting, where a is typically unity, here a must be selected to represent a length scale appropriate for $\boldsymbol{x}(\bar{\boldsymbol{r}})$.

As with the AD approach discussed in Chapter 8, the efficient storage of data is key to the performance of the correlated sampling algorithm. Rather than LIFO stacks, the restoration of correlated sampling data requires first-in, first-out (FIFO) stores which function very similarly; data is initially stored in RAM and, when the capacity of the memory is exceeded, the stores save the contents held in memory to disk. An ideal simulation would be able to perform an entire DMC calculation without the stores exceeding the available RAM and, again, the easy parallelisation of Monte Carlo methods allows the memory requirements to be split over multiple nodes of HPC machines. However, in this case, there is no requirement to perform completely independent parallel DMC calculations; a single equilibrated distribution of walkers can be used as the starting point for multiple subsequent calculations using different streams of random numbers. The total memory demand is given by

Per time step

Per walker

* The weight
* The number of walkers spawned

Per electron:

· If the move was accepted

· *The move acceptance probability and result*

· *The position*

· *The drift velocity*

where the items in italics are only necessary if Metropolis rejection reweighting or a type of drift velocity or separation limiting are required.

9.5 Results

Figure 9.1 displays total energy derivative calculations using correlated sampling finite differences. All Slater-Jastrow trial wave functions for the DMC total energy evaluations are based on HF orbitals using the same optimised aug-cc-pVDZ-CDF Gaussian basis set and Dirac-Fock QMC pseudopotentials as the AD calculations on the alkanes in Chapter 8. The Slater determinant of each trial wave function is generated using GAMESS with Jastrow factors optimised for each O-H separation with CASINO. Each energy evaluation is produced by a simulation containing 1200 walkers performing 120000 time steps of 0.02 a.u. The finite difference derivatives use a step size of 0.004

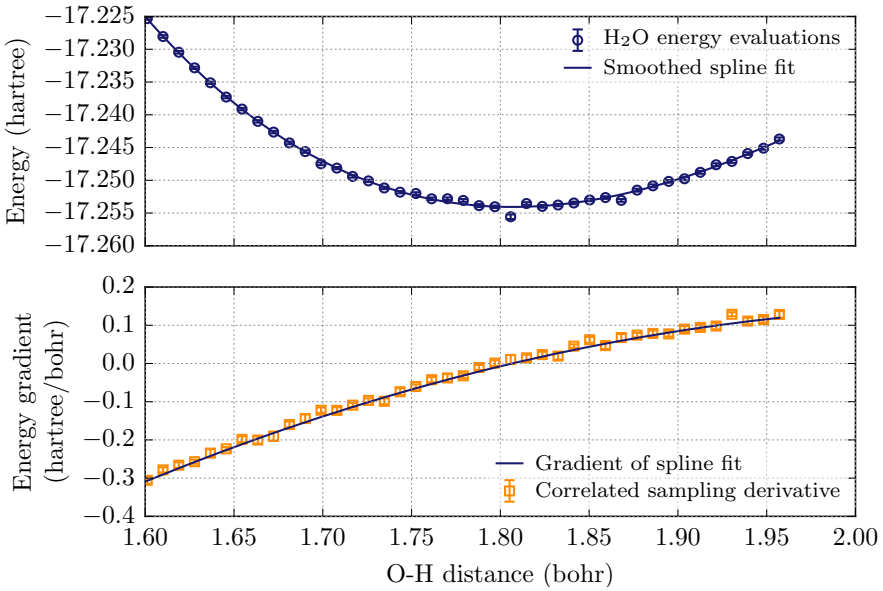


Figure 9.1: The top pane displays the variation in the DMC total energy of a water molecule as a function of the oxygen-hydrogen distance during a symmetric stretch. Shown in the bottom pane are derivatives calculated using finite difference correlated sampling calculations with a step size of 0.004 bohr.

bohr, where the Slater determinant component of the wave function is regenerated at the perturbed geometry. In the absence of correlated sampling the resultant errors in the derivatives are an order of magnitude larger, with the error reduction increasing as the step size decreases. For this particular system, the calculation of derivatives using step sizes smaller than 0.0004 bohr suffers from numerical noise, where the precision of the energy difference decreases and the denominator of equation (9.3) amplifies any discrepancies.

The correlated sampling simulations of Figure 9.1 are performed without any form of limiting described by equation (9.43). Restricting differences in drift velocity or walker separation has no noticeable effect on the results, with the stochastic coherence being sufficient to keep the correlated walker synchronised with the reference trajectories. In this case the total memory

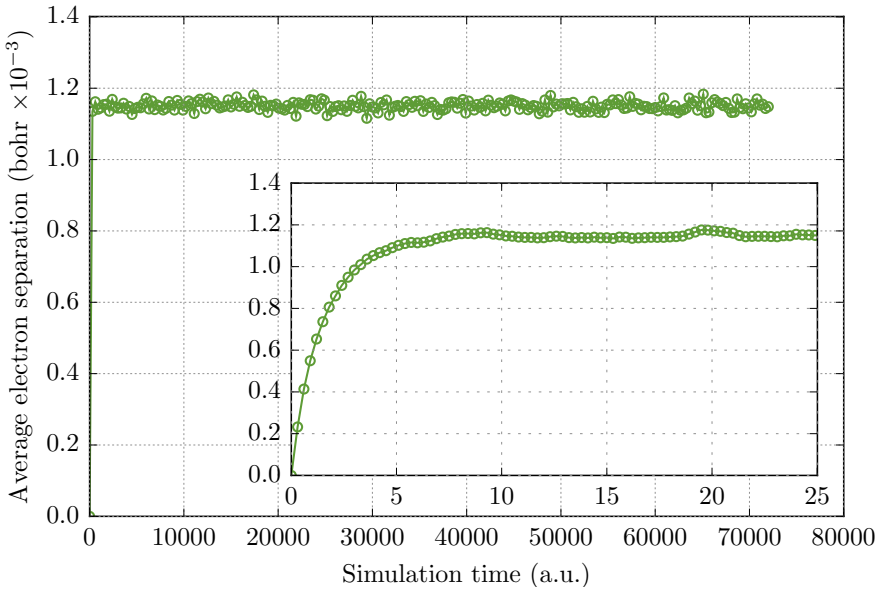


Figure 9.2: This figure shows the separation of the correlated and reference trajectories averaged over all electrons and all walkers on each time step of a DMC simulation of a water molecule. The geometry is taken from figure 9.1 with an initial O-H separation of 1.66 bohr and a correlated displacement of 0.002 bohr. Each simulation contains 1200 walkers performing 3600000 time steps of 0.02 a.u. The inset zooms in on the start of the simulation.

requirement for each of the correlated sampling calculations in Figure 9.1 is 6.34 GB, which is trivially distributed over multiple nodes of an HPC cluster.

Figure 9.2 displays the stochastic coherence between correlated and reference trajectories for an H_2O system taken from Figure 9.1. Here an O-H displacement of 0.002 bohr is used for the correlated perturbation and the number of DMC time steps is increased to 3600000. The first 10 a.u. of simulation time shows an exponentially fast convergence to a state where the average electron separation assumes a constant system and perturbation dependent value, giving the average effect of displacing the nuclei on all electron trajectories. As with the total energy evaluations of Figure 9.1, no limiting of drift velocities or electron separations is required to maintain the

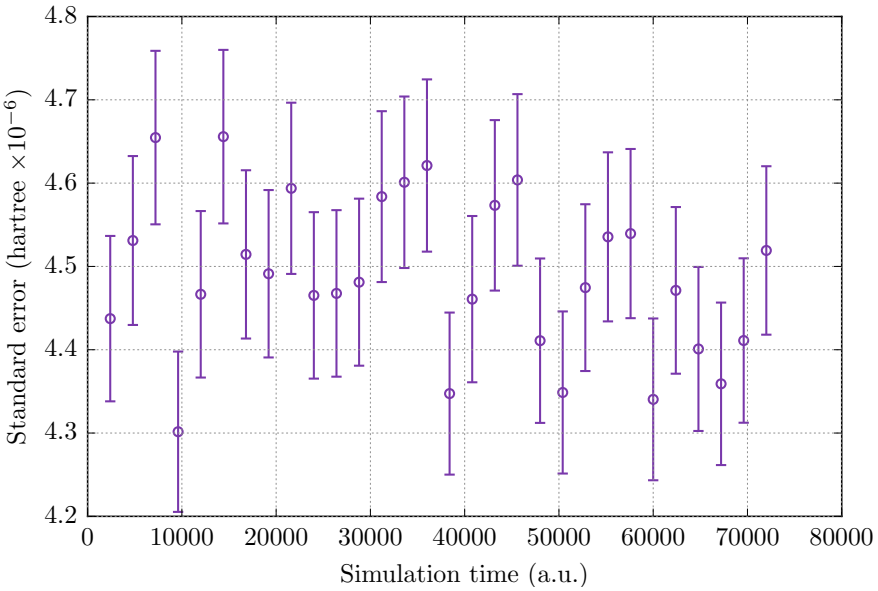


Figure 9.3: The variation of the standard error in the correlated sampling total energy difference computed in Figure 9.2 over a limited history of local energies taken from the previous 2400 a.u. of simulation time.

correlation shown in Figure 9.2.

A feature of existing DMC correlated sampling schemes is an increasing variance in the estimate of a total energy difference with simulation time due to the weighting of electronic configurations. This effect is mitigated by computing the weights, which approximate the ratio of the DMC wave functions in each system, using information obtained over only a limited history of time steps. In simulations using stochastic coherence to correlate walkers this weighting is not required, and the variance of energy difference estimators remains constant over the course of a simulation. Figure 9.3 displays the variation in the standard error of the energy difference computed during the simulation producing Figure 9.2. Here each standard error is obtained from the difference in local energies of the previous 2400 a.u. of simulation

time, showing that with increasing simulation time the fluctuations in the correlated energy difference remain constant.

Despite these promising results for a single water molecule, correlated sampling DMC simulations of more complex systems are problematic. When using this technique to study periodic, high pressure hydrogen structures, it is observed that walkers lose synchronisation catastrophically a short way into a correlated calculation. Removing the periodic boundary conditions, so that the system becomes a collection of 8 closely spaced pseudo-hydrogen atoms, shows no significant improvement in performance. In this case limiting either the drift velocity difference or the electron separations delays the point of catastrophic failure, but even very tightly limited simulations eventually fail. The same behaviour is found for a system containing more than a single water molecule, indicating that this instability is not system specific.

Logging proposed configuration updates that cross the nodal surface of the trial wave function in both the correlated and reference systems reveals no clear link between node crossings and decorrelation, but it seems possible that the divergence of the drift velocities in regions near the nodes is responsible. A better correlated sampling strategy would be to propagate the reference and correlated walkers simultaneously. This would allow potentially problematic configuration updates to be re-proposed in the reference system, stabilising the calculation at the expense of a small time step error. Such an approach would also eliminate the substantial memory requirement of larger correlated sampling DMC calculations. However, the small set of established DMC codes are all ill suited to handling two concurrent trial wave functions and, as the trial wave function is used throughout the DMC algorithm, it is

difficult to manually encapsulate it into a self contained quantity.

