Imperial College London
Department of Chemistry

# Investigating Quantum Monte Carlo Methods in Slater Determinant Bases

William Andrew Vigor

June 2015

Submitted in part fulfilment of the requirements for the degree of
Doctor of Philosophy in Chemistry of Imperial College London
and the Diploma of Imperial College London

# Abstract

In this thesis we investigate the recently developed Full Configuration Interaction Quantum Monte Carlo (FCIQMC) method. This method, unlike the traditional methods in Quantum Monte Carlo (QMC), doesn't require the use of the uncontrolled fixed node approximation and so potentially it can yield far more accurate results. This means that it can be thought of as a hybrid between the methods used by quantum chemists and QMC, and thus has spawned a new field of stochastic quantum chemistry. The work described in this thesis can be split into three distinct but interrelated parts.

We begin with an investigation of the underlying FCIQMC stochastic process. We show that FCIQMC is an example of Markov Chain Monte Carlo. This means we can compute a stochastic matrix from which all details about the Monte-Carlo simulation can be obtained. Unfortunately the size of the space scales unfavourably as a function of system size meaning that we have only managed to compute the matrix for the smallest interesting two determinant system. We then use these results to quantify population control bias in FCIQMC for a two determinant system supplementing these analytical results with empirical results to investigate more realistic systems.

We then attempt to quantify the efficiency of the FCIQMC algorithms, defining a measure of efficiency. After this we investigate the dependence of our measure on the system size. The error bar of the most efficient FCIQMC algorithm will decay fastest as a function of computer time. We then draw conclusions about the applicability of the FCIQMC method.

Finally we describe an implementation of FCIQMC on a novel data flow computer architecture. In our implementation we made a modification to the FCIQMC algorithm to fit the data flow paradigm. We investigate the efficiency of the modified FCIQMC algorithm.

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

# Acknowledgements

This work would not have been possible without the love and support of my parents and brother. I would like to thank EPSRC financial support.

A special thanks to Robin Bruce, Craig Davies and Georgi Gaydadjiev at Maxeler Technologies for teaching me to use data flow technology. Also to the past and present members of the group Mike Robb, David Mendive-Tapia, Lee Thompson, Juan Jose Serrano-Perez, Stefano Serapian, Clyde Fare, Morgane Vacher, Valentina Santolini, Joao Malhado, Jan Meisner, Andrew Jenkins and Marine Bouduban. This work owes so much to their friendship and advice. I also wish to thank Brian Ward, Precious Ugbomah, Jordan Muscatello, Clare Ashworth, Richard Matthews, Matt Clough, Sarah Smith and Becky Rowe who have helped make an office with no natural light a nice place to work.

I wish to thank James Spencer for teaching me that writing a good scientific program is important both for me and for everyone else, and for his expert advice. Finally I would like to thank my supervisors Mike Bearpark and Alex Thom whose guidance and support have been indispensable.

I declare that the work within this thesis is my own and nothing is the result of collaboration unless explicitly stated otherwise. The work in Chapters 2 and 3 is based upon the work published in W. A. Vigor, J. S. Spencer, M. J. Bearpark, and A. J. W. Thom, J. Chem. Phys. **142**, 104101 (2015). The work in Chapter 6 is built on top of a library of DFE functions developed and owned by Maxeler Technologies.

William Andrew Vigor
June 2015

# Contents


**1. Introduction** **13**

1.1. A Review of Computational Chemistry . . . . . 15

1.1.1. Hartree–Fock Theory . . . . . 16

1.1.2. Configuration Interaction . . . . . 18

1.1.3. Coupled Cluster Theory . . . . . 20

1.1.4. Møller Plesset Perturbation Theory . . . . . 21

1.1.5. Other Methods . . . . . 22

1.1.6. Why do we need high accuracy? . . . . . 22

1.2. Quantum Monte Carlo . . . . . 24

1.2.1. Monte-Carlo Integration . . . . . 24

1.2.2. Variational Monte Carlo . . . . . 25

1.2.3. Diffusion Monte Carlo . . . . . 27

1.3. Full Configuration Interaction Quantum Monte Carlo . . . . . 31

1.3.1. The Initiator Approximation . . . . . 34

1.3.2. Random Excitation Generators . . . . . 34

1.3.3. The Sign Problem . . . . . 36

1.3.4. An Aside: FCIQMC and the Hubbard model . . . . . 37

**2. The FCIQMC Stochastic Process** **40**

2.1. General Markov Chain Monte Carlo Theory . . . . . 41

2.2. The FCIQMC chain . . . . . 43

2.2.1. The Stochastic Matrix for a two Determinant System . . . . . 44

2.2.2. The Stationary Distribution of $H_2$ in a STO-3G basis set . . . 46

2.2.3. The Critical Timestep . . . . . 51

2.3. Conclusion . . . . . 52

**3. Population Control Bias in FCIQMC** **54**

3.1. $H_2$ in a STO-3G Basis Set . . . . . 57

3.2. The Neon Atom . . . . . 58

3.3. Discussion . . . . . 62

**4. The Efficiency of FCIQMC** **64**
4.1. The Computational Cost of a Calculation . . . . . 65
4.2. Minimising the Error Bar . . . . . 67
4.2.1. Measuring $a$ . . . . . 69
4.2.2. Varying $N_0$ and $\xi$ . . . . . 71
4.2.3. Scaling with $\langle N_p \rangle$ . . . . . 71
4.2.4. Scaling with $\delta\tau$ . . . . . 75
4.3. The scaling of $a$ as a function of system size . . . . . 77
4.4. Conclusion . . . . . 83

**5. A Review of Computer Architecture** **84**
5.1. CPU Architecture . . . . . 84
5.2. Graphical Processing Units . . . . . 87
5.3. Data Flow Architecture . . . . . 89

**6. A Data Flow Approach to FCIQMC** **91**
6.1. General Programing for Data Flow . . . . . 91
6.2. A Data Flow Implementation of FCIQMC . . . . . 98
6.2.1. Annihilation . . . . . 100
6.2.2. Calculating the Projected Energy . . . . . 101
6.2.3. Spawning and Diagonal Death . . . . . 101
6.3. Efficiency . . . . . 105
6.3.1. DFE time to process . . . . . 105
6.3.2. $a$ as a function of $\delta\tau$ for the Hubbard Model . . . . . 111
6.3.3. Improving Data-Flow FCIQMC . . . . . 113
6.4. Data Flow for Molecular FCIQMC . . . . . 114
6.5. Conclusion . . . . . 115
6.6. Addendum . . . . . 116

**7. Conclusion** **117**

**A. Plateau Heights and the Stochastic Error** **126**

**B. DFE Annihilation** **136**

# List of Tables

1.1. The Slater Condon rules for evaluating matrix elements between determinants and a one electron operator. . . . . 19
1.2. The Slater Condon rules for evaluating matrix elements between determinants and a two electron operator. . . . . 20
1.3. The MPn energy from zeroth to second order.[13] . . . . . 21

# List of Figures

1.1. The Metropolis algorithm used to sample a function with an unknown normalisation. . . . . 26
1.2. A diagrammatic representation of an example of a single iteration of the FCIQMC algorithm. . . . . 32
1.3. An FCIQMC calculation of the ground state energy of the neon atom in an aug-cc-pVDZ basis set . . . . . 35

2.1. The instantaneous shift, $S$, as a function of the instantaneous psip population $N$, as defined in Eq. 1.48 for $H_2$. . . . . 47
2.2. The effect of population control on the stationary distributions of the number of psips, $N$, and the shift, $S$ for $H_2$. . . . . 48
2.3. The effect of population control on the stationary distribution of the numerator and denominator of the projected energy for $H_2$. . . . . 50
2.4. Top: The stationary distribution breaks down once the time step becomes close to the critical point. Here $\delta\tau_c = 0.6061$. Bottom: We see the population explodes once $\delta\tau \approx 1.2$ after being metastable for a very short period of time. This metastability could be caused by a fluctuation in the shift. We see no population explosion for any other point even if run for a large number of timesteps. . . . . 53

3.1. The energy estimators as a function of $1/\langle N\rangle$ for $H_2$. . . . . 55
3.2. The energy estimators calculation from the means of the stationary distribution as a function of $1/\langle N\rangle$ for $H_2$ . . . . . 56
3.3. Projected energy estimate from FCIQMC calculations on the Ne atom (cc-pVDZ basis set) as a function of (left) the shift damping parameter. 59
3.4. The bias in $\langle E_{\mathrm{Proj}}\rangle$ relative to $E_{\mathrm{FCI}}$ for all calculations in Sec. 3 plotted as a function of the variance of the shift. . . . . 60

3.5. Reweighted projected energy estimate from FCIQMC calculations on the Ne atom (cc-pVDZ basis set) for different $\langle N \rangle$ as a function of $W$, the number of iterations reweighted over. Four independent FCIQMC calculations were performed for each $\langle N \rangle$, and each point shows the mean of the reweighted $\langle E_{\text{proj}} \rangle$ for the four runs. . . . . 61

4.1. The computer time per timestep per psip as a function of the timestep. 67
4.2. The error bar in the projected energy as a function of $N$. . . . . 70
4.3. The inefficiency as a function of $\gamma$ (the shift damping factor in Eq. 1.47). 72
4.4. The inefficiency as a function of the average number of psips. . . . . 73
4.5. Top: The inefficiency as a function of $\delta\tau$ for Ne cc-pVDZ and Ne aug-cc-pVDZ. Bottom: The plateau height as a function of the timestep for Ne in a aug-cc-pVDZ basis set. . . . . 76
4.6. The effect of increasing the timestep on the inefficiency $a$ for two different $\langle N_p \rangle$. . . . . 77
4.7. Top: The inefficiency factor $a$ for the neon atom in a cc-pVDZ, aug-cc-pVDZ, cc-pVTZ, aug-cc-pVTZ and a cc-pVQZ basis set as a function of $\langle N_p \rangle$. Bottom: the minimum value of $a$ as a function of basis set size. . . . . 80
4.8. The inefficiency $a$ for HF in a cc-pVDZ basis as a function of $\langle N_p \rangle$ and bond length ($R_0 = 0.91622$ Å). . . . . 81
4.9. Top: The ineffciency $a$ as a function of $\langle N_p \rangle$ and number of atoms in a helium chain. Bottom: The minimum ineffciency $a$ as function of the number of helium atoms, for helium in a 6-31G basis set. . . . . 82

5.1. The number of transistors, clock speeds, performance per clock cycle and power consumption for successive generations of Intel CPUs. Reproduced from [93]. . . . . 88

6.1. A simple example of a stream algorithm with a pipeline depth of 3. . 92
6.2. An input of a DFE. . . . . 93
6.3. An output of a DFE. . . . . 94
6.4. Some fundamental stream operations. . . . . 96
6.5. A kernel which sums the rows of a matrix. . . . . 97
6.6. An overview of the DFE implementation of FCIQMC. . . . . 100
6.7. A kernel which sums an array of floating point numbers. . . . . 102

6.8. The pipeline which computes the diagonal matrix elements for the two site Hubbard model. . . . . 103
6.9. The FPGA excitation generator routine for the Hubbard Model. . . . 104
6.10. Top: The time to perform spawning and diagonal death as a function of the number of cycles on a MAX3 Vectis card. The number of cycles scheduled grows proportionally to $l_m + l_s$ (some extra cycles are required to remove determinants where all the psips have annihilated). To get different cycle lengths we grow the population until $\langle N_p \rangle \approx$ 1800000 for different values of U/t and $\delta\tau$ (for each system we use 10 electrons). Bottom: From a fit of the above data the effective clock speed (the reciprocal of the gradient) and the intercept. . . . . 107
6.11. The average time spent performing a timestep for an equilibrated mFCIQMC calculations on a CPU (right bars) and a DFE (left bars). We see a speed up of between 1.89(2) times (for U/t=2 $\delta\tau = 0.009$) and 7(2) times (for U/t=2 $\delta\tau = 0.0009$). . . . . 108
6.12. Flip Flop FF and Look Up Table LUT (the two most important type of DFE resource) usage as a function of the number of sites. . . . . 110
6.13. Top: $a$ as a function of $\delta\tau$ for the 9 site Hubbard model comparing the mFCIQMC and FCIQMC Markov chains. Bottom: $a$ as a function of $\delta\tau$ for the 16 site Hubbard model comparing the mFCIQMC and FCIQMC Markov chains. . . . . 112
6.14. The effect of spawn onto multiple determinants on $a$ as a function of $\delta\tau$ for the 9 site Hubbard model. . . . . 114

A.1. Top: The inefficiency $a$ remains constant until $\delta\tau \approx 0.001$. Bottom: The plateau height can be seen to increase after $\delta\tau \approx 0.001$. . . . . 127
A.2. Top: The inefficiency $a$ remains constant until $\delta\tau \approx 0.001$. Bottom: The plateau height can be seen to increase after $\delta\tau \approx 0.001$. . . . . 128
A.3. Top: The inefficiency $a$ remains constant until $\delta\tau \approx 0.01$. Bottom: The plateau height can be seen to increase after $\delta\tau \approx 0.01$. . . . . 129
A.4. Top: The inefficiency $a$ remains constant until $\delta\tau \approx 0.01$. Bottom: The plateau height can be seen to increase after $\delta\tau \approx 0.01$. . . . . 130
A.5. Top: The inefficiency $a$ remains constant until $\delta\tau \approx 0.0042$. Bottom: The plateau height can be seen to increase after $\delta\tau \approx 0.0042$. . . . . 131
A.6. Top: The inefficiency $a$ remains constant until $\delta\tau \approx 0.0035$. Bottom: The plateau height can be seen to increase after $\delta\tau \approx 0.0035$. . . . . 132

A.7. Top: The inefficiency $a$ remains constant until $\delta\tau \approx 0.0005$. Bottom: The plateau height can be seen to increase after $\delta\tau \approx 0.0005$. . . . . 133
A.8. Top: The inefficiency $a$ remains constant until $\delta\tau \approx 0.0021$. Bottom: The plateau height can be seen to increase after $\delta\tau \approx 0.0021$. . . . . 134
A.9. Top: The inefficiency $a$ remains constant until $\delta\tau \approx 0.00175$. Bottom: The plateau height can be seen to increase after $\delta\tau \approx 0.00175$. . . . . 135

B.1. The input control stage of the DFE annihilation algorithm which merges two input psip arrays into a third output psip array. . . . . 138
B.2. The algorithm which merges the two controlled input streams into a single stream of determinants and psips. . . . . 139

# 1. Introduction

We begin by briefly reviewing the wide range of standard methods used by computational chemists to predict properties of chemical systems such as reaction energies. The method used in this thesis, Full Configuration Interaction Quantum Monte Carlo (FCIQMC) is based on these methods. They form a hierarchy, the higher up the hierarchy the more accurate the answer, but this comes at the expense that the CPU and memory requirements scale increasingly poorly as a function of the number of electrons. Highly accurate calculations are much more computationally expensive (they require many more CPU hours and may require much more memory) than less accurate calculations.

One of the goals of computational chemistry is to develop methods accurate enough to make predictions with a similar accuracy to (or better accuracy than) experiments on large molecules (hundreds of atoms). For energies this experimental or chemical accuracy is often quoted to be about 1 kcal $\text{mol}^{-1}$, or in atomic units used within this thesis 0.0016 Hartree (we denote this unit as $E_h$) [1]. Unfortunately the methods which are capable of achieving this accuracy scale unfavourably with system size. The CPU time for Coupled Cluster Singles Doubles and perturbative Triples (CCSD(T))[2] scales as the seventh power of the number of electrons, but has been shown to give results to within chemical accuracy for a wide range of small molecules [3]. For this reason it is known as the gold standard in the computational chemistry community [4]. Approaches such as Density Functional Theory (DFT) provide energies which are often nearly as accurate as CCSD(T). This scales as the third power of the number of electrons and so can be applied to large molecules [5]. Unfortunately DFT relies heavily on the choice of functional: some functionals are very accurate for some systems. However it has become increasingly difficult to systematically improve a single functional which is universally applicable. For this reason DFT does not fall into the standard hierarchy of quantum chemical methods,[6] although it is widely used in many practical applications.

To provide further context to the Monte-Carlo method used in this thesis we discuss two common Quantum Monte Carlo (QMC) methods, Diffusion Monte Carlo

(DMC) and Variational Monte Carlo (VMC). These don't require the storage of the entire wavefunction and are parallelisable onto thousands of processors. Unfortunately this comes at the cost of the introduction of a statistical error bar (in addition to the underlying approximations in the method itself which cause a systematic error). In DMC an uncontrollable fixed node approximation is also introduced. Finally we discuss the method used in this thesis, FCIQMC, which is both a Quantum Monte Carlo method and falls into the computational chemistry hierarchy. This is done both to provide context to this work and to develop the notation used in the later chapters. FCIQMC has the advantage over DMC that it doesn't require the use of the fixed node approximation.

## 1.1. A Review of Computational Chemistry

Even though the Schrödinger equation which describes systems of interacting electrons and nuclei has been known for the last 80 or so years,[7] finding cheap and accurate approximate solutions remains one of the principal challenges in computational chemistry. The time independent Schrödinger Equation is:

$$\hat{H}\,|\psi\rangle = E\,|\psi\rangle \tag{1.1}$$

where $|\psi\rangle$ is the wavefunction which describes everything about the system in question and $E$ is the energy of $\psi$. For a system of $N$ electrons and $M$ nuclei the Hamiltonian $\hat{H}$ in atomic units is:

$$\hat{H} = -\sum_{i=1}^{N}\frac{\nabla_i^2}{2} - \sum_{A=1}^{M}\frac{\nabla_A^2}{2M_A} + \sum_{i=1}^{N}\sum_{j>i}^{N}\frac{1}{|\mathbf{r}_i-\mathbf{r}_j|} - \sum_{i=1}^{N}\sum_{A=1}^{M}\frac{Z_A}{|\mathbf{r}_i-\mathbf{R}_A|} + \sum_{A=1}^{M}\sum_{B>A}^{M}\frac{Z_A Z_B}{|\mathbf{R}_A-\mathbf{R}_B|} \tag{1.2}$$

where $M_A$ is the ratio of the mass of the nucleus $A$ to the mass of an electron, $\mathbf{r}_i$ is the position vector of electron $i$, $\mathbf{R}_A$ is the position vector of nucleus $A$, $Z_A$ is the atomic number of nuclei $A$. The first two terms are the kinetic energy of the electrons and the nuclei respectively, the third is the electron-electron interaction, the fourth term is the interaction between the nuclei and the electrons, and the final term is the nucleus-nucleus interaction.

The Schrödinger equation has only been solved analytically for hydrogenic atoms (atoms with a single electron and differing nuclear charge), systems with a single particle and a model potential[8] and certain systems with more than one electron but with a lower dimensionality.[9] However, a number of numerical methods have been invented to solve this equation approximately. Some of these are discussed in the rest of this chapter.

The clamped nuclei approximation serves as the basis for all of the methods discussed in this review. We assume that the nuclei in our system are fixed and solve the Schrödinger equation with a constant nuclear potential at fixed geometry, thus $|\psi\rangle$ no longer has an explicit dependence on the nuclear coordinates. We neglect the nuclear kinetic energy and then solve to find an electronic wavefunction at fixed nuclear geometry.

$$\hat{H} = -\sum_{i=1}^{N}\frac{\nabla_i^2}{2} + \sum_{i=1}^{N}\sum_{j>i}^{N}\frac{1}{|\mathbf{r}_i-\mathbf{r}_j|} - \sum_{i=1}^{N}\sum_{A=1}^{M}\frac{Z_A}{|\mathbf{r}_i-\mathbf{R}_A|} + \langle\sum_{A=1}^{M}\sum_{B>A}^{M}\frac{Z_A Z_B}{|\mathbf{R}_A-\mathbf{R}_B|}\rangle \tag{1.3}$$

The postulates of Quantum mechanics place an additional constraint on the solution $|\psi\rangle$. Each electron must also be described additionally by either being in a spin up state or a spin down state. We can group the spin into electron coordinates:

$$\mathbf{x}_i = (\mathbf{r}_i, S) \tag{1.4}$$

where $S =\uparrow$ for an up spin electron and $S =\downarrow$ for a down spin electron. The $|\psi\rangle$ must be antisymmetric with respect to permutations of any two electronic coordinates:

$$|\psi(\ldots, \mathbf{x}_i, \ldots \mathbf{x}_j)\rangle = -|\psi(\ldots, \mathbf{x}_j, \ldots \mathbf{x}_i)\rangle \tag{1.5}$$

### 1.1.1. Hartree–Fock Theory

Hartree–Fock Theory provides a starting point for the many accurate post Hartree–Fock methods used by computational chemists. For most chemical systems Hartree–Fock obtains more than 95 percent of the total energy of a molecule, and using direct methods Hartree–Fock can be made to scale as the third power of the number of electrons[10]. Using a density fitting approximation (which often works very well) it can be improved further to scale linearly with the number of electrons.[11] For a more detailed introduction see [12] or [13].

The simplest possible wavefunction which obeys antisymmetry Eq. 1.5 is a single Slater determinant:

$$|\psi\rangle = \frac{1}{\sqrt{N!}} \begin{vmatrix} \chi_1(\mathbf{x}_1) & \chi_2(\mathbf{x}_1) & \cdots & \chi_N(\mathbf{x}_1) \\ \chi_1(\mathbf{x}_2) & \chi_2(\mathbf{x}_2) & \cdots & \chi_N(\mathbf{x}_2) \\ \vdots & \vdots & \ddots & \vdots \\ \chi_1(\mathbf{x}_N) & \chi_2(\mathbf{x}_N) & \cdots & \chi_N(\mathbf{x}_N) \end{vmatrix} = |\chi_1, \chi_2, \ldots \chi_N\rangle \tag{1.6}$$

Using the variational principle it can be shown that the set of $\chi_i(\mathbf{x})$ which provide the variational minimum solution to Eq. 1.3, under the additional constraint that the spin orbitals remain orthonormal, are eigenfunctions of the Fock operator:

$$\hat{f}(\mathbf{r}_1) = -\frac{\nabla_1^2}{2} - \sum_{A=0}^{M} \frac{Z_A}{|\mathbf{r}_1 - \mathbf{R}_A|} + \sum_{b}^{occ} \int \chi_b(\mathbf{x}_2) \frac{1}{|\mathbf{r}_1 - \mathbf{r}_2|} \left(1 - \hat{P}_{12}\right) \chi_b^*(\mathbf{x}_2) \mathrm{d}\mathbf{r}_2 \tag{1.7}$$

where $\sum_b^{occ}$ represents the sum over the occupied eigenfunctions i.e. the $N$ lowest energy eigenfunctions and $\hat{P}_{12}$ interchanges electrons 1 and 2. From this point onwards we refer to eigenfunctions of Eq. 1.7 as molecular orbitals. This equation

depends on the set of molecular orbitals themselves and thus must be solved using iterative methods.

Analytic solutions have only been produced for atoms [12] and certain model Hamiltonians. In practice in computational chemistry (where one doesn't have the luxury of spherical symmetry) one has to introduce a finite set of basis functions (basis set). These basis functions are generally formed as a linear combination of Gaussian functions in an attempt to fit Slater orbitals which arise naturally from the solution of the hydrogenic atom. This is done because the integration of Slater functions is much more computationally expensive than Gaussian functions. The set of molecular orbitals is constrained to be a linear combination of these basis functions (or atomic orbitals). By the variational theorem the molecular orbitals are a variational minimum solution (of the Hartree–Fock equation) for a given finite basis set. Thus increasing the size of the finite basis set allows us to converge monotonically towards the so called Hartree–Fock limit. In this thesis we use both the Pople basis sets [14] and the Correlation Consistent basis sets of Dunning et al. [15].

Although Hartree–Fock recovers most of the energy of a chemical system it doesn't take into account the correlation between electrons. Electrons should be correlated i.e. the probability of one electron being at one specific point in space should depend on the position of every other electron. The functional form of a single determinant (of one electron functions) means that electrons of opposite spin are uncorrelated. Although an anti-symmetric wavefunction does mean that electrons of the same spin are correlated, it does not describe every possible way the electrons could be correlated. As per the convention in computational chemistry from this point onwards we ignore this exchange based correlation and describe Hartree–Fock as an uncorrelated method. A correlation energy $E_c$ is defined using the Hartree–Fock energy:

$$E_c = E - \langle D_0 | \hat{H} | D_0 \rangle \tag{1.8}$$

Here $E$ is the exact clamped nuclear solution to the Schrödinger equation and $|D_0\rangle$ is the determinant made up of the set of occupied molecular orbitals.

There are two widely used formulations of Hartree–Fock. In Restricted Hartree–Fock (RHF) [12] each molecular orbital is constrained to be doubly occupied with one spin up and one spin down electron. In Unrestricted Hartree–Fock (UHF) [16] this constraint is relaxed meaning the molecular orbitals for spin up and spin down electrons can be different. RHF fails to describe open shell molecules. A classic example of this is the hydrogen molecule pulled apart: both electrons should be

described by an equivalent wavefunction centred on each atom, but are forced to occupy the same molecular orbital. This problem is not present in UHF.[13]

Differing amounts of electron correlation at different nuclear geometries means that Hartree–Fock theory recovers more of the total energy at some geometries than at others. Unfortunately this means that Hartree–Fock theory often produces inaccurate energy differences when studying chemical reactions. For this reason a method which allows the electrons to be correlated is often required. Post Hartree–Fock methods allow electrons to be correlated by adding in additional determinants built from the Hartree–Fock molecular orbitals to the wavefunction. Some of these post Hartree–Fock methods are discussed in the next few sections.

### 1.1.2. Configuration Interaction

The most conceptually simple post Hartree–Fock method is Configuration Interaction (CI). Here the wave function is simply formed as a linear combination of Slater determinants. The CI wavefunction ansatz is:

$$|\psi\rangle = C_0 |D_0\rangle + \sum_{a=1} \sum_{i=1} C_i^a |D_i^a\rangle + \sum_{a>b} \sum_{i>j} C_{ij}^{ab} |D_{ij}^{ab}\rangle + \sum_{a>b>c} \sum_{i>j>k} C_{ijl}^{abc} |D_{ijk}^{abc}\rangle + \dots \tag{1.9}$$

Where the determinant $|D_i^a\rangle$ differs from the Hartree–Fock determinant by molecular orbital $\chi_i$ (occupied in the Hartree–Fock determinant) being replaced with $\chi_a$ (unoccupied in the Hartree–Fock determinant). If Eq. 1.9 is truncated at the first two terms, the method is referred to as Configuration Interaction Singles CIS. If the third term is also included then the method is referred to as Configuration Interaction Singles and Doubles CISD etc. If Eq. 1.9 is left untruncated then the method is called Full Configuration Interaction FCI. In FCI all possible determinants built from the set of Hartree–Fock molecular orbitals are included in the wavefunction ansatz.

The coefficients $C$ can be found by diagonalising the matrix:

$$\langle\psi| \hat{H} |\psi\rangle = \begin{pmatrix} \langle D_0| \hat{H} |D_0\rangle & \langle D_0| \hat{H} |D_1^2\rangle & \dots & \langle D_0| \hat{H} |D_{12}^{34}\rangle & \dots \\ \langle D_1^2| \hat{H} |D_0\rangle & \langle D_1^2| \hat{H} |D_1^2\rangle & \dots & \langle D_1^2| \hat{H} |D_{12}^{34}\rangle & \dots \\ \vdots & & & & \\ \langle D_{12}^{34}| \hat{H} |D_0\rangle & \langle D_{12}^{34}| \hat{H} |D_1^2\rangle & \dots & \langle D_{12}^{34}| \hat{H} |D_{12}^{34}\rangle & \dots \\ \vdots & & \ddots & & \dots \end{pmatrix} \tag{1.10}$$

The matrix elements can be calculated using the Slater-Condon rules see tables 1.1

| | $\hat{O} = \sum_i o(\mathbf{x}_i)$ |
|---|---|
| $\langle D\vert \hat{O} \vert D\rangle$ | $\sum_m^N \int \chi_m^*(\mathbf{x}_1) o(\mathbf{x}_1) \chi_m(\mathbf{x}_1) \mathrm{d}\mathbf{x}_1$ |
| $\langle D\vert \hat{O} \vert D_i^a\rangle$ | $\int \chi_i^*(\mathbf{x}_1) o(\mathbf{x}_1) \chi_a(\mathbf{x}_1) \mathrm{d}\mathbf{x}_1$ |
| $\langle D\vert \hat{O} \vert D_{ij}^{ab}\rangle$ | 0 |

Table 1.1.: The Slater Condon rules for evaluating matrix elements between determinants and a one electron operator.

and 1.2.

FCI recovers all the electron correlation in a finite basis set. The storage requirements for Full CI (FCI) scale as:[1]

$$\mathcal{O}\left(\binom{N_\alpha}{m_\alpha}\binom{N_\beta}{m_\beta}\right) \tag{1.11}$$

Where $N_\alpha$ ($N_\beta$) is the number of alpha (beta) spin electrons and $m_\alpha$ ($m_\beta$) is the number of alpha (beta) molecular orbitals.

Brillouin's theorem states that the matrix elements between the Hartree-Fock determinant and single excitations with respect to the Hartree–Fock determinant are equal to zero:

$$\langle D_0\vert \hat{H} \vert D_i^a\rangle = 0 \;\forall\; i, a \tag{1.12}$$

Thus CID is the lowest level of Configuration interaction (for calculations of the ground state energy). [2]

Configuration Interaction is not widely used by computational chemists because FCI is too expensive for all but the smallest of molecules and truncated CI is not size consistent (the energy of two molecules separated by a large distance is not twice that of an isolated molecule).[13] In the next section we discuss coupled cluster theory which uses a different wavefunction ansatz such that it can be truncated in a size consistent way and scales reasonably.[19]

[1] If an iterative diagonalisation technique such as Lanczos [17] or Davidson [18] is used to diagonalise Eq. 1.10.

[2] CISD provides an improvement over CID as the matrix elements between the single and double excitations lower the energy.

### 1.1.3. Coupled Cluster Theory

Coupled Cluster (CC) theory uses an exponential wavefunction ansatz which allows truncation in a size consistent way.

$$|\psi\rangle = e^{\hat{T}} |D_0\rangle \tag{1.13}$$

Where:

$$\hat{T} = \sum_i \sum_a t_i^a \hat{a}_i^a + \sum_{i>j} \sum_{a>b} t_{ij}^{ab} \hat{a}_{ij}^{ab} + \sum_{i>j>k} \sum_{a>b>c} t_{ijk}^{abc} \hat{a}_{ijk}^{abc} + \dots \tag{1.14}$$

Where $\hat{a}_i^a$ is an operator which operates on a determinant such that it unoccupies spin orbital $i$, and occupies spin orbital $a$ (if $i$ is already unoccupied or $a$ occupied then the operation results in zero). The exponential operator means that even if $T$ is truncated to singles and doubles, the coupled cluster wavefunction will contain terms such as:

$$\sum_{i>j} \sum_{a>b} \sum_{k>l} \sum_{c>d} t_{ij}^{ab} t_{kl}^{cd} |D_{ijkl}^{abcd}\rangle \tag{1.15}$$

This involves a quadruple excitation relative to the Hartree-Fock determinant. To find the set of $t$ we solve a simultaneous set of equations:

$$\langle D_0| e^{-\hat{T}} \hat{H} e^{\hat{T}} |D_0\rangle = E, \langle D_i^a| e^{-\hat{T}} \hat{H} e^{\hat{T}} |D_0\rangle = 0, \langle D_{ij}^{ab}| e^{-\hat{T}} \hat{H} e^{\hat{T}} |D_0\rangle = 0 \dots \tag{1.16}$$

As $e^{-\hat{T}} \neq \left(e^{\hat{T}}\right)^\dagger$ coupled cluster theory is not variational (although approximations to variational coupled cluster theory have been investigated [20]).[3] This means that the total energy from a truncated coupled cluster calculation might be lower than the FCI energy. In practise CCSD provides very good energies close to the equilibrium geometry and it scales as $\mathcal{O}(n^6)$ making it useful to study small molecules. CCSDT

[3] Solving $\langle D_0| e^{\hat{T}^\dagger} \hat{H} e^{\hat{T}} |D_0\rangle$ (which is variational) results in the number of terms scaling factorially with system size even if truncated at second order. [20]

| | $\hat{O} = \sum_i \sum_{j>i} o(\mathbf{x}_i, \mathbf{x}_j)$ |
|---|---|
| $\langle D\| \hat{O} \|D\rangle$ | $\frac{1}{2} \sum_m^N \sum_o^N \int\int \chi_m^*(\mathbf{x}_1)\chi_o^*(\mathbf{x}_2) o(\mathbf{x}_1, \mathbf{x}_2) \chi_m^*(\mathbf{x}_1)\chi_o^*(\mathbf{x}_2) \mathrm{d}\mathbf{x}_1 \mathrm{d}\mathbf{x}_2$ |
| $\langle D\| \hat{O} \|D_i^a\rangle$ | $\sum_i^N \int\int \chi_m^*(\mathbf{x}_1)\chi_o^*(\mathbf{x}_2) o(\mathbf{x}_1, \mathbf{x}_2) \chi_m^*(\mathbf{x}_1)\chi_o^*(\mathbf{x}_2) \mathrm{d}\mathbf{x}_1 \mathrm{d}\mathbf{x}_2$ |
| $\langle D\| \hat{O} \|D_{ij}^{ab}\rangle$ | $\int\int \chi_m^*(\mathbf{x}_1)\chi_o^*(\mathbf{x}_2) o(\mathbf{x}_1, \mathbf{x}_2) \chi_m^*(\mathbf{x}_1)\chi_o^*(\mathbf{x}_2) \mathrm{d}\mathbf{x}_1 \mathrm{d}\mathbf{x}_2$ |

Table 1.2.: The Slater Condon rules for evaluating matrix elements between determinants and a two electron operator.

scales as $\mathcal{O}(n^8)$ making it expensive for all but small molecules. CCSD(T) includes a perturbative approximation to the triples and scales as $\mathcal{O}(n^7)$, this is known as the 'Gold Standard' in computational chemistry because of its accurate energies and scaling which is modest enough to make it applicable to small molecules. [3]

### 1.1.4. Møller Plesset Perturbation Theory

Another commonly used post Hartree–Fock method is Møller Plesset Perturbation Theory (MPPT). MPPT is based on Rayleigh-Schrödinger Perturbation Theory (RSPT) which is discussed in detail in most undergraduate quantum mechanics text books (e.g. [8]). In RSPT the Hamiltonian is written as the sum of two terms $\hat{H} = \hat{H}_0 + \lambda V$ where the first term $\hat{H}_0$ has known eigenfunctions and eigenvalues. By Taylor's expanding the wavefunction and energy, (around $\lambda = 0$) and equating terms in the same power of $\lambda$ one can derive corrections to the energy of arbitrary order in terms of the eigenfunctions of $H_0$. In Møller Plesset perturbation theory the Hamiltonian is written as:

$$\hat{H} = \hat{F} + \lambda \left( \hat{H} - \hat{F} \right) \tag{1.17}$$

where $\hat{F} = \sum_i \hat{f}(\mathbf{r}_i)$, which is only correct if $\lambda = 1$. The first two orders of the energy correction are shown in Table. 1.1.4.

There is no guarantee that the perturbation series converges.[4]

In practise second order Møller Plesset perturbation theory (MP2) often recovers a significant amount of the correlation energy near the equilibrium geometry and scales modestly as $\mathcal{O}(N^5)$.[19] Using an extremely accurate local approximation MP2 can be made to scale linearly. [21] For this reason it is often used when CCSD is too expensive. MP3 scales as $\mathcal{O}(N^6)$ [19] which is the same as CCSD but is less reliable and for this reason corrections to the energy beyond MP2 are seldom used.

[4] The convergence of the Taylor's expansion of the energy can be predicted using complex analysis. A discussion of this is beyond the scope of this thesis.[19]

| Order | Energy |
|---|---|
| 0 | $\langle D_0 \vert \hat{F} \vert D_0 \rangle$ |
| 1 | $\langle D_0 \vert \hat{H} \vert D_0 \rangle$ |
| 2 | $\langle D_0 \vert \hat{H} \vert D_0 \rangle + \sum_{j>i} \sum_{b>a} \frac{\langle D_0 \vert \hat{H} \vert D_{ij}^{ab} \rangle^2}{\langle D_0 \vert \hat{F} \vert D_0 \rangle - \langle D_{ij}^{ab} \vert \hat{F} \vert D_{ij}^{ab} \rangle}$ |

Table 1.3.: The MPn energy from zeroth to second order.[13]

### 1.1.5. Other Methods

This review would be left incomplete without mention of Density Functional Theory (DFT). DFT is the most widely used method in computational chemistry. Often it can produce results as accurate as a low level coupled cluster method such as CCSD, but with far more modest scaling $\mathcal{O}(N^3)$. The Hohenberg-Kohn theorems prove that the energy is a functional of the electron density. Unfortunately this universal functional is unknown.[22] Only the functional of the classical electrostatic interaction is well known. The kinetic energy can be calculated approximately by reintroducing a single determinant wavefunction, the so called Kohn-Sham ansatz.[23] The rest of the terms are grouped together and called the exchange and correlation functional.

As the form of the exchange and correlation functional is unknown, approximations based on systems such as the uniform electron gas [24] or the helium atom [25] are used. The most commonly used function B3LYP [26, 27] mixes this approach with an empirical amount of exchange from Hartree-Fock. B3LYP provides accurate results on a wide range of systems, for example it provides good agreement with the G2 set of experimental data.[28] However it has problems modelling weak interactions and excited states. Recently, improved functionals have been developed to solve these problems.[29] Finding a universal functional which is accurate for a broad range of different molecules has so far proved elusive.

### 1.1.6. Why do we need high accuracy?

We have reviewed some of the multitude of methods commonly used by computational chemists. We now briefly discuss when we need the highly accurate methods as described in this thesis to accurately predict experimental results.

The atomisation energy of a molecule is defined as the difference between the energy of the ground state of its constituent atoms and the energy of the molecule at the equilibrium geometry. Atomisation energies are difficult to calculate accurately because all the bonds are broken in a molecule. Helgaker et al. [30] calculated the atomisation energies of 17 closed shell organic molecules using a very large cc-pCV6Z basis set and found a mean absolute deviation from experiment of 423.0, 36.2, 37.2 and 4.6 kJ mol$^{-1}$ using Hartree–Fock, MP2, CCSD and CCSD(T) respectively. So CCSD(T) with a very large basis set is needed to get close to chemical accuracy.

The dipole moment is obtained by differentiating the total energy with respect to the electric field strength. An investigation of BH and HF by Halkier et al. showed that to get agreement with experimental values one had to use CCSD(T) as well as

an extrapolation to the complete basis set limit. [31]

High accuracy is also required to converge bond lengths to experimental accuracy. Helgaker et al. [30] calculated the mean absolute error relative to experiment for 27 small molecules. They found that using CCSD(T) and a cc-pVQZ basis they could converge bond distances to 0.16 pm, although this high accuracy results from error cancellation (bond distances are shortened by about 0.1 pm by increasing the basis to cc-pV6Z and by 0.02 pm by going to CCSDT). Again for highly accurate harmonic forces we need to use a large basis set and a high coupled cluster truncation level. For example to get within 1 $cm^{-1}$ of the exact non relativistic limit for $N_2$ we need to use CCSDTQ5 and a cc-pCV5Z basis. [30] Experimental measurements at this accuracy can be challenging even for stable molecules. However the aim of these highly accurate calculations is to also make measurements of compounds which are hard to measure experimentally such as transient intermediates in chemical reaction mechanisms, or in extreme environments such as the interstellar medium.

## 1.2. Quantum Monte Carlo

Quantum Monte-Carlo (QMC) differs from the quantum chemical approaches discussed in the previous section. As with all Monte-Carlo methods a pseudo-random sequence is used to sample an underlying mathematical equation (the exact equation depends on the type of QMC).

Monte-Carlo simulations can be split into two classes: static and dynamic Monte-Carlo. In static Monte-Carlo simulations the probability of making a move has no dependence on either the present state of the simulation or any states that have been visited in the past. Otherwise the process is classed as dynamic Monte-Carlo. Monte-Carlo integration, which we discuss in the next section, is an example of static Monte-Carlo, where as VMC, DMC and FCIQMC which we discuss in Secs. 1.2.2, 1.2.3 and 1.3 respectively are all examples of dynamic Monte-Carlo.