It is also possible that the wave functions of larger systems are simply too complex for stochastic coherence to work. Equation (9.21) shows that *on average* correlated trajectories are driven together, but this may not be sufficient to guarantee convergence. As the dimensionality increases, the probability of finding a divergent degree of freedom in the dynamics at each time step also increases. Thus, maintaining correlated dynamics becomes far more difficult, and the cohesive action of the synchronised noise is insufficient to converge the trajectories of the correlated walkers. Further work is required to better understand the failure of the correlated sampling scheme, and to determine if, via some careful approximations, the method can be successfully applied to larger systems.

9.6 Extrapolation to zero step size

The finite difference approach of the previous section can be used to formulate a direct differentiation of the DMC algorithm by considering the divergence of two correlated walkers with similar trial wave functions as the finite difference step size tends to zero. Equations (9.11) and (9.14) show that, when coupled by the diffusive component of their dynamics, the divergence of two trajectories is governed solely by the difference between the correlated, $\Psi_{\mathrm{T}}^{\Delta\lambda} = \Psi_{\mathrm{T}}(\lambda + \Delta\lambda,\ \bar{\boldsymbol{r}})$, and reference, $\Psi_{\mathrm{T}}(\lambda,\ \bar{\boldsymbol{r}})$, trial wave functions. As $\Delta\lambda$ tends to zero the correlated total energy tends to the reference total energy

and exact derivatives can be calculated using

$$\left.\frac{dE_{\mathrm{DMC}}}{d\lambda}\right|_{\lambda} \;=\; \lim_{\Delta\lambda\to 0}\left\{\frac{E_{\mathrm{DMC}}(\lambda+\Delta\lambda) \;-\; E_{\mathrm{DMC}}(\lambda)}{\Delta\lambda}\right\} \;. \tag{9.44}$$

Each time step the divergence, $\Delta\bar{\boldsymbol{r}}$, of two initially coincident correlated walkers is given by

$$\begin{aligned} \Delta\bar{\boldsymbol{r}}_0 \;&=\; 0 \\ \Delta\bar{\boldsymbol{r}}_{n+1} \;&=\; \Delta\bar{\boldsymbol{r}}_n + \left[\; \boldsymbol{v}\left(\lambda+\Delta\lambda,\; \bar{\boldsymbol{r}}_n+\Delta\bar{\boldsymbol{r}}_n\right) - \boldsymbol{v}\left(\lambda,\; \bar{\boldsymbol{r}}_n\right) \;\right] \Delta t \;. \end{aligned} \tag{9.45}$$

For small $\Delta\lambda$ a Taylor expansion of the first term in the square brackets yields

$$\Delta\bar{\boldsymbol{r}}_{n+1} \;=\; \Delta\bar{\boldsymbol{r}}_n + \left[\; \left.\frac{\partial\boldsymbol{v}}{\partial\lambda}\right|_{\lambda,\bar{\boldsymbol{r}}_n} \Delta\lambda \;\;+\;\; \left.(\boldsymbol{\nabla}\otimes\boldsymbol{v})^T\right|_{\lambda,\bar{\boldsymbol{r}}_n} \cdot \Delta\bar{\boldsymbol{r}}_n \;\right] \Delta t \;, \tag{9.46}$$

generating an expression for the variation of the walker paths with respect to λ as $\Delta\lambda \to 0$,

$$\begin{aligned} \frac{d\bar{\boldsymbol{r}}_0}{d\lambda} \;&=\; 0 \\ \frac{d\bar{\boldsymbol{r}}_{n+1}}{d\lambda} \;&=\; \frac{d\bar{\boldsymbol{r}}_n}{d\lambda} \;+\; \left[\; \left.\frac{\partial\boldsymbol{v}}{\partial\lambda}\right|_{\lambda,\bar{\boldsymbol{r}}_n} \;+\; \left.(\boldsymbol{\nabla}\otimes\boldsymbol{v})^T\right|_{\lambda,\bar{\boldsymbol{r}}_n} \cdot \frac{d\bar{\boldsymbol{r}}_n}{d\lambda} \;\right] \Delta t \;. \end{aligned} \tag{9.47}$$

The divergence of the correlated walks causes a corresponding divergence in the weights accumulated along each trajectory. Conventional weight accumulation is governed by

$$G_w(\bar{\boldsymbol{r}}' \leftarrow \bar{\boldsymbol{r}},\; \Delta t) \;=\; \exp\left(-\frac{1}{2}[E_{\mathrm{L}}(\bar{\boldsymbol{r}}') \;+\; E_{\mathrm{L}}(\bar{\boldsymbol{r}}) \;-\; 2E_{\mathrm{T}}]\Delta t\right) \;, \tag{9.48}$$

from equation (4.27). Working with the natural logarithm of the weight, μ_i, gives expressions that are amenable to a treatment similar to that performed on the walker dynamics,

$$\begin{aligned} \mu_0 &= 0 \\ \mu_{n+1} &= \mu_n - \frac{\Delta t}{2}\left[\, E_L\left(\lambda,\ \bar{\boldsymbol{r}}_{n+1}\right) + E_L\left(\lambda,\ \bar{\boldsymbol{r}}_n\right) - 2E_{Tn} \,\right] . \end{aligned} \tag{9.49}$$

By applying the same energy shifts to both the reference and correlated walks, the divergence of the weights is given by

$$\begin{aligned} \Delta\mu_0 &= 0 \\ \Delta\mu_{n+1} &= \Delta\mu_n - \frac{\Delta t}{2}\Big[\, E_L\left(\lambda+\Delta\lambda,\ \bar{\boldsymbol{r}}_{n+1}+\Delta\bar{\boldsymbol{r}}_{n+1}\right) - E_L\left(\lambda,\ \bar{\boldsymbol{r}}_{n+1}\right) \\ &\qquad + E_L\left(\lambda+\Delta\lambda,\ \bar{\boldsymbol{r}}_n+\Delta\bar{\boldsymbol{r}}_n\right) - E_L\left(\lambda,\ \bar{\boldsymbol{r}}_n\right) \,\Big] \end{aligned} \tag{9.50}$$

where a Taylor expansion of the terms inside the square brackets produces

$$\begin{aligned} \Delta\mu_{n+1} = \Delta\mu_n - \frac{\Delta t}{2}\Bigg[& \left.\frac{\partial E_L}{\partial\lambda}\right|_{\lambda,\bar{\boldsymbol{r}}_{n+1}} \Delta\lambda + \boldsymbol{\nabla} E_L|_{\lambda,\bar{\boldsymbol{r}}_{n+1}} \cdot \Delta\bar{\boldsymbol{r}}_{n+1} \\ & + \left.\frac{\partial E_L}{\partial\lambda}\right|_{\lambda,\bar{\boldsymbol{r}}_n} \Delta\lambda + \boldsymbol{\nabla} E_L|_{\lambda,\bar{\boldsymbol{r}}_n} \cdot \Delta\bar{\boldsymbol{r}}_n \Bigg] . \end{aligned} \tag{9.51}$$

The variation of the accumulated weight with λ is thus given by

$$\begin{aligned}
\frac{d\mu_0}{d\lambda} &= 0 \\
\frac{d\mu_{n+1}}{d\lambda} &= \frac{d\mu_n}{d\lambda} - \frac{\Delta t}{2}\left[\left.\frac{\partial E_L}{\partial \lambda}\right|_{\lambda,\bar{\boldsymbol{r}}_{n+1}} + \boldsymbol{\nabla} E_L|_{\lambda,\bar{\boldsymbol{r}}_{n+1}} \cdot \frac{d\bar{\boldsymbol{r}}_{n+1}}{d\lambda} \right. \\
&\qquad \left. + \left.\frac{\partial E_L}{\partial \lambda}\right|_{\lambda,\bar{\boldsymbol{r}}_n} + \boldsymbol{\nabla} E_L|_{\lambda,\bar{\boldsymbol{r}}_n} \cdot \frac{d\bar{\boldsymbol{r}}_n}{d\lambda} \right],
\end{aligned} \tag{9.52}$$

which, expressed in terms of the weights, is

$$\begin{aligned}
\frac{dw_0}{d\lambda} &= 0 \\
\frac{dw_{n+1}}{d\lambda} &= w_{n+1}\left(\frac{1}{w_n}\frac{dw_n}{d\lambda} - \frac{\Delta t}{2}\left[\left.\frac{\partial E_L}{\partial \lambda}\right|_{\lambda,\bar{\boldsymbol{r}}_{n+1}} + \left.\frac{\partial E_L}{\partial \lambda}\right|_{\lambda,\bar{\boldsymbol{r}}_n} \right.\right. \\
&\qquad \left.\left. + \boldsymbol{\nabla} E_L|_{\lambda,\bar{\boldsymbol{r}}_{n+1}} \cdot \frac{d\bar{\boldsymbol{r}}_{n+1}}{d\lambda} + \boldsymbol{\nabla} E_L|_{\lambda,\bar{\boldsymbol{r}}_n} \cdot \frac{d\bar{\boldsymbol{r}}_n}{d\lambda} \right] \right).
\end{aligned} \tag{9.53}$$

Equations (9.47) and (9.53) can now be used to generate an expression for the derivative of the DMC total energy of equation (4.30),

$$\begin{aligned}
\frac{dE_{\mathrm{DMC}}}{d\lambda} &= \frac{d}{d\lambda}\left(\frac{\sum_{i=1}^{N} E_{\mathrm{L}}(\bar{\boldsymbol{r}}_i)\, w_i}{\sum_{j=1}^{N} w_j} \right) \\
&= \frac{\sum_{i=1}^{N}\left(\left[\left.\frac{\partial E_L}{\partial \lambda}\right|_{\lambda,\bar{\boldsymbol{r}}_i} + \boldsymbol{\nabla} E_L|_{\lambda,\bar{\boldsymbol{r}}_i} \cdot \frac{d\bar{\boldsymbol{r}}_i}{d\lambda} \right] w_i + \left[E_L(\lambda,\, \bar{\boldsymbol{r}}_i) - E_{\mathrm{DMC}} \right] \frac{dw_i}{d\lambda} \right)}{\sum_{j=1}^{N} w_j}
\end{aligned} \tag{9.54}$$

which can be manipulated into the form of a typical DMC property estimator,

$$\frac{dE_{\mathrm{DMC}}}{d\lambda} = \frac{\sum_{i=1}^{N}\left(\left.\frac{\partial E_L}{\partial \lambda}\right|_{\lambda,\bar{\boldsymbol{r}}_i} + \boldsymbol{\nabla} E_L|_{\lambda,\bar{\boldsymbol{r}}_i} \cdot \frac{d\bar{\boldsymbol{r}}_i}{d\lambda} + \left[E_L(\lambda,\, \bar{\boldsymbol{r}}_i) - E_{\mathrm{DMC}} \right] \frac{d\mu_i}{d\lambda} \right) w_i}{\sum_{j=1}^{N} w_j}. \tag{9.55}$$

Whilst the introduction of local energy and drift velocity limiting increases the complexity of equation (9.55), its fundamental form remains tractable. The final component of the derivative calculation is the propagation of differential information through walker branching events, which can be handled as described in Section 8.3.

The implementation of this scheme in an established DMC code is not straightforward, as the derivatives of the walker weights require the gradient of the local energy. Second derivatives of the trial wave function are used to evaluate the electronic kinetic energy, but third derivatives, required for the gradient of the kinetic energy, are not part of the standard DMC algorithm. The extraction of these derivatives is a task well suited to AD, which can be applied to the evaluation of the local energy without requiring a comprehensive overhaul of established DMC software packages. Whilst the derivatives required to evaluate equation (9.55) are available from the bespoke DMC algorithm developed for the application of AD in Chapter 8, the direct differentiation strategy outlined in this section has not been rigorously tested.