We now briefly review two of the methods most commonly used in QMC to provide context for comparison with Full Configuration Interaction Quantum Monte Carlo which is used throughout this thesis.

### 1.2.1. Monte-Carlo Integration

Consider the definite integral of a function $f(x)$ of one variable:

$$I = \int_a^b f(x)\mathrm{d}x \tag{1.18}$$

$I$ can be estimated using a sequence of random numbers $x_1, x_2, \ldots x_N$ sampled from a uniform distribution in the interval $[a, b]$.

$$I \approx \tilde{I} = (b-a)\frac{1}{N}\sum_{i=1}^{N} f(x_i) \tag{1.19}$$

$I$ will have variance:

$$\sigma_{\tilde{I}}^2 = \frac{1}{N(N-1)}\sum_{i=1}^{N}(f(x_i) - \tilde{I})^2 \tag{1.20}$$

$\sigma_{\tilde{I}}$ provides an estimate of the standard error meaning $I + \sigma_I > \tilde{I} > I - \sigma_I$ is true 68% of time. [5] This standard error decays as $N^{-1/2}$, the so called square root law. The asymptotic form of the decay as a function of the number of points is faster for Monte-Carlo than using an estimate based on a grid such as Simpson's rule if

[5] We assume that the central limit theorem is obeyed.

the number of dimensions is more than eight.[6] However the prefactor of the decay might well scale unfavourably with the number of dimensions. The square root law can be generalised to almost all Monte-Carlo methods.

By rewriting Eq. 1.18 as:

$$I = \int_a^b \frac{f(x)p(x)}{p(x)} \mathrm{d}x \tag{1.21}$$

we can improve the efficiency of the Monte-Carlo procedure by importance sampling. A sequence of random numbers $x_1, x_2, \ldots x_N$, distributed according to $p(x)$, in the interval $[a, b]$, is used to sample $f(x)/p(x)$:

$$I \approx (b-a)\frac{1}{N}\sum_{i=1}^{N} \frac{f(x_i)}{g(x_i)} \tag{1.22}$$

The flatter the transformation makes $f(x)/p(x)$ the more efficient the importance sampled Monte-Carlo integration is. These equations can be generalised to many variables.

### 1.2.2. Variational Monte Carlo

Variational Monte Carlo (VMC) is one of the most conceptually simple forms of QMC. VMC relies on the variational principle.

$$E \geq \frac{\int \psi^*(\mathbf{R})\hat{H}(\mathbf{R})\psi(\mathbf{R})\mathrm{d}\mathbf{R}}{\int \psi^*(\mathbf{R})\psi(\mathbf{R})\mathrm{d}\mathbf{R}} \tag{1.23}$$

We have grouped the electronic coordinates $\mathbf{R} = (\mathbf{r}_1, \mathbf{r}_2, \ldots \mathbf{r}_N)$ for convenience. Both the numerator and denominator in the above equations are integrals of $3N$ variables.

To sample Eq. 1.23 we define a local energy:

$$E_l(\mathbf{R}) = \frac{\hat{H}(\mathbf{R})\psi(\mathbf{R})}{\psi(\mathbf{R})} \tag{1.24}$$

If $\psi(\mathbf{R})$ is an exact eigenfunction of $\hat{H}(\mathbf{R})$ then $E_l(\mathbf{R})$ is the ground state energy for all values of $\mathbf{R}$. Otherwise $E_l(\mathbf{R})$ has $\mathbf{R}$ dependence, if we substitute the equation

[6] Simpson's rule decays as $N^{-4/d}$, where $d$ is the number of dimensions.[32]

for the local energy into Eq. 1.23, then:

$$E > \frac{\int \psi^*(\mathbf{R})\psi(\mathbf{R})E_l(\mathbf{R})\mathrm{d}\mathbf{R}}{\int \psi(\mathbf{R})\psi(\mathbf{R})\mathrm{d}\mathbf{R}} \tag{1.25}$$

So we sample $E_l(\mathbf{R})$ using a set of random vectors distributed according to $\psi^*(\mathbf{R})\psi(\mathbf{R})/\int \psi(\mathbf{R})\psi(\mathbf{R})\mathrm{d}\mathbf{R}$. Unfortunately we do not know the normalisation $(\int \psi(\mathbf{R})\psi(\mathbf{R})\mathrm{d}\mathbf{R})$ for an arbitrary wavefunction so we may use the Metropolis algorithm (see Fig. 1.1 to sample $E_l(\mathbf{R})$).

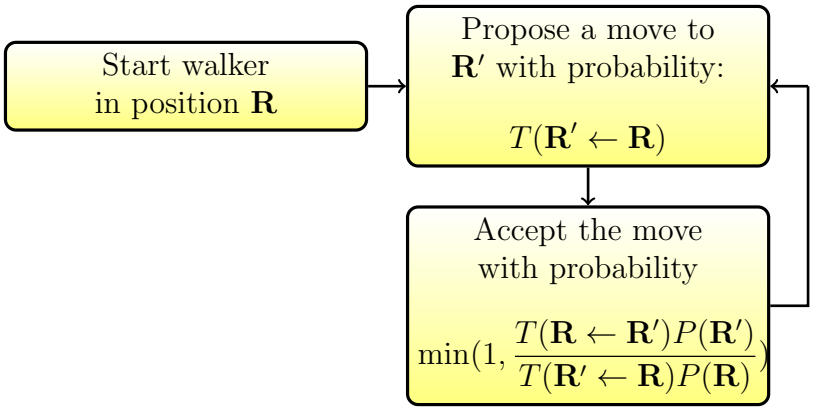



Figure 1.1.: The Metropolis algorithm used to sample a function $P(\mathbf{R})$ with an unknown normalisation, first described in Ref. [33]. An ensemble of walkers samples this function by moving around the space $R$ according to these rules. The probability of a walker proposing a move $T(\mathbf{R}' \leftarrow \mathbf{R})$ only affects the efficiency of the Monte-Carlo algorithm itself. Often one uses a Gaussian function centered around $\mathbf{R}$ with a width such that the average probability of making a move is approximately 50%.

The VMC algorithm depends on the quality of the wavefunction being sampled (if the integrals in Eq. 1.23 were to be solved analytically then this would provide the same energy as VMC in the limit of an infinite number of iterations). The VMC technique is useful for optimising a wavefunction which depends on some free parameters. Techniques such as energy minimisation [34] and variance minimisation [35] have been developed for this purpose. As no analytic integration has to be performed, more complicated wavefunctions which are expensive and/or difficult to integrate can be used. For example a significant amount of electron correlation can be added to a single Slater determinant by multiplying it by a Jastrow factor [36, 37]. The Jastrow factor contains terms which depend on the separation between two electronic coordinates and also the separation between an electronic coordinate and the position of a fixed nucleus. It is well known that the exact wavefunction

must satisfy a set of conditions when these separations tend towards zero [38]. The use of a Jastrow factor automatically satisfies these conditions.

Often VMC is used to optimise the wavefunction as a precursor to Diffusion Monte-Carlo DMC. DMC (the topic of the next section) can improve the quality of the wavefunction.

### 1.2.3. Diffusion Monte Carlo

DMC belongs to a class of projector Monte-Carlo algorithms For a more detailed explanation of the concepts presented here see [39] or [40]. DMC relies on the imaginary time Schrödinger equation:

$$-\frac{\partial}{\partial\tau}\psi(\mathbf{R},\tau) = \left(\hat{H} - S\right)\psi(\mathbf{R},\tau) \tag{1.26}$$

Where $\tau$ is the imaginary-time (substituting $\tau = it$ into Eq. 1.26 gives the time-dependent Schrödinger equation). $S$ is used to control the normalisation. If one expands $\psi(\mathbf{R},\tau)$ in the basis of eigenstates of $\hat{H}$, which we denote $\Psi_0, \Psi_1 \ldots$ (which have no $\tau$ dependence) and have eigenvalues $E_0, E_1 \ldots$:

$$\psi(\mathbf{R},\tau) = a_0(\tau)\Psi_0(\mathbf{R}) + a_1(\tau)\Psi_1(\mathbf{R}) + \ldots \tag{1.27}$$

Substituting this expansion into Eq. 1.26, multiplying by $\Psi_i(\mathbf{R})$ and integrating over the electronic coordinates gives:

$$a_i(\tau) = e^{-(E_i - S)\tau}a_i(0) \tag{1.28}$$

Thus all the excited states die faster than the ground state $\Psi_0(\mathbf{R})$. If we additionally control $S$ such that $\psi(\tau)$ remains normalised (we will discuss this later) then in the limit of infinite imaginary time the ground state will emerge.

In order to perform this projection we consider the imaginary time Schrödinger equation in two parts. Firstly the kinetic energy:

$$-\frac{\partial}{\partial\tau}\psi(\tau) = -\sum_i \frac{\nabla_i^2}{2}\psi(\tau) \tag{1.29}$$

The equation above can be rewritten in terms of its Green's function:

$$\psi(\mathbf{R},\tau+\delta\tau) = \int G(\mathbf{R} \leftarrow \mathbf{R}',\delta\tau)\psi(\mathbf{R}',\tau)d\mathbf{R}' \tag{1.30}$$

In the case of the kinetic energy:

$$G(\mathbf{R} \leftarrow \mathbf{R}', \delta\tau) = \frac{1}{(2\pi\delta\tau)^{3N/2}} e^{-\frac{(\mathbf{R}-\mathbf{R}')^2}{2\delta\tau}} \tag{1.31}$$

This Green's function obeys the same equation as the wavefunction with the boundary condition that when $\delta\tau = 0$ , $G(\mathbf{R} \rightarrow \mathbf{R}', \delta\tau) = \delta(\mathbf{R} - \mathbf{R}')$. For the second term in the Hamiltonian:

$$-\frac{\partial}{\partial\tau}\psi(\tau) = V(\mathbf{R})\psi(\tau) \tag{1.32}$$

we have:

$$G(\mathbf{R} \rightarrow \mathbf{R}', \delta) = e^{-(v(\mathbf{R})-v(\mathbf{R}'))\delta\tau} \tag{1.33}$$

The actual Hamiltonian is the sum of these two terms; unfortunately the Green's function for this sum cannot be written in a compact form. [7]

So instead in DMC the Trotter-Suzuki formula for the exponential of a sum of operators provides an approximation for the Green's function: [47]

$$e^{\tau(\hat{A}+\hat{B})} = e^{\tau\hat{B}/2} e^{\tau\hat{A}} e^{\tau\hat{B}/2} + \mathcal{O}(\tau^3) \tag{1.34}$$

So the Green's function for the DMC process is:

$$G(\mathbf{R} \rightarrow \mathbf{R}', \delta\tau) = \frac{1}{(4\pi\delta\tau)^{3N/2}} e^{-\frac{(\mathbf{R}-\mathbf{R}')^2}{2\delta\tau}} e^{-(v(\mathbf{R})-v(\mathbf{R}')-S)/2} + \mathcal{O}(\tau^3) \tag{1.35}$$

The DMC algorithm can be greatly improved by making the importance sampling transformation: $\psi(\mathbf{R}, \tau) = \psi_t(\mathbf{R})/f(\mathbf{R}, \tau)$. The imaginary time of evolution of $f(\mathbf{R}, \tau)$ can be found by multiplying Eq. 1.26 by $\psi_t(\mathbf{R})$.

$$\frac{\partial}{\partial\tau} f(\mathbf{R}, \tau) = -\frac{1}{2}\nabla^2 f(\mathbf{R}, \tau) + \nabla.[\mathbf{V}(\mathbf{R}) f(\mathbf{R}, \tau)] + [E_l(\mathbf{R}) - S] f(\mathbf{R}, \tau) \tag{1.36}$$

Where $\mathbf{V}$ is the drift velocity:

$$\mathbf{V}(\mathbf{R}) = \frac{\nabla\psi_t(\mathbf{R})}{\psi_t(\mathbf{R})} \tag{1.37}$$

[7] Green's Function Quantum Monte Carlo (GFQMC) has been developed to sample the exact Green's function using a linear projector instead of Eq. 1.26.[41, 42] GFQMC can produce results of a similar quality to that of DMC (it has been successfully applied to a number of small molecules [43], solids [44] and model systems [45] ) and doesn't have a timestep error. As the timestep error is easily extrapolated away in DMC and GFQMC is more expensive, it has not found as widespread applicability as DMC. [46]

In a similar way to before, an approximate Green's function for Eq. 1.36 can be constructed:

$$G(\mathbf{R} \rightarrow \mathbf{R}', \delta\tau) = \frac{1}{(4\pi\delta\tau)^{3N/2}} e^{-\frac{(\mathbf{R}-\mathbf{R}'-\tau\mathbf{V}(\mathbf{R}'))^2}{2\delta\tau}} e^{-(E_l(\mathbf{R})-E_l(\mathbf{R}')-S))/2} + \mathcal{O}(\tau^3) \quad (1.38)$$

The application of this Green's function Eq. 1.38 is simulated by a discrete set of walkers in the space of the electron coordinates $\mathbf{R}$. This is done in two stages, firstly each walker attempts to move to a new set of electronic coordinates $\mathbf{R}'$ where $\mathbf{R}'$ is distributed according to:

$$P_m(\mathbf{R} \rightarrow \mathbf{R}') = \frac{1}{(4\pi\delta\tau)^{3N/2}} e^{-\frac{(\mathbf{R}-\mathbf{R}'-\tau\mathbf{V}(\mathbf{R}'))^2}{2\delta\tau}} \quad (1.39)$$

The change in normalisation caused by the other term is simulated using a birth/death process:

$$P_d = e^{-\delta\tau(E_l(\mathbf{R})-E_l(\mathbf{R}')-2S)/2} \quad (1.40)$$

The walker continues its evolution with probability $P_d - \lfloor P_d \rfloor$ and $\lfloor P_d \rfloor$ new walkers are created ($\lfloor x \rfloor$ denotes the largest integer less than $x$).

There are two possible estimators of the energy. Firstly $S$ can be controlled such that the population of walkers remains constant on average. If this condition is satisfied then the average $S$ will be the ground state energy due to Eq. 1.28. The population has to be controlled slowly in order to prevent the introduction of a significant bias. Secondly we can use the mixed estimator:

$$E = \lim_{\tau\rightarrow\infty} \frac{\int \psi_t(\mathbf{R})\hat{H}e^{-\hat{H}\tau}\psi_t(\mathbf{R})d\mathbf{R}}{\int \psi_t(\mathbf{R})e^{-\hat{H}\tau}\psi_t(\mathbf{R})d\mathbf{R}} = \lim_{\tau\rightarrow\infty} \frac{\int f(\mathbf{R},\tau)E_l(\mathbf{R})d\mathbf{R}}{\int f(\mathbf{R},\tau)f(\mathbf{R},\tau)d\mathbf{R}} \quad (1.41)$$

This can be estimated by sampling $E_l(\mathbf{R})$ using the set of walkers from the DMC calculation once a significant amount of imaginary time has elapsed. This works because these walkers are distributed according to $f(\mathbf{R}, \infty)$.

So far the algorithm we have described ignores the fact that the wavefunction must be anti-symmetric on exchange of two electronic coordinates. All practical DMC calculations [8] use the fixed node approximation where the space $\mathbf{R}$ is partitioned 'a priori' into regions where the wavefunction is constrained to be positive or negative. This is done using the nodes of $\psi_t(\mathbf{R})$.[39] Moves in which the walkers cross a nodal

[8] In a few model systems it has been possible to get round the fixed node approximation using DMC.[45]

boundary are rejected.

Recently backflow transformations have been used in an attempt to optimise the nodal surface during a DMC calculation.[48] Although this has been shown to improve the nodal surface greatly of both VMC and DMC wavefunctions, it cannot completely capture the exact nodal surface. This means that fixed node approximation introduces an error which is only systematically improvable by using the nodes from a better trial wavefunction. Eventually this will involve using a method from computational chemistry which is more expensive than DMC. Fixed node DMC calculations obey the fixed node variation principle [39] meaning that the wavefunction sampled has the lowest possible energy for a given nodal surface.[9]

[9]For some simple molecular [43] and solid systems [44] release node GFQMC has been used to systematically improve the nodal surface during a calculation.

## 1.3. Full Configuration Interaction Quantum Monte Carlo

We now turn to the more recently developed method FCIQMC which the work in this thesis is based on. FCIQMC marries together FCI discussed in Sec. 1.1.2 with QMC. FCIQMC doesn't require the use of a fixed node approximation (because it is performed in Slater Determinant Space) and a FCIQMC calculation should converge to the ground state FCI energy in the limit of an infinite number of iterations. This means in principle it is possible to estimate the non-relativistic energy of the system by extrapolating to the complete basis set limit.[49] Unfortunately the storage requirements almost certainly scale in the same way as FCI with the number of electrons and the basis set size.[50] This potentially limits the applicability to small molecules, however FCIQMC with the initiator approximation (discussed in Sec. 1.3.1) has provided energies of FCI quality for some of the largest molecules to date.[51]

Like DMC, FCIQMC is a projector method but unlike DMC, FCIQMC uses a linear projector:

$$|\psi(\tau+\delta\tau)\rangle = \left(1-\left(\hat{H}-S\right)\delta\tau\right)|\psi(\tau)\rangle \tag{1.42}$$

to update the representation of the wavefunction on every iteration. [10]

We begin with the (CI) ansatz where the wavefunction is a linear combination of Slater determinants:

$$|\psi\rangle = \sum_i C_i |D_i\rangle . \tag{1.43}$$

It is convenient (though not necessary[52]) to represent the coefficients $C_i$ by a discrete set of signed particles or psips. [53] To make the distinction between continuous and discrete quantities we shall rewrite Eq. 1.42 as:

$$d_i(\tau+\delta\tau)|D_i\rangle = \sum_i \left(\langle D_i|\hat{H}|D_j\rangle - S\delta_{ij}\right)\delta\tau d_j(\tau) + d_i(\tau) \tag{1.44}$$

In order to sample the sum on the right hand side of Eq. 1.44 the algorithm proposed by Booth *et al.*[49] is used.

The off-diagonal term is sampled by each psip (we denote the determinant it is on

[10] Deterministic powers methods based on Eq. 1.42 are often used in FCI when storage of the entire FCI matrix is too computationally expensive. This forms the basis of the Lanczos [17], and Davidson methods [18] often used by computational chemists to find the eigenvectors corresponding to a few of the smallest eigenvalues in deterministic CI. These require the storage space of a few times the number of desired eigenvectors.

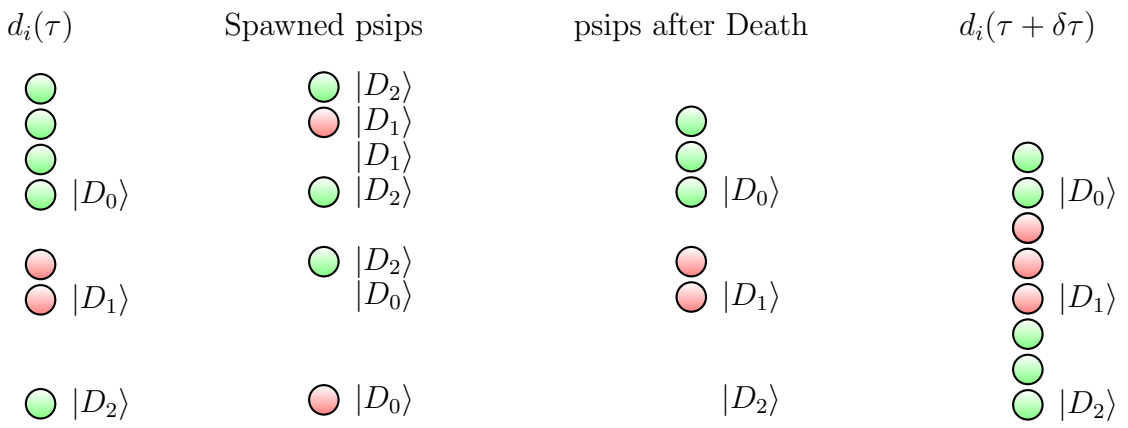



Figure 1.2.: A diagrammatic representation of an example of a single iteration of the FCIQMC algorithm. A green (red) circle represents a positive (negative) psip. The action of Eq. 1.42 on the representation of the wavefunction from the end of the last step (first column) is sampled to update the representation (fourth column). This is done in three stages: firstly each psip attempts to spawn a child psip onto a random determinant (this is shown in the second column). Secondly each psip dies or is cloned and these psips are added/removed from the representation on the last iteration (the result of this is shown in the third column). Finally newly spawned psips (second column) and surviving parent psips (third column) are annihilated to form the complete updated representation (fourth column).

by $i$) attempting to spawn a child psip on a randomly selected determinant $j$ with probability:

$$P_s(i \to j) = \frac{\left|\langle D_i| \hat{H} |D_j\rangle\right| \delta\tau}{p_{gen}(i \to j)} \tag{1.45}$$

Where $p_{gen}(i \to j)$ is the probability that a psip on determinant $i$ selects determinant $j$ to spawn onto. The sign of any successfully spawned child psip is the same as its parent if $\langle D_i| \hat{H} |D_j\rangle < 0$ or opposite to its parent if $\langle D_i| \hat{H} |D_j\rangle > 0$.

The diagonal term is sampled by each psip dying or cloning. This occurs with probability:

$$P_d(i) = \left|\langle D_i| \hat{H} |D_i\rangle - S\right| \delta\tau \tag{1.46}$$

If $P_d > 0$ then the psip dies (the population of positive or negative psips moves closer to zero) and if $P_d < 0$ the psip is cloned (the population moves away from zero).

The final step is annihilation, psips of opposite signs on the same determinant annihilate so that each determinant has a population of psips with a unique sign

after every step of $\delta\tau$. This term is essential so that the sign structure of the wavefunction emerges.

Even so for a finite population of psips it is not immediately obvious that the ground state should emerge as we do not sample every possible term in the sum in Eq. 1.44 every iteration. A detailed discussion of the sign problem in FCIQMC is left until Sec. 1.3.3, here we simply make the observation as in [49] that setting the shift to $\langle D_0| \hat{H} |D_0\rangle$ initially means that the population will grow exponentially, plateau and then grow exponentially again (a single exponential growth is seen for systems where every $\langle D_i| \hat{H} |D_j\rangle > 0$). Only after the plateau phase does $\langle d_i(\tau)\rangle_\tau$ become the ground state eigenvector (see Fig. 1.3).

The choice of $p_{gen}$ must balance the computational cost of its computation with the cost of attempting and failing to spawn onto a determinant for which $\langle D_i| \hat{H} |D_j\rangle = 0$. A detailed discussion of this is left until Sec. 1.3.2.

Following an equilibration phase (after the simulation has exited the plateau phase), the shift is periodically updated every $A$ steps to control the psip population using[49]

$$S(\tau + A\delta\tau) = S(\tau) - \frac{\gamma}{A\delta\tau} \log \frac{N(\tau + A\delta\tau)}{N(\tau)}, \tag{1.47}$$

where $\gamma$ is a damping factor and $N(\tau)$ is the total number of psips at time $\tau$. Repeated substitution of Eq. 1.47 into itself yields:

$$S(\tau + A\delta\tau) = S(0) - \xi \log \frac{N(\tau + A\delta\tau)}{N_s}, \tag{1.48}$$

where $S(0)$ is the initial value of the shift (in this work the Hartree–Fock energy), $N_s$ is the population at the end of the equilibration phase and $\xi = \gamma/(A\delta\tau)$ is usually fixed during a simulation. Eq. 1.48 implies that FCIQMC is an example of Markov Chain Monte Carlo (MCMC), the implications of which we shall discuss in Chapter 2.

The correlation energy can also be found by:

$$E_{\text{Proj}} = \frac{\langle D_0|\hat{H}|e^{-\hat{H}\tau}D_0\rangle}{\langle D_0|e^{-\hat{H}\tau}|D_0\rangle} = \frac{\sum_{i\neq 0} H_{0i}n_i}{n_0}, \tag{1.49}$$

where the trial state, $|D_0\rangle$, is typically the Hartree–Fock determinant. The variance of the projected estimator is generally smaller than that of the shift and can be reduced further by a multi-determinant trial function.[52]

Both estimators are serially correlated as the state of the simulation at one

timestep is heavily dependent on the state at the previous timestep. We use an automated iterative blocking algorithm[54, 55, 56, 57] to accurately estimate the stochastic error in all FCIQMC calculations presented in this thesis.

### 1.3.1. The Initiator Approximation

The FCIQMC is only applicable to systems where the vector $d_i(\tau)$ can be stored. Plateau height has been shown to scale linearly with the size of the Hilbert space for a number of light atoms ($Z \leq 11$) [59]. Some model systems have displayed a sublinear scaling [50]. If the height of the plateau is large enough so that $d_i(\tau)$ cannot be stored then FCIQMC is essentially a memory limited process. Thus approximations to reduce the sign problem (which we discuss in detail in Sec. 1.3.3) in FCIQMC could be effective in making the method applicable to much larger molecules.

In the initiator approximation [60] the Hilbert space is partitioned into two subspaces on each iteration. Determinants with a population above a certain threshold (the initiator threshold) are placed into what we shall call the initiator space and determinants with less than this population are placed into the non-initiator space. The spawning step is modified such that determinants in the non-initiator space can only spawn onto determinants which are already occupied. Determinants in the initiator space can spawn onto unoccupied determinants. The diagonal death and the annihilation step remain the same as in FCIQMC.

The approximation greatly reduces the magnitude of the sign problem by making it much less likely that a determinant will change sign. This introduces an error in the energy which is systematically improvable (in the limit of a large population of psips all determinants which are occupied in the ground state have a population greater than the initiator threshold). In order to remove the initiator error, multiple calculations are performed with increasing $N_p$ until the energy becomes within statistical errors of the point before. There is no guarantee that error decays monotonically towards the ground state and thus initiator FCIQMC (iFCIQMC) energy produced from this process may not be variational. However for most systems of interest a monotonic [59] or near monotonic decay [61] has been observed.

### 1.3.2. Random Excitation Generators

The random generation of the child determinant from the parent determinant can have a large effect on the efficiency of an FCIQMC calculation. Ideally one would

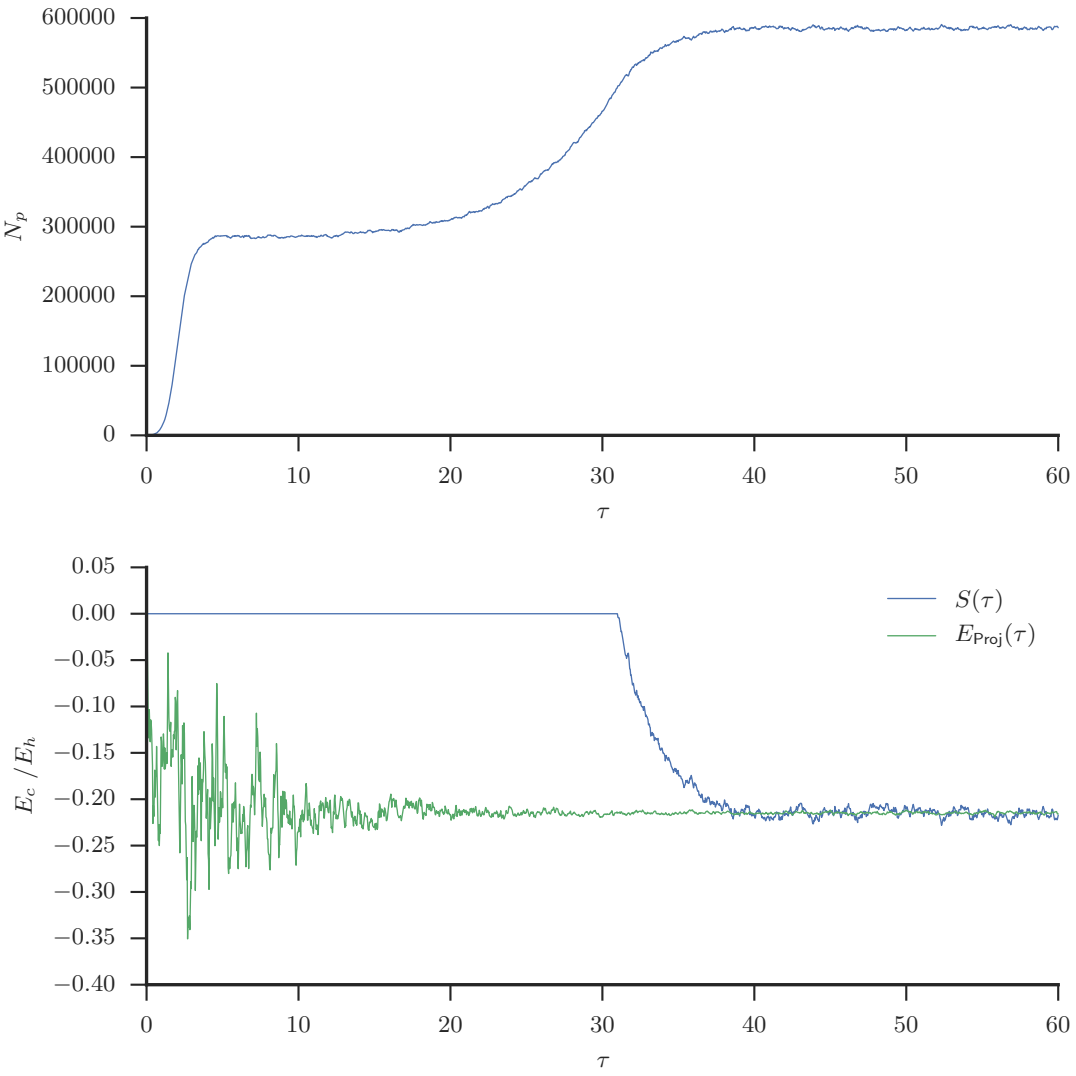



Figure 1.3.: An FCIQMC calculation of the ground state energy of the neon atom in an aug-cc-pVDZ basis set, $\delta\tau = 0.003$, $A = 10$, and $\xi = 1.39$. Top: The total population of psips $N_p$ as a function of $\tau$. Bottom: The shift, $S(\tau)$, and the projected energy, $E_{\text{Proj}}$, as a function of $\tau$. The population initially grows exponentially before slowing down to a plateau where the sign structure of the wave function emerges. After this the population grows exponentially again and the projected energy becomes a valid estimator of the ground state. Once the population reaches a desired level ($N_0 = 500000$ in this case) population control stabilises the population and the shift becomes a valid estimator of the projected energy. The calculation was performed with the HANDE Quantum Monte-Carlo package [58].

select a child determinant with a probability proportional to its matrix element. This would involve computing every possible matrix element and would thus be extremely expensive but the error bar would converge fastest as a function of the total number of iterations.

A simple optimisation (at least for the Hamiltonians of interest in chemistry and condensed matter physics which have at most two electron operators) is to only attempt spawning onto child determinants which are a single or double excitation with respect to the parent determinant. As per the Slater-Condon rules (Table. 1.1 and Table. 1.2) any matrix element for which the bra and the ket differ by more than a double excitation is zero. The number of single and double excitations can easily be computed using this excitation scheme.

In addition to this the symmetry of the occupied molecular orbitals can also be taken into account. If the sum of the irreducible representation of each molecular orbital is not equal in the bra and the ket then the matrix element will be zero. Unfortunately counting the number of child determinants where this is not true is non-trivial (it scales linearly with the number of molecular orbitals), so the approach of computing all possible determinants and explicitly testing this symmetry constraint is taken. For small molecules of high symmetry this additional expense is often worthwhile. By this we mean that the error bar converging faster as a function of the number of iterations more than makes up for the extra time taken to perform each iteration. If a large enough basis set is used then this will no longer be true.

### 1.3.3. The Sign Problem

The sign problem in FCIQMC was first explained in [62]. The original FCIQMC paper [49] made the observation that the sign structure emerges after the plateau and the average of the energy estimators becomes the ground state energy after this point. This section briefly summarises the points made in [62].

If the annihilation step is not performed, both positive and negative psips can exist on the same determinant. These positive and negative psips can both spawn child psips and perform diagonal death. We denote the number of positive (negative) psips as $d_i^+$ ($d_i^-$) on determinant $i$. We also denote the negative elements of the shifted Hamiltonian matrix $\left(\hat{H} - S\right)$ as $T_{ij}^-$ and the positive elements as $T_{ij}^+$.

$$\frac{\partial d_i^+}{\partial \tau} = \sum_j \left( T_{ij}^+ d_j^+ + T_{ij}^- d_j^- \right) \tag{1.50}$$

$$\frac{\partial d_i^-}{\partial \tau} = \sum_j \left( T_{ij}^+ d_j^- + T_{ij}^- d_j^+ \right) \tag{1.51}$$

If we uncouple these equations:

$$\frac{\partial (d_i^+ + d_i^-)}{\partial \tau} = \sum_j \left( T_{ij}^+ + T_{ij}^- \right) (d_j^+ + d_j^-) \tag{1.52}$$

$$\frac{\partial (d_i^+ - d_i^-)}{\partial \tau} = \sum_j \left( T_{ij}^+ - T_{ij}^- \right) (d_j^+ - d_j^-) \tag{1.53}$$

then the largest eigenvalue of $\left( T_{ij}^+ + T_{ij}^- \right)$ is always larger than the largest eigenvalue of $\left( T_{ij}^+ - T_{ij}^- \right)$ and thus as $\tau \to \infty$ the $\left( T_{ij}^+ + T_{ij}^- \right)$ signal dominates.

If we now add an annihilation step it is difficult to write down exactly, so instead we use a finite flux of positive and negative walkers which annihilate.

$$\frac{\partial d_i^+}{\partial \tau} = \sum_j \left( T_{ij}^+ d_j^+ + T_{ij}^- d_j^- \right) - k d_i^+ d_i^- \tag{1.54}$$

$$\frac{\partial d_i^-}{\partial \tau} = \sum_j \left( T_{ij}^+ d_j^- + T_{ij}^- d_j^+ \right) - k d_i^+ d_i^- \tag{1.55}$$

Again uncoupling the equations:

$$\frac{\partial (d_i^+ + d_i^-)}{\partial \tau} = \sum_j \left( T_{ij}^+ + T_{ij}^- \right) (d_j^+ + d_j^-) - 2k d_i^+ d_i^- \tag{1.56}$$

$$\frac{\partial (d_i^+ - d_i^-)}{\partial \tau} = \sum_j \left( T_{ij}^+ - T_{ij}^- \right) (d_j^+ - d_j^-) \tag{1.57}$$

The annihilation term is quadratic in the number of psips and eventually this suppresses the growth in the $d_i^+ + d_i^-$ signal relative to the $d_i^+ - d_i^-$. In this way the eigenvalue of $d_i^+ - d_i^-$ gets projected out after many iterations of the FCIQMC algorithm.

### 1.3.4. An Aside: FCIQMC and the Hubbard model

In Chapter 6 we test an implementation of FCIQMC on a novel data flow computer architecture. To avoid the complexity of summing the one and two electron integrals when computing the matrix elements of the Hamiltonian (using the Slater-Condon rules) we have instead investigated the Hubbard model. [63] The Hubbard model is

important in solid state physics: it describes the metal insulator transition and is thought to describe a superconducting phase. [64]

The model consists of a lattice of sites in two dimensions (we only investigate the Hubbard model in two dimensions in this thesis). Each site has four neighbours (we use periodic boundary conditions in all Hubbard model calculations in this thesis).

The Hubbard Hamiltonian is:

$$\hat{H} = -t \sum_{\langle ij \rangle, \sigma} (a^{\dagger}_{i,\sigma} a_{j,\sigma} + a^{\dagger}_{j,\sigma} a_{i,\sigma}) + U \sum_{i}^{N} n_{i\uparrow} n_{j\downarrow}, \tag{1.58}$$

the operator $a^{\dagger}_{i,\sigma}$ creates an electron on site $i$ (on the lattice) with a spin $\sigma$ ($\sigma$ may be spin up $\uparrow$ or down $\downarrow$), the operator $a_{i,\sigma}$ unoccupies a site, $n_{i\uparrow}$ is the number of electrons on site $i$ which are spin up. Each site can be occupied by at most two electrons, of opposite spins. The sum over $\langle i, j \rangle$ only includes terms when $i$ is a nearest neighbour to $j$.

The operators obey the anti-commutator relations:

$$\left\{ a_{i\sigma}, a^{\dagger}_{j\rho} \right\} = \delta_{i,j} \delta_{\sigma,\rho} \;\; \{a_{i\sigma}, a_{j\rho}\} = 0 \;\; \left\{ a^{\dagger}_{i\sigma}, a^{\dagger}_{j\rho} \right\} = 0 \tag{1.59}$$

So that any product of creation operators obeys the anti-symmetry relations and behaves in the same way as a Slater determinant.

We can change basis from the real space basis in Eq. 1.58 to a reciprocal space basis using the transformation:

$$c^{\dagger}_{k,\sigma} = \frac{1}{\sqrt{M}} \sum_{i} a^{\dagger}_{i,\sigma} e^{i\mathbf{k}.\mathbf{i}}, \tag{1.60}$$

where $\mathbf{i}$ is the vector from an arbitrary defined origin to site $i$ and $\mathbf{k}$ is a vector in reciprocal space (in the first Brillouin zone) to the reciprocal lattice point $K$ and $M$ is the number of sites in the lattice. For a discussion of reciprocal space in a solid state physics context see [65]. The Hamiltonian can be written in reciprocal space as:

$$\hat{H} = \sum_{k,\sigma} \epsilon_k c^{\dagger}_{k,\sigma} c_{k,\sigma} + \frac{U}{M} \sum_{k_1,k_2,k_3} c^{\dagger}_{k_1,\uparrow} c^{\dagger}_{k_2,\downarrow} c_{k_3,\uparrow} c_{k_1+k_2+k_3,\downarrow}, \tag{1.61}$$

the vector $\mathbf{k}_1 + \mathbf{k}_2 + \mathbf{k}_3$ is mapped back into the first Brillouin zone if necessary and $\epsilon_k = -2t\left(\cos(k_x a) + \cos(k_y a) + \cos(k_z a)\right)$, where $a$ is the lattice parameter and $k_x$, $k_y$ and $k_z$ are the components of $\mathbf{k}$ in $x$, $y$ and $z$ directions respectively. This is the basis where the Hartree–Fock determinant (formed from the $N_\uparrow + N_\downarrow$

creation operators in reciprocal space with the smallest $\epsilon_k$) is the exact ground state eigenfunction when $U = 0$. Thus the reciprocal space basis is a compact representation of the wavefunction when $U/t$ is small as most of the weight will be on the Hartree–Fock determinant.

In FCIQMC plateau height is known to scale linearly with $U/t$ at half filling i.e. when the number of electrons equals the number of sites in a reciprocal space basis. [62] Thus we have a model Hamiltonian with a way to control one measure of the difficulty of the FCIQMC algorithm.

# 2. The FCIQMC Stochastic Process

In this chapter[1] we show that FCIQMC is an example of Markov Chain Monte Carlo (MCMC).[2] All details about a FCIQMC calculation can be inferred from its Markov Chain. Originally the idea behind this work was to use the Markov chain to understand how best to design an FCIQMC simulation. Unfortunately the space scales so unfavourably as a function of system size, that we have only been able to investigate the two determinant system (which cannot have a sign problem). To our knowledge no one has performed this analysis before for a Quantum Monte-Carlo simulation (possibly because the space scales so unfavourably as a function of system size).

In chapter 3 we infer details directly from the Markov chain to investigate population control bias. We also empirically investigate a larger system using multiple FCIQMC calculations. Originally the idea was to use the Markov chain to understand how best to perform an FCIQMC simulation i.e. minimise how quickly the statistical error bar reduces as a function of the parameters effecting the population dynamics: $\delta\tau$, $\xi$, $N_s$, $S(0)$. Unfortunately this proved difficult to do. So instead we empirically investigate these ideas in chapters 4 and 6. We hope these ideas in this chapter provide a way of understanding FCIQMC which complements the more traditional approach which we reviewed in Sec. 1.3.