Chapter 10

Conclusions

10.1 Summary

The important properties of small atoms and molecules are determined by the quantum mechanical interactions of the electrons and nuclei. Extremely accurate predictions of these properties can often be obtained using the post-Hartree–Fock methods of quantum chemistry, but the computational cost of these methods scales particularly poorly with system size, making applications to collections of molecules impracticable. Furthermore, despite recent progress in this area,[69–71] the treatment of solids using quantum chemistry methods remains problematic. There is, therefore, a fairly hard limit to the predictive power of quantum chemistry, whereas the extended quantum properties of large and periodic systems underpin many of the important fundamental processes of chemistry, physics and biology.

An alternative, and comparably accurate, method of determining electronic structure is provided by QMC methods. Compared to quantum chem-

istry approaches the computational cost of QMC scales relatively modestly with system size, allowing the quantum treatment of systems which are orders of magnitude larger. Furthermore, QMC methods are inherently easy to parallelise, making them far more suitable for modern HPC architectures. Despite this significant advantage, and the ability to treat solids, a more widespread adoption of QMC methods has been hindered by a lack of accurate and efficient atomic forces, which are required for efficient structure relaxation and molecular dynamics simulations. Without the ability to calculate forces it is not possible to obtain many temperature dependent or bulk properties which are essential to modelling a system on a larger, physically realisable scale.

This thesis provides a new, efficient procedure for calculating exact derivatives within QMC methods, using the programming technique of AD. By applying chain rule differentiation to the source code underlying a computed function, AD allows access to derivatives of functions that are too complex to tackle algebraically. In particular, the reverse mode of AD can obtain derivatives with a spectacular efficiency when treating algorithms with a large number of inputs, as all the derivatives of a single selected program output can be obtained, simultaneously, in a small multiple of the time taken to compute the output itself. However, a straightforward application of AD to the DMC algorithm produces derivatives that neglect the important contribution from the stochastic decisions inherent to Monte Carlo methods.

The output from algorithms which use random numbers to make probabilistic decisions typically contains discontinuities, as, for a fixed stream of random numbers, changes to an input parameter can affect the outcome

of the decisions. This causes step changes in the output, where either side of the discontinuity represents a calculation where a different decision was made. Taking a numerical derivative of the output, using AD or any other approach, neglects the important contribution to the desired, macroscopic gradient from the discontinuous jumps. The macroscopic gradient is obtained from differentiating the expectation value of the function, which is shown to differ from to the expectation value of a numerical derivative. Thus, even in the limit where a stochastic function has an infinite length, a straightforward differentiation cannot capture the gradient of the expectation value of the function.

This thesis contains a strategy for calculating derivatives of stochastic functions which incorporates the effect of probabilistic decisions. Applying this approach to the AD expressions corresponding to the DMC algorithm yields exact and efficient estimations of DMC atomic forces. These forces are validated by performing DMC force calculations involving 26 electrons in a non-trivial system of non-local pseudopotentials, which demonstrate that applications to even larger systems containing thousands of electrons could be possible. The efficiency of the force calculations also enables DMC molecular dynamics simulations, making it possible to watch chemical reactions with unprecedented accuracy and determine properties of multiple, dynamically interacting molecules. These developments increase the utility and scope of DMC simulations by providing new functionality, and will spur the adoption of QMC methods by a far larger circle of the scientific community.

The biggest drawback to the calculation of DMC derivatives using AD is the difficulty of an implementation. To calculate exact derivatives AD

must be applied to the entirety of the DMC algorithm which, if not initially designed for this procedure, can be a prohibitively complex process. Additionally, following a successful implementation of AD, subsequent software development can be far more burdensome.

This thesis also explores a different way of calculating derivatives of DMC calculations using correlated sampling and the phenomenon of stochastic coherence. The primary benefit of this method is that an incorporation into existing DMC codes requires only minor modifications. However, the rather significant drawback is that the proposed correlated sampling approach breaks down for systems larger than a single water molecule. An alternative approach is the direct calculation of derivatives obtained from taking the limit of a correlated sampling step size to zero. This process produces iterative expressions for calculating estimates of derivatives on each time step of a DMC simulation, allowing derivatives to be treated on the same footing as conventional DMC property estimators. Whilst these expressions necessitate higher order derivatives of functions than those typically required by the DMC algorithm, the calculation of these derivatives is primarily an exercise in algebra.

10.2 Further work

Having demonstrated the potential of DMC force calculations using AD, an obvious extension of the work in this thesis is to apply the method to bigger and more interesting systems. Any effects which rely upon the correlated action of multiple electrons, such as van der Waals interactions, are diffi-

cult to capture using cheaper, single-electron methods like HF or DFT, and become more important at system sizes too large to be treated with the multi-electron methods of quantum chemistry. There exists many approximate extensions to DFT which incorporate dispersion effects,[161–174] but these typically require empirical information or first-principles reference data from calculations beyond the original level of theory. This makes QMC methods the best suited tools for obtaining high accuracy results in solids or large molecules where dispersion interactions play a major role in determining the properties of the system.

Whilst this thesis has concentrated on the calculation of atomic forces, the application of reverse mode AD simultaneously provides total energy derivatives with respect to *all* other simulation parameters. This presents a unique opportunity to optimise the trial wave function within DMC using the derivatives of the total energy with respect to its basis set and basis set coefficients. Compared to conventional wave function optimisation in VMC, which by construction focusses on improving the wave function where its magnitude is large, an optimisation in DMC would target the nodal surface directly, allowing a more systematic improvement of the DMC total energy. However, conventional gradient based optimisation strategies are ill suited to applications with stochastic errors in the derivatives and, compounding the problem, the errors in the derivatives do not go to zero with the magnitude of the derivatives. Structural relaxations using DMC forces face exactly the same difficulties, with previous determinations of equilibrium geometries employing a scheme, based on Bayesian inference, which uses evaluations of the total energy only.[175] This scheme can be extended to incorporate

stochastic energy gradients, allowing optimisations of systems with a large number of degrees of freedom, but a working implementation is yet to be demonstrated.

Another useful quantity obtained automatically from reverse mode AD is the total energy derivative with respect to the DMC time step. To obtain highly accurate DMC properties it is necessary to extrapolate results to a time step of zero size,[73] where the associated time step error vanishes. This procedure typically involves fitting a trend to multiple DMC calculations performed using different time steps. Whilst it is unlikely that the computational overhead of AD would make this process any more efficient in isolation, any gradient calculations provide information about the variation of the total energy with the time step for zero extra cost.

As discussed in Section 9.5, the correlated sampling technique presented in this thesis may be improved if walkers in the correlated system are able to evolve in parallel with those in the reference system. However, the direct differentiation scheme derived in the following section has the potential to be far more useful, as repeated evaluations of correlated calculations are no longer required to calculate energy gradients with multiple degrees of freedom. All the functionality required to implement and test this method is provided by the AD expressions used for the DMC force calculations in this thesis. If accurate and reliable derivatives can be obtained this way, then integration into existing QMC software packages is relatively straightforward, providing an alternate route to atomic forces without the implementation and maintenance burden of AD.

Appendix A

The diffusion Monte Carlo code

A.1 Obtaining the code

The DMC code used to perform the AD force calculations presented in this thesis can be obtained from the author, `thomas.poole10@imperial.ac.uk`, on receipt of a valid CASINO developers licence.

A.2 Code information

Total number of commits: 951 (average 2.7 commits per active day)

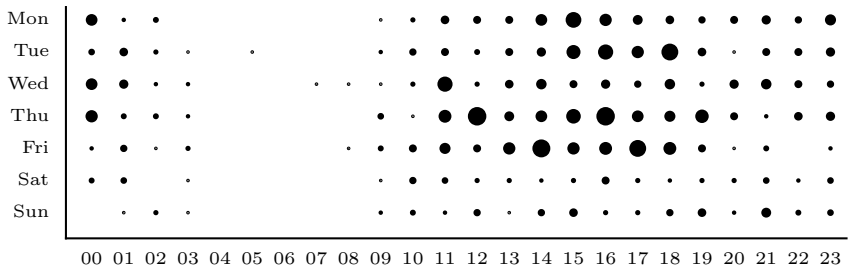


Figure A.1: A representation of the relative number of commits made during each hour of the week.

Total lines of (non-library) code: 28832

- 87% Fortran 90
- 13% Python

Libraries used:

- dSFMT:[176,177] double precision SIMD-oriented Fast Mersenne Twister pseudorandom number generator
- Tapenade:[143] automatic AD engine

Dependencies:

- An MPI implementation: a message passing interface for parallelisation
- GAMESS:[148,149] to generate trial wave functions
- CASINO:[12] to optimise trial wave functions

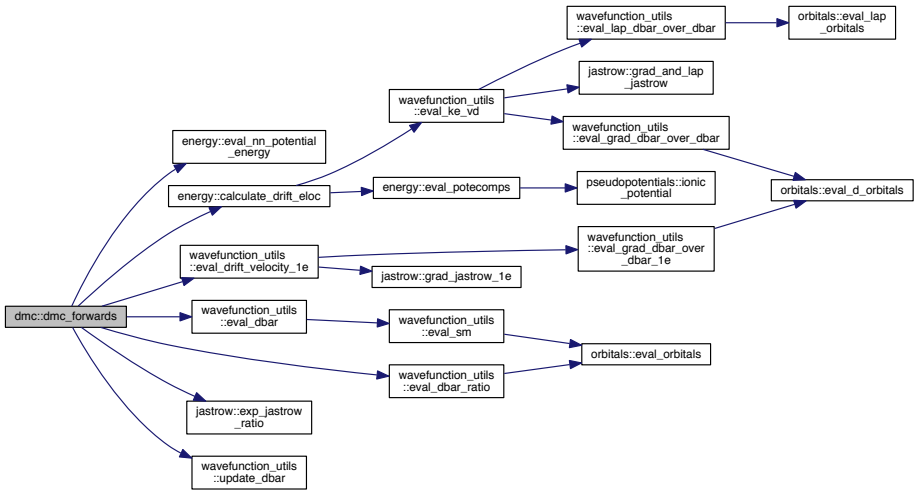


Figure A.2: A call graph of the significant functions in the forwards pass of the DMC code (a duplicate of Figure 8.2).

A.3 Function definitions

The functions shown in Figure A.2 are separated into a hierarchy of modules dependent upon they type of information they provide. Each function is specified by a module name, either `dmc`, `energy`, `wavefunction_utils`, `jastrow`, `pseudopotentials` or `orbitals`, followed by a double colon and the function name. Terminal functions are those which directly use the inputs to the DMC algorithm, including the nuclear positions, the orbital coefficients, and the Jastrow factor parameters.

The orbital evaluation functions, `eval_orbitals`, `eval_d_orbitals` and `eval_lap_orbitals`, are all supplied with an integer specifying an electron and a position in space. In turn they calculate the orbital value, the gradient of the orbital with respect the electron position, and the Laplacian of the or-

bital with respect the electron position. These quantities are predominantly used in combination with the $\bar{D}_{ji}$ matrix of Section 4.4.3, which provides an efficient means of updating electronic configurations and calculating the local energy and drift velocity of equations (4.4) and (4.20) respectively.

The orbital values propagate through a Slater determinant in `eval_sm` to the initial evaluation of the $\bar{D}_{ji}$ matrix in `eval_dbar`. After the first time step of the DMC simulation, updates to the electronic configuration trigger corresponding updates to $\bar{D}_{ji}$ via `update_dbar` and `eval_dbar_ratio`. Here the latter function is combined with `exp_jastrow_ratio` to obtain the ratio of trial wave functions either side of a single electron move, which is required for the Metropolis rejection.

Orbital gradients and Laplacians feed into expressions for the drift velocity and local energy via ratios of $\bar{D}_{ji}$ matrices[107] computed in the functions `eval_lap_dbar_over_dbar` and `eval_grad_dbar_over_dbar`, with the function `eval_grad_dbar_over_dbar_1e` treating single elections. The calculation of the single electron drift velocity in `eval_drift_velocity_1e` requires the gradient of the Jastrow factor from `grad_jastrow_1e`, in addition to the determinental component from the orbitals. The same is true for the efficient evaluation of the electronic kinetic energy in `eval_ke_vd`, which necessitates a previous evaluation of the drift velocity. Thus, both the gradient and Laplacian of the Jastrow factor are required from `grad_and_lap_jastrow`, and evaluations of the local energy return the drift velocity at no extra cost. In addition to the kinetic energy, the local energy in `calculate_drift_eloc` also requires the electron-electron and electron-nucleus potential energies, obtained from `eval_potecomps` and `ionic_potential` respectively, with the

latter handling the contributions from any pseudopotentials. The final energy component comes from `eval_nn_potential_energy`, which is evaluated once to determine the static interaction energy between the nuclei.

Appendix B

Extended energy curve fits

This appendix displays the extent of the fits to DMC energies displayed in Figures 8.7, 8.8 and 8.9.

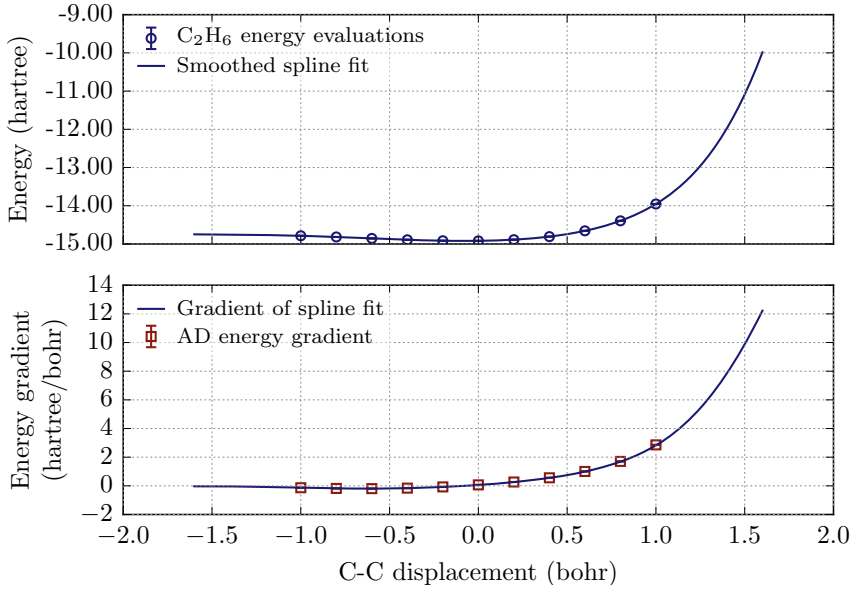


Figure B.1: The full extent of the average fit and gradient of 30 independent series of DMC total energies for the ethane molecule in Figure 8.7. Each series was composed of 17 total energy evaluations of the same precision as the data series shown, spaced evenly over the range of the fit.

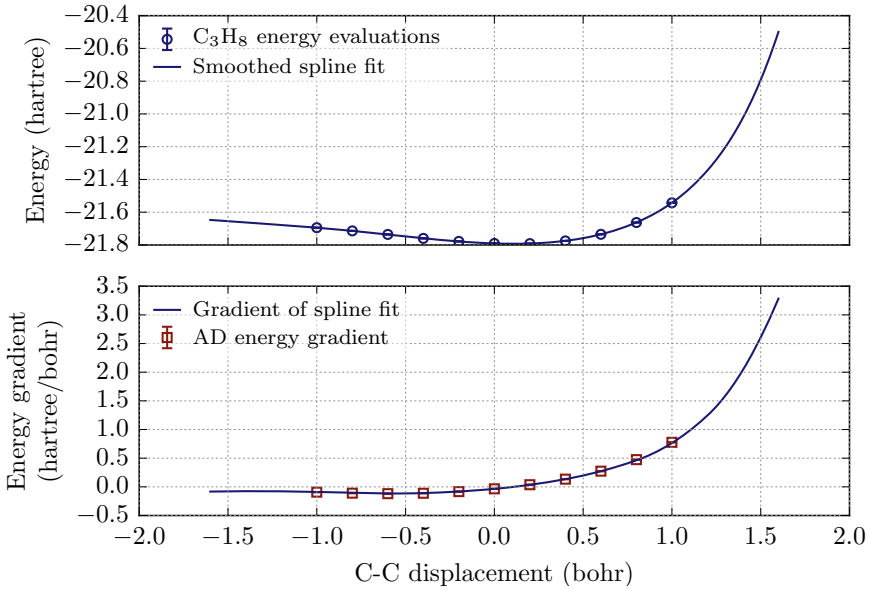


Figure B.2: The same information as in Figure B.1 for the propane molecule in Figure 8.8.

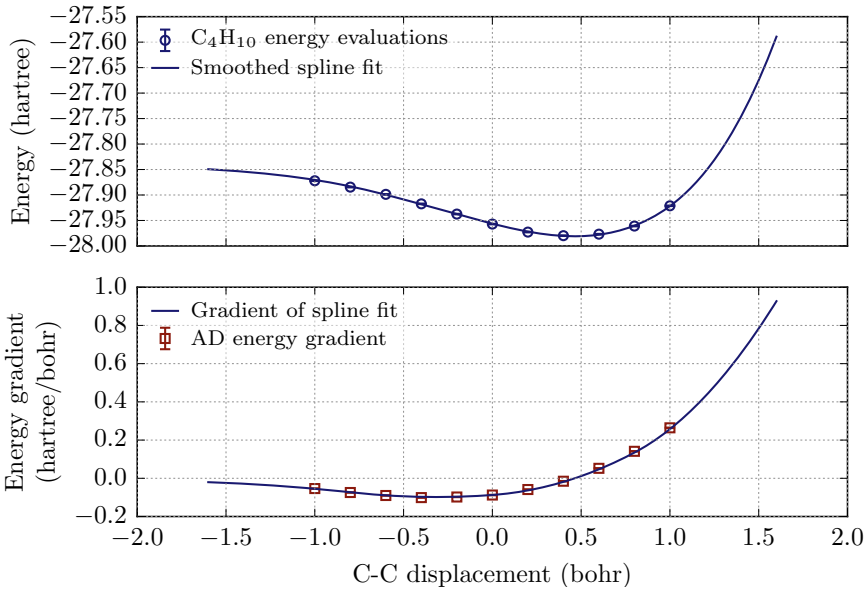


Figure B.3: The same information as in Figure B.1 for the butane molecule in Figure 8.9.

Appendix C

Stochastic coherence examples

This appendix contains snapshots from the videos displayed on our website, `http://www.cmth.ph.ic.ac.uk/people/t.poole/research.html`.

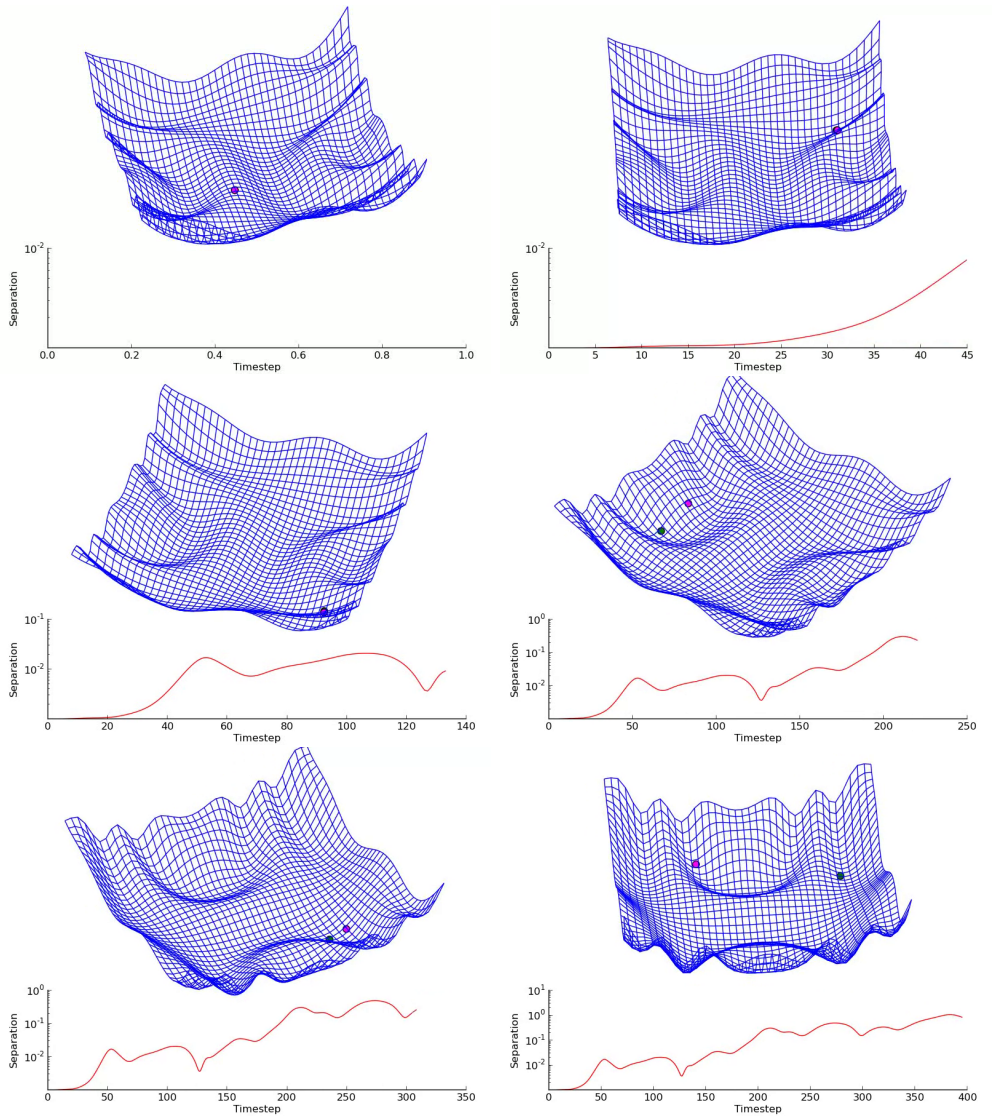


Figure C.1: Snapshots from a simulation of two non-interacting, frictionless balls sliding over an energy landscape. The balls are initially coincident, with a tiny difference in their starting velocities. Below the visualisation of the dynamics is a plot of the logarithmic distance between the balls.