We begin by reviewing general MCMC theory in Sec. 2.1, before describing the process for FCIQMC in Sec. 2.2. We describe the formulae for a solution for a two determinant system in Sec. 2.2.1. After this we numerically compute some solutions for $H_2$ in a STO-3G basis set in Sec. 2.2.2. Finally we draw some conclusions about the FCIQMC Markov chain in Sec. 2.3.

[1] The content of this chapter has been taken from the work published in [66]. All calculations in this chapter were performed by the author. The discussion in Sec. 2.1 has been extended and Sec. 2.2.3 is not present in the published work.

[2] Only in the limit that $A = 1$ can the stochastic process describing projected energy be described by a Markov Chain (although a process with the same mean can be described with a Markov Chain). Shift can be described as a Markov Chain for all $A$.

## 2.1. General Markov Chain Monte Carlo Theory

A discrete time Markov chain is the most simple class of dynamic Monte-Carlo (we discussed dynamic Monte-Carlo in Sec. 1.2). In a discrete time Markov Chain the probability of transitioning from one state to another (in one discrete time step) depends only upon the current state of the simulation and not any of the history. This has a number of implications which we shall now discuss in the remainder of this section. For a more in-depth review of MCMC than that presented here see Ref. [32, 67].

We shall denote the Markov states of the simulation using Greek indices $\alpha, \beta \ldots$ (to avoid confusion with Slater determinants for which we shall continue to use Roman indices $i, j, \ldots$). The stochastic matrix, $\Gamma$, consists of elements $\Gamma_{\alpha,\beta}$ which give the probability that the system transitions from state $\alpha$ to state $\beta$ in one step in the Markov chain and is in general not symmetrical. We use the convention that the row defines the state in which the chain starts and the column defines the state in which the chain ends.

As the Markov chain must transition from one state to another or remain in the same state,

$$\sum_{\beta} \Gamma_{\alpha\beta} = 1 \quad \forall \quad \alpha, \tag{2.1}$$

and $\Gamma$ is non-negative, we can infer some properties of the eigenvectors and eigenvalues of the stochastic matrix. The Perron–Frobenius theorem [68, 69, 70, 71] proves that the $\Gamma$ must have one or more eigenvectors with eigenvalue one and all other eigenvalues must be less than one. The eigenvectors with a unit eigenvalue are called the Perron-Frobenius eigenvectors. As is the convention in probability theory and to be consistent with our definition of the stochastic matrix and Eq. 2.1, the eigenvectors are left eigenvectors:

$$\sum_{\alpha} \gamma_{\alpha} \Gamma_{\alpha\beta} = \gamma_{\beta}. \tag{2.2}$$

$\gamma_{\alpha}$ gives the probability that the Markov chain will be in state $\alpha$ if the chain is in equilibrium. The Perron-Frobenius eigenvector is unique and the chain will converge towards this distribution if (*i*) all the states are aperiodic i.e. $\Gamma^{N}_{\alpha\beta} > 0$ for all values of large $N$; (*ii*) the stochastic matrix consists of a single communicating class (every state can be reached from every other state in some number of steps). This condition

is known as ergodicity in the physics/chemistry community.[3]

When doing MCMC we are usually interested in some functions of the state space. For an ergodic system, the mean of a generic function $f(\alpha)$ defined on the state space, which has value $f_t$ on step $t$ of a specific Markov chain, is given by

$$\mu_f = \langle f_t \rangle_t = \sum_\alpha f(\alpha)\gamma_\alpha. \tag{2.3}$$

The Perron-Frobenius eigenvector specifies the distribution of an ensemble of independent Markov chains, and by computing it we may find expectation values of interest in this system.

We can get an estimate of $\mu_f$ from a single calculation which takes $N$ steps:

$$\bar{f} = \frac{1}{N}\sum_{t=1}^{N} f_t \tag{2.4}$$

Now:

$$\lim_{N\to\infty} \bar{f} = \mu_f \tag{2.5}$$

We would like to measure how quickly $\bar{f}$ decays towards $\mu_f$. This will tell us the prefactor of the square root law. However the distribution of an ensemble of independent Markov chains ($\gamma_\alpha$) is different to the distribution of a single chain starting from a single specific state $\alpha$ having taken a single step ($\Gamma_{\alpha,\beta}$). The standard way to estimate the error bar from a correlated single Markov chain is to use blocking analysis [54].

We can define the unnormalised autocorrelation function [32] (the covariance between a signal and itself $t$ steps later):

$$C_{ff}(t) = \sum_{\alpha,\beta} \gamma_\alpha \Gamma_{\alpha\beta}^{|t|} \left( f(\alpha) - \sum_\delta \gamma_\delta f(\delta) \right) \left( f(\beta) - \sum_\delta \gamma_\delta f(\delta) \right) \tag{2.6}$$

We have used the fact that a chain (with stochastic matrix $\Gamma_{\alpha\beta}$) taking $t$ steps is a Markov chain with Stochastic matrix $\Gamma_{\alpha\beta}^t$. Thus $\gamma_\alpha \Gamma_{\alpha\beta}^t$ is the probability of the Markov chain ending in a state $\beta$ after $t$ steps if it starts in the state $\alpha$ with probability $\gamma_\alpha$ (the stationary distribution).

[3] In the mathematics literature of Markov chains ergodicity is a stronger condition. [32]

This can be simplified to:

$$C_{ff}(t) = \sum_{\alpha,\beta} f(\alpha)\gamma_\alpha \Gamma^{|t|}_{\alpha\beta} f(\beta) - \gamma_\alpha f(\alpha)\gamma_\beta f(\beta). \tag{2.7}$$

by multiplying out the brackets and substituting in Eq. 2.2 and Eq. 2.1.

Using the error propagation rules for correlated data[4] and noting that $C_{ff}(0)$ is the variance, we can write:

$$\sigma_{\bar{f}}^2 = \frac{1}{N^2} \sum_{r,s=1}^{n} C_{ff}(r-s) \tag{2.8}$$

Grouping terms with the same $r - s$:

$$\sigma_{\bar{f}}^2 = \frac{1}{N} \sum_{t=-(N-1)}^{N-1} (1 - \frac{|t|}{N}) C_{ff}(t) \tag{2.9}$$

The term in the sum is the prefactor of the square root law.

## 2.2. The FCIQMC chain

It follows immediately from the FCIQMC algorithm in Sec. 1.3 that the probability of transitioning from the state at a given time step to another state at the next time step depends only on the current stochastic representation of the eigenvector (i.e. the number of psips on each determinant) and the current value of the shift. Given Eq. 1.48, it is clear that the shift depends only on the number of psips on the last shift update step (i.e. we do not require the entire history of the simulation to calculate the shift). We can hence describe FCIQMC as a Markov chain taking one step every $A$ time steps (although if $A > 1$ the projected energy estimator is no longer well defined as it is averaged over every step between shift update steps). To make the equations more mathematically convenient and avoid the computational cost of multiplying the size of the state space by $A$ we shall henceforth assume that the simulation takes one time step between shift updates (i.e. $A = 1$). Thus the transition probability only depends on the state of the simulation on the previous step.

A state $\alpha$ in FCIQMC is represented by the signed number of psips on each

[4] $\sigma^2_{\sum_i f_i} = \sum_i \sigma^2_{f_i} + \sum_{j \neq i} \mathrm{Cov}(f_i, f_j)$, where the set of $f_i$ are correlated random variables and $\mathrm{Cov}(f_i, f_j)$ is the covariance between $f_i$ and $f_j$. [72]

determinant $(n_a, n_b, \dots)$.

$$\alpha := (n_a, n_b, \dots). \tag{2.10}$$

We may omit the shift $S$ as it is simply a function of the total number of particles and not an independent variable. The FCIQMC chain is in an absorbing state (meaning the probability of leaving this state is zero) when there are no psips on any of the determinants, as all events which change the psip population require an existing non-zero population.

Starting with a non-zero number of psips, once the population reaches $N_s$ psips, the shift is allowed to vary. After some equilibration time, the shift oscillates about the correlation energy of the system. It is after this point that the FCIQMC simulation is converged and statistics can be gathered. The Markov process hence changes after the shift turns on i.e in a conventional FCIQMC calculation if the population drops below this point the population is still controlled (the shift will simply become positive to restore the population to the previous level).

The functions of the state space of interest in FCIQMC are the shift and the numerator and denominator of the projected energy:

$$S(\alpha) = S(t=0) - \xi \log \frac{N(\alpha)}{N_s} \tag{2.11}$$

$$E_{\text{Numer}}(\alpha) = \sum_{i \neq 0} H_{0i} n_i(\alpha) \tag{2.12}$$

$$N_{\text{Denom}}(\alpha) = n_0(\alpha) \tag{2.13}$$

where $N(\alpha) = \sum_i |n_i(\alpha)|$.

From this we may determine the average projected energy as $\langle E_{\text{Proj}} \rangle = \langle E_{\text{Numer}} \rangle / \langle N_{\text{Denom}} \rangle$ (to compute the average from the Markov chain $\langle S \rangle = \sum_\alpha S(\alpha)\gamma_\alpha$). Calculating the projected energy using a single function of the state space:

$$E_{\text{Proj Single}}(\alpha) = \frac{\sum_{i \neq 0} H_{0i} n_i(\alpha)}{n_0(\alpha)} \tag{2.14}$$

leads to a bias because the numerator and denominator of the projected energy are correlated.

### 2.2.1. The Stochastic Matrix for a two Determinant System

Computing the stochastic matrix for an arbitrarily large system of determinants is computationally infeasible as the Markov state space scales as the power of the

number of determinants, so we restrict ourselves to the simplest possible (interesting) system. Consider a system of two determinants $a$ and $b$. $\alpha$ and $\beta$ represent two Markov states with a given signed number of psips on each determinant:

$$\alpha = (n_a, n_b) \tag{2.15}$$

$$\beta = (n'_a, n'_b). \tag{2.16}$$

On each step, each psip independently attempts to spawn and die. The probability that $n$ psips succeed out of $N$ attempts is:

$$B(n, N, p) = \binom{N}{n} p^n (1-p)^{N-n} \tag{2.17}$$

where $p$ is the probability of one psip spawning or dying independently and $B(n, N, p)$ is the probability mass function of the binomial distribution. We can work out $p$ for spawning and dying using Eq. 1.45 and 1.46 respectively.

Using the sign convention for spawning and diagonal death, we can calculate the change on determinant $a$[73]:

$$n'_a - n_a = -\text{sgn}(H_{ba})\text{sgn}(n_b)n_{sa} - \text{sgn}(H_{aa} - S)\text{sgn}(n_a)n_{da} \tag{2.18}$$

where $n_{sa}$ ($n_{da}$) is the number of psips spawning onto (dying on) $a$ and sgn is the sign function,

$$\text{sgn}(x) = \begin{cases} -1 & \text{if } x < 0, \\ 1 & \text{if } x \geq 0. \end{cases} \tag{2.19}$$

Similarly the change on determinant $b$:

$$n'_b - n_b = -\text{sgn}(H_{ab})\text{sgn}(n_a)n_{sb} - \text{sgn}(H_{bb} - S)\text{sgn}(n_b)n_{db} \tag{2.20}$$

We can rearrange Eq. 2.18 and Eq. 2.20 to specify $n_{da}$ for a given $n_{sa}$:

$$n_{da} = -\frac{n'_a - n_a + \text{sgn}(H_{ba})\text{sgn}(n_b)n_{sa}}{\text{sgn}(H_{aa} - S)\text{sgn}(n_a)} \tag{2.21}$$

$$n_{db} = -\frac{n'_b - n_b + \text{sgn}(H_{ab})\text{sgn}(n_a)n_{sb}}{\text{sgn}(H_{bb} - S)\text{sgn}(n_b)}. \tag{2.22}$$

As the change on each determinant is independent we can calculate the probability

$p_{cn_a,n'_a}$ that the number of psips on $a$ goes from $n_a$ to $n'_a$:

$$p_{cn_a,n'_a} = \sum_{n_{sa}} B(n_{sa}, n_b, P_s(a|b))B(n_{da}, n_a, P_d(a)). \tag{2.23}$$

Similarly the probability $p_{cn_b,n'_b}$ that the number of psips on $b$ goes from $n_b$ to $n'_b$:

$$p_{cn_b,n'_b} = \sum_{n_{sb}} B(n_{sb}, n_a, P_s(b|a))B(n_{db}, n_b, P_d(b)). \tag{2.24}$$

Unfortunately we know of no simplification of the sum of the product of two binomial distributions with different independent probabilities. As the change on one determinant is independent of the change on the other determinant, the desired stochastic matrix element is simply

$$\Gamma_{\alpha,\beta} = p_{cn_a,n'_a} p_{cn_b,n'_b}. \tag{2.25}$$

We have constructed $\Gamma_{\alpha,\beta}$ for some simple systems and determined (by direct[74] and iterative[75] diagonalization dependent upon the size of the state space) the stationary distributions $\gamma_\alpha$ which correspond to the Perron-Frobenius eigenvector. With these we may investigate the properties of the ensemble of FCIQMC simulations possible.

### 2.2.2. The Stationary Distribution of $H_2$ in a STO-3G basis set

In this section we describe how the stationary distributions of the estimators of the projected energy, the shift and the number of psips can be adjusted by the parameters which control the Markov chain. A simple choice of two determinant system is $H_2$ in a STO-3G basis with an internuclear separation of 0.7122 Å and we use the set of closed shell determinants based on Hartree–Fock reference orbitals: $\sigma_g^2$ and $\sigma_u^2$. The absolute Hartree–Fock energy of this system is $-1.1175058843\ E_h$. We present some results here to illustrate some of the properties of the transition matrix for a two determinant system.

We begin by looking at factors which control the population of psips in the simulation: $N_s$ defines the number of psips at which the shift is allowed to vary (we will take the shift to be equal to the Hartree–Fock energy at this number) and $\xi$ controls how heavy the shift damping is. With these two degrees of freedom, we

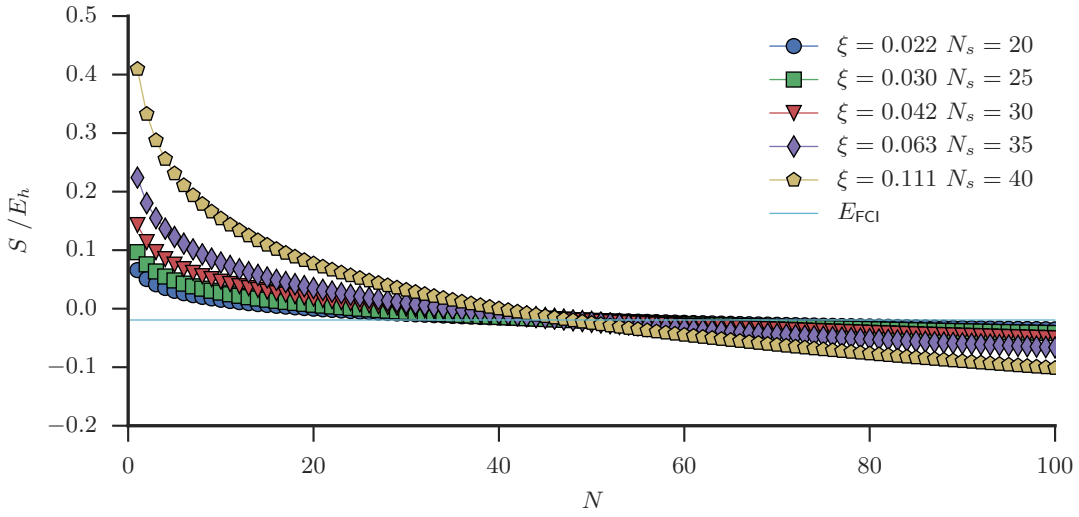



Figure 2.1.: The instantaneous shift, $S$, as a function of the instantaneous psip population $N$, as defined in Eq. 1.48 for $H_2$ (STO-3G basis, internuclear separation 0.7122Å). The pairs of population control parameters, $\xi$ and $N_s$, were chosen such that $S$ equals the correlation energy at the same number of psips (47.63). Lines are drawn to guide the eye.

may adjust both $N_s$ and $\xi$ at the same time to achieve a given number of psips ($N$ in Eq. 1.48) but with different population dynamics. The total population of psips should be stable when the shift is equal to the correlation energy if FCIQMC is not biased. We may hence use Eq. 1.48 to determine pairs of $\xi$ and $N_s$ which have the same average population $N$. Fig. 2.1 shows the shift as a function of population for pairs of $\xi$ and $N_s$ such that the average population is identical and $\approx 50$. $\xi$ and $N_s$ in turn affect the stationary distributions of the shift and the number of psips. If $\xi$ is large the stationary distribution of the number of psips is narrow because the population is well controlled. However, the stationary distribution of the shift is broad (Fig. 2.2). Of course we can only choose $\xi$ and $N_s$ to achieve a desired population if we know the correlation energy beforehand. In practice one could use an estimate of the correlation energy from a less expensive method, such as coupled cluster theory, to reach an approximate desired population.

The projected energy estimator is also affected by the choice of shift damping parameter. The population on each determinant is proportional to the overlap of that determinant with the ground state of the FCI wavefunction (again assuming everything is unbiased). Thus we expect both the stationary distributions of the denominator and numerator of the projected energy (Fig. 2.3) to look similar to the

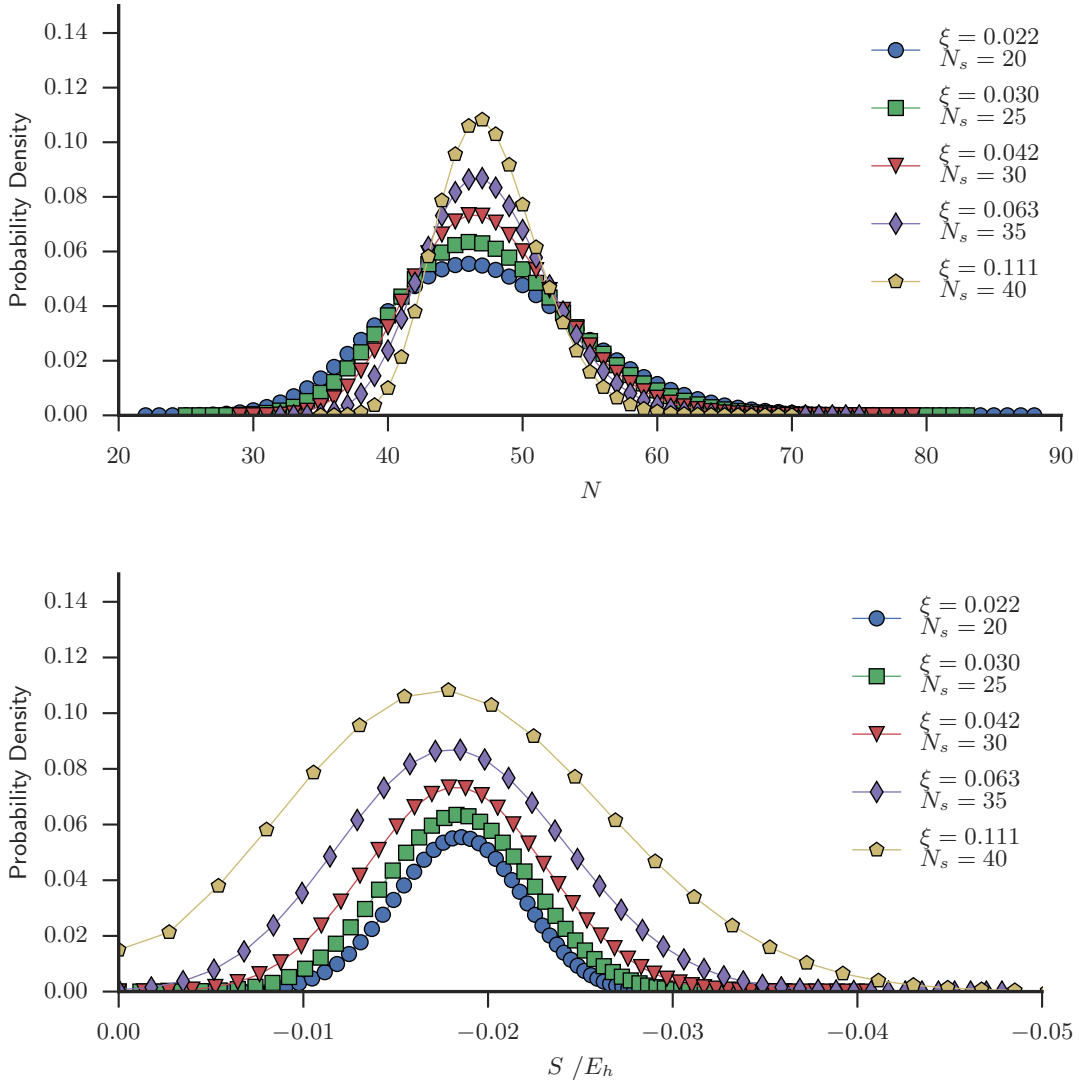



Figure 2.2.: The effect of population control on the stationary distributions of the number of psips, $N$, and the shift, $S$ for $H_2$ (STO-3G basis, internuclear separation 0.7122Å). Only states with less than 100 psips on each determinant were included in the transition matrix. The variance of $N$ increases as $\xi$ decreases whereas the variance of $S$ increases as $\xi$ increases. Lines are drawn to guide the eye.

stationary distribution of the number of psips (Fig. 2.2), though the numerator is additionally scaled by the matrix element between the Hartree–Fock determinant and the excited state. Changing the shift damping parameters seems to have little effect on the numerator of the projected energy (Fig. 2.3) though the change in variance of the denominator indicates that the projected energy will have a smaller variance as $\xi$ increases.

Our tests indicate that the time step has very little effect on the stationary distribution. This is unsurprising as it simply scales the probabilities that a single psip spawns, or undergoes diagonal death/cloning and has no effect on the relative probabilities. Importantly once the time step is set beyond the point which causes multiple psips to be spawned or undergo diagonal death/cloning from a single parent we expect the stationary distribution to change. The structure of the Markov matrix will change at this point connecting different states together. This is the subject of the next section.

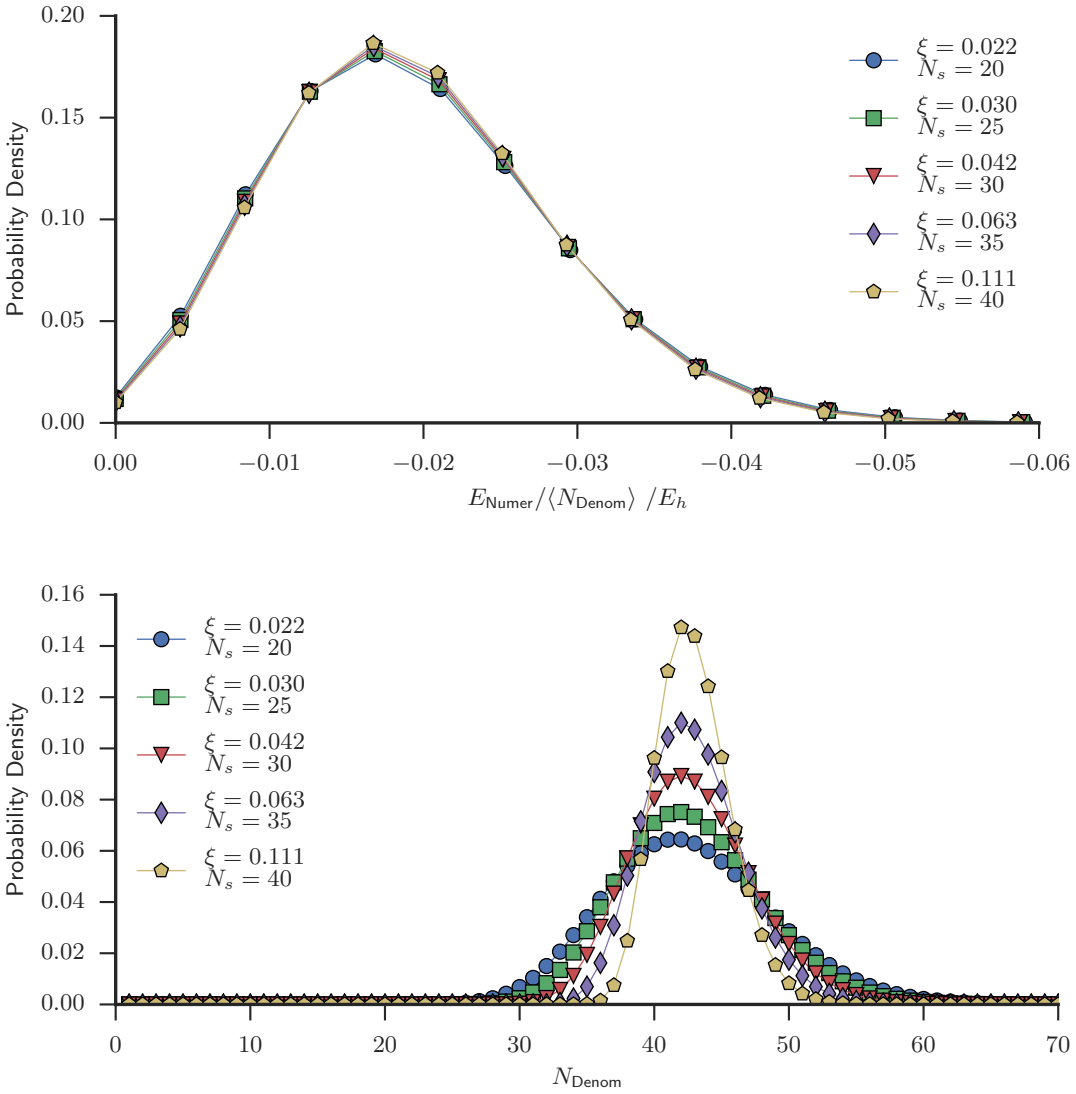



Figure 2.3.: The effect of population control on the stationary distribution of the numerator and denominator of the projected energy for $H_2$ (STO-3G basis, internuclear separation 1.4244Å). Only states with less than 100 psips on each determinant were included in the transition matrix. The variance of the denominator increases as $\xi$ increases. Each numerator has been divided by the mean denominator for that set of parameters to place the plots on a common scale. The variance of the normalized numerator has only a small dependence on the population control parameter. Lines are drawn to guide the eye.

### 2.2.3. The Critical Timestep

It has been remarked previously in [62] that a necessary condition for FCIQMC to decay to the ground state is that:

$$\delta\tau \leq \frac{2}{E_0 - E_{max}}, \tag{2.26}$$

where $E_{max}$ is the largest eigenvalue of the FCI matrix. Thus one would expect that if $\delta\tau$ was set greater than this value the stationary distribution will change. We have investigated this for $H_2$ (STO-3G basis, internuclear separation 1.4244Å) effect and found that the stationary distribution breaks down before this point. For this $H_2$ system the stationary distribution breaks down when $\delta\tau$ becomes close to 0.6061 (see Fig. 2.4). This is when the probability of diagonal death on the determinant connected to the Hartree–Fock becomes one. This suggests that FCIQMC becomes unstable when:[5]

$$\delta\tau = \delta\tau_c = \frac{1}{max(H_{ij})}. \tag{2.27}$$

We have also performed multiple FCIQMC calculations at different timesteps around $\delta\tau_c$. In Fig. 2.4 we find that the population dynamics are stable even above $\delta\tau_c$ (the population eventually explodes at about $2\delta\tau_c$ for this system. This is still at a point less than the value suggested by Eq. 2.26). This may mean that FCIQMC is metastable above $\delta\tau_c$ with a very long lifetime if $\delta\tau$ is close to $\delta\tau_c$ (if there a very small but finite probability we expect the stationary distribution to break down but an FCIQMC calculation could take an extremely long time to reach this state). We seem to see for $\delta\tau = 1.2$ that the population is metastable for a while before exploding in Fig. 2.4 bottom. In Chapter 4 where we investigate the effect of the timestep on the stochastic error bar, we found that after a certain point the population explodes. We suspect that this effect is related to the behaviour we see here.

[5] This analysis assumes that spawning can also cause this instability i.e. We assume that if the largest matrix element was an off diagonal, and also that the timestep was large enough to cause the probability of spawning (when sampling this diagonal matrix element) to become greater than one, the population would explode in a similar way. Before we can be completely certain about Eq. 2.27 we should investigate this.

## 2.3. Conclusion

We have demonstrated that FCIQMC is an example of Markov Chain Monte Carlo and computed the stochastic matrix for a two determinant system. We have explored the effect of population control on the stationary distribution and how setting the timestep greater than a certain critical point causes the stationary distribution to break down.

In this chapter we assumed that the excitation generators always generate a valid excitation i.e. for our two determinant system a psip on determinant $a$ always attempts to spawn onto determinant $b$. In reality, as we discussed in Sec. 1.3.2, if we use unrenormalised excitation generators often there is a failure rate associated with this process. It would be interesting to investigate the effect of this failure rate on the stationary distribution. One could do this by modifying the probabilities in Eq. 2.18.

It would also be interesting to see what effect, if any, a change of basis of the Hamiltonian matrix has on the stochastic matrix.

It would be interesting to extend these ideas to investigate the sign problem using a three determinant system. The size of the stochastic matrix scales as:

$$S = (2N_{\text{pmax}})^d \tag{2.28}$$

Where $d$ is the number of determinants, $N^d_{\text{pmax}}$ is the maximum number of positive or negative psips allowed in the stochastic matrix which might make it very difficult to investigate a reasonable sized three determinant system.

For systems with more than three determinants, scaling of the size of the stochastic matrix as the power of the number of determinants may make this very difficult to study.

Petruzielo *et al.* proposed a method to combine FCIQMC with a deterministic power method [52]. The Hamiltonian matrix is partitioned into a deterministic and a stochastic space. The projection equation, Eq. 1.42, remains the same, but the way the stochastic sampling is performed differs. Spawning and diagonal death do not take place using matrix elements from the deterministic space. Instead, a contribution from all determinants in this space is made by an exact matrix (containing only the deterministic space) vector product every timestep. This process will remain a Markov Chain, but the rules for constructing the Markov matrix will differ (having to take into account the exact projection). The state space of the process remains the same (containing the population on all determinants).

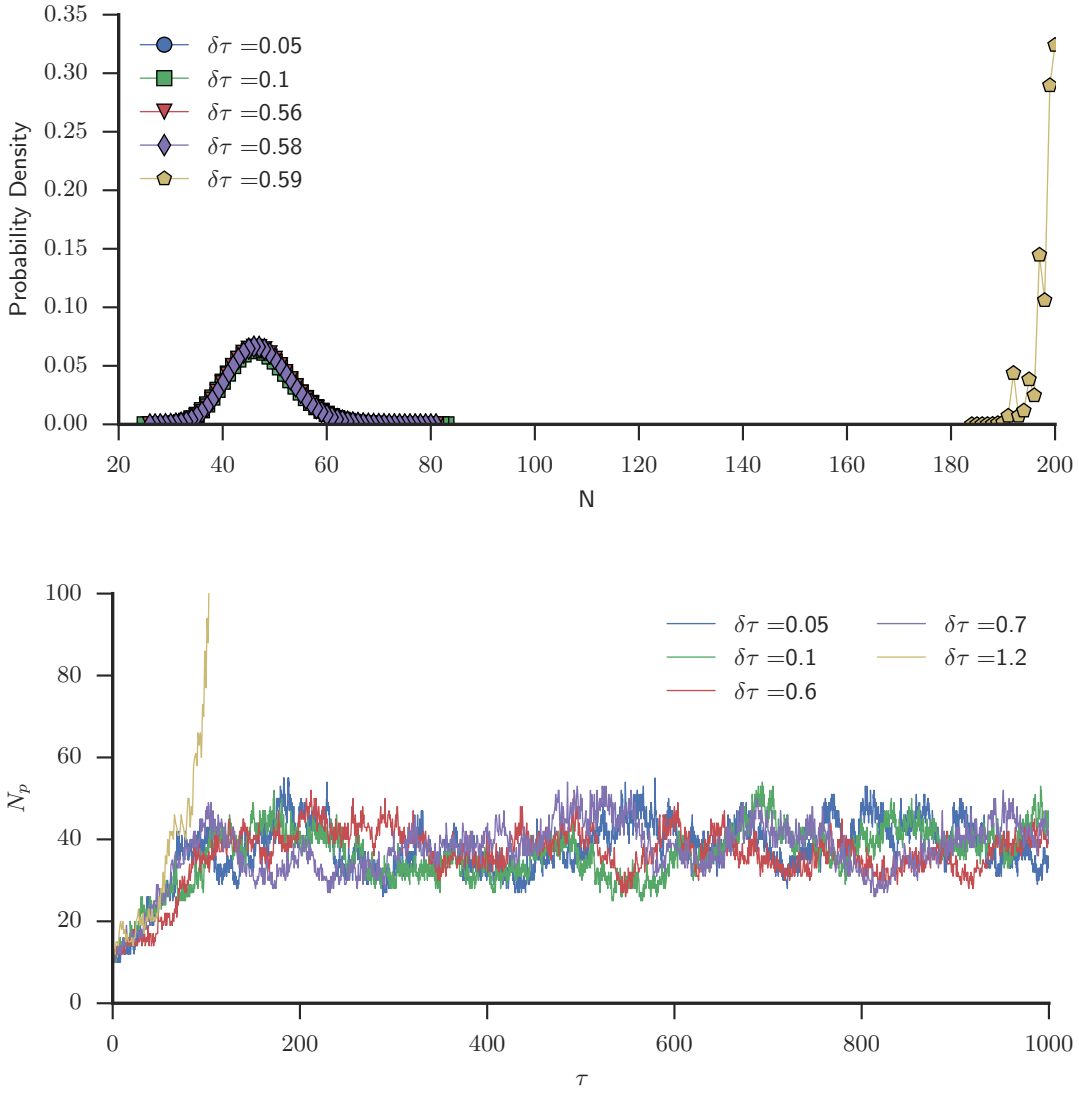



Figure 2.4.: Top: The stationary distribution breaks down once the time step becomes close to the critical point. Here $\delta\tau_c = 0.6061$. Bottom: We see the population explodes once $\delta\tau \approx 1.2$ after being metastable for a very short period of time. This metastability could be caused by a fluctuation in the shift. We see no population explosion for any other point even if run for a large number of timesteps.

# 3. Population Control Bias in FCIQMC

In the last chapter we showed the FCIQMC in an example of Markov Chain Monte Carlo (MCMC).[1] This enables us to compute the exact mean of both the shift and projected energy with no statistical error. In Sec. 3.1 we find that the mean of both of the estimators of the energy (the shift and projected energy) are larger than the correlation energy determined by FCI (we investigate two geometries of $H_2$ in an STO-3G basis). This bias is caused by population control. It is important to understand when population control bias will be significant compared to the stochastic error bar and in Sec. 3.2 we investigate population control bias in a more realistic Neon atom in a cc-pVDZ basis. We then investigate a method for removing population control bias known to be effective in DMC. We show that this technique also works well for FCIQMC.

In order to achieve a finite population in a simulation, we must resort to population control by introducing a shift which itself is dependent upon the current total population. However, this process introduces a feedback into the propagator and hence a systematic bias[46]. Random fluctuations causing the population to increase (i.e. the psip distribution enters a low energy region of phase space) or decrease (higher energy region of phase space) are moderated by a corresponding decrease or increase in the shift. Both actions lead to an increase in the time-averaged energy estimators.

Population control bias occurs because the population control algorithm dynamically changes the shift and thus the diagonal elements of the Hamiltonian throughout the calculation in response to the population fluctuations. This correlation of the changes in the Hamiltonian with the instantaneous population leads to a bias in the psip populations and therefore to the energy estimators. As the responsiveness of the population control decreases we expect the bias to also decrease and there will

[1] The content of this chapter has been taken from the work published in [66]. All calculations were performed by the author of this thesis. The discussion in Sec. 3.3 has been extended in this thesis.

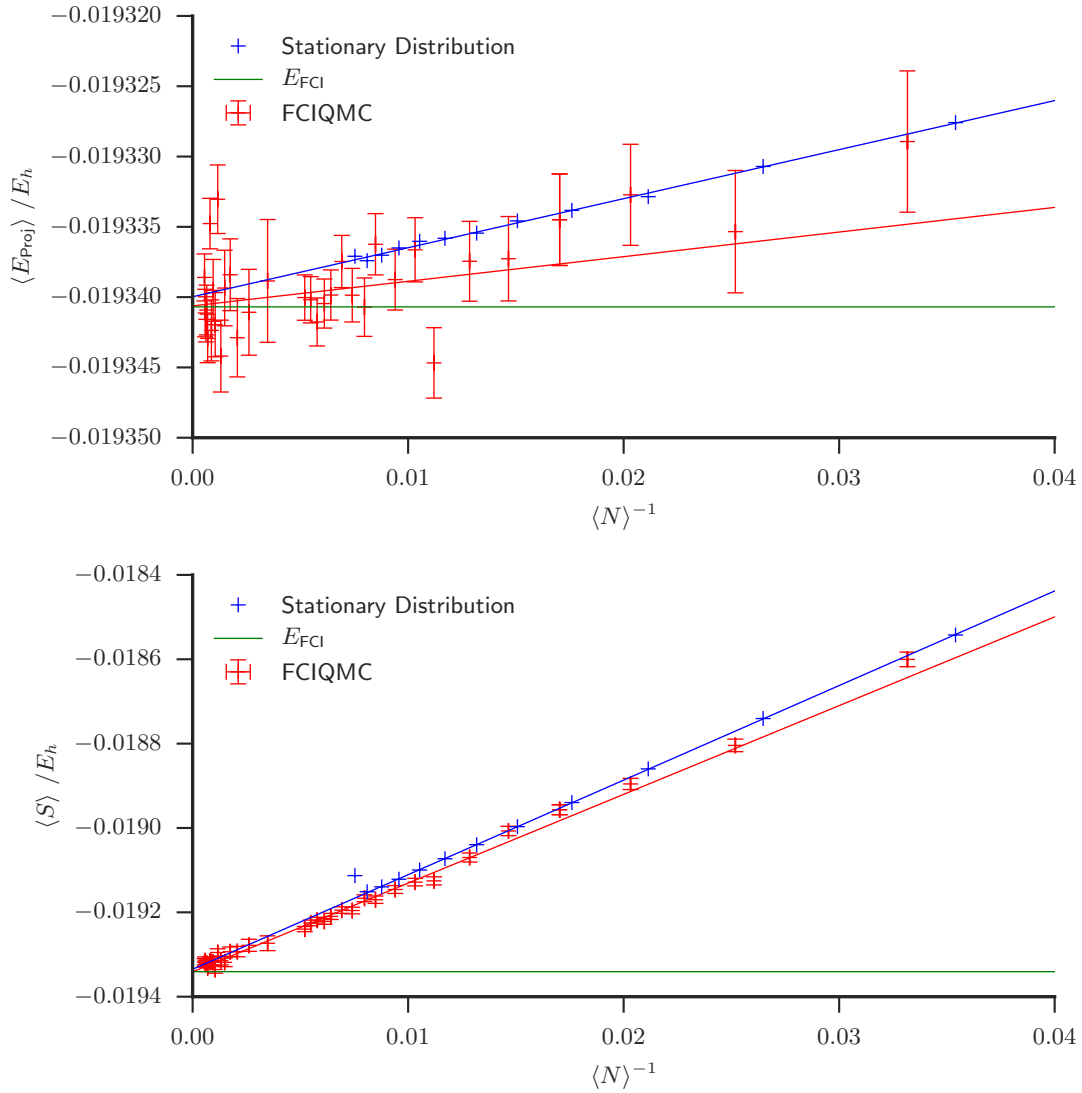



Figure 3.1.: The energy estimators as a function of $1/\langle N \rangle$ for $H_2$ (STO-3G basis, internuclear separation 0.7122Å). The exact means were calculated from the means of the stationary distributions. Only states with up to 150 psips on each determinant were included in the transition matrix calculations. The FCIQMC estimates were calculated from a single chain and the error estimated by blocking[54]. The bias in both estimates of the correlation energy decays with the inverse of the average number of psips. Linear fits were performed with numpy[76]. Errors were weighted in the fits for the FCIQMC data using the sum of the variance of $1/\langle N \rangle$ and variance of the energy estimator.

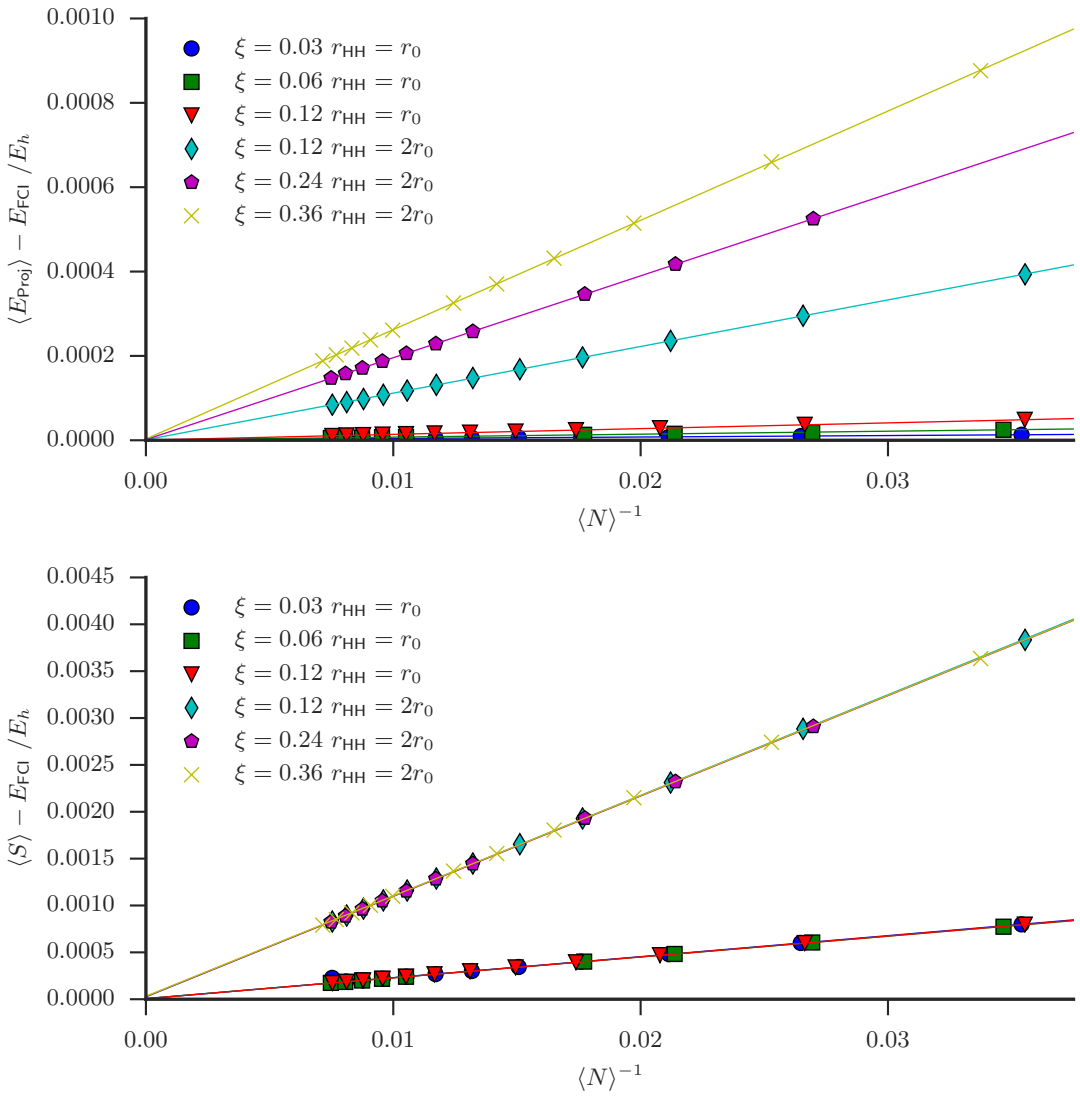



Figure 3.2.: The energy estimators calculation from the means of the stationary distribution as a function of $1/\langle N \rangle$ for $H_2$ (STO-3G basis, internuclear separation $r_{\text{HH}} = r_0$ and $2r_0$ where $r_0 = 0.7122$Å) for different values of $\xi$, the shift damping parameter. Only states with up to 150 psips on each determinant were included in the transition matrix calculations. The bias in the projected energy can be reduced by decreasing $\xi$ whereas the bias in the shift remains the same. Fits with $1/\langle N \rangle$ were performed with numpy[76].

be no population control bias in the limit of a constant value of the shift.

In DMC this bias is known[46, 77] to scale as $\langle N \rangle^{-1}$. In FCIQMC we suspect that the effect has most likely been obscured by the stochastic error in all previous studies. With the aid of exact energy estimators from the transition matrix, in this chapter we investigate the magnitude of any bias present. We feel it is important to understand where population control bias is likely to cause a problem if a small stochastic error is desired. In addition to the energy estimators from the transition matrix, we shall also investigate them from single chains via blocking analyses of single FCIQMC calculations to compare both methods and use them to quantify the factors controlling population control bias. This chapter contains the first quantification of population control bias in FCIQMC.

## 3.1. $H_2$ in a STO-3G Basis Set

For different values of $N_s$, transition matrix and single chain calculations were performed on $H_2$ in a STO-3G basis at the equilibrium geometry of 0.7122 Å (Hartree–Fock energy $-1.1175058843\,E_h$), and the energy estimators evaluated. Fig. 3.1 shows the bias of the projected energy and shift estimators decreases with $1/\langle N \rangle$. Though the single-chain calculations have relatively large stochastic errors, a similar bias in the energy and decay is also notable, and there is good agreement between the single chain and transition matrix results. The fits for the transition matrix calculations, however, do not exactly intercept the $y$-axis at the correlation energy. The worst extrapolation is for the projected energy, which disagrees by $2.3 \pm 0.5\,\mu E_h$. It is difficult to tell if this is caused by a loss of numerical precision, truncation in the transition matrix calculations or if there are higher order effects with small $\langle N \rangle$.

The prefactor in the $1/N$ scaling of the bias in the projected energy is affected by $\xi$, and damping less hard, i.e. decreasing $\xi$, reduces the prefactor (Fig. 3.2). Population control bias also appears to be made worse in strongly correlated systems. Fig. 3.2 shows population control bias as a function of $\xi$ for both $H_2$ in a STO-3G basis set at bond lengths of 0.7122 Å and 1.4244 Å (Hartree–Fock energy $-0.9338980552\,E_h$). We may explain this by reviewing DMC, where, in the limit of a perfect trial function, there are no branching processes and thus no population control bias. Equivalently in FCIQMC if there is no spawning there can be no population control bias. Although true only in the limit in which the Hilbert space is the set of eigenvectors of the Hamiltonian, this indicates that in the weakly correlated limit, there will be less population control bias, which is exactly what we observe.

## 3.2. The Neon Atom

Population control bias is also potentially a significant source of a systematic error in systems which are large enough not to be trivially soluble (rendering transition matrix calculations computationally infeasible). We now turn to the neon atom in a cc-pVDZ basis[78][2] which has a Hilbert space of 50000 determinants. This is small enough such that it is straightforward to compute the FCI energy via iterative diagonalisation but large enough such that most determinants have a small contribution to the wavefunction. It was necessary to oversample the Hilbert space in $H_2$ (i.e. more psips than the number of determinants) whereas FCIQMC calculations in this neon system are stable with a significant undersampling of the space. We shall investigate the effect of population control bias in this regime.

Changing the population control parameters affects both estimators of the correlation energy in the same way as $H_2$. Fig. 3.3a shows the projected energy decaying towards the FCI energy as $\xi$ decreases until the estimator of the energy becomes within error bars. Fig. 3.3b shows the bias in the projected energy decaying as $1/\langle N \rangle$. This intercepts the $y$-axis at -0.1921066(12) $E_h$ which is within errors of the FCI energy of -0.192105578 $E_h$, suggesting we converge to the exact ground state as expected. The population control bias is however significant and with about 10000 psips is about 20 $\mu E_h$.

Whilst increasing $\xi$ and $N$ separately to achieve a reduction in population control bias is possible, a more general prescription for any arbitrary FCIQMC calculation would be preferable. It is with this in mind we return to the relationship between the varying of the shift and the bias, by plotting the bias against the variance of the shift for all the calculations performed in this section (Fig. 3.4). For both geometries of $H_2$ there is an approximately linear correlation between bias and variance of the shift, though with slight variation as $\xi$ is changed. For $H_2$ then it appears that the variance of the shift can be used as a predictor of the population control bias.

The results for Ne are less promising however, as calculations with approximately the same values of shift variance, but different combinations of $\xi$ and $N$ have significantly different values of bias.

We conclude therefore that in more complex systems the variance of the shift alone is not a sufficient predictor of the population control bias. It is likely that not only extent, but also speed of variation of the shift is important, and such values are considerably more difficult to calculate, so we will leave investigation of this

[2]Hartree–Fock energy: $-128.4887755516\,E_h$.

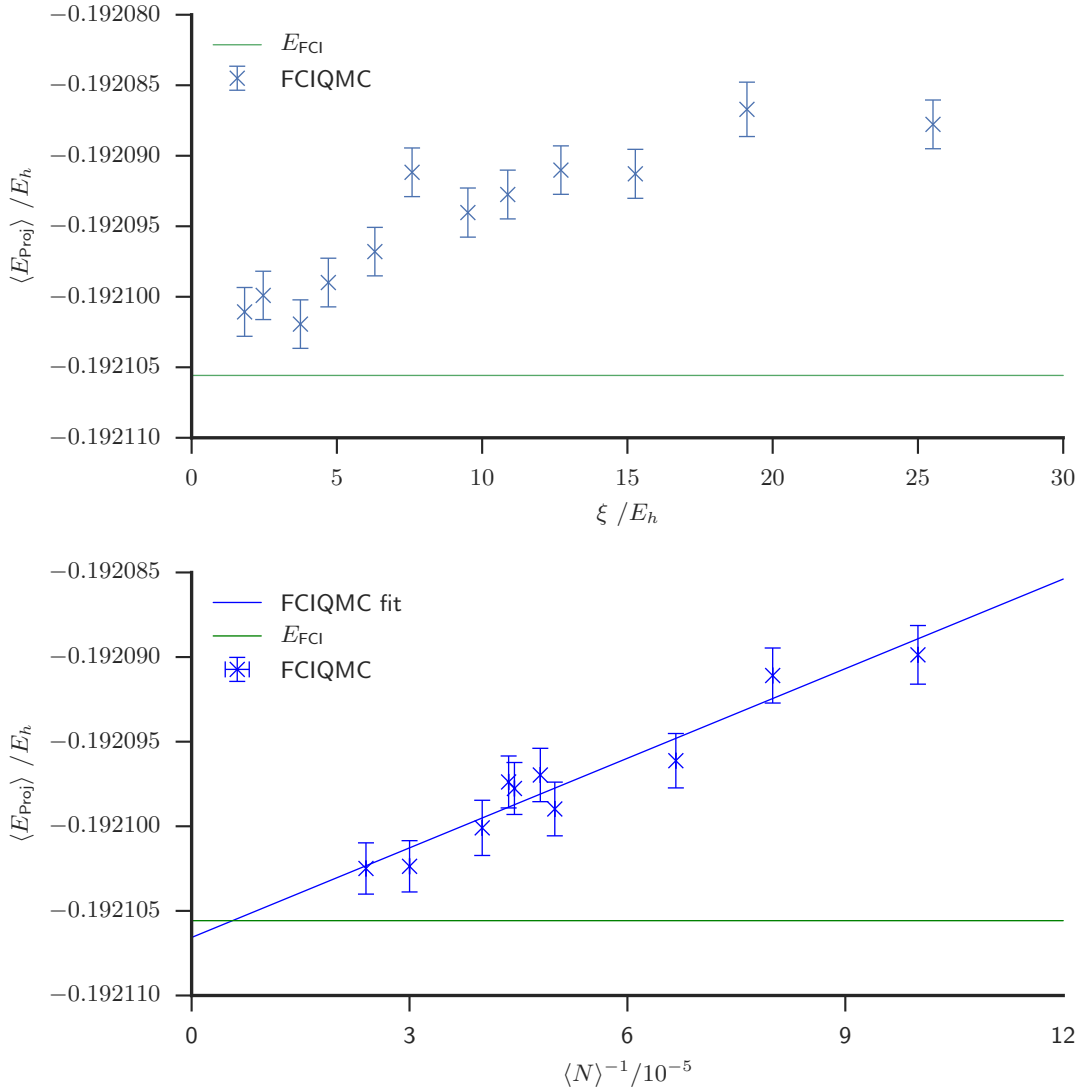



Figure 3.3.: Projected energy estimate from FCIQMC calculations on the Ne atom (cc-pVDZ basis set) as a function of (left) the shift damping parameter, $\xi$, (calculation details: $\delta\tau = 0.005$, 20000000 iterations where $\langle N \rangle$ has the value 10000) and (right) as a function of $1/\langle N \rangle$, where $N$ is the number of psips ($\xi = 1$, $\delta\tau = 0.005$, 200000000 iterations or 72 hours on 12 cores whichever was shorter). Linear fits were performed with numpy.[76].

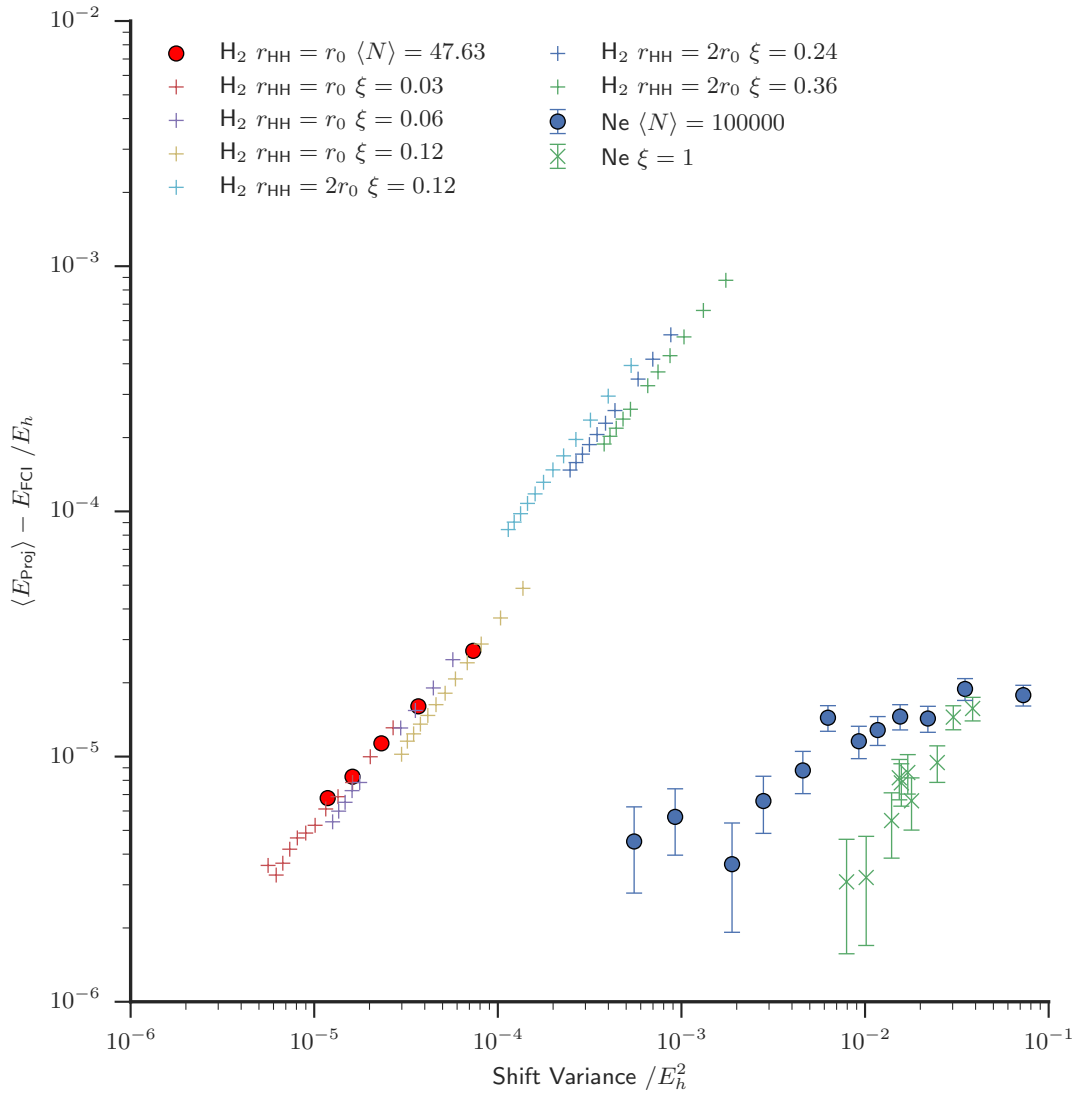



Figure 3.4.: The bias in $\langle E_{\mathrm{Proj}} \rangle$ relative to $E_{\mathrm{FCI}}$ for all calculations in Sec. 3 plotted as a function of the variance of the shift. $r_0 = 0.7122$Å. There is a clear correlation between these variables for $H_2$, where a power law fit indicates a linear relationship between bias and shift variance. For neon, while values for fixed $\xi$ can be linearly fitted, there is no such convincing fit for fixed $N$ which appears to be sublinear.

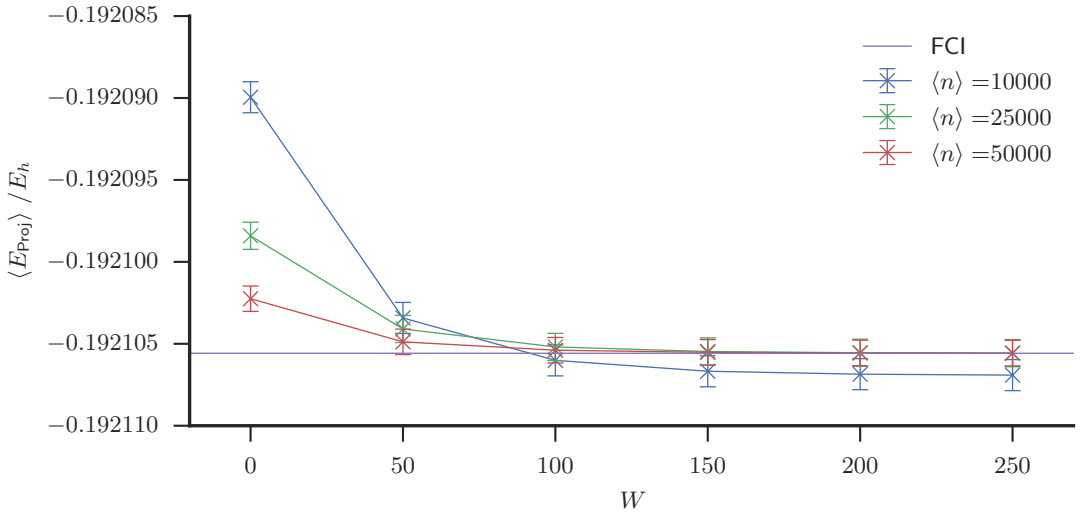



Figure 3.5.: Reweighted projected energy estimate from FCIQMC calculations on the Ne atom (cc-pVDZ basis set) for different $\langle N \rangle$ as a function of $W$, the number of iterations reweighted over. Four independent FCIQMC calculations were performed for each $\langle N \rangle$, and each point shows the mean of the reweighted $\langle E_{\text{proj}} \rangle$ for the four runs.

connection to a future investigation.

The population bias can be both quantified and reduced by a reweighting technique based upon the history of the shift[46]. This technique is commonly used in DMC. The contribution at a given time, $\tau$, to the numerator and denominator of the projected energy is weighted by taking into account the shift of the preceding $W$ iterations. For $S_m$ denoting the shift $m$ iterations previously, the weight is given by:

$$w(\tau, W) = \prod_{m=1}^{W} e^{-\delta\tau(S_m - \langle S \rangle)}. \tag{3.1}$$

The reweighting is implemented as a post-processing step on the output of a calculation. The population control bias is effectively removed for sufficiently large $W$ at the cost of increasing the stochastic error and, as can be seen in Fig. 3.5, the residual bias is of the order of the stochastic error bars. [3]

The value of $W$ ($\approx 250$) required for this procedure to converge is of the order of the serial correlation length. We note that it is not possible to apply this method to

[3]FCIQMC projects out the ground state using $(1 - (\hat{H} - S))\delta\tau$ whereas $\exp(-\delta\tau(\hat{H} - S))$ is used in DMC. This means weighting with Eq. 3.1 is an approximation. The error introduced is second order in $\delta\tau$ (this can be shown using a Taylor expansion). The reweighting requires the projected energy and population at every individual time step.

the Markov Chain approach as the dependence of expectation values on calculation history makes the process non-Markovian.

## 3.3. Discussion

Recently Petruzielo *et al.* proposed an adaptation of the FCIQMC method.[52] It was proposed to use floating point numbers to represent the population of psips on a determinant and to adapt the population dynamics of FCIQMC for non-integer weights. This adaptation results in an uncountably infinite state space of the Markov chain. They also proposed to partition the determinant space into deterministic and stochastic subspaces, where the action of the Hamiltonian in the deterministic subspaces is applied exactly using sparse matrix multiplication and the action in the stochastic subspaces is sampled in the same way as in FCIQMC. Using floating point numbers as walker weights might reduce the prefactor of the $1/\langle N \rangle$ scaling in finite population bias.

The population control algorithm in DMC, as recommended in Ref. [46], is slightly different from that used in FCIQMC: the shift is updated from the 'best current estimate' of the energy rather than from the previous value of the shift. Using this population control algorithm would render FCIQMC non Markovian. Nonetheless we could use the stochastic matrix technique presented here to calculate the probability distribution of the shift in the limit of convergence of the projected energy. It would be interesting to investigate if this is a better method of population control for FCIQMC.

Using the population control approach given in Ref. [49] (i.e. using Eq. 1.48 with $\gamma$ set in the region 0.01 to 0.05), may introduce finite population bias if the time step needed to converge a calculation needs to be small due to the factor of $\frac{1}{\delta\tau}$ in Eq. 1.48. This means finite population control bias is likely to be more of a problem for calculations which require smaller time steps, such as strongly correlated systems, or calculations using coupled cluster Monte Carlo[79].

We also note that attempts to converge FCIQMC calculations to $\mu$Hartree accuracy have previously been attempted (see Ref. [80]) and in this regime, population control bias could also potentially become important (similar in magnitude to the stochastic error bar).

We recommend that one reweights the projected energy estimator, as suggested in Ref. [46], as it does not involve multiple expensive runs. If a large enough population is used the resultant estimate of the energy would be unbiased albeit with

a larger stochastic error (though this has appeared negligibly larger in our tests). Alternatively one should use a large population of psips and set $\xi$ to be as small as possible, such that the number of psips does not drop below the system-dependent critical population. Doubling the number of psips in a simulation increases the equilibration time and probably also the memory requirements. It is also important to perform enough steps to get an accurate estimate of the error. In choosing an appropriate value of $\xi$ there is a compromise to be made; it is tempting to increase $\xi$ because it reduces the fluctuations in the total number of psips and, for larger systems, this can reduce the maximum amount of memory used during the calculation. However too large a $\xi$ will cause population control bias to become significant.

In conclusion, we caution users of FCIQMC and related methods to be aware that population control can introduce a significant bias in calculated energies. We recommend that post-processing reweighting is used to quantify its magnitude and the psip population and damping parameters be modified as suggested in this chapter if needed. Population control bias is generally small compared to the stochastic error unless extreme accuracy is required. Running in constant shift mode can also be effective in removing the bias, by setting the shift approximately 20% above the correlation energy. This causes the population to grow slowly throughout the simulation.

# 4. The Efficiency of FCIQMC

In this chapter we investigate the efficiency of FCIQMC. For the most efficient FCIQMC calculation, the error bar will decay fastest as a function of the computer time. In chapter 2 we showed that FCIQMC is a Markov chain. We can group the set of parameters which define the FCIQMC chain into two. Firstly the molecular geometry and basis set which define the FCI matrix are specified by the problem that we are trying to solve. Secondly the parameters $\delta\tau$, $\xi$, $N_s$, $S(0)$ control the psip population dynamics. These parameters control the systematic biases (the initiator error discussed in Sec. 1.3.1 and population control bias discussed in Chapter 3) and the statistical error which we shall discuss in this chapter.

We believe that understanding the efficiency of a FCIQMC calculation is important for a number of reasons. We can provide guidance so that one can use a computational budget as effectively as possible. Ideally one could use these ideas to make the process a 'black box', meaning FCIQMC could be performed automatically without careful tuning of the population dynamics. Secondly it can let us compare different FCIQMC algorithms. In chapter 6 we discuss an implementation of FCIQMC on novel hardware. In our implementation we modified the population dynamics, and we use the methodology developed in this chapter to compare this modified FCIQMC algorithm to the original. The efficiency is the important factor if a calculation is CPU limited, and the best FCIQMC algorithm maximises the efficiency. If a calculation is memory limited then plateau height (which has been extensively investigated previously [62, 50]) is important as if the plateau is large enough so that the list of determinants and psips cannot be stored then the calculation cannot be performed.

We begin by investigating how the computer time depends on the parameters which control the population dynamics. After this we explore the dependence of the prefactor of the square root law as a function of the parameters which control the psip population. We discuss how the plateau height affects the stochastic error bar and draw some conclusions about how to choose parameters to maximise the efficiency of the FCIQMC algorithm 'a priori' from the plateau height. After this

we investigate the scaling of the error bar of FCIQMC as a function of the system size (basis set size, and number of electrons).

Previous investigations of the efficiency of FCIQMC have investigated the standard error as a function of computer time and focused on comparing different algorithms [52, 81]. We instead investigate separately how the error decays as the number of timesteps and how the computer time per timestep increases as a function of the population control parameters. From knowledge of both of these factors we find a lower bound on the most efficient calculation.

## 4.1. The Computational Cost of a Calculation

An efficient FCIQMC implementation[1] uses a sparse storage scheme to store the list of psips. A representation of a determinant along with the number of positive or negative psips on the determinant are stored together. Diagonal death involves looping through the list and adjusting the population on each site. The cost of this step scales proportionally to the number of occupied sites. Spawning involves each psip attempting to spawn onto a child determinant. This scales proportionally to the total number of psips. The newly spawned walkers are appended to a new list. During annihilation this list is sorted and all psips on the same determinant are all accumulated onto a single copy.

For molecular FCIQMC let us assume that the expensive steps are attempting to spawn onto a child determinant and diagonal death because both of these steps involve summing over the one- and two-electron integrals when applying the Slater Condon rules (Tables 1.1 and 1.2). The number of spawning attempts scales proportionally to the total number of psips as on each iteration each psip attempts to spawn children onto a single determinant; we denote the prefactor of this step as $C_1$. The diagonal matrix elements are stored for reuse so that diagonal death can be performed quickly. This means that the total cost of generating the diagonal matrix elements will scale proportionally to the number of successful spawning attempts. As the number of successful spawning attempts scales in proportion to the timestep and the population, we denote the prefactor of this contribution as $C_2$.

Under these assumptions the average total amount of computer time to perform a single timestep $\langle t \rangle$ is:

$$\langle t \rangle = C_1 \langle N_p \rangle + C_2 \delta\tau \langle N_p \rangle \tag{4.1}$$

[1] We use HANDE for all FCIQMC calculations in this thesis [58].

$C_1$ and $C_2$ and depend on both the computer architecture the calculation is running on and the compiler. We would expect a change of processor to only change $C_1$ and $C_2$ by some scale factors but not to have an order of magnitude change on the ratio of $C_1$ to $C_2$. We take care here to run all our timing calculations on one processor (core) of the same machine.[2] We have control over $\langle N_p \rangle$ and $\delta\tau$ by adjusting the parameters which control the psips population dynamics.

In Fig. 4.1 we have plotted the cost per timestep per psips $\langle t \rangle / \langle N_p \rangle$ for a range of different populations and systems ($C_1$ will be the intercept and $C_2$ will be the gradient of the fit lines). If Eq. 4.1 was obeyed exactly by the simulation then we would expect there to be no dependence between the number of psips and either the intercept or gradient of Fig. 4.1 for a given system. We see that for a large $\langle N_p \rangle$ Eq. 4.1 seems to nearly hold (the lines seem to converge) but for small $\langle N_p \rangle$ there seems to be a higher order effect.

This higher order effect means that the cost of an FCIQMC calculation per timestep step per psip is more expensive for a small population than a large one. Perhaps looping over the list of determinants and psips has become the dominant factor. This would make an interesting topic for future investigation but would require running many calculations. This term scales proportionally with the number of occupied sites $\langle N_{\text{occ}} \rangle$ (the cost of looping through the main list of psips). We know that $\langle N_{\text{occ}} \rangle$ must increase as a function of $\langle N_p \rangle$ and eventually reach a maximum once every determinant which has a finite CI coefficient is occupied with at least one psip at every timestep. The exact form of this monotonically increasing function will be complicated and depend on the sparsity of the CI wavefunction in determinant space.

We could run more calculations to quantify this higher-order effect, however we can draw some conclusions with the data we have. We know that doubling $\langle N_p \rangle$ approximately doubles the total computer time of running for a single timestep (if $\langle N_p \rangle$ is small then the total computer time slightly less than doubles) see Fig. 4.1. Doubling $\delta\tau$ is cheaper than doubling $\langle N_p \rangle$ because it only increases the runtime by a factor of $C_2 \langle N_p \rangle$ according to the analysis in Eq. 4.1 (ignoring higher order effects in $\langle N_p \rangle$). It is unlikely for any realistic calculation that this higher order effect will become important enough to mean that doubling $\langle N_p \rangle$ becomes cheaper than doubling $\delta\tau$.[3]

[2] By running on a single core we avoid complications resulting from $\langle N_p \rangle$ affecting load balancing, which will in turn affect $\langle t \rangle$. Load balancing has been extensively investigated in [82].

[3] For example in Fig. 4.1 bottom if we increase $\delta\tau$ from 0.0015 to 0.015 then the cost of the calculation per time step for $\langle N_p \rangle = 50000$ increases from 500 ns $\times 50000 = 0.025$ Seconds

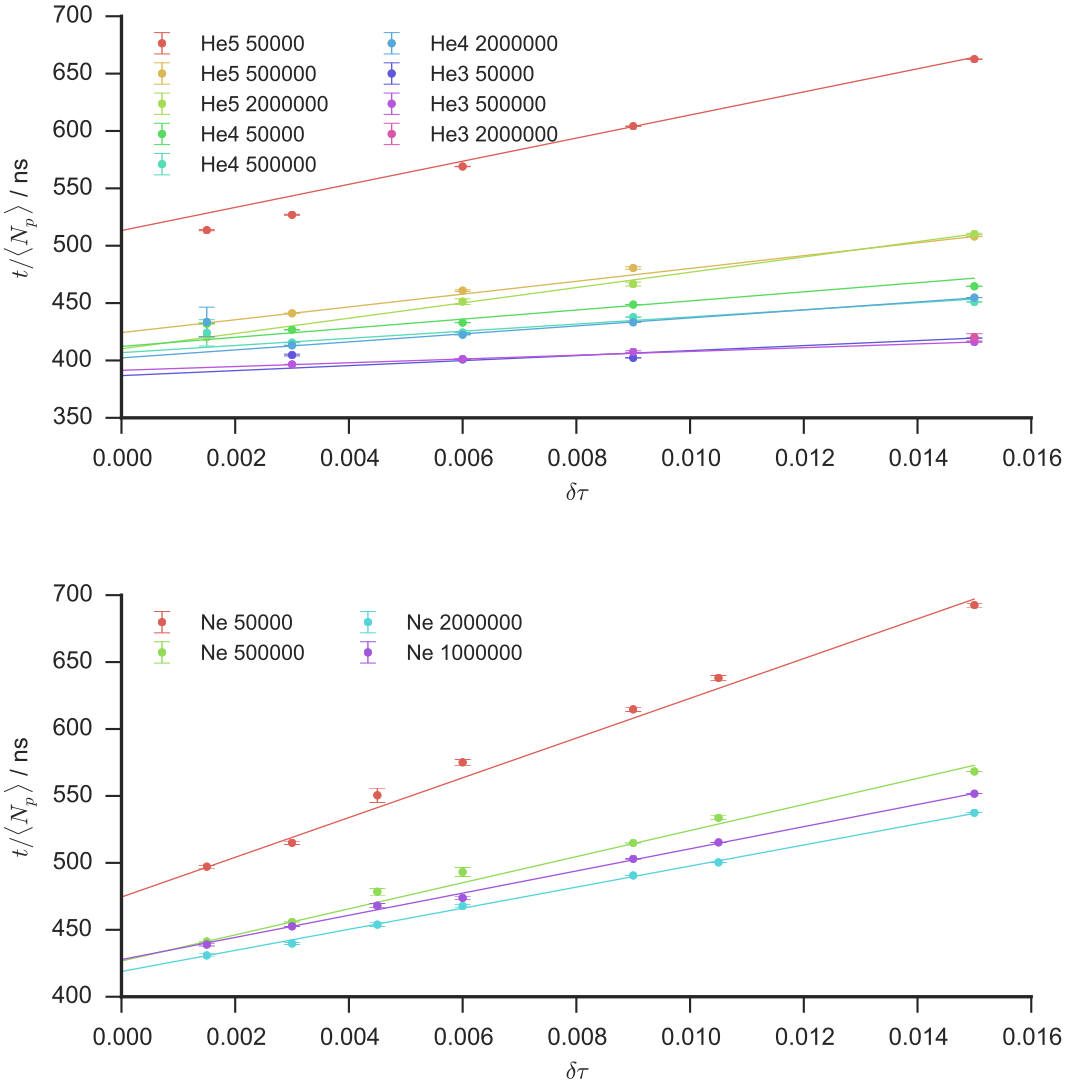



Figure 4.1.: The computer time per timestep per psip as a function of the timestep. Showing the linear relationship (Eq. 4.1) with the timestep and a near linear relationship with the number of psips for a large population. All calculations were run on a single core of an Intel Core i7-2600 processor.

## 4.2. Minimising the Error Bar

In the last section we discussed the scaling law which describes how the computational cost of a FCIQMC calculation scales with both $\delta\tau$ and $\langle N_p \rangle$. In this section we investigate how the prefactor of the square root law decays as a function of the

to 700 ns $\times 50000 = 0.035$. Where as increasing $\langle N_p \rangle$ from 50000 to 500000 for $\delta\tau = 0.0015$ increases the cost of the calculation per timestep to 430 ns $\times 500000 = 0.215$. So it is much cheaper to increase the $\delta\tau$ by a factor of ten than increase the $\langle N_p \rangle$ by a factor of ten.

parameters defining the Markov chain. In theory this could be computed exactly from the Markov chain using Eq. 2.9 although we found this impractical even for the two determinant system described in Sec. 2.2.1 (this requires summing over the elements of powers of the stochastic matrix). As we discussed in the last section doubling $\langle N_p \rangle$ doubles (or slightly less than doubles due to the higher order effects) the computer time of a single timestep whereas doubling $\delta\tau$ only causes the computer time to increase by $C_2\langle N_p \rangle$. $\delta\tau$ cannot be increased indefinitely as eventually the population will explode as discussed in Sec. 2.2.3.

For a system free of the sign problem one might expect the error bar to decay as:

$$\sigma_E = \frac{a}{\sqrt{\langle N_p \rangle N \delta\tau}}, \tag{4.2}$$

where $N$ is the number of timesteps taken, as in a sign problem free system (meaning that it is impossible for psips of opposite signs to appear on the same determinant) the psips do not interact with each other by annihilation. So we would expect that doubling $\langle N_p \rangle$ is the same as running with the original population for double the number of timesteps. We would also expect a similar behaviour in terms of $\delta\tau$: Doubling $\delta\tau$ makes every psip twice as likely to spawn a child and undergo diagonal death and thus doubles the amount of work done in one timestep.

$a$ is related to the sample standard deviation (often denoted $\sigma$ in the statistics literature). The standard error in the mean $\epsilon$ of an uncorrelated process can be written:

$$\epsilon = \frac{\sigma}{\sqrt{N}}, \tag{4.3}$$

where $N$ is the number of uncorrelated samples and $\sigma$ is an intrinsic property of a Monte-Carlo process. In FCIQMC the process is complicated by serial correlation: the correlation length scales proportionally to $\delta\tau$ and so the equivalent number of uncorrelated samples scales as $N\delta\tau$. Thus $a$ is analogous to $\sigma$ per psip. $\epsilon$ is analogous to $\sigma_E$ as $\sigma_E$ is the standard error in the mean projected energy.

If we assume that a FCIQMC calculation obeys both Eq. 4.2 and Eq. 4.1 then we can draw some conclusions about when a FCIQMC simulation will be most efficient. This problem can be phrased in another way: say we have a fixed number of CPU hours and want to minimise $\sigma_E$ as a function of $\langle N_p \rangle$, $N$ and $\delta\tau$. Eq. 4.2 states that $\sigma_E$ depends on the product $\langle N_p \rangle N \delta\tau$ i.e. increasing $N$, $\langle N_p \rangle$ or $\delta\tau$ decreases $\sigma_E$ by the same amount. In Eq. 4.1 increasing $\delta\tau$ increases the computer time less than increasing $\langle N_p \rangle$ by the same factor. Both increasing $\langle N_p \rangle$ and increasing $N$ cost the

same amount in terms of the computer time (doubling $\langle N_p \rangle$ doubles the cost of a single timestep and doubling $N$ doubles the number of timesteps). Thus we should use a large $\delta\tau$ to fill up the computational budget and run for a single step with a single psip.

Obviously this analysis is unrealistic as it is not possible to estimate $\sigma_E$ from a single step. Later in this chapter we concern ourselves with measuring $a$ as a function of $\langle N_p \rangle$ and $\delta\tau$ for a range of realistic systems which do not obey Eq. 4.2. Firstly we describe a method for measuring $a$ from a FCIQMC calculation. We will call $a$ the inefficiency, as $a$ is the hypothetical error bar which would result from running a FCIQMC for a single timestep, with a single psip, with $\delta\tau = 1$ and thus a small $a$ results in an efficient calculation.

### 4.2.1. Measuring $a$

Rearranging Eq. 4.2 we can see that $a$ depends on $\sigma_E$ and $\langle N_p \rangle$ (also $\delta\tau$ and $N$ but these are specified at runtime). $\langle N_p \rangle$ is a function of the state space (as discussed in Sec. 2.2). In FCIQMC we have two estimators of the energy, the projected energy and the shift. The shift is a function of the state space and the projected energy is a function of the numerator and denominator of the projected energy (two functions of the state space). Throughout this section we use the projected energy when measuring $\sigma_E$ as usually $\sigma_E$ for the projected energy is significantly smaller than $\sigma_E$ for the shift.

In the output of an FCIQMC calculation we obtain an estimate of each of these functions of the state space for every timestep. Serial correlation causes the standard error calculated from the estimates to be an underestimate of the statistical uncertainty (a systematic error). To provide an unbiased estimate we use the blocking algorithm described in [54]. This works by averaging together successive estimates of each function of the state space and computing the standard error of these blocks. Once the length of the block size becomes large enough the blocks become uncorrelated with each other and thus the standard error becomes an unbiased estimate of the uncertainty in the mean (the length of a block at this point might well be different for different functions of the state space). However if the block size is large enough such that the estimate of the standard error of the function is calculated using only a few blocks, then this will cause a large statistical error in the standard error of the function.