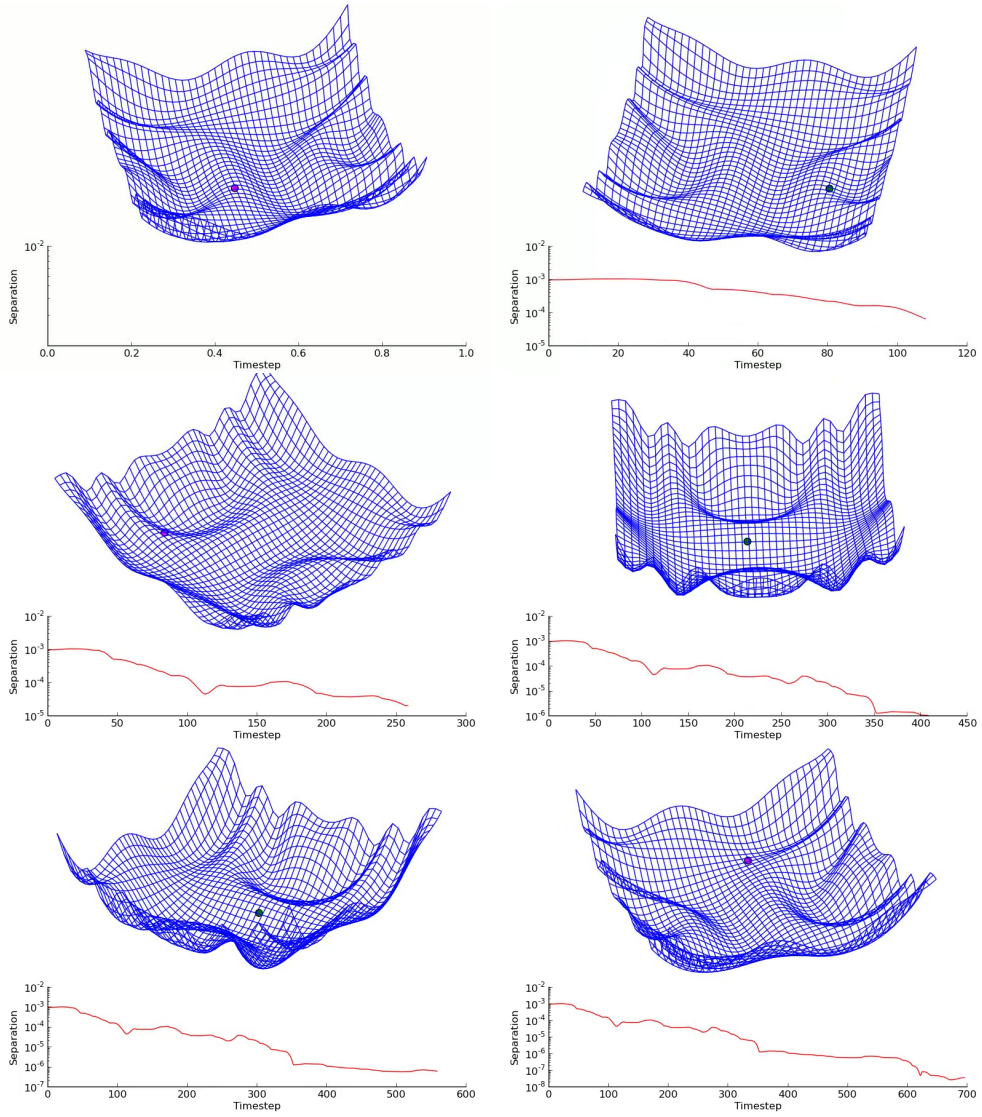


Figure C.2: Snapshots from a simulation identical to that shown in Figure C.1 but where the balls are stopped every few time steps and restarted with identical, randomly selected velocities.

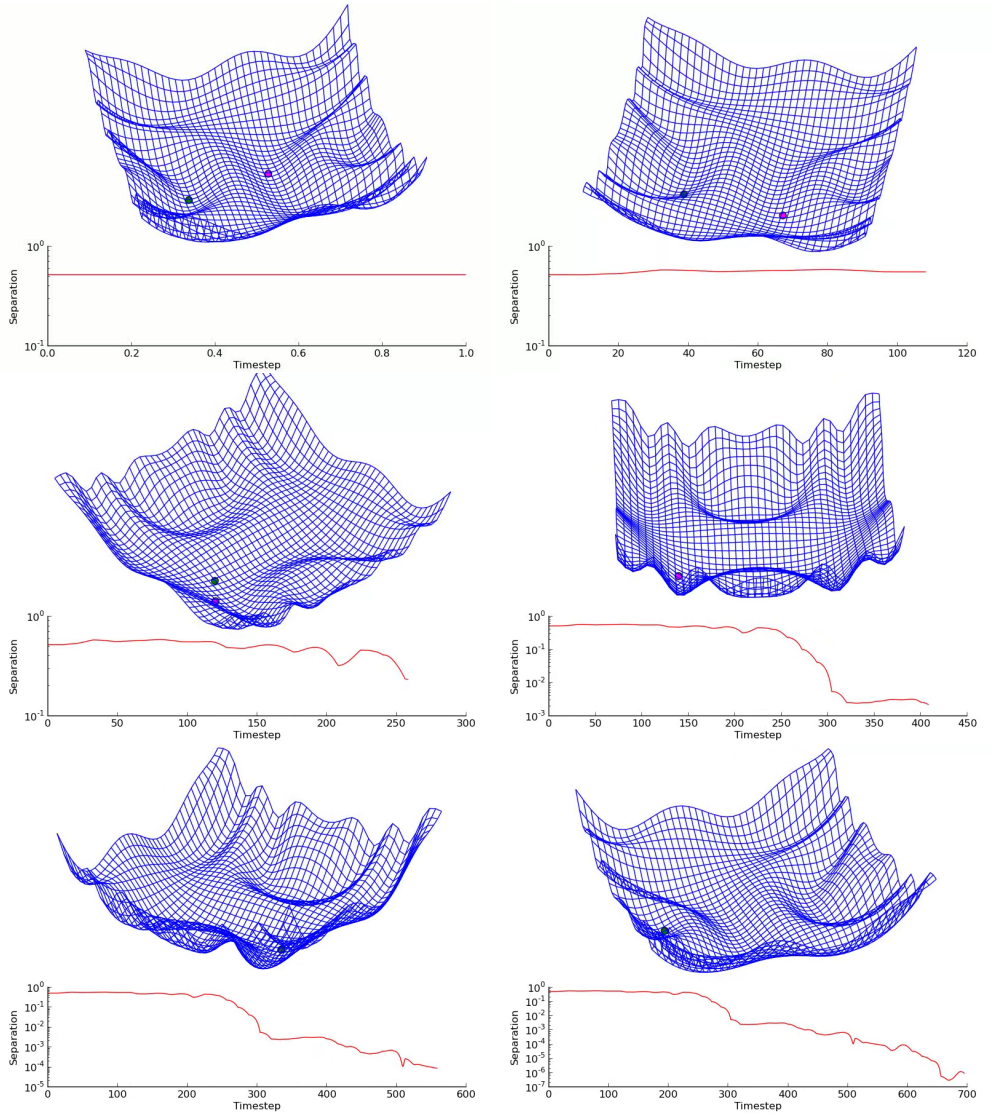


Figure C.3: Snapshots from a simulation identical to that shown in Figure C.2 but where the balls are initially placed in different, well separated minima in the potential.

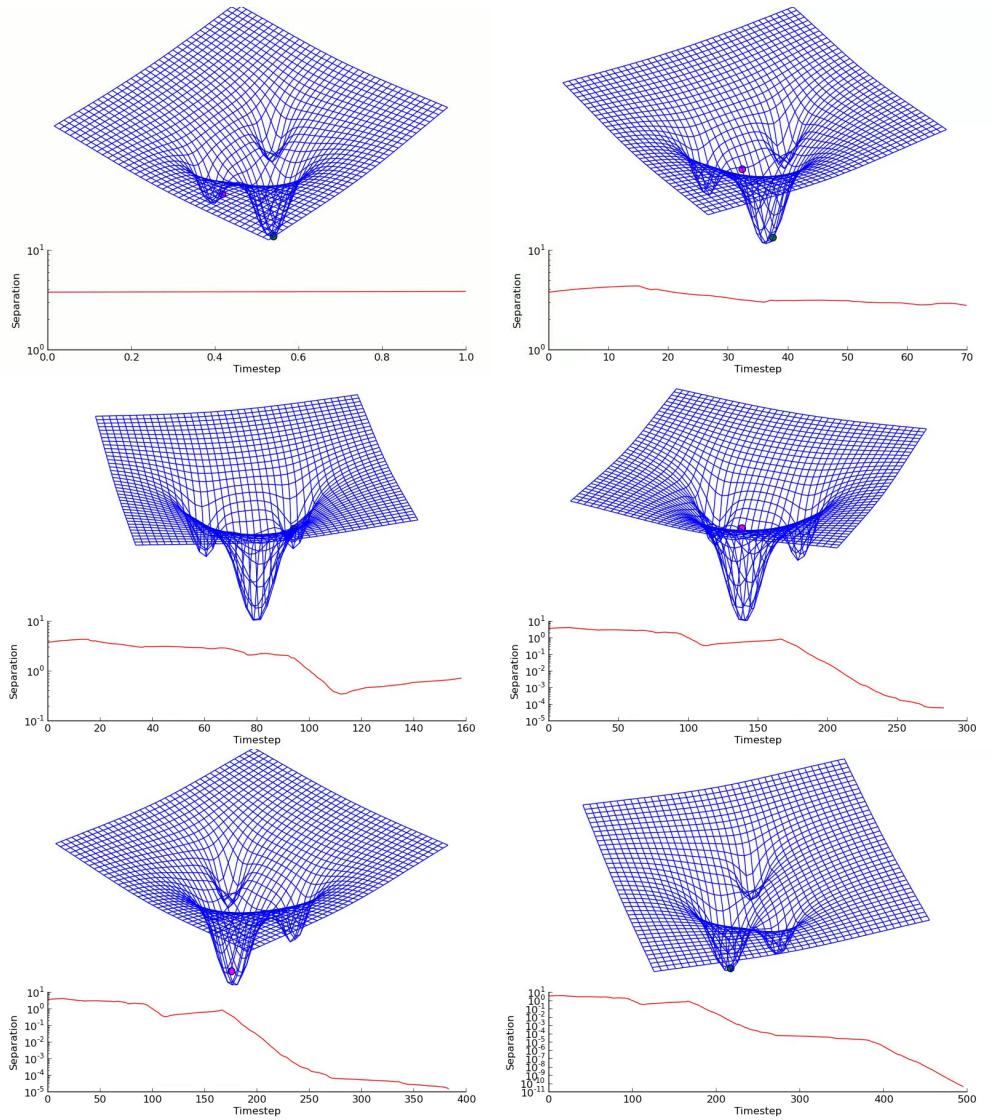


Figure C.4: Snapshots of two balls sampling a 2D function (shown inverted) using Langevin dynamics, where the same diffusive component is used for both balls. Below the visualisation of the dynamics is a plot of the logarithmic distance between the balls.

Bibliography

[1] Loos, P. F. & Gill, P. M. W. Exact Wave Functions of Two-Electron Quantum Rings. *Phys. Rev. Lett.* **108**, 083002 (2012).

[2] Glasserman, P. *Monte Carlo Methods in Financial Engineering* (Springer, 2004).

[3] Mode, C. J. (ed.) *Applications of Monte Carlo Methods in Biology, Medicine and Other Fields of Science* (InTech, 2011).