So the choice of block size must be appropriate to balance the statistical and

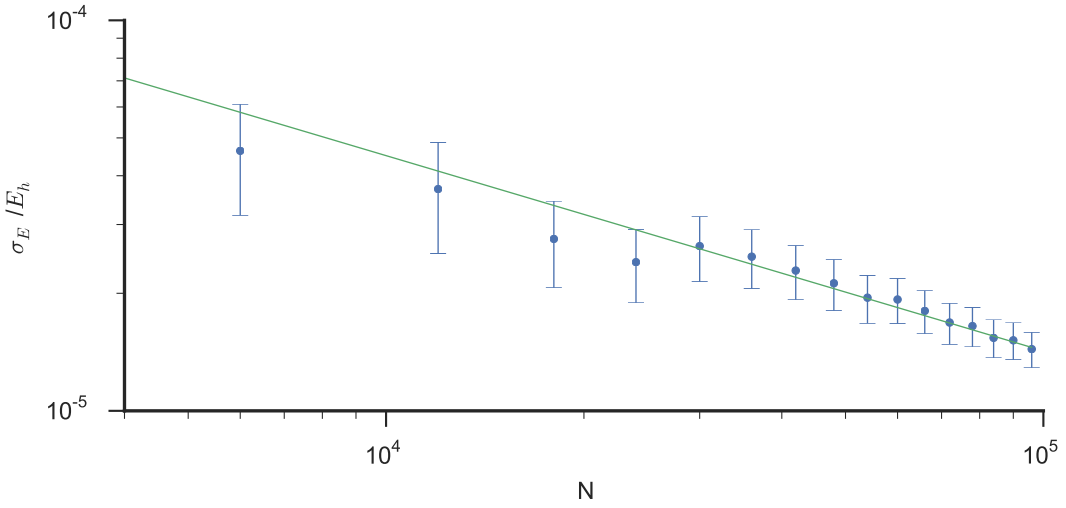



Figure 4.2.: The error bar in the projected energy as a function of $N$ (the number of timesteps included in the blocking analysis) for Ne cc-pVDZ with $\langle N_p \rangle = 84904(46)$ and $\delta\tau = 0.0015$.

systematic error in $\sigma_E$. For consistency we use an automatic approach to calculate the optimal block length as described in [55] which has previously been applied to DMC [56].[4] We have used this method throughout this thesis but it is especially important here as we need to consistently block many calculations.

The simplest way to measure $a$ from an FCIQMC calculation is to block the estimates of the numerator and denominator of the projected energy and $\langle N_p \rangle$ for every timestep after equilibrium. One can then calculate $a$ by rearranging Eq. 4.2. Another way to measure $a$ is to block an increasingly larger subset of the total data starting at the timestep after equilibration increasing the amount of data blocked at each iteration. This can then be fitted to a square root-law as a function of $N$ and $a$ can be extracted this way by multiplying the coefficient of the fitted square root by $\langle N_p \rangle$ and $\delta\tau$. In theory these two methods should be equivalent: we test this in Fig. 4.2. As the last point in Fig. 4.2 passes through the fit line it seems to suggest that the second method of calculating $a$ is as accurate as the first.

We calculate the error in $a$ using the error propagation rules and Eq. 4.2:

$$\sigma_a = \sigma_E \sqrt{\langle N_p \rangle N \delta\tau} \left( \left( \frac{\sigma_{\sigma_E}}{\sigma_E} \right)^2 + \left( \frac{\sigma_{N_p}}{2\langle N_p \rangle} \right)^2 \right)^{0.5}. \tag{4.4}$$

[4]Blocking analysis was performed using pyblock [57].

$\sigma_E$, $\sigma_{\sigma_E}$ and $\sigma_{N_p}$ are all calculated from the blocking analysis.

### 4.2.2. Varying $N_0$ and $\xi$

The analysis in Eq. 4.2 implies that $N_0$ and $\xi$ only have an impact on $\sigma_E$ by adjusting $\langle N_p \rangle$ and thus provide an extra degree of freedom (in Chapter 2 we showed that the population control parameters $N_0$ and $\xi$ can be set to give a specific $\langle N_p \rangle$ if the correlation energy is known 'a priori'). In Fig. 4.3 we show that this is indeed the case even for systems with a significant sign problem such as Ne aug-cc-pVDZ. A choice of too large $\xi$ and too small a $\langle N_p \rangle$ will introduce a bias as we investigated in Chapter 3. Perhaps if we were to converge $\sigma_E$ to the same extent as in our investigations of population control bias then the line in Fig. 4.3 might not be flat and any such bias is masked by statistical error. We expect this flat behaviour to apply universally to FCIQMC when population control bias is negligible.

We note that the stationary distribution of the projected energy was almost invariant to a change in population control parameters in Fig. 2.3 for $H_2$ in a STO-3G basis set supporting this hypothesis.

### 4.2.3. Scaling with $\langle N_p \rangle$

If annihilation is important then we expect different behaviour from Eq. 4.2. Let us first consider how $a$ behaves as a function of $N$. We expect FCIQMC to obey the square root law (i.e. $\sigma_E = \mathcal{O}(1/\sqrt{N})$). To understand this imagine averaging estimates of the energy from different timesteps such that enough imaginary time has passed so that the estimates of the energy are independent. As the number of independent estimates must scale linearly with the number of timesteps $N$ this must decay as $\mathcal{O}(1/\sqrt{N})$. We tested this when we investigated $\sigma_E$ as a function of $N$ in Fig. 4.2. In this section and the next we investigate how $a$ in Eq. 4.2 varies as a function of $\langle N_p \rangle$ and $\delta\tau$.

If $a$ followed Eq. 4.2 then $a$ would be independent of $\langle N_p \rangle$ and $\delta\tau$. In Fig. 4.1 we saw that (in terms of the CPU time) increasing $\langle N_p \rangle$ is slightly cheaper than increasing $N$ by the same factor and in the limit of large $\langle N_p \rangle$ the cost of both of these processes becomes the same (because of higher order effects). So increasing $\langle N_p \rangle$ would make the calculation more efficient in terms as the computational cost per step per psip decreases.[5]

[5]We ignore the time taken for equilibration in this analysis. This is usually small compared to the time taken to gather statistics.

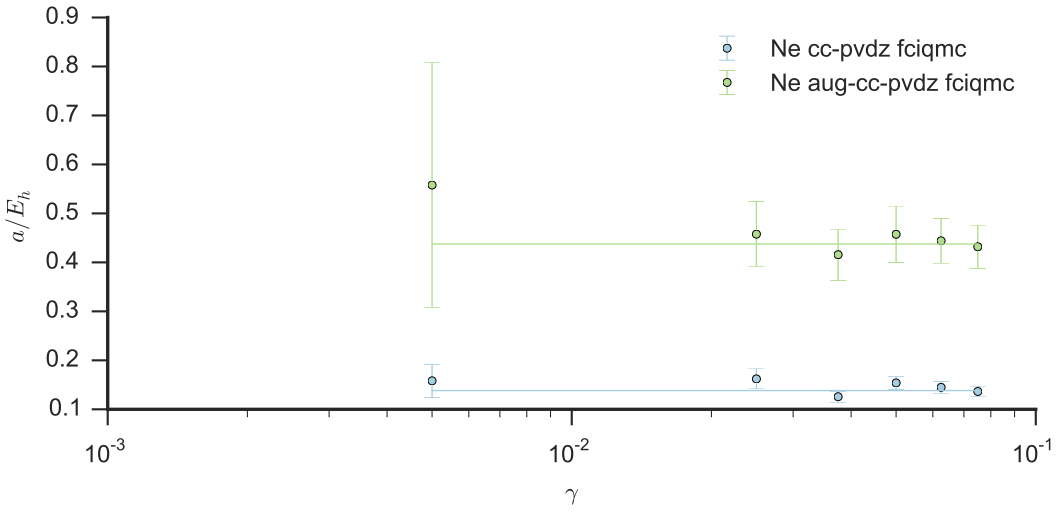



Figure 4.3.: The inefficiency $a$ as a function of $\gamma$ (the shift damping factor in Eq. 1.47). Using an estimate of the correlation energy we have adjusted $N_0$ and $\xi$ together (in the same way as Fig. 2.1) to produce the same $\langle N_p \rangle$. For Ne cc-pVDZ we used $E_c \approx -0.19211$ $(\xi, N_0) =$ (0.1,146446), (0.5,680982), (0.75,774028), (1,825216), (1.25,857540), (1.5,879789) to give $\langle N_p \rangle = 1000000$. For Ne aug-cc-pVDZ we used $E_c \approx -0.21313$ $(\xi, N_0) =$ (0.1,118683), (0.5,652947), (0.75,752636), (1,808051), (1.25,843240), (1.5,867546) to give $\langle N_p \rangle = 1000000$. $\delta\tau$ was set to 0.005 for both Ne cc-pVDZ and Ne aug-cc-pVDZ.

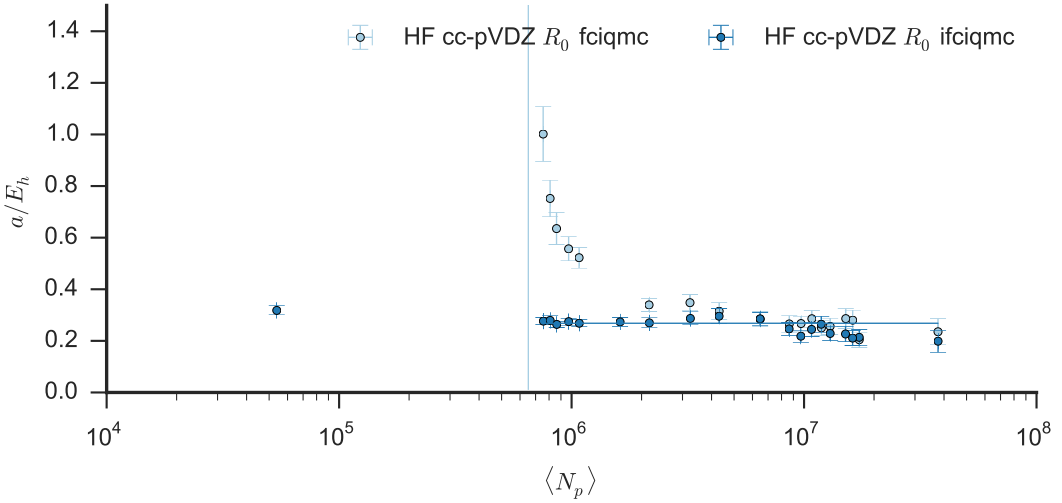



Figure 4.4.: The inefficiency $a$ as a function of the average number of psips. For HF in a cc-pVDZ basis with a bond length $R_0 = 0.91622$ Å (the Hartree–Fock equilibrium bond length) with $\delta\tau = 0.00175$ for both FCIQMC and iFCIQMC.

For FCIQMC we see in Fig. 4.4 (we see this behaviour in a wide range of systems in Sec. 4.3) that close to the plateau $a$ decays as a function of $\langle N_p \rangle$ with an asymptote at the plateau. As observed in the original FCIQMC paper [49] due to the sign problem which we discussed in Sec. 1.3.3 only once the population plateaus does the vector of psips and determinants become a stochastic representation of the eigenvector. So one would expect either that $\sigma_E$ diverges at this point or the average correlation energy to be incorrect which would probably have a knock on effect on $\sigma_E$.[6] In the limit where $\langle N_p \rangle$ is significantly larger than the plateau $a$ becomes a constant.

For initiator FCIQMC (iFCIQMC) we see that $a$ is a constant as a function of $\langle N_p \rangle$. The constant $a$ for iFCIQMC calculations (with the same $\delta\tau$ as for FCIQMC) is the same as $a$ for FCIQMC calculations in the large $\langle N_p \rangle$ limit (we call this the iFCIQMC limit). The point significantly below the plateau (at about $\langle N_p \rangle = 50000$) in Fig. 4.4 doesn't fit the fit line[7], this might be because the initiator error hasn't converged at this point and this in turn has an effect on the standard error of the projected energy. We see this behaviour for many systems which are discussed in Sec. 4.3. A power law behaviour between $\langle N_p \rangle$ and $\sigma_E$ has previously been observed

[6] We have managed to control the population below the plateau for example in Fig. A.2 bottom at $\delta\tau = 0.007$. $\sigma_E$ becomes very large at this point.
[7] $a = 0.2682(34)$ from the fit and $a = 0.318(18)$ at this point.

for iFCIQMC in [50] for a model Hamiltonian.

Fig. 4.4 shows that it may be worth using the initiator approximation well beyond the plateau as $a$ for FCIQMC doesn't hit the iFCIQMC limit (where $a$ is constant as a function of $\langle N_p \rangle$) until about $\langle N_p \rangle = 2000000$ for this system (other systems show the same behaviour which we discuss latter). So if even one can afford to do FCIQMC calculation (can afford to store the population at the plateau) a significant reduction in the error bar may be possible by using the initiator approximation if the population required to converge $a$ to the iFCIQMC limit is unaffordable. In Fig. 4.4 this would correspond to being able to store $\langle N_p \rangle = 700000$ but not $\langle N_p \rangle = 2000000$. It may however be difficult to understand if the introduction of a (systematic) initiator error is a price worth paying for a possibly significant reduction in statistical error bar.

Fig. 4.4 has implications about how best to parallelise FCIQMC and iFCIQMC. The simplest way to parallelise a Monte-Carlo algorithm is to use 'dumb' parallelism mode. Here one runs an independent FCIQMC calculation on each processor (with the pseudo-random number generator for each processor initialised with a different seed). Then statistics for each of the calculations can be combined together. Running in parallel in this way drops the error bar by a factor of $1/\sqrt{N_p}$ where $N_p$ is the number of processors (as running on $N_p$ processors for $n$ steps is equivalent to running for $NN_p$ steps on one processor).

In the traditional parallel FCIQMC implementation as detailed in [82] the Hilbert space is partitioned over the processors. Psips spawned from one parent determinant in one part of the Hilbert space onto a child determinant in another part of the space are communicated between processors. This process potentially limits the parallel scalability of FCIQMC.

This method should be used if it allows us to increase $N_p$ to get to the limit where $a$ becomes flat (about $\langle N_p \rangle = 2000000$ in Fig. 4.4) or to converge the initiator error for iFCIQMC.[8] After this point we should use 'dumb' parallelism mode i.e. for HF cc-pVDZ run multiple independent calculations with a population of about $\langle N_p \rangle = 2000000$ to minimise the effect of interprocessor communication.[9]

[8] We assume that at this point interprocessor communication doesn't significantly slow down the calculation i.e. that the decrease in $a$ is not cancelled by an increase in the cost of one time step.

[9] In the worse case interprocessor communication will be negligible and both methods of parallelism will be equally as efficient.

### 4.2.4. Scaling with $\delta\tau$

In Fig. 4.5 for small $\delta\tau$ we see that $a$ is a constant function of $\delta\tau$ and for large $\delta\tau$ we see $a$ increases for Ne aug-cc-pVDZ. A wide range of systems showing the same behaviour are discussed in Appendix A suggesting that this is universal in FCIQMC. We see the same behaviour whether or not the initiator approximation is used.

A previous investigation [83] has shown that when one increases $\delta\tau$ the plateau height remains constant until a certain point after which it increases. We wondered whether these two effects are caused by the same underlying process. This could potentially be useful as plateau height is cheaper to measure than $a$ as one can stop the calculation once the population reaches a plateau. Fig. 4.5 shows that the plateau height is a good metric for when $a$ remains constant as a function of $\delta\tau$. We have investigated a wide range of systems in Appendix A. Once $\delta\tau$ is large enough to raise the plateau height above $\langle N_p \rangle$ we expect $a$ to diverge in a similar way to in Sec. 4.2.3. The largest $\delta\tau$ such that $a$ is constant we denote $\delta\tau_0$.

If again we assume that Eq. 4.1 is true we know that increasing the timestep makes the calculation a factor of $C_2$ more expensive. Thus it is cheaper to double the $\delta\tau$ than to double $\langle N_p \rangle$ or $N$. We know that for small $\delta\tau$ $a$ remains constant as a function of $\delta\tau$ thus the calculation can be made more efficient by increasing $\delta\tau$. Once we get to the point where $a$ begins to increase with increasing $\delta\tau$ the situation is complicated by both Eq. 4.1 and how quickly $a$ increases as a function of $\delta\tau$. Thus $\delta\tau_0$ forms a lower bound for the most efficient timestep. Finding the actual most efficient timestep and not just a lower bound depends on how quickly $a$ increases as a function of $\delta\tau$ and Eq. 4.1.

In Fig. 4.6 have investigated whether $\langle N_p \rangle$ has an impact on $\delta\tau_0$ and find this is not case. We do however find that $a$ increases faster as a function of $\delta\tau$ when $\delta\tau > \delta\tau_0$ for a small $\langle N_p \rangle$ than a large $\langle N_p \rangle$. Meaning that the most efficient timestep depends on $\langle N_p \rangle$. To avoid running many expensive calculations when we compare different systems in the next section we will measure $a$ using a $\delta\tau < \delta\tau_0$ i.e. we measure $a$ where $a$ is a constant (as a function of $\delta\tau$).

As we discussed in Sec. 2.2.3 the population dynamics become metastable after the timestep is set to be greater than a certain critical value. Obviously it is undesirable for the timestep to be set such that the population is likely to explode during the calculation. We find that the point at which the population explodes is beyond $\delta\tau_0$ for the systems studied in this chapter with the exception of Ne cc-pVDZ (see Fig. 4.5).[10]

[10]When running the calculations for Fig. 4.5 we found that the population explodes when $\delta\tau$ is

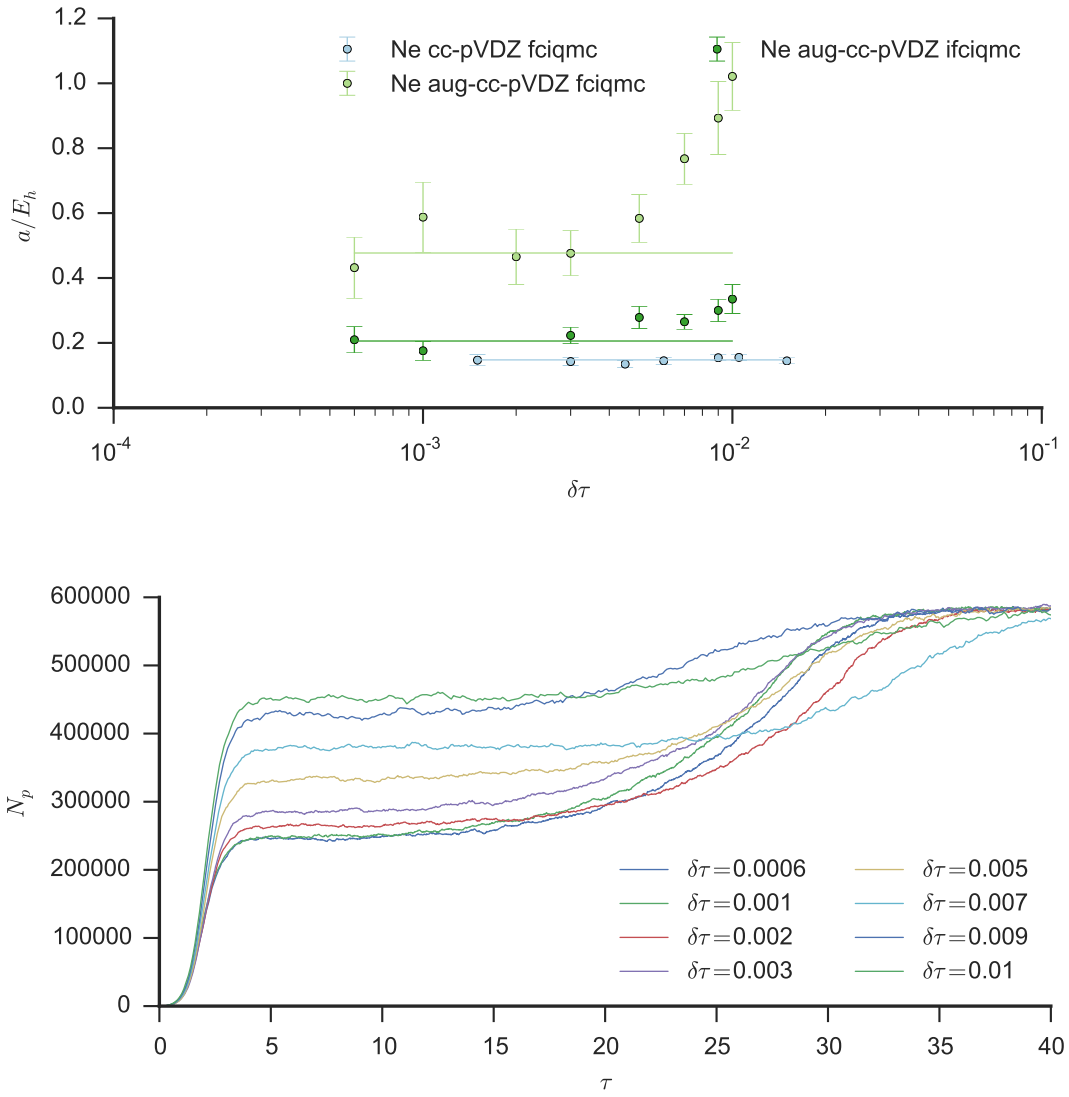



Figure 4.5.: Top: The inefficiency $a$ as a function of $\delta\tau$ for Ne cc-pVDZ and Ne aug-cc-pVDZ (for a fixed $\langle N_p \rangle \approx 46100$ and $\langle N_p \rangle \approx 590000$ respectively). Bottom: The plateau height can be seen to increase after $\delta\tau_0 \approx 0.003$ as a function of the timestep for Ne in a aug-cc-pVDZ basis set.

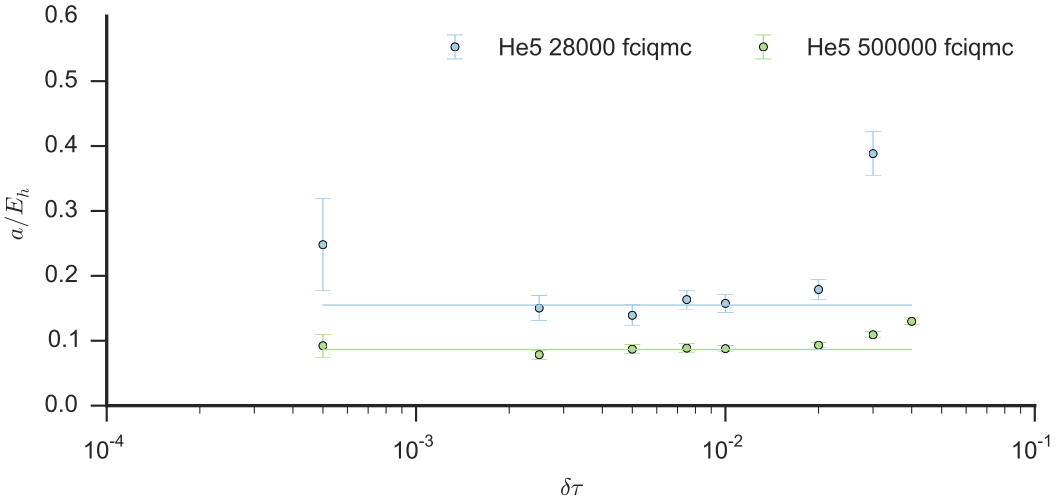



Figure 4.6.: The effect of increasing the timestep on the inefficiency $a$ for two different $\langle N_p \rangle$. There is no dependence on where $a$ diverges from the fit line for the different populations suggesting that $\delta\tau_0$ is a universal property for as given geometry and basis set.

## 4.3. The scaling of $a$ as a function of system size

In the previous sections we investigated both the scaling of $a$ as a function of $\langle N_p \rangle$ and $\delta\tau$. From the data in the last two sections we are now in a position to define a metric to let us compare how the error bar behaves for a range of systems.

The simplest such metric is to measure $a$ when it is a constant as a function of $\langle N_p \rangle$ and $\delta\tau$. This means using a $\delta\tau < \delta\tau_0$ and either the initiator approximation (making sure the initiator error is converged) or a large $\langle N_p \rangle$. This is the minimum $a$ for a given system. This is the hypothetical minimum error bar for running FCIQMC with $\tau = N\delta\tau = 1$ and $\langle N_p \rangle = 1$.

What we would really like to understand is how many core hours are required to converge an FCIQMC calculation to within a certain statistical error bar. To understand this we need to also understand how $C_1$, $C_2$ and $\delta\tau_0$ scale as a function of system size (number of electrons and basis set size) as well as $a$. An investigation of how $C_1$ and $C_2$ scale as a function of system size is left for future work. Some data on $\delta\tau_0$ is presented in Appendix A but we have only sampled enough points to check that the calculations for $a$ in this sections were performed with $\delta\tau < \delta\tau_0$ and not to accurately determine $\delta\tau_0$. In this section we attempt to quantify how $a$

slightly greater than 0.05 for Ne cc-pVDZ.

depends on basis set size, correlation energy and the number of electrons.

The neon atom has been investigated extensively previously and the plateau in the neon atom is known to be a small fraction ($\approx 0.01\%$) of the size of the Hilbert space [49]. This means FCIQMC works very well in this system, (this is not the case is a chain of helium atoms which we investigate later in this section). For these reasons we choose the neon atom to investigate how the number of basis functions has an impact on $a$. We find that in Fig. 4.7 $a$ scales linearly (within statistical errors) with the number of basis functions for the neon atom. Further investigation is necessary to tell if this behaviour extends to systems in which FCIQMC works less well.

We have also investigated stretching the bond length $R$ in the HF molecule and the effect this has on $a$, as this stretching changes the correlation energy. We start at $R_0$, the Hartree–Fock equilibrium bond distance of the molecule, to see the effect of this on the stochastic error bar (Fig. 4.8). Stretching the HF bond length has the effect of increasing $E_c$, and also increases and then decreases the plateau height. We find that $a$ increases with the bond length but $a$ does not scale linearly with the correlation energy, $R$, or the plateau height.

Finally we have investigated chains of helium atoms at a separation of 3 Å. This is a weakly interacting system and so the aim is to understand how $a$ scales in FCIQMC as a function of system size in this limit. To our knowledge this system has not been previously investigated using FCIQMC. A previous study investigated DMC scaling as a function of the number of hydrogen atoms at a large separation. [84] We attempted to investigate a system of hydrogen chains but found the calculations difficult to converge (for large chains) so we decided to investigate helium chains instead to understand how FCIQMC scales a function of system size. For this system the plateau seems to rise linearly as a function of the size of the Hilbert space. $a$ seems to scale slightly faster than linearly as a function of the number of helium atoms suggesting sublinear scaling with the size of the Hilbert space.

We also investigated the effect of localising the set of molecular orbitals on $a$. We separately localised using Pipek Mezey localisation [85] the occupied (in the Hartree–Fock) and unoccupied molecular orbitals to prevent these from mixing. Localising the orbitals causes many of the electron integrals to become very small (because the overlap between localised molecular orbitals far away on the Helium chain will be small). This breaks symmetry and so increases the size of the Hilbert space however we find that the plateau height decreases. A previous investigation [83] found that plateau height was unaffected by orbital localisation. Localisation

seems to also possibly decrease $a$ slightly compared to using the canonical orbitals (it is difficult to draw conclusions from the data in Fig. 4.8 as the error bars are large).

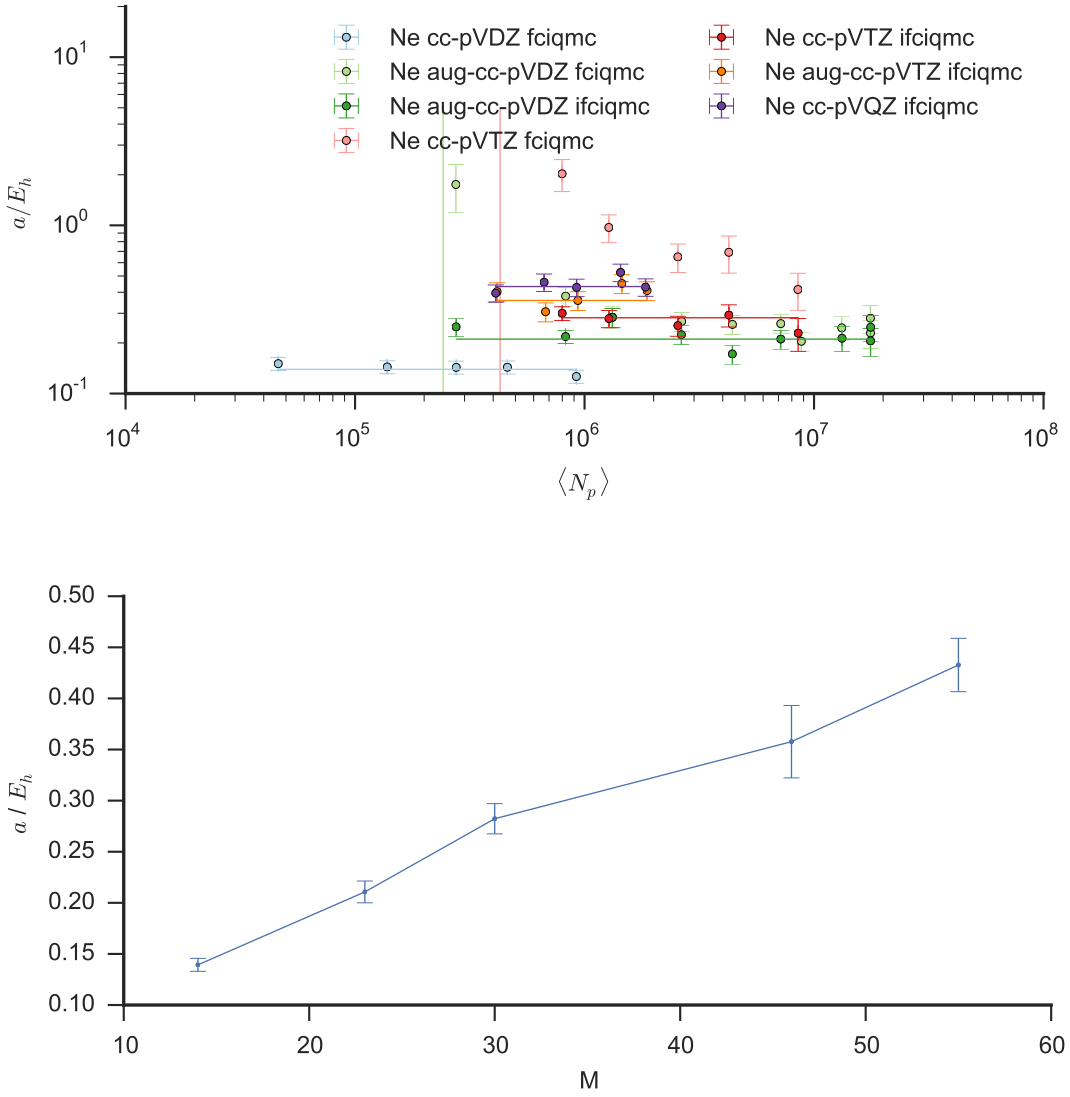
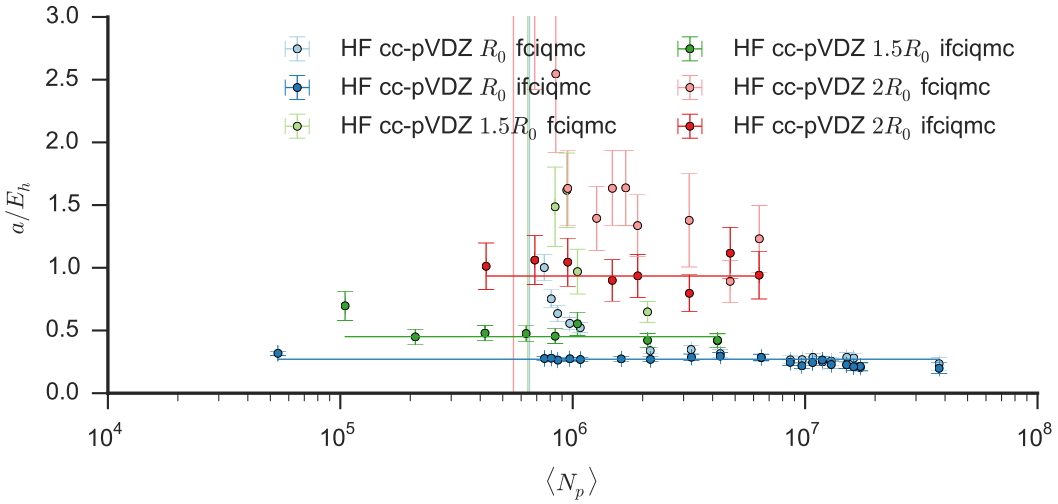



Figure 4.7.: Top: The inefficiency factor $a$ for the neon atom in a cc-pVDZ, aug-cc-pVDZ, cc-pVTZ, aug-cc-pVTZ and a cc-pVQZ basis set as a function of $\langle N_p \rangle$. Ne cc-pVDZ has no significant plateau and we find that $a$ is a constant function for $\langle N_p \rangle$. Bottom: the minimum value of $a$ as a function of basis set size. For Ne cc-pVDZ, Ne aug-cc-pVDZ, Ne cc-pVTZ, Ne aug-cc-pVTZ and Ne cc-pVQZ we used a $\delta\tau$ of 0.003606, 0.0022, 0.001075, 0.00065 and 0.00036 respectively (as determined in Appendix A).




| Bond Length Å | Plateau Height | $E_c$ | $a$ |
|---|---|---|---|
| 0.91622 | 651000(27000) | -0.2111374(21) | 0.2682(34) |
| 1.37433 | 747000(4000) | -0.2341493(34) | 0.471(36) |
| 1.83244 | 553000(27000) | -0.272692(16) | 0.974(45) |

Figure 4.8.: The inefficiency $a$ for HF as a function of $\langle N_p \rangle$ and bond length ($R_0 = 0.91622$ Å).). For $R = R_0$, $R = 1.5R_0$ and $R = 2R_0$ we used a $\delta\tau$ of 0.003, 0.001 and 0.001 respectively. These were determined in Appendix A.

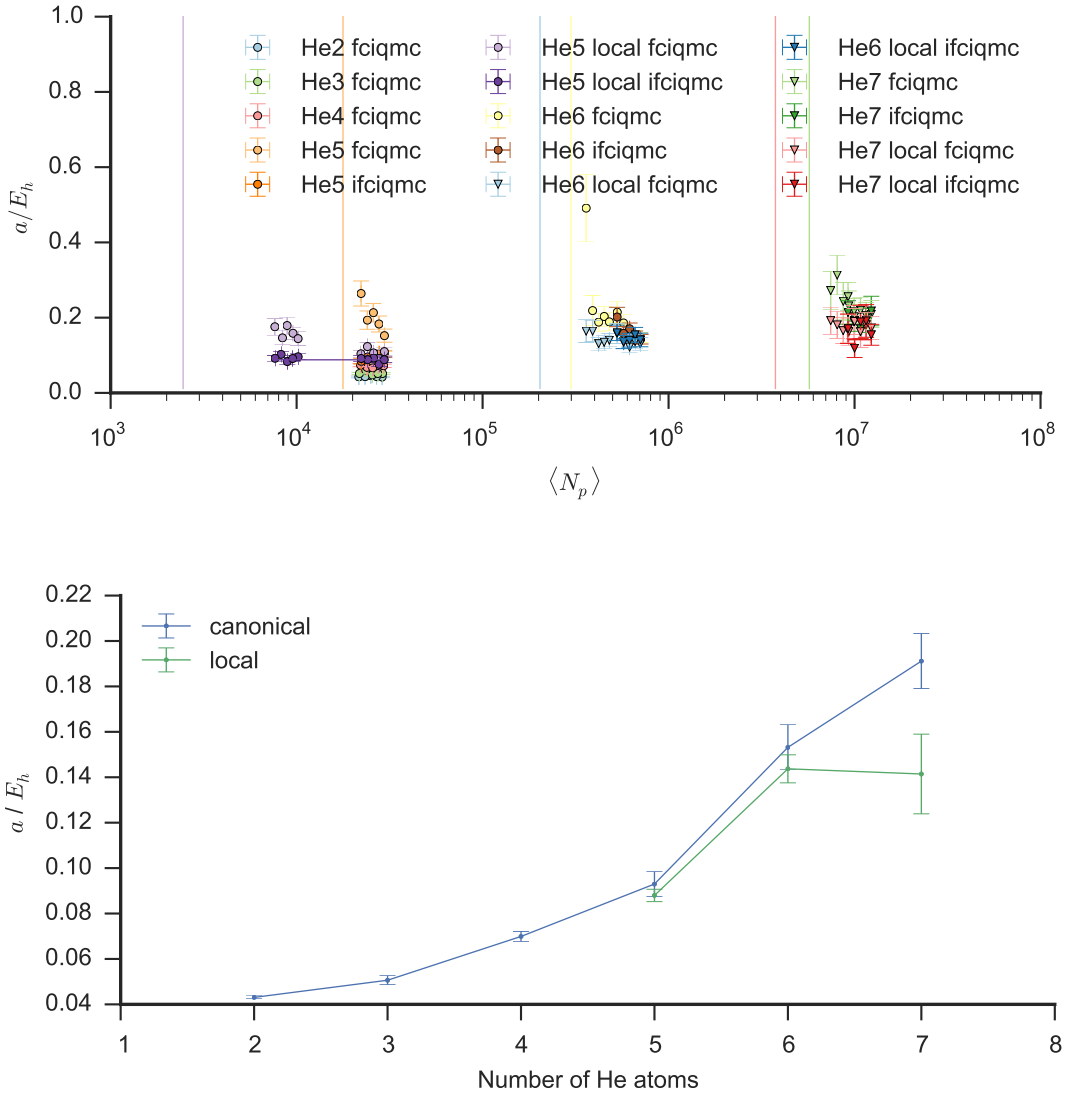



Figure 4.9.: Top: The ineffciency $a$ as a function of $\langle N_p \rangle$ and number of atoms in a helium chain. Bottom: The minimum ineffciency $a$ as function of the number of helium atoms, for helium in a 6-31G basis set.

## 4.4. Conclusion

We have empirically investigated the scaling of the error bar for FCIQMC as a function of the parameters which control the population dynamics. In FCIQMC we have found that a sizeable reduction in the stochastic error is possible by increasing the population of psips. We defined a metric $a$ to measure this. A lower bound on the most efficient timestep was found to be the largest timestep such that the plateau height remains constant. Thus we recommend using such a timestep and either a large population of psips or the initiator approximation.

Using this method of minimising $a$ we investigated how $a$ behaves as a function of system size. $a$ scales near linearly with the number of basis functions and faster than linearly as a function of the number of atoms.

In order to understand the amount of computer time required to converge FCIQMC to certain statistical error we need to understand how $C_1$ and $C_2$ in Sec. 4.1 depend on system size. Understanding this as a function of system size would make an interesting subject for a future investigation.

Orbital localisation was found to be effective for reducing the plateau height and thus the cost of a FCIQMC calculation. This means that it is worthwhile investigating how to perform excitations effectively using localised orbitals. Perhaps it will be favourable to select orbitals which are spatially close together more frequently than orbitals which are far apart (by modifying the generation probability). We expect this to significantly improve the efficiency of FCIQMC. We expect the calculations on the localised helium chain to reject many excitations so perhaps it is surprising that it significantly lowers the plateau as the localised orbital basis increases the size of the Hilbert space dramatically.

It may also be possible to design an unitary orbital rotation to reduce plateau height as the localised orbitals used in this chapter are not necessarily optimal in this regard.

# 5. A Review of Computer Architecture

In the previous chapters we have investigated the FCIQMC algorithm. We now briefly review the design of the computer architecture that simulations run on. We do this to provide context for the next chapter in which we describe an implementation of FCIQMC on the selected data flow architecture.

In the past users developing scientific software could gain speed up (from one generation of CPU to the next) without worrying about the structure of the underlying architecture (if they wanted to exploit a single processor). Between 1986 and 2003 the clock speed of computers increased by 40% per year, between 2003 and 2010 this slowed to 1% per year. [86] This is an overestimation of the decrease in the rate of real world performance because the instructions (fundamental operations that a processor performs at clock speed rate) have become more extensive and pipelined so a modern processor can run an algorithm more quickly than an older processor even if it has the same clock speed. In addition cache sizes have also increased to alleviate the gap between the rate of underlying mathematical events and bus performance. A better measure of single core performance is the SPEC benchmark [87] which shows performance increased by 52% per year between 1986 and 2003 and 22% per year between 2003 and 2010.[1][86]

## 5.1. CPU Architecture

Fundamentally a CPU consists of a set of registers (which store data) and a set of instructions which can perform operations on the values stored in these registers. These instructions may perform mathematical operations such as adding the values in two registers and placing the result in the third, a memory operation such as moving data from a memory address into a register or a control flow operation such as conditionally branching: reading the next instruction from an alternative location

[1] One has to be careful that the benchmark is a realistic approximation to the problem of interest.

based on a certain condition. It is the compiler's task to translate an algorithm, normally written in some high level programming language, into the set of basic instructions. The minimal set of instructions with which it is possible to perform any algorithm is small but typically the choice of instruction set architecture can have a large impact on the performance of a CPU.[2]

Instruction set architectures can be classified according to Flynn's taxonomy depending on how they operate on data. [89] The simplest type is a scalar processor with Single Instruction Single Data (SISD) instructions where each instruction acts on one item of data at a time. Another common processor type uses Single Instruction Multiple Data (SIMD) instructions where each instruction acts on multiple items of data at a time. Adding SIMD instructions to an instruction set architecture can greatly improve performance of algorithms which can be vectorised (meaning multiple operations can be performed at once). Multiple Instruction Single Data (MISD) instruction set architectures are not commonly used because they are inefficient. A contemporary x86 CPU is an example of Multiple Instruction Multiple Data instruction set architecture as it has multiple processors or cores operating in parallel on different items of data at once (we discuss the impact of this later).

Clock speed also has an impact on the performance of a CPU. Clock speed refers to the speed at which data is processed and transfered on a CPU from one operation to the next. A CPU does not necessarily perform one instruction per clock cycle. This, coupled with the different types of instruction (and cache size which we discuss in the next paragraph), means that clock speed is not a direct indicator of performance of a CPU.

By breaking instructions up into multiple partial stages they can be pipelined. This enables different parts of different instructions to be performed in parallel by the hardware. Each stage passes the result to the next stage of the pipeline. Although this process increases the number of instructions per clock cycle if one instruction depends on the result of another instruction still in the pipeline then the pipeline must be stalled and until the result is available wasting clock cycles. There are many optimisations such as out of order execution (reordering instructions to reduce data dependency) and branch prediction (predicting which result of a branch is the most likely and flowing the predicted path to prevent the pipeline from stalling). In the case of branch prediction, if it turns out that the other path should have been followed then the results of the branch are discarded and the other path is taken. A more detailed review of how instruction set architectures affect CPU performance is

[2]Any algorithm can be performed by a processor with a single instruction. [88]

beyond the scope of this thesis (see [86] or [90]).

Obviously it is desirable to raise the clock speed of a computer if possible without compromising on something else. Unfortunately the power consumption of a processor is directly proportional to the clock speed and depends on the square of the voltage across the processor. In the past to increase clock speed processor manufacturers have decreased the voltage that processors use and also allowed the power consumption to increase (see Fig. 5.1). Unfortunately the voltages have now become extremely low such that leakage currents have become a problem so that voltage can no longer be decreased. We have also reached the limit of the amount of heat that can be safely dissipated from a CPU. It is the combination of all these factors that has caused the rate of clock speed increase to stall.

Computational performance has nevertheless increased much more rapidly than DRAM access times and this means that large sections of a modern CPU are devoted to caches. Cache has a much smaller storage capacity than the DRAM but can be accessed in a smaller number of clock cycles. The idea is to store data which is reused in the cache to avoid the performance penalty associated with reading the data from the DRAM. This means that data locality can have a significant impact on performance (randomly updating a large vector is much slower than updating it sequentially). It is often difficult to predict when a certain item of data needs to be in the cache. If the required data must be read from the DRAM then the CPU must stop processing data for several clock cycles (known as a cache miss). Modern CPUs have multiple levels of caches (smaller faster caches at the top and larger slower caches at the bottom), which together with the DRAM and disk form the memory hierarchy. They also attempt to prefetch data that is likely to be needed so it is ready in the cache (it is much easier to predict the data needed for sequential data access patterns than for a random one).

Moore's law, which states that the number of transistors on a CPU doubles every two years, continues to hold (see Fig. 5.1). Much of the improvement of a single computer processing unit (i.e. a single physical chip) is being currently being made by moving to multi processor (Core) designs. The classification of this in Flynn's taxonomy is MIMD (multiple instruction multiple data). This is rather unfortunate because in order for computational chemistry codes to exploit these machines efficiently the algorithms have to be parallelised. The programmer must explicitly specify sections where the processors can operate independently and sections where the processors must communicate. Ideally two processors operating independently in parallel would perform the computation at twice the speed of a single processor.

So it is imperative that the programmer minimises interprocessor communication (as the penalty of these communications can be significant) and performs effective load balancing so that the amount of time one processor waits for another is minimised. There are a number of tools which provide language constructs to specify how the code should run in parallel such as OpenMP [91] (generally used when memory is shared between processors) and Message Passing Interface MPI [92] (used when memory is not shared between processors). Effective use of parallel processing means that computational chemistry software can exploit large parallel machines. However parts of computer algorithms are easier to parallelise than others.

The maximum acceleration which can be achieved by running an algorithm on multiple CPUs can be calculated formally using Amdahl's law. [94] This makes the assumption that any part of the algorithm which can run in parallel on $n$ processors runs at $n$ times the speed of any part of the algorithm which must run in serial. In this way the number of processors which can be effectively exploited can be estimated. This theoretical upper bound in the acceleration can never be practically achieved but Amdahl's law does provide a way of comparing computer algorithms by their theoretical parallel effectiveness.

Monte-Carlo methods have an advantage here as they can often be parallelised effectively and as expected computer architecture continues down the path of an increasing number of processors working in parallel, this might present a significant advantage. The easiest way to parallelise a Monte-Carlo algorithm is to use dumb parallelism mode (we discussed this in the context of FCIQMC in Sec. 4), each processor runs its own independent calculation (using a different sequence of random numbers) and the statistics from each processor are combined together.

These factors all mean that understanding 'a priori' how fast an application will run on a give CPU is difficult to quantify. These factors however do offer directions for optimisation, for example by reducing inter core communication or efficient load balancing or reducing the size of the working set of data to reduce the frequency of cache misses.

## 5.2. Graphical Processing Units

Recently computational scientists have started to exploit architectures based on Graphical Processing Units (GPUs). These architectures have been optimised to perform graphical operations effectively, and so if a problem can be structured to exploit this type of operation, GPUs can greatly speed up the application. On

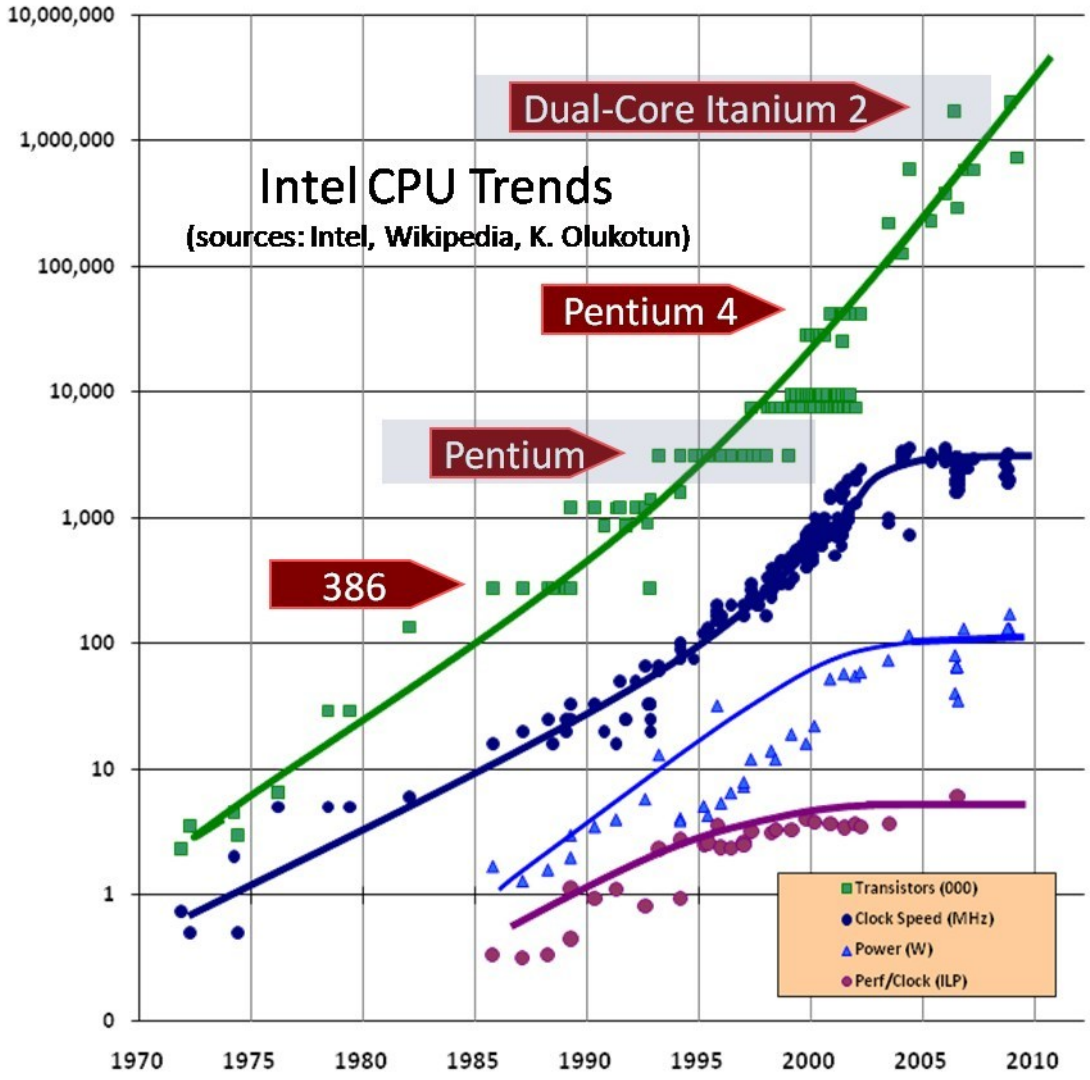



Figure 5.1.: The number of transistors, clock speeds, performance per clock cycle and power consumption for successive generations of Intel CPUs. Reproduced from [93].

a CPU a large section of the chip is dedicated to different memory hierarchies or caches. GPUs use a SIMD paradigm (this width is larger than SIMD instructions on a CPU) and are clocked at a slower speed. This means more area of the chip can be dedicated to performing arithmetic operations as we can hide the latency caused by accessing the DRAM by reading large sequential blocks as much less area is needed for cache.

This leads to a much greater data throughput: data can be processed at a much greater rate, but the operations must be independent or nearly independent for this process to be efficient. Data is typically transfered from the CPU to the GPU across the PCI express bus. This is a very slow operation (compared to reading directly from the DRAM) and so if a large amount of data must be processed on both the CPU and GPU then this can become the limiting factor. Libraries such as NVIDIA's CUDA [95] can be used to extend programs based on a programming language such as C or Fortran. Some of the most powerful supercomputers in the world, such as Titan [96] (currently ranked second in the world according to the top500 rank [97]) use GPU architecture.

Many software packages used by quantum chemists have been ported to a GPU. For example one study showed that the calculation of the two electron integrals has been made 130 times faster using a GPU compared to an AMD Opteron dual core processor. [98] Hartree–Fock and DFT calculations of water molecules have been sped up by 180 times. [99] Hartree–Fock gradient calculations have been sped up by 180 times. [100] Parts of the diffusion Monte-Carlo code QMCPack have successfully been ported to a GPU: Elser et al. [101] reported a speed up of 10-15 times compared to a Quad Core Xeon CPU for DMC calculations on solid state systems.

## 5.3. Data Flow Architecture

In Sec. 6 we will discuss FCIQMC and streaming data flow technology which provides a different computational paradigm than both CPUs and GPUs. Here we briefly discuss this technology. Streaming data flow technology is a system that incorporates a large Field Programmable Gate Array (FPGA) and a lot of DRAM organised in several parallel channels. These devices have a lower clock rate when compared to CPUs and GPUs, but their hardware can be reprogrammed to perform a custom set of logical operations in a long pipeline (the data flows from operation to operation). In this way an FPGA can be programmed to solve a specific problem.

This can be much more efficient than both CPU and GPU implementations as massive parallelism can be exploited, although this is limited by the available resources on the chip. [102]

Data flow architectural design has to take into account that data transfer between the Data Flow Engine (DFE) and CPU is slow as it involves the PCI express bus. Design of a parallel pipeline is more difficult than writing the algorithm in a high level language for use on a general purpose CPU. However the significant performance increase potentially makes this worthwhile. Brisk et al. used a DFE to speed up the alignment of DNA sequences. [103] They found that their implementation was 7 times faster than an Intel Xeon CPU and 3.5 times faster than an implementation on a NVIDIA GTX 580 GPU. To our knowledge no investigations using Data flow architecture for computational chemistry have been published.

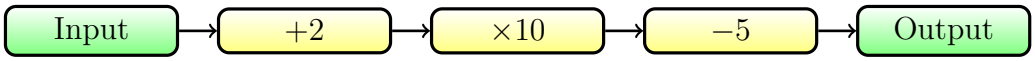
# 6. A Data Flow Approach to FCIQMC

In this chapter we describe a novel implementation of FCIQMC on a DFE. We have chosen the Hubbard Hamiltonian [63] because it only depends on two parameters meaning the one and two electron integrals are easy to compute. A molecular Hamiltonian requires the summing of many differing integrals when computing a matrix element. DFEs differ from CPUs because the DFE can be reconfigured for each individual algorithm and the very long pipelines of the implementation mean we can potentially exploit massive parallelism. These long pipelines can be hundreds or even thousands of operations long. With this in mind, in Sec. 6.2 we describe an implementation which performs FCIQMC with a throughput of one site (containing a number of psips on a determinant) per clock cycle.

In reality DFEs have a finite amount of space available on the chip. The best DFE implementation of FCIQMC will use this space such that the error bar decreases fastest as a function of the computer time used. In Sec. 6.3 we discuss the most efficient use of these resources, including scaling with system size and how many instructions we can actually use for systems of interest. Finally in Sec. 6.4 we discuss the prospects for molecular FCIQMC on a DFE.

## 6.1. General Programing for Data Flow

In this section we describe the fundamental operations from which a highly customised data flow pipeline can be constructed. We do this so that we can show how to organise these fundamental operations to build a design which can perform FCIQMC. DFEs rely on a streaming computation paradigm: a large memory is read from and written to sequentially, a customised stream of data flows from the DFE or CPU memory (via the PCI express bus). Each datum is then processed in




| Cycle | Input | +2 | ×10 | −5 | Output |
|---|---|---|---|---|---|
| 1 | 1 | ? | ? | ? | |
| 2 | 2 | $1+2=3$ | ? | ? | |
| 3 | 3 | $2+2=4$ | $3\times 10=30$ | ? | |
| 4 | 4 | $3+2=5$ | $4\times 10=40$ | $30-5=25$ | 25 |
| 5 | 5 | $4+2=6$ | $5\times 10=50$ | $40-5=35$ | 35 |
| 6 | 6 | $5+2=7$ | $6\times 10=60$ | $50-5=45$ | 45 |
| 7 | 7 | $6+2=8$ | $7\times 10=70$ | $60-5=55$ | 55 |
| 8 | | $7+2=9$ | $8\times 10=80$ | $70-5=65$ | 65 |
| 9 | | | $9\times 10=90$ | $80-5=75$ | 75 |
| 10 | | | | $90-5=85$ | 85 |

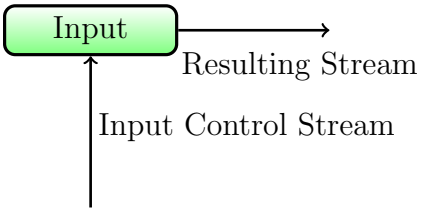
Figure 6.1.: A simple example of a stream algorithm with a pipeline depth of 3. In the diagram the black arrows represent the flow of data from one operation to the next. After an initial latency of three cycles the pipeline becomes fully utilised and so all three operations are performed on one item per cycle. In a more realistic example this pipeline could be thousands of operations deep. To exploit parallelism effectively the stream length should be much longer than the pipeline depth.

a pipeline and then the resulting data is written back to the memory.[1] A simple example of such stream processing with a pipeline depth of 3 is shown in Fig. 6.1. Once the pipeline becomes fully utilised all operations in the pipeline operate in parallel until the pipeline begins to empty. This model is efficient if the pipeline depth is much shorter than the amount of data to be processed. As this pipeline depth can potentially be thousands of operations long we can exploit the massive parallelism of the DFE on large datasets.

The functions which make up a stream process are known as a kernel. An input to a kernel is used to generate a stream by reading an array of data sequentially. By controlling the input one can control how fast the kernel reads the array: on each iteration the input can either read a new item of data from the array or repeat the last item read from the array (see Fig. 6.2). An output of a kernel can be used to do the opposite (convert a stream of data into an array see Fig. 6.3). In Figs. 6.2 and 6.3 an arrow indicates the current position of the stream in an array of data.

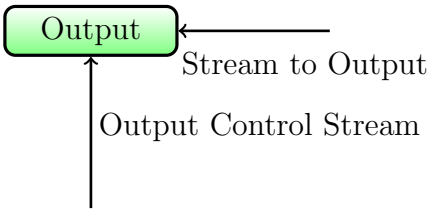
Many operations from a standard programming language can operate on a stream

[1] A stream can randomly access a small amount (about 5 MB in total) of fast memory (see Fig. 6.4). The large memory can hold up to 24 GB on the MAX3 Vectis card used throughout this thesis.




| Cycle | | | | | | Input Control Stream | Resulting Stream |
|---|---|---|---|---|---|---|---|
| | ↓ | | | | | | |
| 1 | $a_1$ | $a_2$ | $a_3$ | $a_4$ | ... | T | $a_1$ |
| | ↓ | | | | | | |
| 2 | $a_1$ | $a_2$ | $a_3$ | $a_4$ | ... | F | $a_1$ |
| | | ↓ | | | | | |
| 3 | $a_1$ | $a_2$ | $a_3$ | $a_4$ | ... | T | $a_2$ |
| | | ↓ | | | | | |
| 4 | $a_1$ | $a_2$ | $a_3$ | $a_4$ | ... | F | $a_2$ |
| | | ↓ | | | | | |
| 5 | $a_1$ | $a_2$ | $a_3$ | $a_4$ | ... | F | $a_2$ |
| | | | ↓ | | | | |
| 6 | $a_1$ | $a_2$ | $a_3$ | $a_4$ | ... | T | $a_3$ |
| | | | | ↓ | | | |
| 7 | $a_1$ | $a_2$ | $a_3$ | $a_4$ | ... | T | $a_4$ |

Figure 6.2.: An input of a DFE can be thought of as a pointer addressing an array of data $a_1, a_2 \ldots$ (this may come from the PCI express bus or or from the DFE RAM). On each cycle the input can be controlled using a stream of booleans: if the stream is true $T$ the pointer will increment to the next value of the array; if false ($F$) the pointer stays at the same point. This results in a stream which can contain repeating elements of the array in the same order as the original array.




| Cycle | Stream to Output | Output Control Stream | |
|---|---|---|---|
| | | | ↓ |
| 1 | $o_1$ | T | $o_1$ |
| 2 | $o_2$ | F | $o_1$ |
| 3 | $o_3$ | F | $o_1$ |
| | | | ↓ |
| 4 | $o_4$ | T | $o_1$ $o_4$ |
| | | | ↓ |
| 5 | $o_5$ | T | $o_1$ $o_4$ $o_5$ |
| 6 | $o_6$ | F | $o_1$ $o_4$ $o_5$ |
| | | | ↓ |
| 7 | $o_7$ | T | $o_1$ $o_4$ $o_5$ $o_7$ |

Figure 6.3.: An output of a DFE works in the opposite way to an input. If the output control stream is true T then an item is appended to the array. If false F it is thrown away. The array may be written to the DFE memory or to the CPU memory via the PCI express bus.

of data. Some simple examples are mathematical operations (e.g. addition) or simple logical operations e.g. testing equality. More complex operations such as the ternary if statement, which takes an input of a Boolean control stream, which controls whether an output is set to one of two possible input streams. Stream offsets act as a delay structure which holds values of the stream from a number of user defined previous cycles. Read Only Memory (ROM) can read an array of data (placing it into an output stream) at an address specified by the input stream. Random Access Memory (RAM) can read an array of data in the same way as a ROM. Additionally the array can be overwritten by a stream of data (the address of where to write the data is specified by another stream). Some of these fundamental operations are shown in Fig. 6.4 along with an example of each processing one or many streams of data.

The input to a kernel creates an implicit loop over the input array of data. Nested

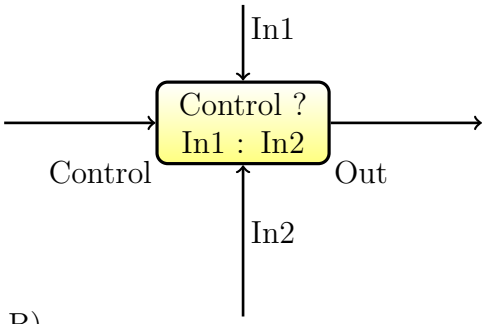
loops can be implemented in a number of ways. Let's denote the number of iterations the nested loop must run for as $n$. Probably the simplest method is to unroll the loop. This involves implementing the operations inside the loop $n$ times. This can be effective if $n$ is both small enough to fit in the logic of the chip and known at compile time.[2] This technique is used extensively in our implementation when looping over basis functions as described in Sec. 6.2. A fully unrolled loop performs the entire loop once per clock cycle.

Nested loops can also be implemented by controlling the inputs and outputs. The input must be set up to read every $n$ cycles and the pipeline must be designed to perform the first iteration of the inner loop every $n$ cycles, the second every $n + 1$ cycles etc. The output must be set to write when the computed data needs to be written. If there are further lower level nested loops or $n$ is difficult to predict, implementing these designs in hardware can be challenging, although algorithms can often be restructured so this is not a problem. Loops nested in this way perform the entire pipeline every $n$ cycles and so loop unrolling is preferable if the resources are available. Often a combination of these two techniques is used so that the inner loop is unrolled enough times to use all the DFE resources, thus providing as much parallelism as possible. Controlled inputs are then used to handle the rest of the iterations of the loop.

Data dependency can also be important when designing a pipeline to perform loops. If each iteration of a loop requires data computed on one or several previous iterations, then the latency of operations in the loop may be important. Data can be sent backwards (as opposed to the standard forward progression through the pipeline) in a kernel to accumulate numbers into a stream. An example of this is shown in Fig. 6.5 which uses this accumulation to sum the rows of a matrix. This example works because integers can be added in a single clock cycle. In general, operations on floating point numbers must take longer than a single cycle because manipulation of the exponent and mantissa requires many operations. We discuss a solution to this problem when we discuss computing the projected energy.

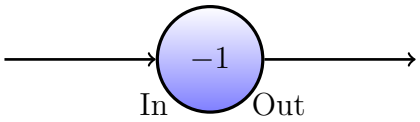
[2]This might also be effective if the maximum possible $n$ is known at compile time.

A)




| Control | In1 | In2 | Out |
|---|---|---|---|
| T | $a_1$ | $b_1$ | $a_1$ |
| F | $a_2$ | $b_2$ | $b_2$ |
| F | $a_3$ | $b_3$ | $b_3$ |
| T | $a_4$ | $b_4$ | $a_4$ |

B)



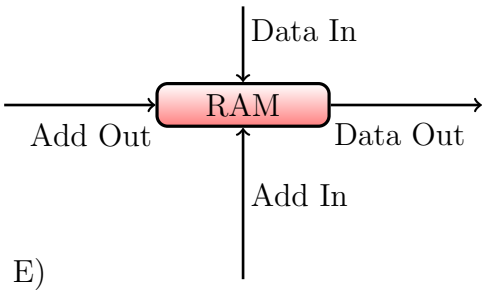

| In | Out |
|---|---|
| $a_1$ | ? |
| $a_2$ | $a_1$ |
| $a_3$ | $a_2$ |
| $a_4$ | $a_3$ |

C)



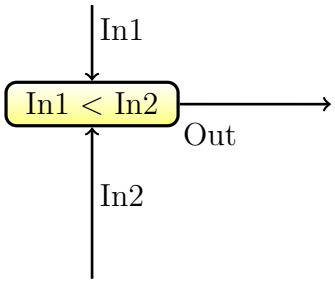

| Address | Out | | | | |
|---|---|---|---|---|---|
| 0 | $r_0$ | $r_0$ | $r_1$ | $r_2$ | $r_3$ |
| 2 | $r_2$ | $r_0$ | $r_1$ | $r_2$ | $r_3$ |
| 4 | $r_4$ | $r_0$ | $r_1$ | $r_2$ | $r_3$ |
| 2 | $r_2$ | $r_0$ | $r_1$ | $r_2$ | $r_3$ |

D)



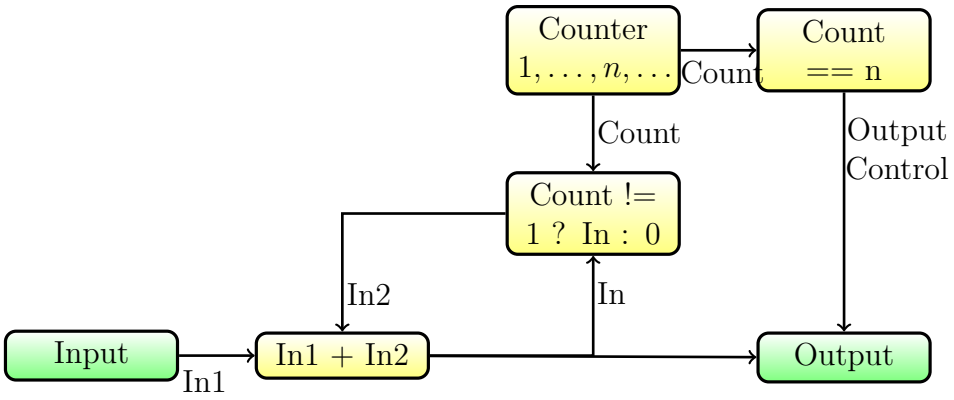

| Address In | Data In | Address Out | Data Out | | | | |
|---|---|---|---|---|---|---|---|
| 0 | $a_0$ | 0 | $a_0$ | $a_0$ | $r_1$ | $r_2$ | $r_3$ |
| 1 | $a_1$ | 2 | $r_2$ | $a_0$ | $a_1$ | $r_2$ | $r_3$ |
| 3 | $a_2$ | 1 | $a_1$ | $a_0$ | $a_1$ | $r_2$ | $a_2$ |
| 1 | $a_3$ | 2 | $a_1$ | $a_3$ | $a_1$ | $r_2$ | $a_2$ |

E)




| In1 | In2 | Out |
|---|---|---|
| 0 | 3 | T |
| 5 | 5 | T |
| 11 | 12 | F |
| 4 | 5 | F |

Figure 6.4.: Some fundamental stream operations. A) Ternary if: an output is set to one of two inputs by a Boolean. B) Stream offset: The data in a stream is delayed by a number of cycles. C) ROM: An array of data can be read. D) RAM: An array of data can be read and written. E) Less than: an output is calculated from two inputs.




| | | | | | | | | Count | In1 | In2 | In1 + In2 | Output Control | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ↓ | | | | | | | | | | | | | | |
| 1 | 2 | 3 | 4 | 6 | 7 | 8 | 9 | 1 | 1 | 0 | 1+0=1 | F | | |
| | ↓ | | | | | | | | | | | | | |
| 1 | 2 | 3 | 4 | 6 | 7 | 8 | 9 | 2 | 2 | 1 | 2+1=3 | F | | |
| | | ↓ | | | | | | | | | | | | |
| 1 | 2 | 3 | 4 | 6 | 7 | 8 | 9 | 3 | 3 | 3 | 3+3=6 | F | | |
| | | | ↓ | | | | | | | | | | ↓ | |
| 1 | 2 | 3 | 4 | 6 | 7 | 8 | 9 | 4 | 4 | 6 | 4+6=10 | T | 10 | |
| | | | | ↓ | | | | | | | | | | |
| 1 | 2 | 3 | 4 | 6 | 7 | 8 | 9 | 1 | 6 | 0 | 6+0=6 | F | 10 | |
| | | | | | ↓ | | | | | | | | | |
| 1 | 2 | 3 | 4 | 6 | 7 | 8 | 9 | 2 | 7 | 6 | 7+6+13 | F | 10 | |
| | | | | | | ↓ | | | | | | | | |
| 1 | 2 | 3 | 4 | 6 | 7 | 8 | 9 | 3 | 8 | 13 | 8+13=21 | F | 10 | |
| | | | | | | | ↓ | | | | | | | ↓ |
| 1 | 2 | 3 | 4 | 6 | 7 | 8 | 9 | 4 | 9 | 21 | 9+21=30 | T | 10 | 30 |

Figure 6.5.: Top: A kernel which sums the rows (of length $n$) of a matrix. The counter repeatedly produces an arithmetic sequence of numbers between 1 and $n$ incrementing by one. Bottom: An example with $n = 4$ summing the rows 1,2,3,4 and 6,7,8,9.

## 6.2. A Data Flow Implementation of FCIQMC

An overview of the DFE implementation of FCIQMC is shown in Fig. 6.6. The implementation has been designed to exploit the massive parallelism of a DFE, using a long pipeline. In this way spawning and diagonal death is performed at the rate of close to one determinant per clock cycle (once the pipeline becomes full).[3]

To decide on the best design for a pipeline which performs one timestep of the FCIQMC algorithm, let us consider the loop structure of the sub-steps, spawning, diagonal death and annihilation. The spawning step requires two important loops as well as some lower level inner loops over spin orbitals which are short enough to be fully unrolled. The first important loop is the outermost loop over the array of determinants occupied after the last timestep. The other important loop is nested a level lower than the outermost loop. This loops over the number of psips on the specific determinant at the position of the outermost loop. Contained within this loop is the algorithm which selects a random child determinant and attempts to spawn onto it. If a spawning event is successful then the newly spawned determinant needs to be appended to a list of newly spawned walkers. The lengths of both of these loops can be predicted at the beginning of the timestep because we know how many psips are on each determinant at this point.

Diagonal death requires a single important loop over the list of determinants occupied after the last timestep. Inside this loop all of the psips on the specific determinant die at the same time with an appropriate weight. On each iteration we can output the determinant and the number of surviving parent psips to a list, unless all psips on a determinant have annihilated. This list of surviving parent psips remains in the same order as the list of determinants occupied after the last timestep.

Annihilation first involves sorting the array of newly spawned walkers into some arbitrarily determinant order. The most efficient sorting algorithms take on average $n \log(n)$ operations, where $n$ is the length of the array to be sorted.[104] After sorting, although this list is in order, multiple psips may have spawned child psips onto the same determinant. This means that after sorting, the spawned psip array may have multiple copies of the same determinant on it. These will be next to each other as the list is now in order. It is efficient to internally annihilate the psip array so as

[3]The limiting factor is the merging together the list of spawned psips with the list of surviving parent psips. This takes $L_M + L_S$ clock cycles where $L_M$ is the length of the main list and $L_S$ is the length of the spawned list. Some extra clock cycles are also used to remove determinants that have died (as the DFE must write a predetermined amount of data).

to only have a single copy of each determinant. This involves looping over the psip array twice. Firstly the psips on the multiple determinants are summed onto the single determinant at the top, after which the list is compressed by removing the holes. Finally we have to merge the spawned psip array into the surviving parent arrays.

Computing both estimates of the energy requires a single important loop over the merged array of determinants and psips.

In a pipeline there is no obvious way to completely sort the newly spawned list and merge it into the main walker list and later on in the pipeline perform spawning and diagonal death. This is because the data access pattern from sorting is different from that of spawning and diagonal death. The sorting algorithm must first sort the entire list before we can guarantee that any one element is in order (using a sorting algorithm which is $n\log(n)$). To get around this problem we sort and internally annihilate the spawned walker list on the CPU and access it from the DFE via the PCI express bus. We discuss this for the Hubbard model in Sec. 6.3. The hope was to process the data on CPU in parallel while the DFE was performing spawning and diagonal death using a sort and merge algorithm. Though we have yet to perform this optimisation we make a prediction about how much of a speed up this optimisation could give in Sec. 6.3.

We can modify the spawning stage so that all psips on the same determinant spawn onto only a single connected determinant with an appropriate weight. We do this because we can then perform all the stages except sorting the newly spawned psips in parallel on the DFE. This will undoubtedly decrease the rate at which the stochastic error converges because the rate at which the simulation explores the Hilbert space will decrease. It will also decrease the computational time to perform a single time step. These consequences are discussed in depth in Sec. 6.3 for both a DFE and CPU. We call this modified FCIQMC algorithm mFCIQMC to distinguish it from the traditional FCIQMC algorithms.

*To summarise:* the DFE first merges two lists of psips and determinants together, performing some annihilation for one timestep. We then use this merged data to calculate the projected energy from this time step. Together all the psips on each determinant attempt diagonal death and spawning for the next timestep. The newly spawned walkers are streamed across the PCI express bus to the CPU where they are sorted. The sorted list is then streamed back to the DFE to process it on the next step. In the next few sections we discuss in more depth how we use data flow components to perform the operations in this pipeline. The merge is the limiting

factor of the number of clock cycles to perform a single timestep.

All the pipelines described within this thesis were implemented using a Java application program interface developed by Maxeler Technologies (who also manufacture the Data Flow engine hardware used to generate the results in a latter section). This design is compiled into a Hardware Description Language which generates the circuits on the Data-Flow engine. Our implementation is linked to the open source stochastic computational chemistry package HANDE which performs the operations on the CPU.[58]

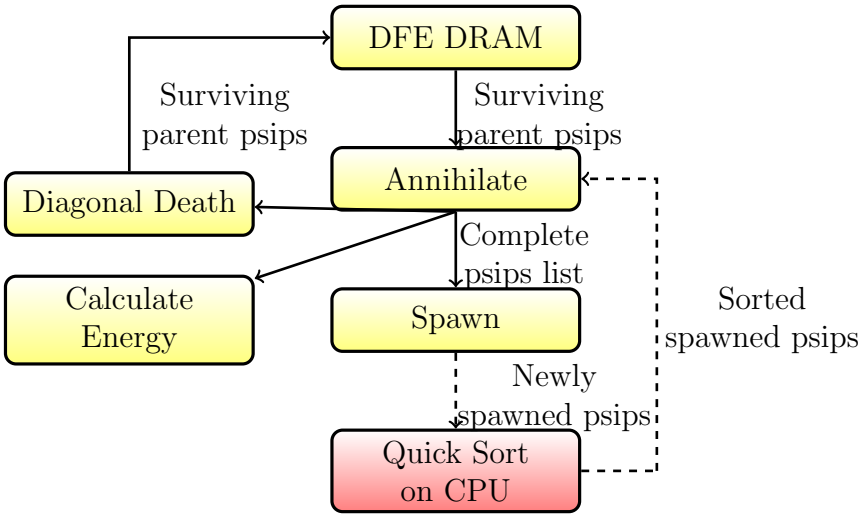



Figure 6.6.: An overview of the DFE implementation of FCIQMC. Solid lines show a stream of data flowing internally on the DFE or between the DFE and the connected DRAM. Dashed lines show data transfer of a stream across the PCI express bus.

### 6.2.1. Annihilation

Annihilation between the surviving parent psips and the newly spawned walkers is performed at the top of the pipeline. This involves merging the sorted array of psips spawned on the last iteration of the DFE and the surviving parent psips from the last iteration of the DFE and annihilating any psips which are on the same determinant. The algorithm which performs this merge is detailed in Appendix B as this pipeline is complicated by many boundary conditions. This algorithm is basically an implementation of the final iteration of a merge sort (this merges two in-order lists).[4]

[4]For a detailed discussion of the merging algorithm see [104] and Appendix B.

### 6.2.2. Calculating the Projected Energy

The denominator of the projected energy (see Eq. 1.49) is computed by storing the number of psips on the Hartree-Fock determinant from the stream of determinants and psips. The numerator of the projected energy involves summing over a floating point factor for each determinant: the matrix element between the determinant and the Hartree-Fock determinant multiplied by the number of psips on the determinant. For the Hubbard model in a reciprocal space basis which we discussed in Sec. 1.3.4 the allowed off-diagonal matrix elements are either $U/N$ or $-U/N$ depending on permutation symmetry of the occupied spin orbitals and the kinetic energy operator.[5]

Summing floating point numbers on a DFE involves a latency of 11 cycles i.e. it takes 11 clock cycles to compute a floating point add. The standard way to accommodate this is to use the algorithm shown in Fig. 6.7. We have to sum the 11 elements of this stream on the CPU to compute the numerator of the projected energy.

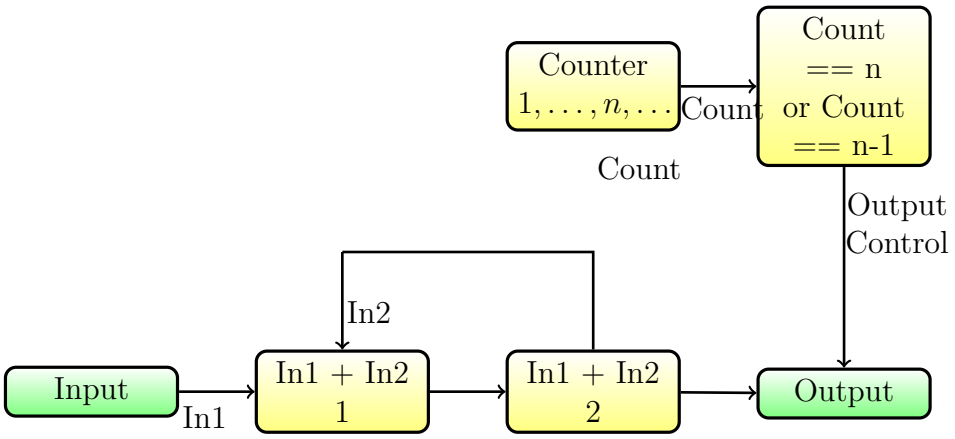
The shift Eq. 1.47 simply involves summing the total number of psips, for which we use a version of the row sum example Fig. 6.5 modified to sum the entire stream.

### 6.2.3. Spawning and Diagonal Death

Due to the modifications to the FCIQMC algorithm as discussed in Sec. 6.2, the algorithm to decide how many psips to spawn on a child determinant as a function of the probability $P_e$ that an event happens is the same for spawning and diagonal death. We spawn/kill/clone $\lfloor P_e \rfloor$ psips deterministically (with probability 1). We use a random number uniformly distributed in the interval $[0, 1]$ to stochastically spawn/kill/clone a single further psip with probability $P_e - \lfloor P_e \rfloor$. The sign convention is the same as conventional FCIQMC (see Sec. 1.3). For spawning $P_e = |n_i \langle D_i | \hat{H} | D_j \rangle \delta\tau|$ and for diagonal death $P_e = |n_i \langle D_i | \hat{H} - S | D_i \rangle \delta\tau|$, where $n_i$ is the number of psips on determinant $i$.

The diagonal matrix element contains a constant (for a specific system) two electron matrix element, and for each occupied spin orbital we add one single electron term (see Fig. 6.8). The shift is delayed by one timestep. We do this because annihilation (from the previous timestep) is occurring in parallel with spawning and

[5] We can calculate the number of permutations to maximally align the determinants by looping over the spin orbitals to find out which ones differ between the bra and the ket. This uses a similar algorithm to Fig. 6.8.




| | | | | Count | In1 | In2 | In1 + In2 1 | In1 + In2 2 | Output Control | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| ↓ | | | | | | | | | | | |
| 1.5 | 2.2 | 3.3 | 4.6 | 1 | 1 | 0 | 1+0=1 | 1.5+0.0=1.5 | F | | |
| | ↓ | | | | | | | | | | |
| 1.5 | 2.2 | 3.3 | 4.6 | 2 | 2 | 1 | 2+0=2 | 2.2+0.0=2.2 | F | | |
| | | ↓ | | | | | | | | | |
| 1.5 | 2.2 | 3.3 | 4.6 | 3 | 3 | 3 | 3+1=4 | 3.3+1.5=4.7 | T | 4.7 | |
| | | | ↓ | | | | | | | | |
| 1.5 | 2.2 | 3.3 | 4.6 | 1 | 4 | 6 | 4+2=6 | 4.6+2.2=6.8 | T | 4.7 | 6.8 |

Figure 6.7.: Top: A kernel which sums an array of floating point numbers (of length $n$). The counter produces an arithmetic sequence of numbers between 1 and $n$ incrementing by one. It assumed that floating point numbers can be summed in two cycles. In reality an extension of this idea is used because it takes 11 cycles to sum a floating point number. The final 11 numbers (2 in this example) must be summed on the CPU. Bottom: An example summing the sequence 1.5,2.2,3.3,4.6.

diagonal death (from this timestep). This means we cannot calculate the population from the previous timestep until this parallel process has completed.