[4] MacCluer, J. W. Monte Carlo Methods in Human Population Genetics: A Computer Model Incorporating Age-Specific Birth and Death Rates. *Am. J. Hum. Genet.* **19**, 303–312 (1967).

[5] Cubasch, U. *et al.* Monte Carlo climate change forecasts with a global coupled ocean-atmosphere model. *Clim. Dyn.* **10**, 1–19 (1994).

[6] Pattabiraman, B. *et al.* A Parallel Monte Carlo Code for Simulating Collisional N-Body Systems. *Astrophys. J. Suppl. Ser.* **204**, 15 (2013).

[7] Young, W. M. & Elcock, E. W. Monte Carlo studies of vacancy migration in binary ordered alloys: I. *Proc. Phys. Soc.* **89**, 735 (1966).

[8] Bortz, A. B., Kalos, M. H. & Lebowitz, J. L. A New Algorithm for Monte Carlo Simulation of king Spin Systems. *J. Comput. Phys.* **17**, 10–18 (1975).

[9] Jansen, A. P. J. Monte Carlo simulations of chemical reactions on a surface with time-dependent reaction-rate constants. *Comput. Phys. Commun.* **86**, 1–12 (1995).

[10] Sjöstrand, T. The Lund Monte Carlo for jet fragmentation and e^+e^- physics - jetset version 6.2. *Comput. Phys. Commun.* **39**, 347–407 (1986).

[11] Foulkes, W. M. C., Mitáš, L., Needs, R. J. & Rajagopal, G. Quantum Monte Carlo simulations of solids. *Rev. Mod. Phys.* **71**, 33 (2001).

[12] Needs, R. J., Towler, M. D., Drummond, N. D. & López Ríos, P. Continuum variational and diffusion quantum Monte Carlo calculations. *J. Phys. Condens. Matter* **22**, 023201 (2010).

[13] McMillan, W. L. Ground state of liquid He^4. *Phys. Rev.* **138**, A442 (1965).

[14] Ceperley, D. M., Chester, G. V. & Kalos, M. H. Monte Carlo Simulation of a Many-Fermion Study. *Phys. Rev. B* **16**, 3081 (1977).

[15] Ceperley, D. M. Ground State of the Fermion One-Component Plasma: A Monte Carlo Study in Two and Three Dimensions. *Phys. Rev. B* **18**, 3126 (1978).

[16] Ceperley, D. M. & Alder, B. J. The calculation of the properties of metallic hydrogen using Monte Carlo. *Phys. B+C* **108**, 875–876 (1981).

[17] Ceperley, D. M. & Alder, B. J. Ground State of the Electron Gas by a Stochastic Method. *Phys. Rev. Lett.* **45**, 566–569 (1980).

[18] Ceperley, D. M. & Alder, B. J. Ground state of solid hydrogen at high pressures. *Phys. Rev. B* **36**, 2092–2106 (1987).

[19] Li, X. P., Ceperley, D. M. & Martin, R. M. Cohesive energy of silicon by the Green's-function Monte Carlo method. *Phys. Rev. B* **44**, 10929–10932 (1991).

[20] Pople, J. A. Nobel lecture: Quantum chemical models. *Rev. Mod. Phys.* **71**, 1267–1274 (1999).

[21] Badinski, A., Haynes, P. D. & Needs, R. J. Nodal Pulay terms for accurate diffusion quantum Monte Carlo forces. *Phys. Rev. B* **77**, 1–8 (2008).

[22] Badinski, A., Haynes, P. D., Trail, J. R. & Needs, R. J. Methods for calculating forces within quantum Monte Carlo simulations. *J. Phys. Condens. Matter* **22**, 074202 (2010).

[23] Badinski, A. & Needs, R. J. Accurate forces in quantum Monte Carlo calculations with nonlocal pseudopotentials. *Phys. Rev. E* **76**, 1–10 (2007).

[24] Badinski, A. & Needs, R. J. Total forces in the diffusion Monte Carlo method with nonlocal pseudopotentials. *Phys. Rev. B* **78**, 1–11 (2008).

[25] Wells, B. H. The differential Green's function Monte Carlo method. The dipole moment of LiH. *Chem. Phys. Lett.* **115**, 89–94 (1985).

[26] Traynor, C. A. & Anderson, J. B. Parallel Monte Carlo calculations to determine energy differences among similar molecular structures. *Chem. Phys. Lett.* **147**, 389–394 (1988).

[27] Umrigar, C. J. Space-Warp Transformation for Evaluating Forces. *Int. J. quantum Chem. Symp.* **23**, 217 (1989).

[28] Filippi, C. & Umrigar, C. J. Correlated sampling in quantum Monte Carlo: A route to forces. *Phys. Rev. B* **61**, R16291–R16294 (2000).

[29] Filippi, C. & Umrigar, C. J. Interatomic forces and correlated sampling in quantum Monte Carlo. In Lester, W. J., Rothstein, S. & Tanaka, S. (eds.) *Recent Adv. Quantum Monte Carlo Methods, Part II*, 12–29 (World Scientific, Singapore, 2002).

[30] Moroni, S., Saccani, S. & Filippi, C. Practical Schemes for Accurate Forces in Quantum Monte Carlo. *J. Chem. Theory Comput.* **10**, 4823–4829 (2014).

[31] Assaraf, R., Caffarel, M. & Kollias, A. C. Chaotic versus Nonchaotic Stochastic Dynamics in Monte Carlo Simulations: A Route for Accurate Energy Differences in N-Body Systems. *Phys. Rev. Lett.* **106**, 150601 (2011).

[32] Per, M. C., Snook, I. K. & Russo, S. P. Efficient calculation of unbiased expectation values in diffusion quantum Monte Carlo. *Phys. Rev. B* **86**, 201107 (2012).

[33] Williamson, A. J., Hood, R. Q. & Grossman, J. C. Linear-Scaling Quantum Monte Carlo Calculations. *Phys. Rev. Lett.* **87**, 246406 (2001).

[34] MacMillan, D. B. Optimization of importance-sampling parameters in Monte Carlo. *Nucl. Sci. Eng.* **48**, 219 (1972).

[35] Umrigar, C. J., Wilson, K. G. & Wilkins, J. W. Optimized trial wave functions for quantum Monte Carlo calculations. *Phys. Rev. Lett.* **60**, 1719–1722 (1988).

[36] Drummond, N. D. & Needs, R. J. Variance-minimization scheme for optimizing Jastrow factors. *Phys. Rev. B* **72**, 1–13 (2005).

[37] Fahy, S. & Hamann, D. R. Transition from chaotic to nonchaotic behavior in randomly driven systems. *Phys. Rev. Lett.* **69**, 761–764 (1992).

[38] Griewank, A. On Automatic Differentiation. In Iri, M. & Tanabe, K. (eds.) *Math. Program. Recent Dev. Appl.*, 83–108 (Kluwer Academic Publishers, Amsterdam, 1989).

[39] Griewank, A. & Walther, A. *Evaluating derivatives: principles and techniques of algorithmic differentiation* (SIAM, Philadelphia IL, 2008), 2 edn.

[40] Sorella, S. & Capriotti, L. Algorithmic differentiation and the calculation of forces by quantum Monte Carlo. *J. Chem. Phys.* **133**, 234111 (2010).

[41] White, A. The Materials Genome Initiative: One year on. *MRS Bulletin* **37**, 715–716 (2012).

[42] Allen, M. P. & Tildesley, D. J. Computer Simulation of Liquids. *Math. Comput.* **57**, 442–444 (1991).

[43] Rapaport, D. C. *The Art of Molecular Dynamics Simulation* (Cambridge University Press., 2004).

[44] Broughton, J. & Li, X. Phase diagram of silicon by molecular dynamics. *Phys. Rev. B* **35**, 9120–9127 (1987).

[45] Matsumoto, M., Saito, S. & Ohmine, I. Molecular dynamics simulation of the ice nucleation and growth process leading to water freezing. *Nature* **416**, 409–413 (2002).

[46] Sheehan, M. E. & Sharratt, P. N. Molecular dynamics methodology for the study of the solvent effects on a concentrated Diels-Alder reaction and the separation of the post-reaction mixture. *Comput. Chem. Eng.* **22**, S27 – S33 (1998).

[47] Liu, X. *et al.* Diffusion Coefficients from Molecular Dynamics Simulations in Binary and Ternary Mixtures. *Int. J. Thermophys.* **34**, 1169–1196 (2013).

[48] Born, M. & Oppenheimer, R. Zur quantentheorie der molekeln. *Ann. Phys.* **389**, 457–484 (1927).

[49] Sayvetz, A. The kinetic energy of polyatomic molecules. *J. Chem. Phys.* **46**, 383–387 (1939).

[50] Haynes, P. D. *Linear-scaling methods in ab initio quantum-mechanical calculations.* Ph.D. thesis, University of Cambridge (1998).

[51] Raghavachari, K. & Anderson, J. B. Electron correlation effects in molecules. *J. Phys. Chem.* **3654**, 12960–12973 (1996).

[52] Sherrill, C. D. & Schaefer III, H. F. The Configuration Interaction Method: Advances in Highly Correlated Approaches. vol. 34 of *Advances in Quantum Chemistry*, 143–269 (Academic Press, 1999).

[53] Crawford, T. D. & Schaefer, H. F. *An Introduction to Coupled Cluster Theory for Computational Chemists*, 33–136 (John Wiley & Sons, Inc., 2007).

[54] DeFrees, D. J. *et al.* Effect of electron correlation on theoretical equilibrium geometries. *J. Am. Chem. Soc.* **101**, 4085–4089 (1979).

[55] Hout, R. F., Levi, B. A. & Hehre, W. J. Effect of electron correlation on theoretical vibrational frequencies. *J. Comput. Chem.* **3**, 234–250 (1982).

[56] Simandiras, E. D., Handy, N. C. & Amos, R. D. On the high accuracy of MP2-optimised geometries and harmonic frequencies with large basis sets. *Chem. Phys. Lett.* **133**, 324–330 (1987).

[57] Raghavachari, K. & Trucks, G. W. Highly correlated systems. Excitation energies of first row transition metals Sc-Cu. *J. Chem. Phys.* **91**, 1062 (1989).

[58] Dewar, M. J. S., Zoebisch, E. G., Healy, E. F. & Stewart, J. J. P. Development and use of quantum mechanical molecular models. 76. AM1: a new general purpose quantum mechanical molecular model. *J. Am. Chem. Soc.* **107**, 3902–3909 (1985).

[59] Stewart, J. J. P. Optimization of parameters for semiempirical methods I. Method. *J. Comput. Chem.* **10**, 209–220 (1989).

[60] Rocha, G. B., Freire, R. O., Simas, A. M. & Stewart, J. J. P. RM1: A reparameterization of AM1 for H, C, N, O, P, S, F, Cl, Br, and I. *J. Comput. Chem.* **27**, 1101–1111 (2006).

[61] Stewart, J. J. P. Optimization of parameters for semiempirical methods V: modification of NDDO approximations and application to 70 elements. *J. Mol. Model.* **13**, 1173–213 (2007).

[62] Kohn, W. Nobel Lecture: Electronic structure of matter-wave functions and density functionals. *Rev. Mod. Phys.* **71**, 1253–1266 (1999).

[63] Hohenberg, P. & Kohn, W. Inhomogeneous Electron Gas. *Phys. Rev.* **155**, B864 – B871 (1964).