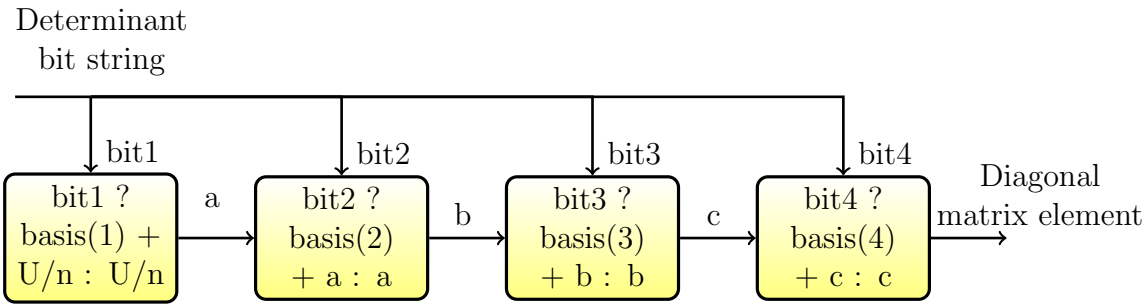



Figure 6.8.: The pipeline which computes the diagonal matrix elements for the two site Hubbard model.

To calculate $P_e$ when spawning we first have to select a determinant $j$ to spawn onto, using the excitation generator algorithm, which we discussed in general for FCIQMC in Sec. 1.3.2. For the Hubbard model in reciprocal space we first select three of the four spin orbitals involved in the excitation before using symmetry to determine the final orbital. There is a possibility that the final spin orbital will already be occupied: if this is the case then we reject the excitation. The excitation generator routines are shown in Fig. 6.9. We compute the off-diagonal matrix element using the method discussed in Sec. 6.2.2.

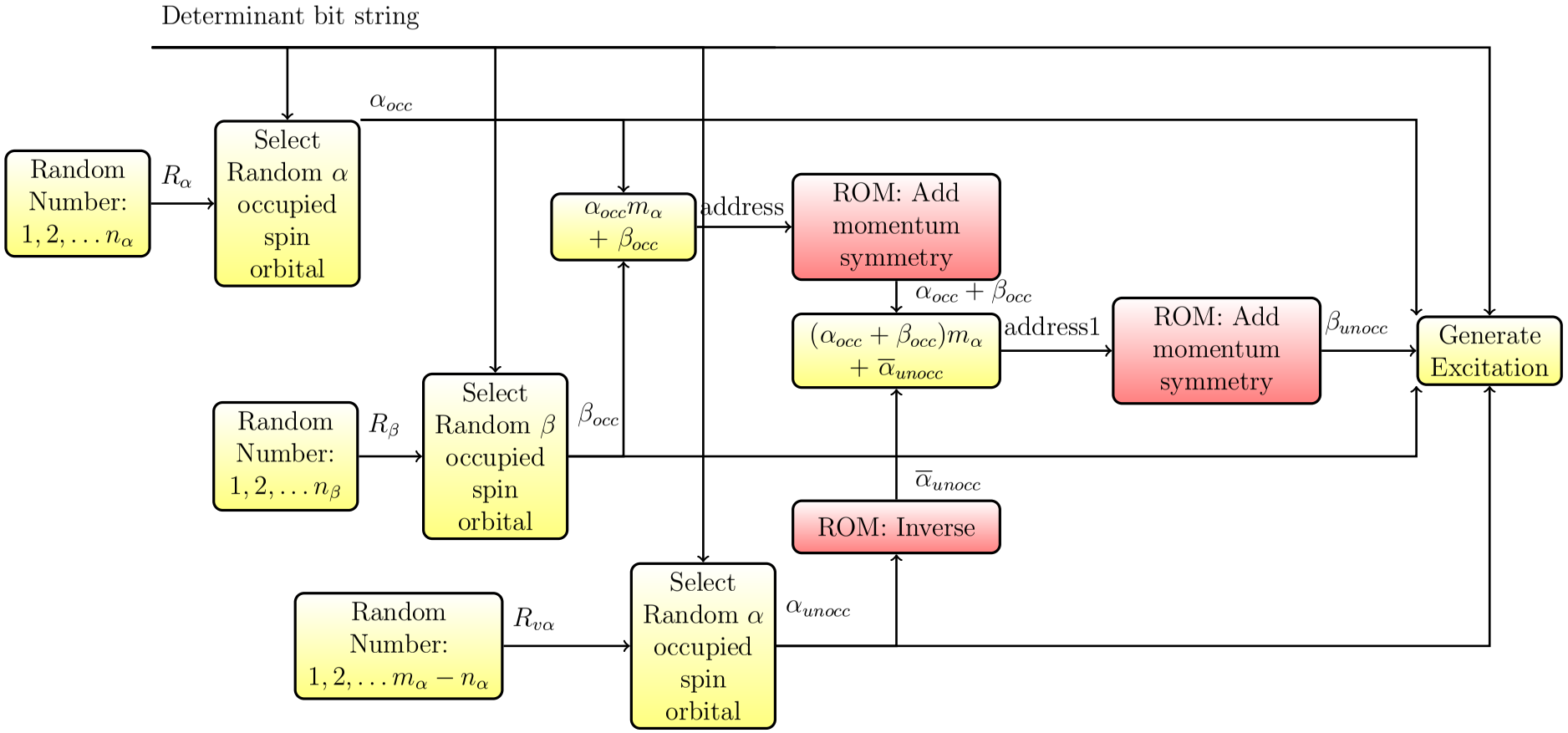



Figure 6.9.: The FPGA excitation generator routine for the Hubbard Model. We first select one occupied (in the bra) alpha, one occupied beta and one unoccupied spin orbital randomly. This is done using three random integers $R_\alpha, R_\beta, R_{v\alpha}$ in the ranges $1, \ldots, n_\alpha$, $1, \ldots, n_\beta$ and $1, \ldots, n_i v\alpha$. Using $R_\alpha$ we loop over the occupied alpha spin orbitals in bit string order to find out which orbital is the $R_\alpha$th occupied alpha spin orbital. We do a similar thing for the unoccupied alpha and occupied beta spin orbital. The excitation must conserve momentum symmetry meaning the total momentum bra and the ket must be the same. Using the 3 selected spin orbitals we can work out the final virtual beta spin orbital which conserves this symmetry. We use mapped ROMs to store the result of symmetry operations, we use a matrix to store the spin orbital which result from summing any two spin orbitals (mapped back into the first Brillouin zone). We use a ROM to store the vector of the spin orbitals which have the inverse momentum of every spin orbital. This symmetry-determined unoccupied beta spin orbital may be already occupied. If this is the case then we reject the excitation (a rejected excitation is represented by a bit string of zeros). The generation probability is $P_{gen} = \frac{2}{n_\alpha n_\beta n_v}$.

## 6.3. Efficiency

In chapter 4 we investigated the efficiency of the FCIQMC algorithm. We separately investigated the computer time taken to perform the calculation and also the statistical error. We now repeat some of this analysis for our DFE implementation to compare our implementation on a DFE to a CPU. As we discussed in Sec. 6.2 we modified the FCIQMC algorithm so we could implement it on a DFE. In Sec. 6.2 we compare a CPU and a DFE implementation of mFCIQMC as this enables us to directly measure the time of both implementations without directly considering the stochastic error bar. Afterwards we compare the modified algorithm with the original FCIQMC algorithm which enables us to conclude whether performing FCIQMC on a DFE is worthwhile compared to a CPU.

### 6.3.1. DFE time to process

On a DFE the computational cost of a single step can be calculated from the clock speed, memory and PCI express bandwidth. There are no complicated architectural consequences such as out of order execution or missing the cache which we reviewed in Chapter 5. The limiting factor (in terms of the number of clock cycles to perform a single timestep) is merging the spawned list (of length $L_s$ which has been sorted on the CPU) with the list of surviving parent walkers (of length $L_m$) and thus to perform a single timestep it costs $L_m + L_s$ clock cycles. We expect $L_m$ to be constant as a function of $\delta\tau$ but $L_s$ to increase with $\delta\tau$ for a given system as $\delta\tau$ scales with the spawning probability. We expect both $L_m$ and $L_s$ to increase as a function of $\langle N_p \rangle$. Thus as for the CPU implementation which we discussed in Sec. 4.1, we expect increasing $\langle N_p \rangle$ to be more expensive (in terms of clock cycles per timestep) than increasing $\delta\tau$ by the same proportion. No matter how many spin orbitals we have, this stage will be performed in a long parallel pipeline until we eventually reach the limit of physical resources available on the DFE. Ideally we would parallelise our implementation enough to use all the resources for any number of spin orbitals, possibly by doing multiple spawning and diagonal death on multiple determinants in parallel to use all the resources on the DFE for a given system size. To do this we would have to extend our annihilation algorithm to operate in parallel, which we believe to be non-trivial.

Once the pipeline becomes full the entire pipeline is performed in parallel once per clock cycle so in theory we can process data at 150 MHz (this is the speed we could build our design at). We could instead be limited by DRAM or PCI

express bandwidth: DRAM has a bandwidth of 38 GB per second and the PCI express has a bidirectional bandwidth of 2 GB per second on the MAX3 vectis card used throughout this thesis. In this thesis for each occupied determinant we use 64 bits: 32 bits for the determinant bit string and 32 bits to store the population of psips.[6] This limits us at present to the 16 site Hubbard model.[7] At 150 MHz this corresponds to 1 GB per second so we should not be limited by either PCI express or DRAM bandwidth. Even if we were to move beyond the 16 site Hubbard model and use 64 bits to store the determinant bit string, we are unlikely to become bound by the PCI express bus for a while because not all determinants spawn a child psip on every cycle.

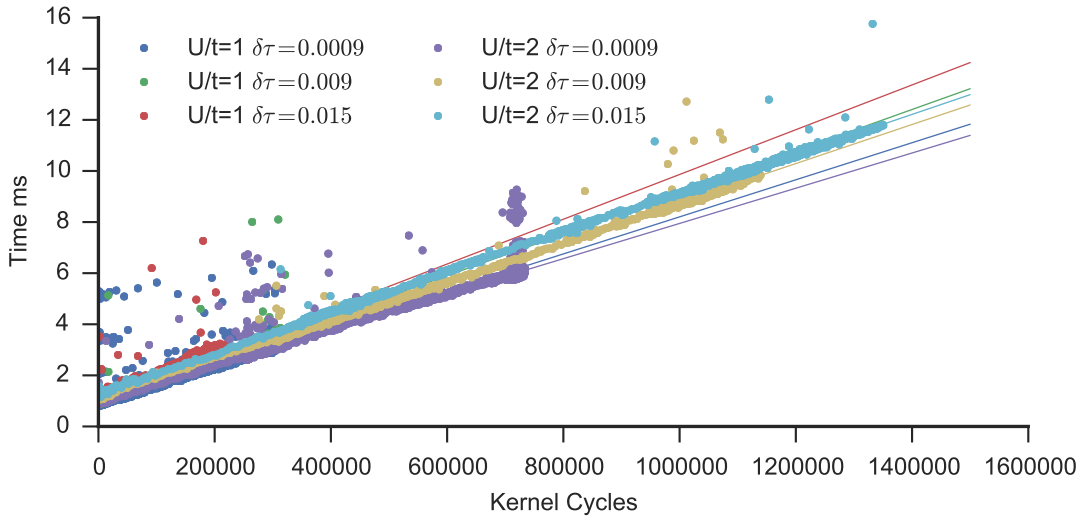
Fig. 6.12 shows how the proportion of used DFE resources scales as the number of sites in the Hubbard model used by different parts of the FCIQMC algorithm. This shows that there is plenty of room on the chip to either study a larger Hubbard model (as we use at most close to 50% of the available resources) or parallelise the algorithm in some way.

In Fig. 6.10 we find that the effective clock rate (as a function of the number of kernel cycles) varies between 145 and 114 MHz. This may be because during annihilation we do not read each input every cycle (on most cycles we read only one of the two inputs, see Appendix B) and a buffer becomes empty causing the DFE to stall, and wasting some of the clock cycles. We see a lower effective clock speed for large $\delta\tau$. This might be because we are doing lots of annihilation, whereas for very small $\delta\tau$ we are probably not spawning many walkers. We expect more of a random data access pattern if $L_s$ is close to $L_m$ because the probability of reading the input of the spawned list of psips becomes greater and the data access pattern becomes more random (see Appendix B for a detailed discussion of this algorithm). If $L_s$ is much smaller than $L_m$ then on most cycles we will read only the input containing the main list of psips.

The intercept of 6.10 tells us the other costs associated with the DFE, such as the cost of transferring the energy estimators, the initial latency of transferring the sorted spawned psips across the PCI express bus etc. The intercept also gives us an estimate of when a system is large enough to be worth computing on a DFE: if this

[6] The sorting routine which runs on the CPU is implemented to use a multiple of 16 bit integers when sorting. Otherwise we could use an integer with the same size as the number of spin orbitals to store the determinant bit string. This would cut down resource usage but might significantly slow down the sorting as CPUs are optimised to perform mathematical operations on 32/64 bit integers.

[7] To go beyond the 16 site Hubbard model the interface to the HANDE QMC code needs to be rewritten.




| System | Effective Clock Speed (MHz) | Intercept ($\mu$s) |
|---|---|---|
| U/t=1 $\delta\tau = 0.0009$ | 137.9(1) | 950.1(8) |
| U/t=1 $\delta\tau = 0.009$ | 123.0(2) | 1024(2) |
| U/t=1 $\delta\tau = 0.015$ | 114.1(3) | 1100(3) |
| U/t=2 $\delta\tau = 0.0009$ | 145.07(4) | 1053(1) |
| U/t=2 $\delta\tau = 0.009$ | 130.9(1) | 1121(5) |
| U/t=2 $\delta\tau = 0.015$ | 129.1(1) | 1376(5) |

Figure 6.10.: Top: The time to perform spawning and diagonal death as a function of the number of cycles on a MAX3 Vectis card. The number of cycles scheduled grows proportionally to $l_m+l_s$ (some extra cycles are required to remove determinants where all the psips have annihilated). To get different cycle lengths we grow the population until $\langle N_p \rangle \approx 1800000$ for different values of U/t and $\delta\tau$ (for each system we use 10 electrons). Bottom: From a fit of the above data the effective clock speed (the reciprocal of the gradient) and the intercept.

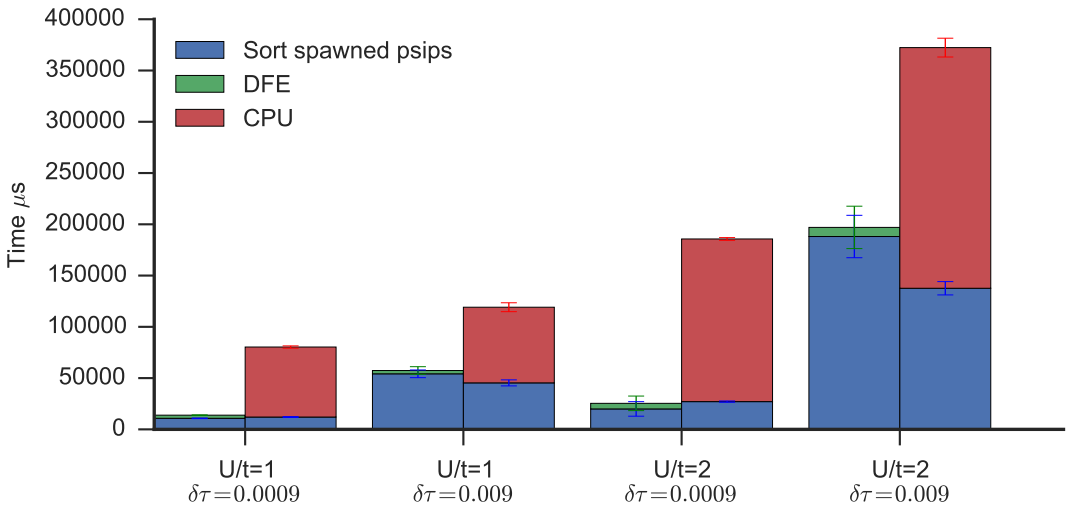



Figure 6.11.: The average time spent performing a timestep for an equilibrated mF-CIQMC calculations on a CPU (right bars) and a DFE (left bars). We see a speed up of between 1.89(2) times (for U/t=2 $\delta\tau = 0.009$) and 7(2) times (for U/t=2 $\delta\tau = 0.0009$).

fixed cost is more expensive than performing the computation on a CPU, using a DFE will be more expensive.

After every timestep we need to sort the list of spawned psips before we can send it back to the DFE. In theory this process could be done in parallel with DFE by sorting the list of spawned psips which come back first, however we have yet to implement this feature. In Fig. 6.11 we quantify how long these two steps take for both a CPU and a DFE.

Fig. 6.11 shows the average time taken to perform one timestep of the FCIQMC algorithm for different systems and timesteps with a fixed $\langle N_p \rangle$. We see a speed up for all values of $\delta\tau$, however we find that for a large $\delta\tau$ a significant proportion of the computer time is spent sorting the spawned walkers. For modified FCIQMC on a DFE we find that for all values of $\delta\tau$, most of the computer time is spent sorting the spawned psips. We could investigate the effect of using a parallel sorting algorithm. Sorting severely limits our speed up. In the traditional FCIQMC algorithm the cost of sorting the newly spawned walkers is insignificant compared to spawning and diagonal death. This suggests that if we do traditional FCIQMC on a DFE then we could obtain a significant speed up.

Fig. 6.12 shows we can easily fit the pipeline we have designed for the 16 site Hubbard model on a MAX3 Vectis card. It looks like the 32 site Hubbard model will

easily fit on the card, although we expect that switching to a 64 bit representation for the determinant bit string may increase the cost in terms of resources for control of access to the DRAM and access to the PCIE bus. There is also plenty of room for potential parallelism.

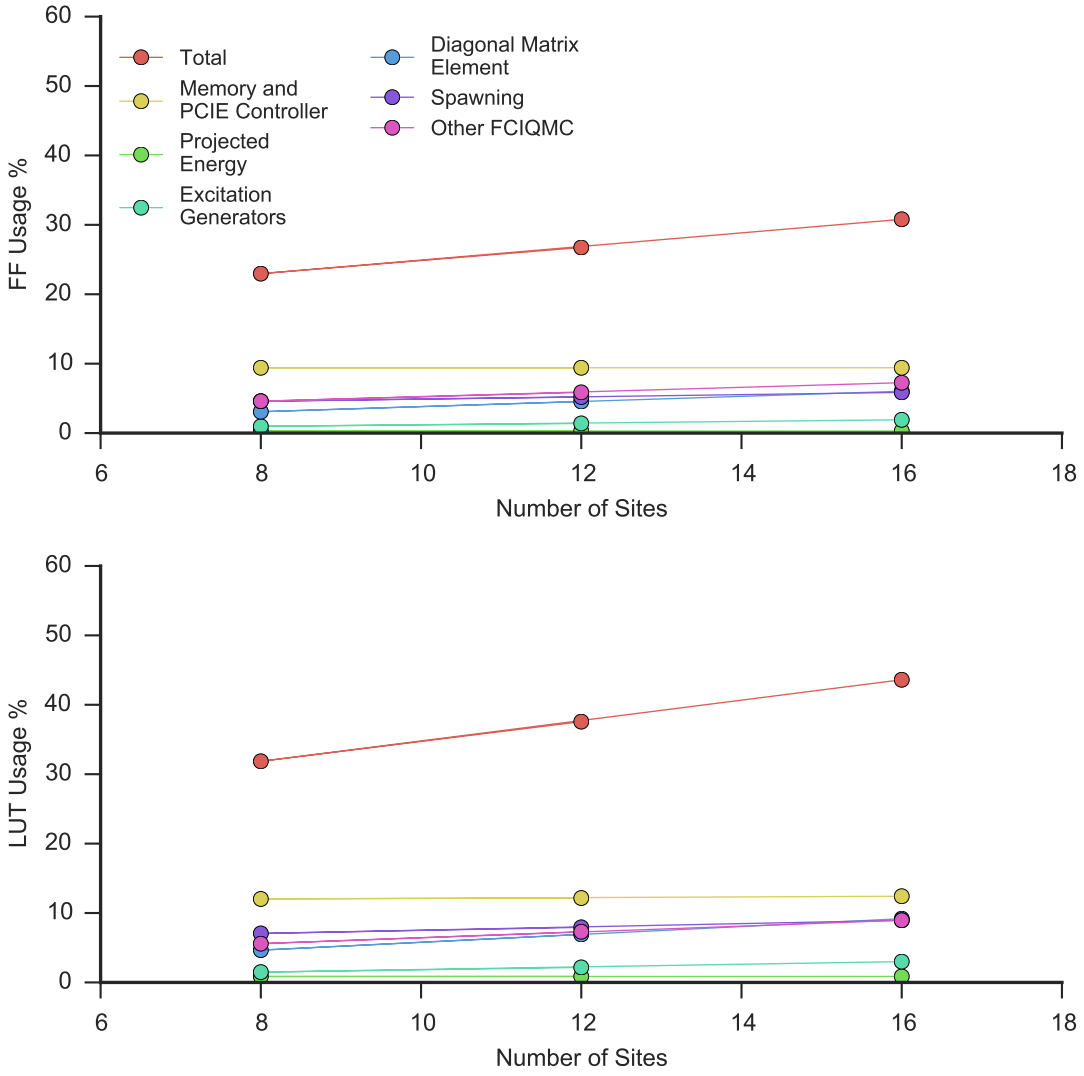



Figure 6.12.: Flip Flop FF and Look Up Table LUT (the two most important type of DFE resource) usage as a function of the number of sites. For our implementation of mFCIQMC on a MAX3 Vectis card.

### 6.3.2. $a$ as a function of $\delta\tau$ for the Hubbard Model

In the last section we investigated how long it takes for the DFE implementation to perform a single timestep for the mFCIQMC algorithm which we compared with a CPU version of mFCIQMC. We now compare the stochastic error for the mFCIQMC algorithm with the traditional FCIQMC algorithm. In chapter 4 we defined a measure of the inefficiency of an algorithm $a$: "$a$ is the hypothetical error bar which would result from running a FCIQMC for a single timestep, with a single psip, with $\delta\tau = 1$ and thus a small $a$ results in an efficient calculation." We investigated how $a$ behaved as a function of $\delta\tau$ for some molecular systems in Sec. 4.2.4 and in Appendix A. We found that $a$ is a constant as a function of $\delta\tau$ for small $\delta\tau$ and for large $\delta\tau$ $a$ increases as a function of $\delta\tau$. For some systems (Ne cc-pVDZ in Sec. 4.2.4) we found that the population dynamics become unstable before any increase in $a$ is seen.

In Fig. 6.13 we have investigated how $a$ behaves as a function of $\delta\tau$ for the traditional FCIQMC and the mFCIQMC algorithms. The removal of the inner loop seems to completely change the behavior of $a$ as a function of $\delta\tau$: $a$ seems to increase as a power law as a function of $\delta\tau$ before becoming constant as a function of $\delta\tau$ for very small $\delta\tau$ for the mFCIQMC algorithm. For the traditional FCIQMC algorithm we find that similar to Ne cc-pVDZ, the population dynamics become unstable before any increase in $a$ is seen.

Even at the extremely small value of $\delta\tau$ when $a$ becomes flat for the mFCIQMC algorithm, $a$ seems to be at least an order of magnitude larger for mFCIQMC than FCIQMC. This, compounded with the fact that we have to get this flat behaviour we have to use a $\delta\tau$ one or two orders of magnitude smaller than $a$, means that the mFCIQMC algorithm is much less efficient than the FCIQMC algorithm. In the last section we showed that the DFE speeds the mFCIQMC algorithm up by about 7 times. We expect the CPU implementation of mFCIQMC to be faster than the traditional FCIQMC on a CPU but not by orders of magnitude. In the next section we discuss the possible ways to improve our DFE implementation to obtain a similar $a$ for a DFE.

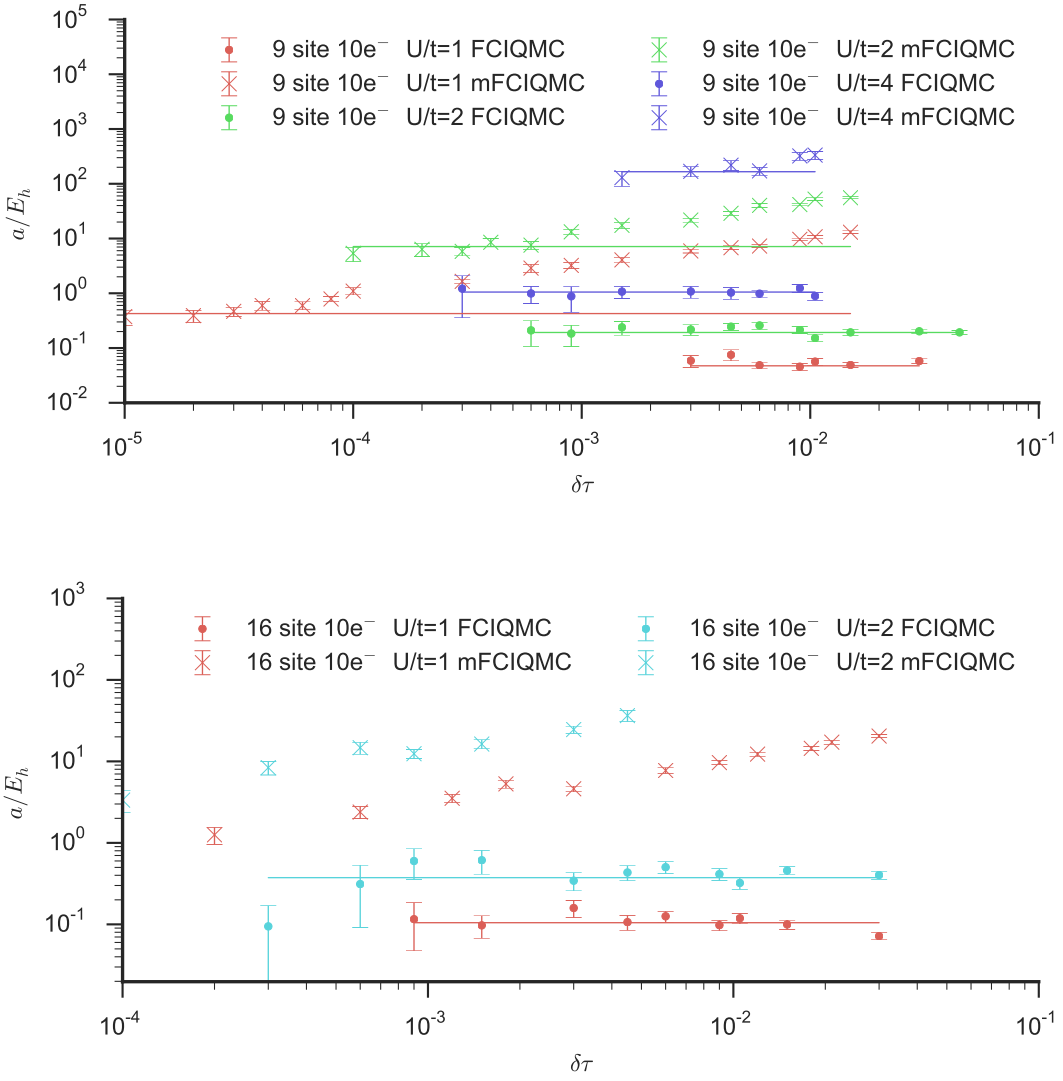



Figure 6.13.: Top: $a$ as a function of $\delta\tau$ for the 9 site Hubbard model comparing the mFCIQMC and FCIQMC Markov chains. Bottom: $a$ as a function of $\delta\tau$ for the 16 site Hubbard model comparing the mFCIQMC and FCIQMC Markov chains.

### 6.3.3. Improving Data-Flow FCIQMC

One potential way to improve the mFCIQMC algorithm is for all the psips on each determinant to spawn onto more than one but still a bounded number of determinants. This could be done on a DFE by spawning onto multiple determinants per clock cycle (this would use more resources than our present implementation). This lets us keep the annihilation algorithm the same. This will increase the amount of data from the spawned walker stream which goes to the CPU and would also mean that each clock cycle could generate up to $n$ times the amount of data, although this would not be a deterministic amount of data (often we would want to stream less than $n$ items of data). The outputs which we described in Fig. 6.3 only allow us to stream a fixed amount of data across the PCIE bus (or no data) per clock cycle so we would have to implement some kind of queue. We have implemented this approach on a CPU in a modified version of HANDE. In Fig. 6.14 we see that although this approach does decrease $a$ we still see a power law like behaviour and more than 20 spawning attempts per determinant are required to get close to the same value of $a$ at a small $\delta\tau$ as the traditional FCIQMC algorithm. This suggests that this approach will not be particularly effective because we will probably run out of DFE resources before we get close to CPU efficiency.

The mFCIQMC has the same stochastic process as FCIQMC in the limit that every occupied determinant is occupied by a single psip. We could use our DFE implementation to perform FCIQMC on the space occupied at most one psips and do FCIQMC on the rest of the space on the CPU. As most of the occupied Hilbert space probably has a small population so we would still potentially gain a significant speed up from this optimisation. We could even combine this idea with spawning multiple times per determinant and moving the entire space above this threshold to the DFE. This threshold could potentially be chosen to be optimal by calculating the speed up ratio for each by looking at the distribution of occupied determinants. We could then store the population with a single bit: 0 for a positive psip and 1 for a negative psip, reducing resource usage.[8]

We could also implement the traditional FCIQMC algorithm by modifying our design to have two modes. The first mode would only perform annihilation writing the merged list to the DRAM. This would be scheduled to run for the same number of cycles as before set proportionally to $L_m + L_s$. The second mode would run for $\langle N_p \rangle$ cycles performing spawning and diagonal death reading the merged list from

[8]The 32 bits used in are implementation to store the population is excessive.

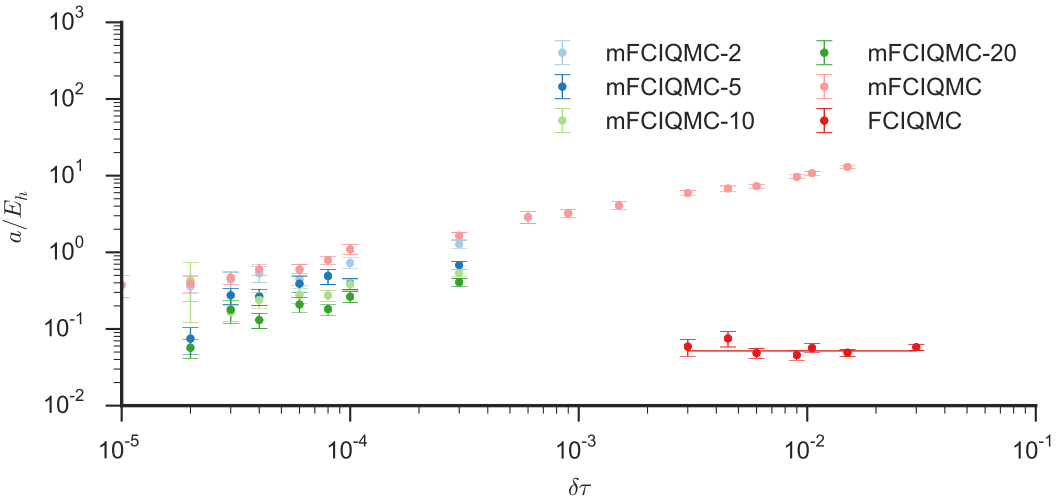



Figure 6.14.: The effect of spawn onto multiple determinants on $a$ as a function of $\delta\tau$ for the 9 site Hubbard model. mFCIQMC-i denotes a version of FCIQMC modified so all the psips on a determinant spawn onto i determinants (with an appropriate weight). mFCIQMC denotes version of FCIQMC modified so all the psips spawn on a determinant spawn onto 1 determinates (we discussed this in the last section). FCIQMC denotes the traditional FCIQMC algorithm where each psip on each determinant spawns onto exactly one connected determinant. Calculations performed on a CPU using a modified version of HANDE

the DRAM. We would have to read the annihilated walkers in a data access pattern which stops reading the input for the next $N-1$ (where $N$ is the number of psips read on this cycle) cycles if more than one psip is read from the input. Potentially this could be parallelised using 'dumb' parallelism mode (which we discussed in Sec. 4.2.3) to perform two calculations at once. Merging the lists for one FCIQMC calculation in parallel whilst performing spawning and diagonal death from another so that all of the resources were active on the DFE at once this would undoubtedly be complicated to implement.

## 6.4. Data Flow for Molecular FCIQMC

In this section we discuss why implementing molecular FCIQMC on a DFE is difficult and possible solutions to these problems.

One way in which molecular FCIQMC differs from the Hubbard model is that both single and double excitations relative to the parent are connected. This means

that a separate excitation generator algorithm is required for both the single and the double excitations. Normally each psip selects either a single or a double excitation with some probability. The easiest way to get around this problem on a DFE is for every psip to select both a single and a double excitation and to attempt to spawn onto both. It is not known if this would make the FCIQMC less efficient as for the DFE.

Another issue is that we need to sum over the one and two electron integrals when computing the off-diagonal matrix element with a randomly selected determinant in the ket (see Tables 1.1 and 1.2). On a DFE we have access to about 5 MB of random access memory that can be accessed with low latency at TB/s bandwidth. We could use this to store the integrals, but this will limit the maximum system size possible. We can only address one item of this random access memory per cycle and scheduling enough cycles to sum the matrix elements may be a non-trivial task. Random access to the large memory is extremely expensive in terms of the latency of the memory address.

## 6.5. Conclusion

We have described an implementation of FCIQMC on novel data flow architecture with the aim to greatly speed up the FCIQMC compared to a CPU. For ease of implementation we made a modification to the spawning stage of the FCIQMC algorithm so that all psips on the same determinant spawn onto only a single connected determinant with an appropriate weight. As we showed in Sec. 6.3.2 this greatly reduced the computational efficiency of the algorithm and as such, at present the DFE implementation is much slower than the CPU implementation. This is unfortunate and so in Sec. 6.3.3 we discussed possible techniques to improve DFE FCIQMC to perform the same algorithm as the unmodified CPU version at the expense that annihilation is not performed in parallel with spawning and diagonal death.

We did however achieve a significant speed up compared to the same algorithm being performed on a CPU. This suggests that if we make these optimisations then a DFE could be effective at performing FCIQMC. Data flow for molecular FCIQMC might be difficult although there may be ways to compress the two electron integrals such that they fit on the DFE fast memory which we can efficiently randomly access. This could be an interesting subject for further exploration.

The initiator approximation (as defined in Sec. 1.3.1) could easily be incorporated into the data flow implementation described in this chapter, by storing an extra bit

for each item on the spawned list to encode whether the parent determinant (that spawned the child psip(s)) was an initiator. The annihilation algorithm (as described in Appendix B) could then be modified to zero the population, if the parent wasn't an initiator and hasn't spawned onto an already occupied determinant.

Our implementation presently uses a significant amount of DFE resources. It is possible to cut down the resources used presently by using fixed point instead of floating point operations. This would be useful if we could use these additional resources for parallelism.

## 6.6. Addendum

After submission of this thesis the work in this chapter was continued by the author as part of a knowledge transfer secondment. We have successfully managed to remove the modification to the spawning stage of the FCIQMC algorithm and thus implemented the traditional FCIQMC algorithm on a DFE for the Hubbard model. This was achieved by modifying the input control such that the input would stop reading data for a number of cycles. This allows enough clock cycles for each psip to spawn a child (number of psips minus one extra cycle(s) are required for each determinant). This process has speeded up FCIQMC by a factor of around 76 times compared to a single core for the 20 site Hubbard model.

# 7. Conclusion

In this thesis we have investigated the recently developed Full Configuration interaction Quantum Monte Carlo (FCIQMC) Method. An investigation of the statistical and systematic errors in the simulation yielded methods to both reduce the systematic error and accelerate convergence of the statistical errors as a function of computer time. We then used these ideas to investigate an implementation of FCIQMC on novel Data Flow architecture. Unfortunately modifications to the algorithm for a simplistic implementation of FCIQMC on Data Flow architecture dramatically increased the statistical error, making this approach ineffective. The increased statistical error more than exceeded the effect of the speed up gained compared to a CPU.

The search for the theoretical basis of the statistical and systematic errors of FCIQMC led us to show that FCIQMC is an example of Markov Chain Monte Carlo. From the stochastic matrix it is possible to compute both the statistical and systematic errors and everything else useful about a simulation. Using the ideas behind Markov Chain theory we constructed the stochastic matrix for a two determinant system. Unfortunately as the space scales poorly with the size of the Hilbert space we could not extend these ideas to investigate a larger system, so instead we concerned ourselves with empirical investigations.

Using these ideas we then investigated population control bias in FCIQMC, a cause of systematic error. Our investigations of population control bias for $H_2$ in a STO-3G basis using the stochastic matrix (which gives an estimate free from statistical errors) and empirical investigations of the more realistic Ne in a cc-pVDZ basis set lead us to conclude that population control bias was of the order of 10 $\mu$ Hartree for these systems. Thus care should be taken to control this bias if convergence of the statistical error to this order of magnitude is required. This bias can be controlled by varying the shift slowly and increasing the number of psips, or alternatively by using the reweighting technique suggested for DMC by Umrigar et al. [46].

How quickly the statistical error decays as a function of computer time is a measure

of the efficiency of the FCIQMC algorithm. An investigation of this efficiency as a function of the timestep yielded a lower bound on the most efficient timestep. For FCIQMC calculations a large population of psips is necessary to maximise the efficiency whereas for iFCIQMC the population had little impact (although it does have an impact on the initiator error). Thus, even if it is practical to store the population at the plateau, the initiators approximation can provide a significant reduction in statistical errors and hence increase the efficiency of a calculation. When performing calculations we recommend this timestep and either a large population or the use of the initiator approximation. An investigation of stretching the bond length in the hydrogen fluoride molecule showed that the statistical error increases as the correlation energy increases but not in a linear fashion. However, the stochastic error scales linearly as a function of the number of basis functions for the neon atom. The stochastic error scales faster than linearly when increasing the number of atoms in a helium chain (at fixed bond length). In this system orbital localisation can be an effective method of reducing both plateau height and possibly the stochastic error bar in a system of helium chains. This is surprising because orbital localisation breaks symmetry, greatly increasing the size of the Hilbert space. As the generation of the matrix elements from these localised orbitals result in zeros, the development of excitation generators for localised molecular orbitals may well be a worthwhile endeavour.

The FCIQMC algorithm can be modified so that it can be performed in a single long pipeline (with the exception of the sorting of the spawned walkers). Unfortunately a design decision to remove a loop (the length of which changes throughout the simulation) made the calculation orders of magnitude less efficient. However FCIQMC on the modified FCIQMC algorithm on a DFE showed a speed up of 7× compared to a CPU implementation of the same algorithm, suggesting if we can redesign the pipeline to add back in this loop then FCIQMC may be effective on a DFE.

# Bibliography


[1] J. A. Pople, M. Head-Gordon, D. J. Fox, K. Raghavachari, and L. A. Curtiss, J. Chem. Phys. **90**, 5622 (1989).

[2] K. Raghavachari, G. W. Trucks, J. A. Pople, and M. Head-Gordon, Chem. Phys. Lett. **157**, 479 (1989).

[3] T. Lee and G. Scuseria, Achieving chemical accuracy with coupled-cluster theory, in *Quantum Mechanical Electronic Structure Calculations with Chemical Accuracy*, edited by S. Langhoff, , Understanding Chemical Reactivity Vol. 13, pp. 47–108, Springer Netherlands, 1995.

[4] J. Řezáč and P. Hobza, J. Chem. Theory Comput. **9**, 2151 (2013).

[5] J. Lee, *Computational Materials Science: An Introduction* (CRC Press, 2011).