[64] Kohn, W. & Sham, L. J. Self-Consistent Equations Including Exchange and Correlation Effects. *Phys. Rev.* **385**, A1133 – A1138 (1965).

[65] Cohen, A. J., Mori-Sánchez, P. & Yang, W. Challenges for density functional theory. *Chem. Rev.* **112**, 289–320 (2012).

[66] Nemec, N. Diffusion Monte Carlo: Exponential scaling of computational cost for large systems. *Phys. Rev. B* **81**, 035119 (2010).

[67] Gillan, M. J., Towler, M. D. & Alfè, D. Petascale computing opens new vistas for quantum Monte Carlo. *Psi-k Newsletter* **103**, 32–69 (2011).

[68] Scemama, A., Caffarel, M., Oseret, E. & Jalby, W. Quantum Monte Carlo for large chemical systems: implementing efficient strategies for petascale platforms and beyond. *J. Comput. Chem.* **34**, 938–951 (2013).

[69] Nolan, S. J., Gillan, M. J., Alfè, D., Allan, N. L. & Manby, F. R. Calculation of properties of crystalline lithium hydride using correlated wave function theory. *Phys. Rev. B* **80**, 165109 (2009).

[70] Marsman, M., Grüneis, A., Paier, J. & Kresse, G. Second-order Møller-Plesset perturbation theory applied to extended systems. I. Within the projector-augmented-wave formalism using a plane wave basis set. *J. Chem. Phys.* **130**, 184103 (2009).

[71] Booth, G. H., Grüneis, A., Kresse, G. & Alavi, A. Towards an exact description of electronic wavefunctions in real solids. *Nature* **493**, 365–70 (2013).

[72] Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H. & Teller, E. Equation of State Calculations by Fast Computing Machines. *J. Chem. Phys.* **21**, 1087–1092 (1953).

[73] Lee, R. M., Conduit, G. J., Nemec, N., López Ríos, P. & Drummond, N. D. Strategies for improving the efficiency of quantum Monte Carlo calculations. *Phys. Rev. E* **83**, 1–11 (2011).

[74] Chandrasekhar, S. Stochastic Problems in Physics and Astronomy. *Rev. Mod. Phys.* **15**, 1–89 (1943).

[75] Flyvbjerg, H. & Petersen, H. G. Error estimates on averages of correlated data. *J. Chem. Phys.* **91**, 461–466 (1989).

[76] Trail, J. R. Heavy-tailed random error in quantum Monte Carlo. *Phys. Rev. E* **77**, 016703 (2008).

[77] Trail, J. R. Alternative sampling for variational quantum Monte Carlo. *Phys. Rev. E* **77**, 016704 (2008).

[78] Assaraf, R. & Caffarel, M. Zero-variance principle for monte carlo algorithms. *Phys. Rev. Lett.* **82**, 4682–4685 (1999).

[79] Umrigar, C. J., Nightingale, M. P. & Runge, K. J. A diffusion Monte Carlo algorithm with very small time-step errors. *J. Chem. Phys.* **99**, 2865 (1993).

[80] Ceperley, D. M. & Alder, B. J. Quantum Monte Carlo for molecules: Greens function and nodal release. *J. Chem. Phys.* **81** (1984).

[81] Grimm, R. C. & Storer, R. G. Monte-Carlo solution of Schrödinger's equation. *J. Comput. Phys.* **7**, 134–156 (1971).

[82] Kalos, M. H., Levesque, D. & Verlet, L. Helium at zero temperature with hard-sphere and other forces. *Phys. Rev. A* **9**, 2178–2195 (1974).

[83] Anderson, J. B. A random-walk simulation of the Schrödinger equation: H^+_3. *J. Chem. Phys.* **63** (1975).

[84] Anderson, J. B. Quantum chemistry by random walk. H ^{2}P, H^+_3 D_{3h} $^1A_1$, H_2 $^3\Sigma^+_u$, H_4 $^1\Sigma^+_g$, Be ^{1}S. *J. Chem. Phys.* **65** (1976).

[85] Reynolds, P. J., Ceperley, D. M., Alder, B. J. & Lester Jr, W. A. Fixed-node quantum Monte Carlo for molecules. *J. Chem. Phys.* **77**, 5593 (1982).

[86] DePasquale, M. F., Rothstein, S. M. & Vrbik, J. Reliable diffusion quantum Monte Carlo. *J. Chem. Phys.* **89** (1988).

[87] Garmer, D. R. & Anderson, J. B. Potential energies for the reaction F+H2→HF+H by the random walk method. *J. Chem. Phys.* **89** (1988).

[88] Ceperley, D. M. Fermion nodes. *J. Stat. Phys.* **63**, 1237–1267 (1991).

[89] Anderson, J. B. Symmetry of the nodal hypersurface for 1s2p ^3P helium. *Phys. Rev. A* **35**, 3550–3553 (1987).

[90] Barnett, R. N., Reynolds, P. J. & Lester Jr, W. A. H + H2 reaction barrier: A fixed-node quantum Monte Carlo study. *J. Chem. Phys.* **82**, 2700 (1985).

[91] Scott, T. C., Lüchow, A., Bressanini, D. & Morgan III, J. D. Nodal surfaces of helium atom eigenfunctions. *Phys. Rev. A* **75**, 9–12 (2007).

[92] Glauser, W. A. *et al.* Random-walk approach to mapping nodal regions of N-body wave functions: Ground-state Hartree–Fock wave functions for Li-C. *J. Chem. Phys.* **97**, 9200 (1992).

[93] Mitáš, L. Structure of fermion nodes and nodal cells. *Phys. Rev. Lett.* **96**, 1–4 (2006).

[94] Jastrow, R. Many-body problem with strong forces. *Phys. Rev.* **98**, 1479–1484 (1955).

[95] Drummond, N. D., Towler, M. D. & Needs, R. J. Jastrow correlation factor for atoms, molecules, and solids. *Phys. Rev. B* **70**, 1–11 (2004).

[96] Goddard, W. A. Improved Quantum Theory of Many-Electron Systems. I. Construction of Eigenfunctions of S^2 Which Satisfy Pauli's Principle. *Phys. Rev.* **157**, 73–80 (1967).

[97] Bressanini, D. & Morosi, G. On the nodal structure of single-particle approximation based atomic wave functions. *J. Chem. Phys.* **129**, 054103 (2008).

[98] Bressanini, D., Morosi, G. & Tarasco, S. An investigation of nodal structures and the construction of trial wave functions. *J. Chem. Phys.* **123**, 204109 (2005).

[99] Casula, M. & Sorella, S. Geminal wave functions with Jastrow correlation: A first application to atoms. *J. Chem. Phys.* **119**, 6500 (2003).

[100] Casula, M., Attaccalite, C. & Sorella, S. Correlated geminal wave function for molecules: An efficient resonating valence bond approach. *J. Chem. Phys.* **121**, 7110–26 (2004).

[101] Bajdich, M., Mitáš, L., Drobný, G., Wagner, L. K. & Schmidt, K. E. Pfaffian pairing wave functions in electronic-structure quantum Monte Carlo simulations. *Phys. Rev. Lett.* **96**, 1–4 (2006).

[102] Bajdich, M., Mitas, L., Wagner, L. K. & Schmidt, K. E. Pfaffian pairing and backflow wavefunctions for electronic structure quantum Monte Carlo methods. *Phys. Rev. B* **77**, 115112 (2008).

[103] Holzmann, M., Ceperley, D. M., Pierleoni, C. & Esler, K. Backflow correlations for the electron gas and metallic hydrogen. *Phys. Rev. E* **68**, 1–15 (2003).

[104] López Ríos, P., Ma, A., Drummond, N. D., Towler, M. D. & Needs, R. J. Inhomogeneous backflow transformations in quantum Monte Carlo calculations. *Phys. Rev. E* **74**, 1–15 (2006).

[105] Sherman, J. & Morrison, W. J. Adjustment of an Inverse Matrix Corresponding to Changes in the Elements of a Given Column or a Given Row of the Original Matrix (abstract). *Ann. Math. Stat.* **20**, 620–624 (1949).

[106] Press, W. H., Teukolsky, S. A., Vetterling, W. T. & Flannery, B. P. *Numerical Recipes 3rd Edition: The Art of Scientific Computing* (Cambridge University Press, New York, NY, USA, 2007), 3 edn.

[107] Fahy, S., Wang, X. W. & Louie, S. G. Variational quantum Monte Carlo nonlocal pseudopotential approach to solids: Formulation and application to diamond, graphite, and silicon. *Phys. Rev. B* **42**, 3503–3522 (1990).

[108] Ceperley, D. M. The statistical error of Green's function Monte Carlo. *J. Stat. Phys.* **43** (1986).

[109] Ma, A., Drummond, N. D., Towler, M. D. & Needs, R. J. All-electron quantum Monte Carlo calculations for the noble gas atoms He to Xe. *Phys. Rev. E* **71**, 1–5 (2005).

[110] Kent, P. R. C., Needs, R. J. & Rajagopal, G. Monte Carlo energy and variance-minimization techniques for optimizing many-body wave functions. *Phys. Rev. B* **59**, 12345–12351 (1999).

[111] Umrigar, C. J., Toulouse, J., Filippi, C., Sorella, S. & Hennig, R. G. Alleviation of the fermion-sign problem by optimization of many-body wave functions. *Phys. Rev. Lett.* **98**, 1–4 (2007).

[112] Toulouse, J. & Umrigar, C. J. Optimization of quantum Monte Carlo wave functions by energy minimization. *J. Chem. Phys.* **126**, 084102 (2007).

[113] Nightingale, M. P. & Melik-Alaverdian, V. Optimization of ground-and excited-state wave functions and van der Waals clusters. *Phys. Rev. Lett.* **87**, 2–5 (2001).

[114] Snajdr, M. & Rothstein, S. M. Are properties derived from variance-optimized wave functions generally more accurate? Monte Carlo study of non-energy-related properties of H_2, He, and LiH. *J. Chem. Phys.* **112**, 4935 (2000).

[115] Galvez, F. J., Buenda, E. & Sarsa, A. Atomic properties from energy-optimized wave functions. *J. Chem. Phys.* **115**, 1166 (2001).

[116] Sandler, P., Buch, V. & Sadlej, J. Ground and excited states of the complex of CO with water: A diffusion Monte Carlo study. *J. Chem. Phys.* **105**, 10387 (1996).

[117] Ramilowski, J. A. & Farrelly, D. Computation of nodal surfaces in fixed-node diffusion Monte Carlo calculations using a genetic algorithm. *Phys. Chem. Chem. Phys.* **12**, 12450–6 (2010).

[118] Lüchow, A., Petz, R. & Scott, T. C. Direct optimization of nodal hypersurfaces in approximate wave functions. *J. Chem. Phys.* **126**, 144110 (2007).

[119] Reboredo, F. A., Hood, R. Q. & Kent, P. R. C. Self-healing diffusion quantum Monte Carlo algorithms: Direct reduction of the fermion sign error in electronic structure calculations. *Phys. Rev. B* **79**, 1–15 (2009).

[120] Reboredo, F. A. Systematic reduction of sign errors in many-body problems: Generalization of self-healing diffusion Monte Carlo to excited states. *Phys. Rev. B* **80**, 1–11 (2009).

[121] Ramilowski, J. A. & Farrelly, D. Fixed node diffusion Monte Carlo using a genetic algorithm: a study of the CO–^{4}He$_N$ complex, N = 1...10. *Phys. Chem. Chem. Phys.* (2012).

[122] Bajdich, M., Tiago, M. L., Hood, R. Q., Kent, P. R. C. & Reboredo, F. A. Systematic reduction of sign errors in many-body calculations of atoms and molecules. *Phys. Rev. Lett.* **104**, 1–4 (2010).

[123] Hammond, B. L., Reynolds, P. J. & Lester Jr, W. A. Valence quantum Monte Carlo with ab initio effective core potentials. *J. Chem. Phys.* **87**, 1130 (1987).

[124] Mitáš, L., Shirley, E. L. & Ceperley, D. M. Nonlocal pseudopotentials and diffusion Monte Carlo. *J. Chem. Phys.* **95**, 3467 (1991).

[125] Casula, M. Beyond the locality approximation in the standard diffusion Monte Carlo method. *Phys. Rev. B* **74**, 161102 (2006).

[126] Dierckx, P. *Curve and Surface Fitting with Splines* (Oxford University Press, Inc., New York, NY, USA, 1993).

[127] Hellmann, H. *Einführung in die Quantenchemie* (Franz Deuticke, Leipzig, 1937).

[128] Feynman, R. P. Forces in molecules. *Phys. Rev.* **56**, 340–343 (1939).

[129] Pulay, P. Ab initio calculation of force constants and equilibrium geometries in polyatomic molecules: I. Theory. *Mol. Phys.* **17**, 197–204 (1969).

[130] Reynolds, P. J., Barnett, R. N., Hammond, B. L. & Lester Jr, W. A. Molecular physics and chemistry applications of quantum Monte Carlo. *J. Stat. Phys.* **43**, 1017–1026 (1986).

[131] Reynolds, P. J., Barnett, R. N., Hammond, B. L., Grimes, R. M. & Lester Jr, W. A. Quantum chemistry by quantum Monte Carlo: Beyond ground-state energy calculations. *Int. J. Quantum Chem.* **29**, 589–596 (1986).

[132] Barnett, R. N., Reynolds, P. J. & Lester Jr, W. A. Monte Carlo algorithms for expectation values of coordinate operators. *J. Comput. Phys.* **96**, 258–276 (1991).

[133] Baroni, S. & Moroni, S. Reptation quantum Monte Carlo: a method for unbiased ground-state averages and imaginary-time correlations. *Phys. Rev. Lett.* **82**, 4745–4748 (1999).

[134] Assaraf, R. & Caffarel, M. Computing forces with quantum Monte Carlo. *J. Chem. Phys.* **113**, 4028 (2000).

[135] Assaraf, R. & Caffarel, M. Zero-variance zero-bias principle for observables in quantum Monte Carlo: Application to forces. *J. Chem. Phys.* **119**, 10536 (2003).

[136] Badinski, A., Trail, J. R. & Needs, R. J. Energy derivatives in quantum Monte Carlo involving the zero-variance property. *J. Chem. Phys.* **129**, 224101 (2008).

[137] Chiesa, S., Ceperley, D. M. & Zhang, S. Accurate, Efficient, and Simple Forces Computed with Quantum Monte Carlo Methods. *Phys. Rev. Lett.* **94**, 1–4 (2005).

[138] Vrbik, J. & Rothstein, S. M. Infinitesimal differential diffusion quantum Monte Carlo study of diatomic vibrational frequencies. *J. Chem. Phys.* **96**, 2071 (1992).

[139] Gaudoin, R. & Pitarke, J. M. Hellman-Feynman operator sampling in diffusion Monte Carlo calculations. *Phys. Rev. Lett.* **99**, 1–4 (2007).

[140] Mazzola, G., Yunoki, S. & Sorella, S. Unexpectedly high pressure for molecular dissociation in liquid hydrogen by electronic simulation. *Nat. Commun.* **5**, 3487 (2014).

[141] Griewank, A. Some Bounds on the Complexity of Gradients, Jacobians, and Hessians. In Pardalos, P. M. (ed.) *Complex. Nonlinear Optim.*, 128–161 (World Scientific Pub Co Inc, 1993).

[142] Capriotti, L. Fast Greeks by algorithmic differentiation. *J. Comput. Financ.* **14**, 3–36 (2011).

[143] Hascoët, L. & Pascual, V. The Tapenade Automatic Differentiation tool: Principles, model, and specification. *ACM Trans. Math. Softw.* **39** (2013).

[144] Walther, A. & Griewank, A. Getting started with ADOL-C. In Naumann, U. & Schenk, O. (eds.) *Comb. Sci. Comput.*, chap. 7, 181–202 (Chapman-Hall CRC Computational Science, 2012).

[145] Bendtsen, C. & Stauning, O. FADBAD, a Flexible C++ Package for Automatic Differentiation. Technical Report IMM–REP–1996–17, Department of Mathematical Modelling, Technical University of Denmark, Lyngby, Denmark (1996).

[146] Faure, C. Adjoining Strategies for Multi-layered Programs. *Optim. Methods Softw.* **17**, 129–164 (2002).

[147] Griewank, A. Achieving logarithmic growth of temporal and spatial complexity in reverse automatic differentiation. *Optim. Methods Softw.* **1**, 35–54 (1992).

[148] Schmidt, M. W. *et al.* General Atomic and Molecular Electronic Structure System. *J. Comput. Chem.* **14**, 1347–1363 (1993).

[149] Gordon, M. S. & Schmidt, M. W. Advances in electronic structure theory: GAMESS a decade later. In *Theory Appl. Comput. Chem.*, 1167–1189 (Elsevier, 2005).

[150] Swope, W. C., Andersen, H. C., Berens, P. H. & Wilson, K. R. A computer simulation method for the calculation of equilibrium constants for the

formation of physical clusters of molecules: Application to small water clusters. *J. Chem. Phys.* **76** (1982).

[151] Attaccalite, C. & Sorella, S. Stable Liquid Hydrogen at High Pressure by a Novel Ab Initio Molecular-Dynamics Calculation. *Phys. Rev. Lett.* **100**, 114501 (2008).

[152] Neugebauer, A. & Häfelinger, G. Prediction of experimentally unknown r_e distances of organic molecules from Dunning basis set extrapolations for *ab initio* post-HF calculations. *J. Phys. Org. Chem.* **19**, 196–205 (2006).

[153] Xu, J., Deible, M. J., Peterson, K. A. & Jordan, K. D. Correlation Consistent Gaussian Basis Sets for H, B-Ne with Dirac-Fock AREP Pseudopotentials: Applications in Quantum Monte Carlo Calculations. *J. Chem. Theory Comput.* **9**, 2170–2178 (2013).

[154] Trail, J. R. & Needs, R. J. Norm-conserving Hartree–Fock pseudopotentials and their asymptotic behavior. *J. Chem. Phys.* **122**, 14112 (2005).

[155] Trail, J. R. & Needs, R. J. Smooth relativistic Hartree–Fock pseudopotentials for H to Ba and Lu to Hg. *J. Chem. Phys.* **122**, 174109 (2005).

[156] Becke, A. D. Density-functional thermochemistry. III. The role of exact exchange. *J. Chem. Phys.* **98** (1993).

[157] Lee, C., Yang, W. & Parr, R. G. Development of the Colle-Salvetti correlation-energy formula into a functional of the electron density. *Phys. Rev. B* **37**, 785–789 (1988).

[158] Vosko, S. H., Wilk, L. & Nusair, M. Accurate spin-dependent electron liquid correlation energies for local spin density calculations: a critical analysis. *Can. J. Phys.* **58**, 1200–1211 (1980).

[159] Stephens, P. J., Devlin, F. J., Chabalowski, C. F. & Frisch, M. J. Ab Initio Calculation of Vibrational Absorption and Circular Dichroism Spectra Using Density Functional Force Fields. *J. Phys. Chem.* **98**, 11623–11627 (1994).

[160] Levien, R. B. & Tan, S. M. Double pendulum: An experiment in chaos. *Am. J. Phys.* **61** (1993).

[161] Wu, Q. & Yang, W. Empirical correction to density functional theory for van der Waals interactions. *J. Chem. Phys.* **116**, 515 (2002).

[162] Dion, M., Rydberg, H., Schröder, E., Langreth, D. C. & Lundqvist, B. I. Van der Waals Density Functional for General Geometries. *Phys. Rev. Lett.* **92**, 246401 (2004).

[163] Grimme, S. Accurate description of van der Waals complexes by density functional theory including empirical corrections. *J. Comput. Chem.* **25**, 1463–73 (2004).

[164] Zimmerli, U., Parrinello, M. & Koumoutsakos, P. Dispersion corrections to density functionals for water aromatic interactions. *J. Chem. Phys.* **120** (2004).

[165] Grimme, S. Semiempirical GGA-Type Density Functional Constructed with a Long-Range Dispersion Correction. *J. Comput. Chem.* **27**, 1787–1799 (2006).

[166] Ortmann, F., Bechstedt, F. & Schmidt, W. Semiempirical van der Waals correction to the density functional description of solids and molecular structures. *Phys. Rev. B* **73**, 205101 (2006).

[167] Jurečka, P., Černý, J., Hobza, P. & Salahub, D. R. Density functional theory augmented with an empirical dispersion term. Interaction energies and geometries of 80 noncovalent complexes compared with ab initio quantum mechanics calculations. *J. Comput. Chem.* **28**, 555–569 (2007).

[168] Thonhauser, T. *et al.* Van der Waals density functional: Self-consistent potential and the nature of the van der Waals bond. *Phys. Rev. B* **76**, 125112 (2007).

[169] Sato, T., Tsuneda, T. & Hirao, K. Long-range corrected density functional study on weakly bound systems: Balanced descriptions of various types of molecular interactions. *J. Chem. Phys.* **126** (2007).

[170] Tkatchenko, A. & Scheffler, M. Accurate Molecular Van Der Waals Interactions from Ground-State Electron Density and Free-Atom Reference Data. *Phys. Rev. Lett.* **102**, 073005 (2009).

[171] Sato, T. & Nakai, H. Density functional method including weak interactions: Dispersion coefficients based on the local response approximation. *J. Chem. Phys.* **131**, 224104 (2009).

[172] Grimme, S., Antony, J., Ehrlich, S. & Krieg, H. A consistent and accurate ab initio parametrization of density functional dispersion correction (DFT-D) for the 94 elements H-Pu. *J. Chem. Phys.* **132** (2010).

[173] Klimeš, J., Bowler, D. R. & Michaelides, A. Chemical accuracy for the van der Waals density functional. *J. Phys. Condens. Matter* **22**, 022201 (2010).

[174] Lee, K., Murray, E. D., Kong, L., Lundqvist, B. I. & Langreth, D. C. Higher-accuracy van der Waals density functional. *Phys. Rev. B* **82**, 81101 (2010).

[175] Wagner, L. K. & Grossman, J. Quantum Monte Carlo Calculations for Minimum Energy Structures. *Phys. Rev. Lett.* **104**, 1–4 (2010).

[176] Matsumoto, M. & Nishimura, T. Mersenne Twister: A 623-dimensionally Equidistributed Uniform Pseudo-random Number Generator. *ACM Trans. Model. Comput. Simul.* **8**, 3–30 (1998).

[177] Saito, M. & Matsumoto, M. SIMD-Oriented Fast Mersenne Twister: a 128-bit Pseudorandom Number Generator. In *Monte Carlo and Quasi-Monte Carlo Methods 2006*, 607–622 (2008).