[6] B. G. Johnson, Development, implementation and applications of efficient methodologies for density functional calculations, in *Modern Density Functional Theory A Tool for Chemistry*, edited by J. Seminario and P. Politzer, , Theoretical and Computational Chemistry Vol. 2, pp. 169 – 219, Elsevier, 1995.

[7] P. A. M. Dirac, Proc. R. Soc. A **123**, pp. 714 (1929).

[8] A. Rae, *Quantum mechanics* (Taylor & Francis Group, 2008).

[9] P.-F. Loos and P. M. W. Gill, Phys. Rev. Lett. **108**, 083002 (2012).

[10] J. Almlf, K. Faegri, and K. Korsell, J. Comput. Chem. **3**, 385 (1982).

[11] R. Polly, H.-J. Werner, F. R. Manby, and P. J. Knowles, Mol. Phys. **102**, 2311 (2004).

[12] C. C. J. Roothaan, Rev. Mod. Phys. **23**, 69 (1951).

[13] A. Szabo and N. Ostlund, *Modern Quantum Chemistry: Introduction to Advanced Electronic Structure Theory* (Dover Publications, 2012).

[14] R. Ditchfield, W. J. Hehre, and J. A. Pople, J. Chem. Phys. **54**, 724 (1971).

[15] T. H. Dunning, J. Chem. Phys. **90**, 1007 (1989).

[16] J. A. Pople and R. K. Nesbet, Selfconsistent orbitals for radicals, 1954.

[17] C. Lanczos, J. Research Nat. Bur. Standards **45**, 255 (1950).

[18] E. R. Davidson, J. Comput. Phys. **17**, 87 (1975).

[19] T. Helgaker, J. Olsen, and P. Jørgensen, *Molecular electronic-structure theory* (Wiley, 2013).

[20] J. B. Robinson and P. J. Knowles, J. Chem. Phys. **136**, (2012).

[21] M. Schütz, G. Hetzer, and H.-J. Werner, J. Chem. Phys. **111**, 5691 (1999).

[22] P. Hohenberg and W. Kohn, Phys. Rev. **136**, B864 (1964).

[23] W. Kohn and L. J. Sham, Phys. Rev. **140**, A1133 (1965).

[24] J. P. Perdew, K. Burke, and M. Ernzerhof, Phys. Rev. Lett. **77**, 3865 (1996).

[25] C. Lee, W. Yang, and R. G. Parr, Phys. Rev. B **37**, 785 (1988).

[26] A. D. Becke, J. Chem. Phys. **98**, 5648 (1993).

[27] P. J. Stephens, F. J. Devlin, C. F. Chabalowski, and M. J. Frisch, J. Phys. Chem. **98**, 11623 (1994).

[28] J. Foresman and A. Frisch, *Exploring chemistry with electronic structure methods* (Gaussian, Inc., 1996).

[29] Y. Zhao and D. Truhlar, Theor. Chem. Acc. **120**, 215 (2008).

[30] T. Helgaker, T. A. Ruden, P. Jørgensen, J. Olsen, and W. Klopper, J. Phys. Organic Chem. **17**, 913 (2004).

[31] A. Halkier, W. Klopper, T. Helgaker, and P. Jørgensen, J. Chem. Phys. **111**, 4424 (1999).

[32] A. Sokal, NATO ASI Series **361**, 131 (1997).

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

[34] C. J. Umrigar, J. Toulouse, C. Filippi, S. Sorella, and R. G. Hennig, Phys. Rev. Lett. **98**, 110201 (2007).

[35] N. D. Drummond and R. J. Needs, Phys. Rev. B **72**, 085124 (2005).

[36] N. D. Drummond, M. D. Towler, and R. J. Needs, Phys. Rev. B **70**, 235119 (2004).

[37] R. Jastrow, Phys. Rev. **98**, 1479 (1955).

[38] T. Kato, Comm. Pure Appl. Math. **10**, 151 (1957).

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

[40] W. A. L. Jr., L. Mitas, and B. Hammond, Chem. Phys. Lett. **478**, 1 (2009).

[41] G. Senatore and N. March, Rev. Mod. Phys. **66**, 445 (1994).

[42] D. Ceperley, The stochastic solution of the many-body schroedinger equation for fermions, in *Recent Progress in Many-Body Theories*, edited by J. Zabolitzky, M. de Llano, M. Fortes, and J. Clark, , Lecture Notes in Physics Vol. 142, pp. 262–269, Springer Berlin Heidelberg, 1981.

[43] D. M. Ceperley and B. J. Alder, J. Chem. Phys. **81**, 5833 (1984).

[44] D. Ceperley and B. Alder, Phys. Rev. B **36**, 2092 (1987).

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

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

[47] M. Suzuki, Comm. Math. Phys. **51**, 183 (1976).

[48] P. Lopez Rios, A. Ma, N. Drummond, M. Towler, and R. Needs, Phys. Rev. E **74**, 066701 (2006).

[49] G. H. Booth, A. J. W. Thom, and A. Alavi, J. Chem. Phys. **131**, 054106 (2009).

[50] J. J. Shepherd, G. E. Scuseria, and J. S. Spencer, Phys. Rev. B **90**, 155130 (2014).

[51] C. Daday, S. Smart, G. H. Booth, A. Alavi, and C. Filippi, J. Chem. Theory Comput. **8**, 4441 (2012).

[52] F. R. Petruzielo, A. A. Holmes, H. J. Changlani, M. P. Nightingale, and C. J. Umrigar, Phys. Rev. Lett. **109**, 230201 (2012).

[53] J. B. Anderson, J. Chem. Phys. **63**, 1499 (1975).

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

[55] U. Wolff, Comput. Phys. Commun. **156**, 143 (2004).

[56] R. M. Lee, G. J. Conduit, N. Nemec, P. Lopez Rios, and N. D. Drummond, Phys. Rev. E **83**, 066706 (2011).

[57] See `https://github.com/jsspencer/pyblock` for code.

[58] `http://www.hande.org.uk/`.

[59] D. M. Cleland, G. H. Booth, and A. Alavi, J. Chem. Phys. **134**, (2011).

[60] D. Cleland, G. H. Booth, and A. Alavi, J. Chem. Phys. **132**, 41103 (2010).

[61] J. J. Shepherd, G. Booth, A. Grüneis, and A. Alavi, Phys. Rev. B **85**, 081103 (2012).

[62] J. S. Spencer, N. S. Blunt, and W. M. C. Foulkes, J. Chem. Phys. **136**, 054110 (2012).

[63] J. Hubbard, Proc. R. Soc. A **276**, 238 (1963).

[64] Nat. Phys. **9**, 523 (2013).

[65] J. Hook and H. Hall, *Solid State Physics* Manchester Physics Series (Wiley, 2013).

[66] W. A. Vigor, J. S. Spencer, M. J. Bearpark, and A. J. W. Thom, J. Chem. Phys. **142**, 104101 (2015).

[67] J. Kemeny and J. Snell, *Finite Markov Chains: With a New Appendix "Generalization of a Fundamental Matrix"* Undergraduate Texts in Mathematics (Springer, 1976).

[68] O. Perron, Math. Ann. **64**, 248 (1907).

[69] G. Frobenius, Sitzungsber. Königl. Preuss. Akad. Wiss , 471476 (1908).

[70] G. Frobenius, Sitzungsber. Königl. Preuss. Akad. Wiss , 514518 (1909).

[71] G. Frobenius, Sitzungsber. Königl. Preuss. Akad. Wiss , 456477 (1912).

[72] K. Riley, P. Hobson, and S. Bence, *Mathematical Methods for Physics and Engineering: A Comprehensive Guide* (Cambridge University Press, 2006).

[73] We assume that the independent psip probabilities are less than or equal to one for simplicity; the generalisation is straightforward.

[74] E. Anderson *et al.*, *LAPACK Users' Guide*, Third ed. (Society for Industrial and Applied Mathematics, Philadelphia, 1999).

[75] R. B. Lehoucq, D. C. Sorensen, and C. Yang, Arpack users guide: Solution of large scale eigenvalue problems by implicitly restarted arnoldi methods., 1997.

[76] T. E. Oliphant, Comput. Sci. Eng. **9**, 10 (2007).

[77] N. Cerf and O. C. Martin, Phys. Rev. E **51**, 3679 (1995).

[78] D. E. Woon and T. H. Dunning, J. Chem. Phys. **101**, 8877 (1994).

[79] A. J. W. Thom, Phys. Rev. Lett. **105**, 263004 (2010).

[80] S. Sharma, T. Yanai, G. H. Booth, C. J. Umrigar, and G. K.-L. Chan, J. Chem. Phys. **140**, (2014).

[81] N. S. Blunt *et al.*, J. Chem. Phys. **142**, (2015).

[82] G. H. Booth, S. D. Smart, and A. Alavi, Mol. Phys. **112**, 1855 (2014).

[83] G. Booth, *A novel quantum Monte Carlo method for molecular systems*, PhD thesis, University of Cambridge, 2010.

[84] N. Nemec, Phys. Rev. B **81**, 035119 (2010).

[85] J. Pipek and P. G. Mezey, J. Chem. Phys. **90**, 4916 (1989).

[86] J. Hennessy and D. Patterson, *Computer Architecture: A Quantitative Approach: Fifth Multimedia Edition* The Morgan Kaufmann Series in Computer Architecture and Design (Elsevier Science, 2016).

[87] https://www.spec.org/.

[88] W. Gilreath and P. Laplante, *Computer Architecture: A Minimalist Perspective* The Springer International Series in Engineering and Computer Science (Springer US, 2003).

[89] M. Flynn, Computers, IEEE Transactions on **C-21**, 948 (1972).

[90] D. Patterson and J. Hennessy, *Computer Organization and Design: The Hardware/Software Interface* The Morgan Kaufmann Series in Computer Architecture and Design (Elsevier Science, 2013).

[91] http://www.openmp.org/mp-documents/OpenMP4.0.0.pdf.

[92] http://www.mpi-forum.org/docs/mpi-3.0/mpi30-report.pdf.

[93] http://www.gotw.ca/publications/concurrency-ddj.htm.

[94] G. M. Amdahl, Validity of the single processor approach to achieving large scale computing capabilities, in *Proceedings of the April 18-20, 1967, Spring Joint Computer Conference*, AFIPS '67 (Spring), pp. 483–485, New York, NY, USA, 1967, ACM.

[95] J. Nickolls, I. Buck, M. Garland, and K. Skadron, Queue **6**, 40 (2008).

[96] https://www.olcf.ornl.gov/titan/.

[97] http://www.top500.org/lists/2014/11/.

[98] I. S. Ufimtsev and T. J. Martinez, J. Chem. Theory Comput. **4**, 222 (2008).

[99] I. S. Ufimtsev and T. J. Martinez, J. Chem. Theory Comput. **5**, 1004 (2009).

[100] I. S. Ufimtsev and T. J. Martinez, J. Chem. Theory Comput. **5**, 2619 (2009).

[101] K. P. Esler, J. Kim, D. M. Ceperley, and L. Shulenburger, Comput. Sci. Eng. **14**, 40 (2012).

[102] M. Flynn, R. Dimond, O. Mencer, and O. Pell, Finding speedup in parallel processors, in *Parallel and Distributed Computing, 2008. ISPDC '08. International Symposium on*, pp. 3–7, 2008.

[103] J. Arram, K. Tsoi, W. Luk, and P. Jiang, Hardware acceleration of genetic sequence alignment, in *Reconfigurable Computing: Architectures, Tools and Applications*, edited by P. Brisk, J. de Figueiredo Coutinho, and P. Diniz, , Lecture Notes in Computer Science Vol. 7806, pp. 13–24, Springer Berlin Heidelberg, 2013.

[104] D. Knuth, *The Art of Computer Programming: Sorting and Searching* The Art of Computer Programming (Addison-Wesley, 1998).

# Appendix A.

# Plateau Heights and the Stochastic Error

In this appendix we investigate how the timestep effects the plateau height and the inefficiency $a$ which we defined in chapter 4. In chapter 4 we found in Fig. 4.5 that for Ne aug-cc-pVDZ that $a$ is a constant as a function of $\delta\tau$ until a certain timestep $\delta\tau = \delta\tau_0$. The purpose of this appendix is two fold: firstly we want to justify the fact that the behavior we observed in Fig. 4.5 is universal in FCIQMC and iFCIQMC for systems which have a significant plateau (we do this by testing many such systems) and secondly that the $\delta\tau$ used in figures 4.7, 4.8 and 4.9 is smaller than $\delta\tau_0$, and thus we are in the regime where $a$ is constant as a function of $\delta\tau$.

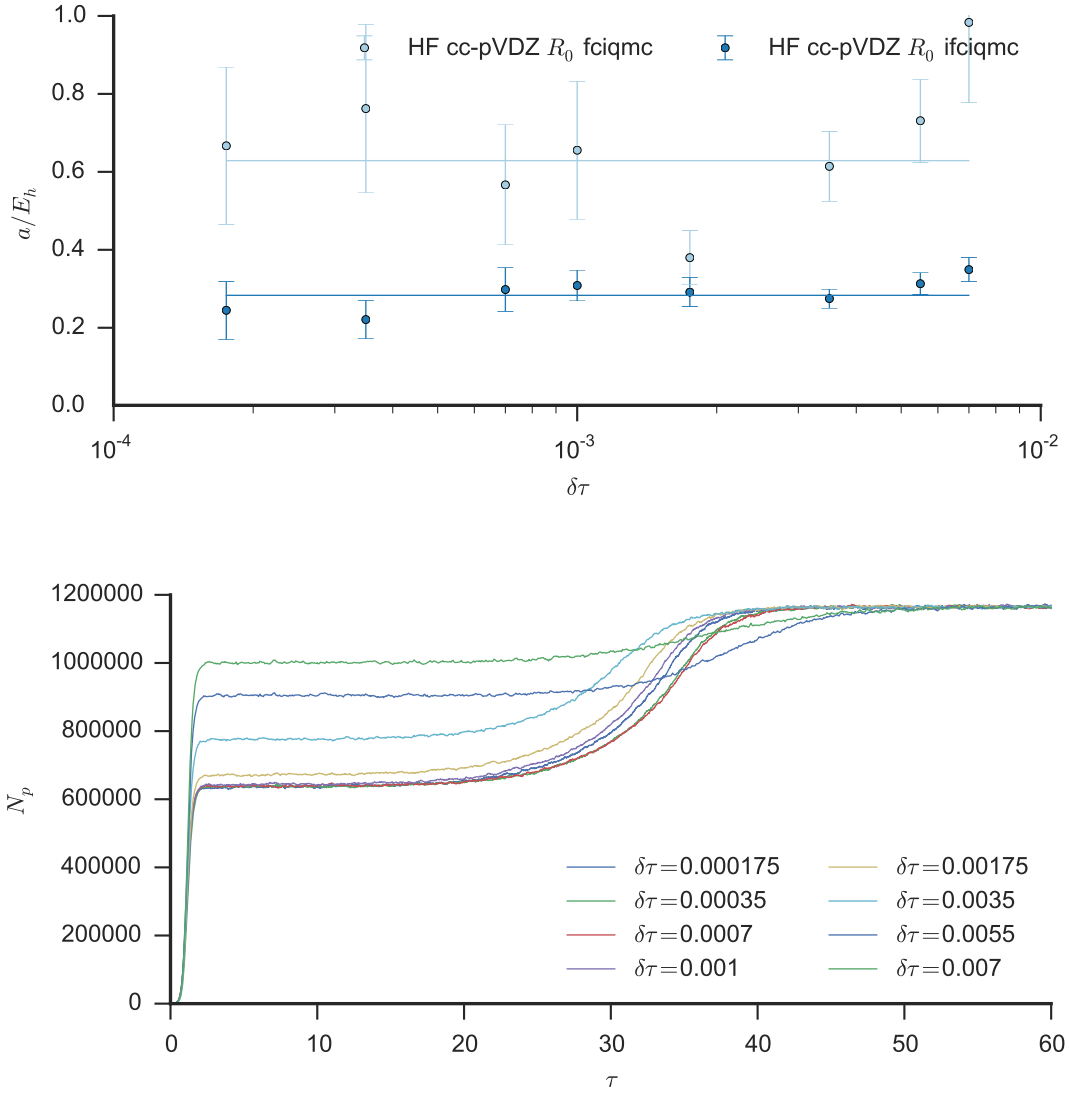



Figure A.1.: Top: The inefficiency $a$ remains constant until $\delta\tau \approx 0.001$. Bottom: The plateau height can be seen to increase after $\delta\tau \approx 0.001$.

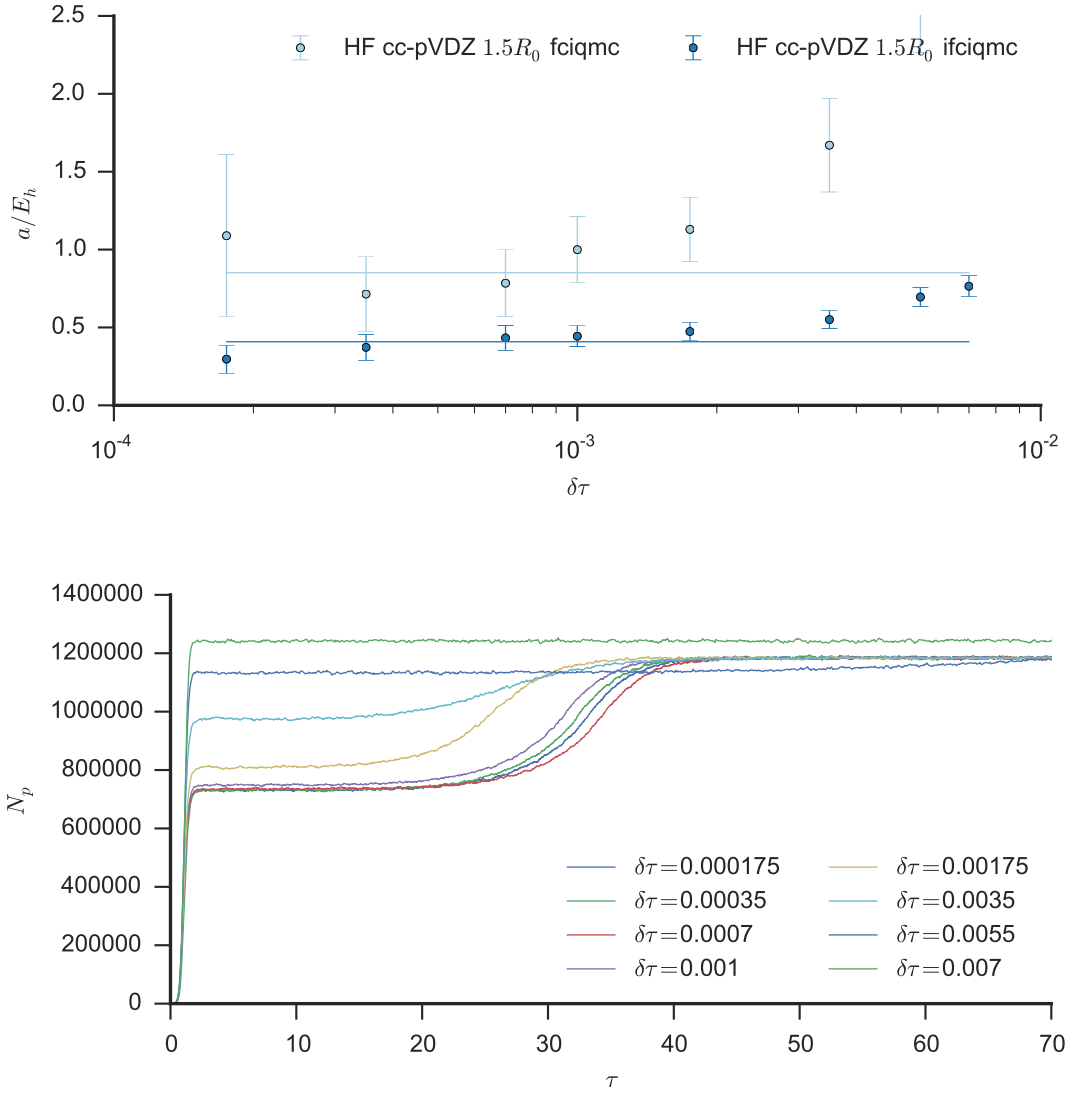



Figure A.2.: Top: The inefficiency $a$ remains constant until $\delta\tau \approx 0.001$. Bottom: The plateau height can be seen to increase after $\delta\tau \approx 0.001$.

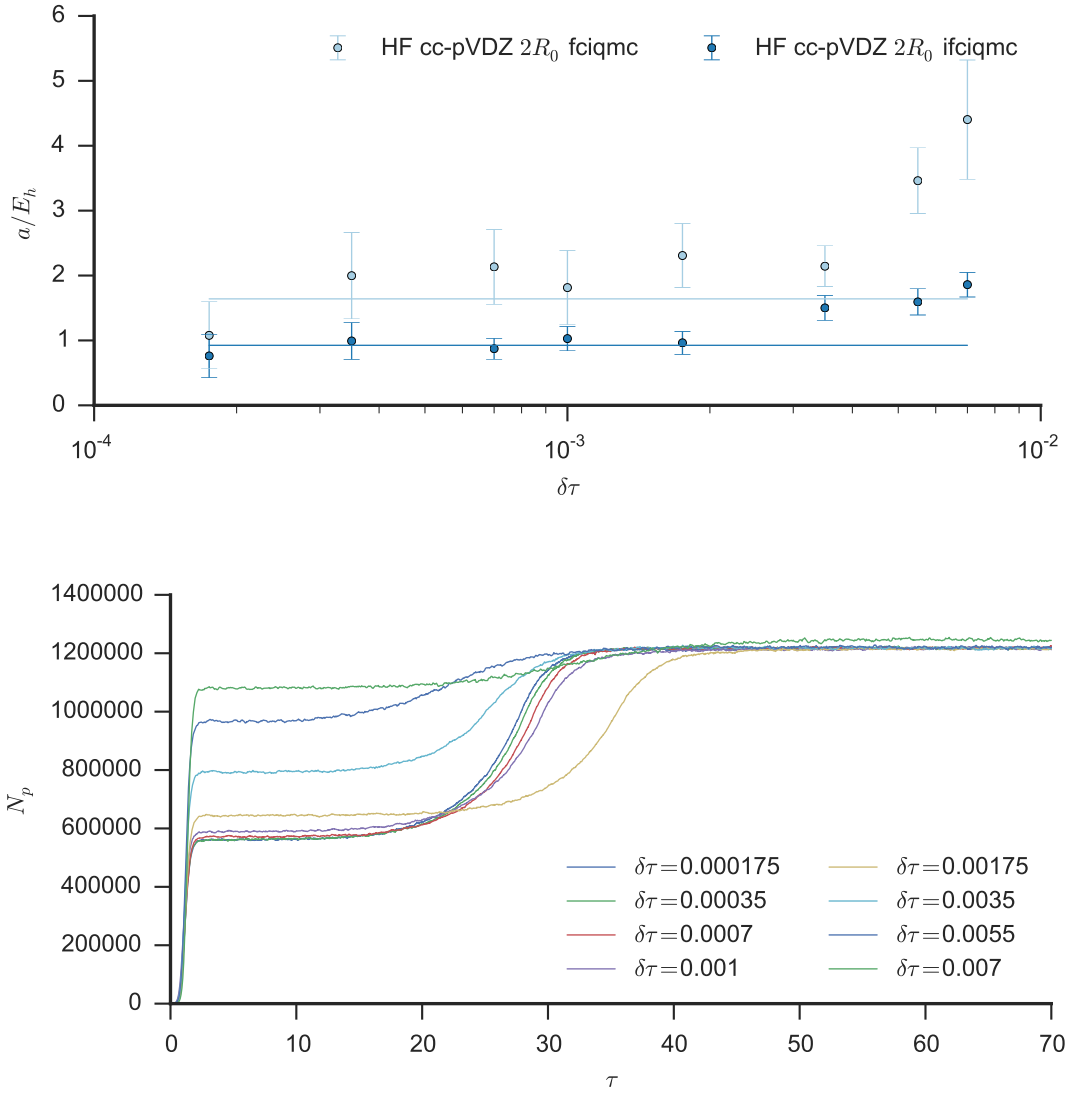



Figure A.3.: Top: The inefficiency $a$ remains constant until $\delta\tau \approx 0.01$. Bottom: The plateau height can be seen to increase after $\delta\tau \approx 0.01$.

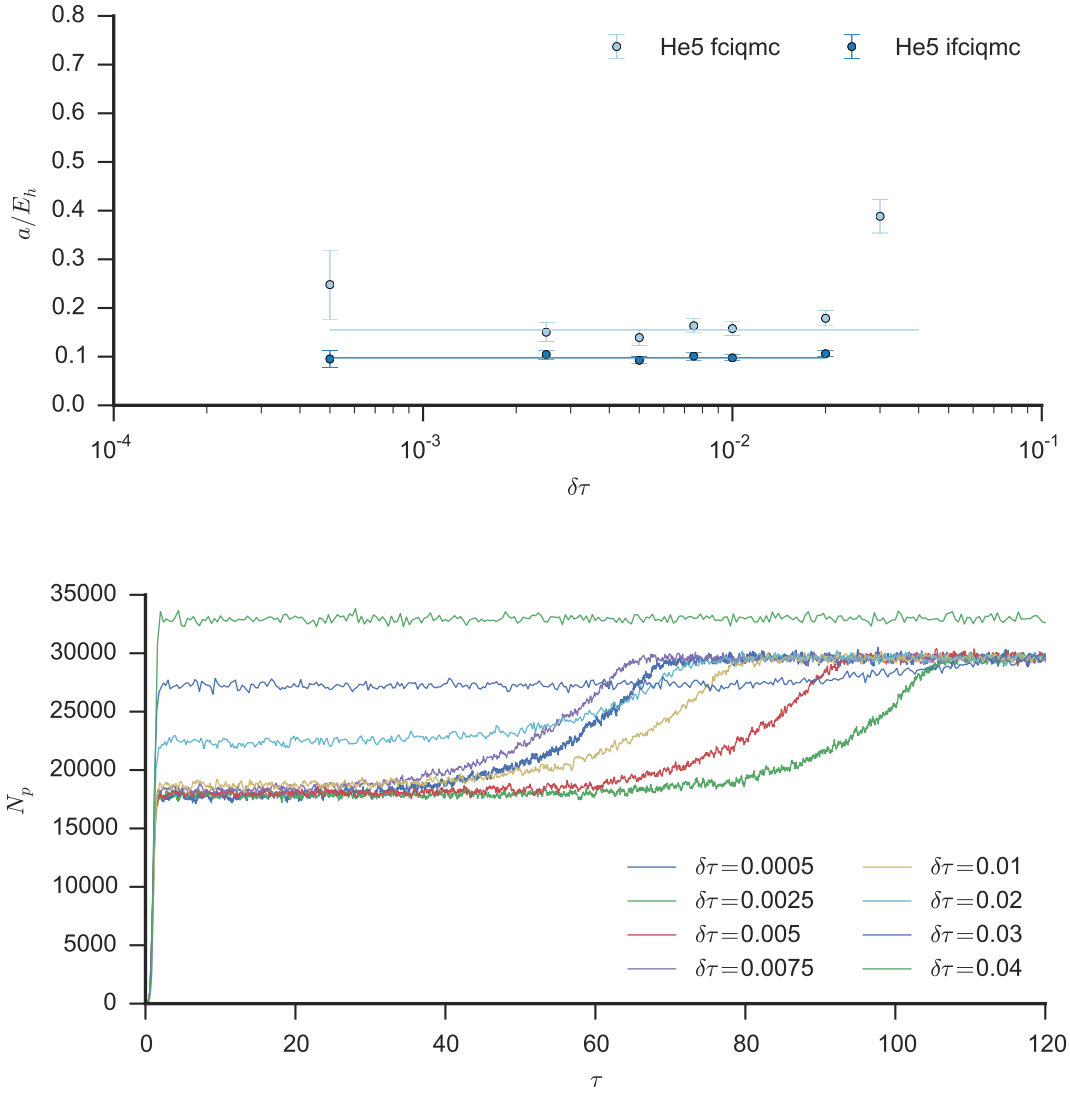



Figure A.4.: Top: The inefficiency $a$ remains constant until $\delta\tau \approx 0.01$. Bottom: The plateau height can be seen to increase after $\delta\tau \approx 0.01$.

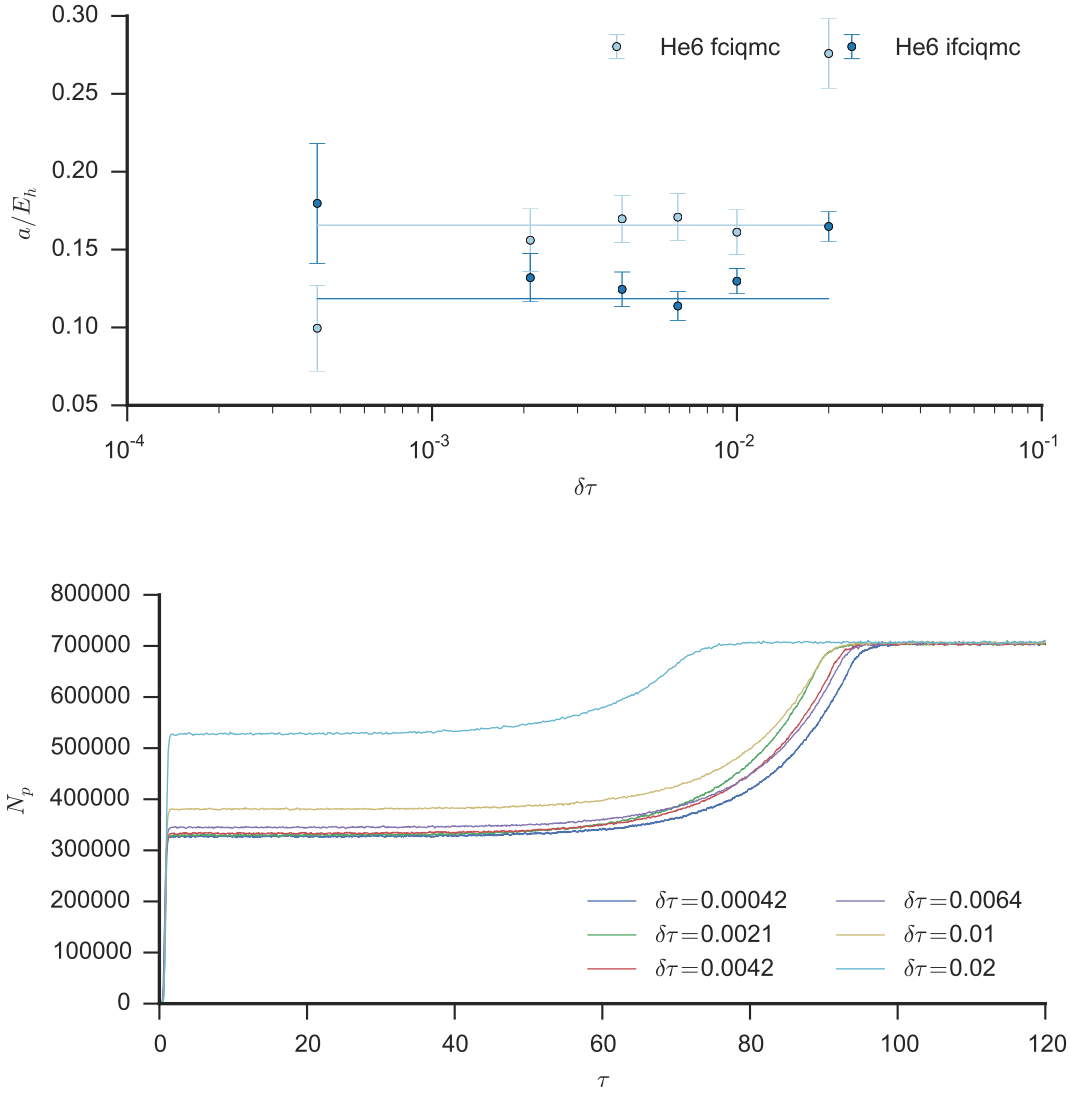



Figure A.5.: Top: The inefficiency $a$ remains constant until $\delta\tau \approx 0.0042$. Bottom: The plateau height can be seen to increase after $\delta\tau \approx 0.0042$.

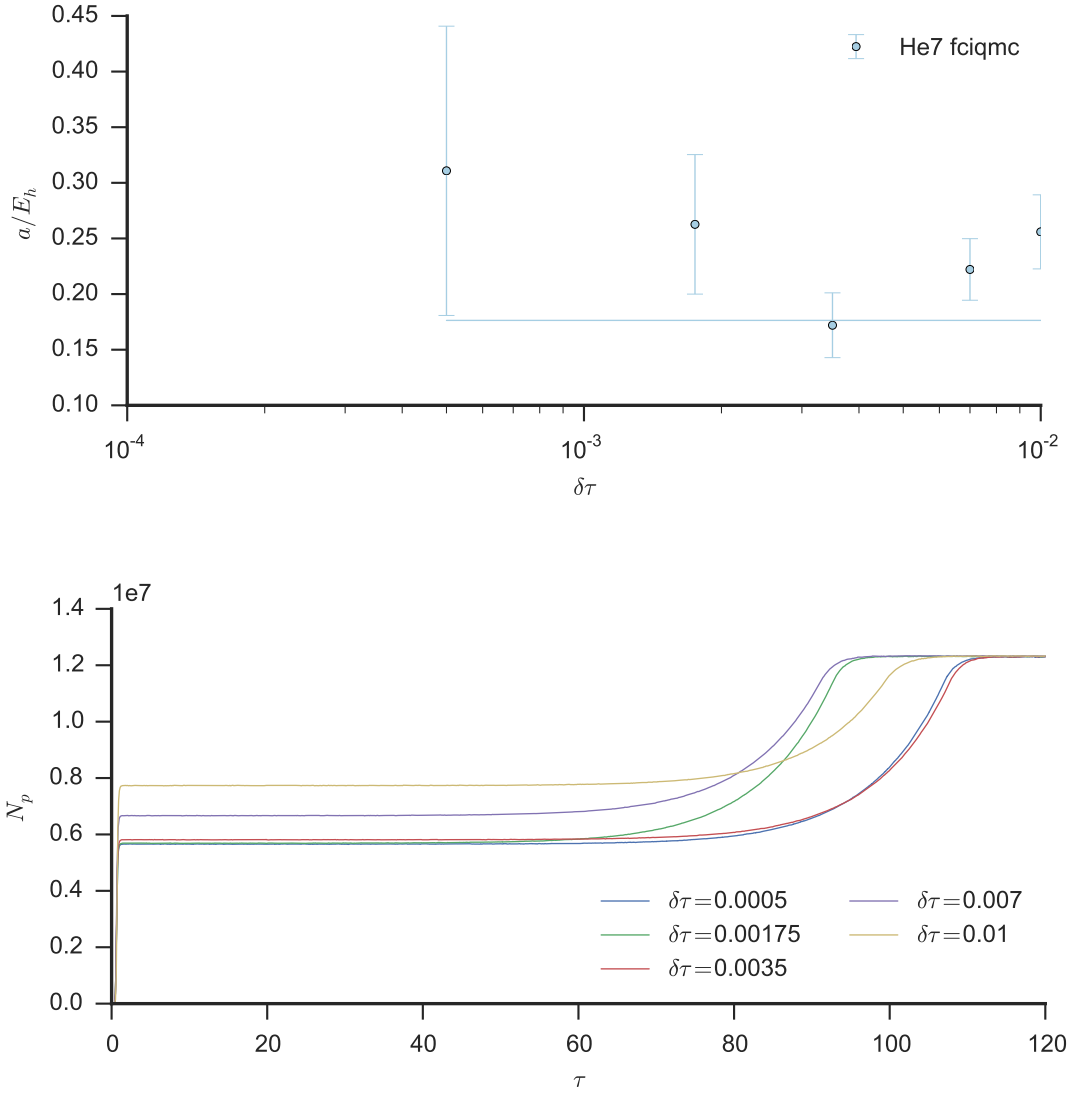



Figure A.6.: Top: The inefficiency $a$ remains constant until $\delta\tau \approx 0.0035$. Bottom: The plateau height can be seen to increase after $\delta\tau \approx 0.0035$.

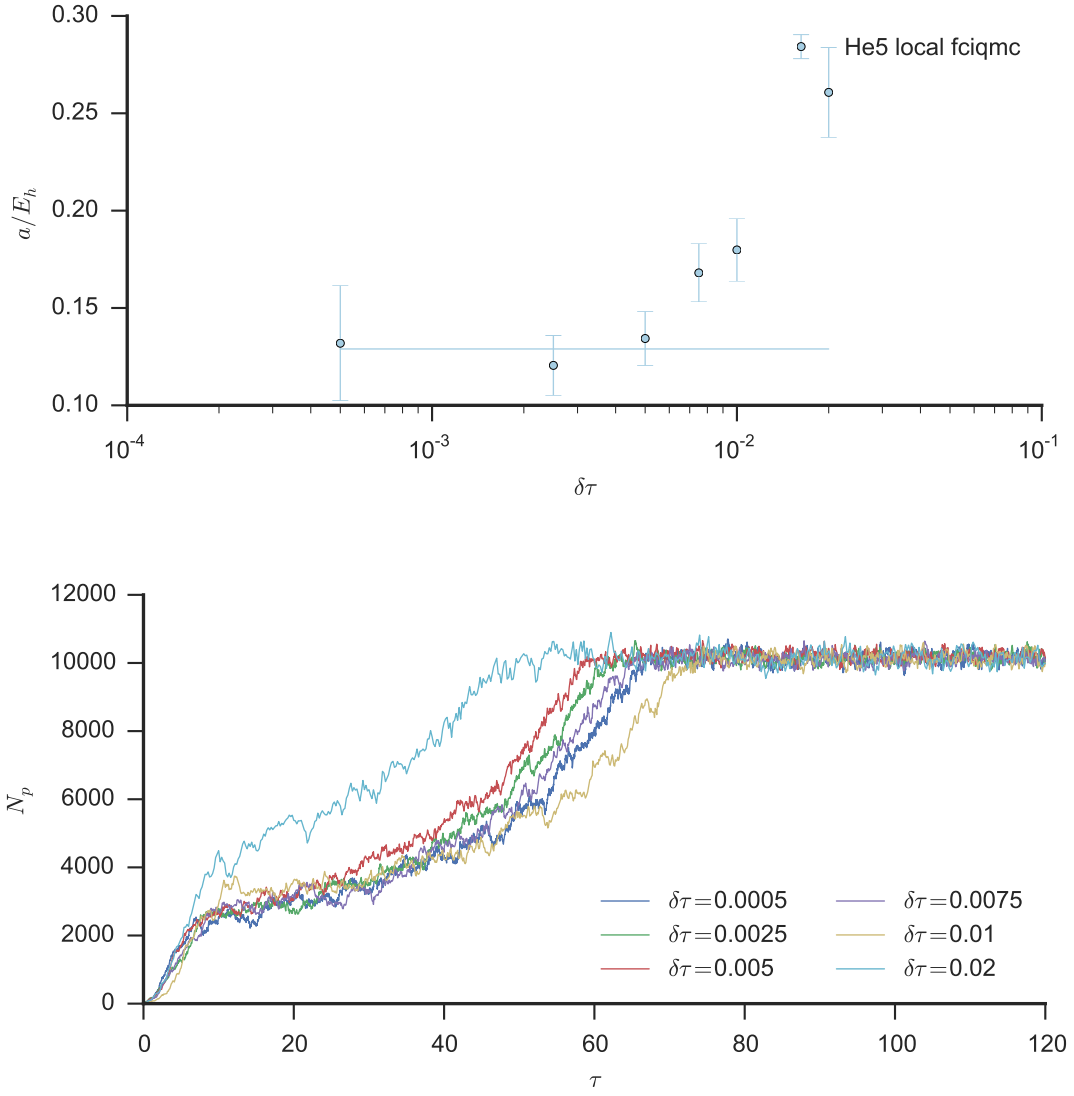



Figure A.7.: Top: The inefficiency $a$ remains constant until $\delta\tau \approx 0.0005$. Bottom: The plateau height can be seen to increase after $\delta\tau \approx 0.0005$.

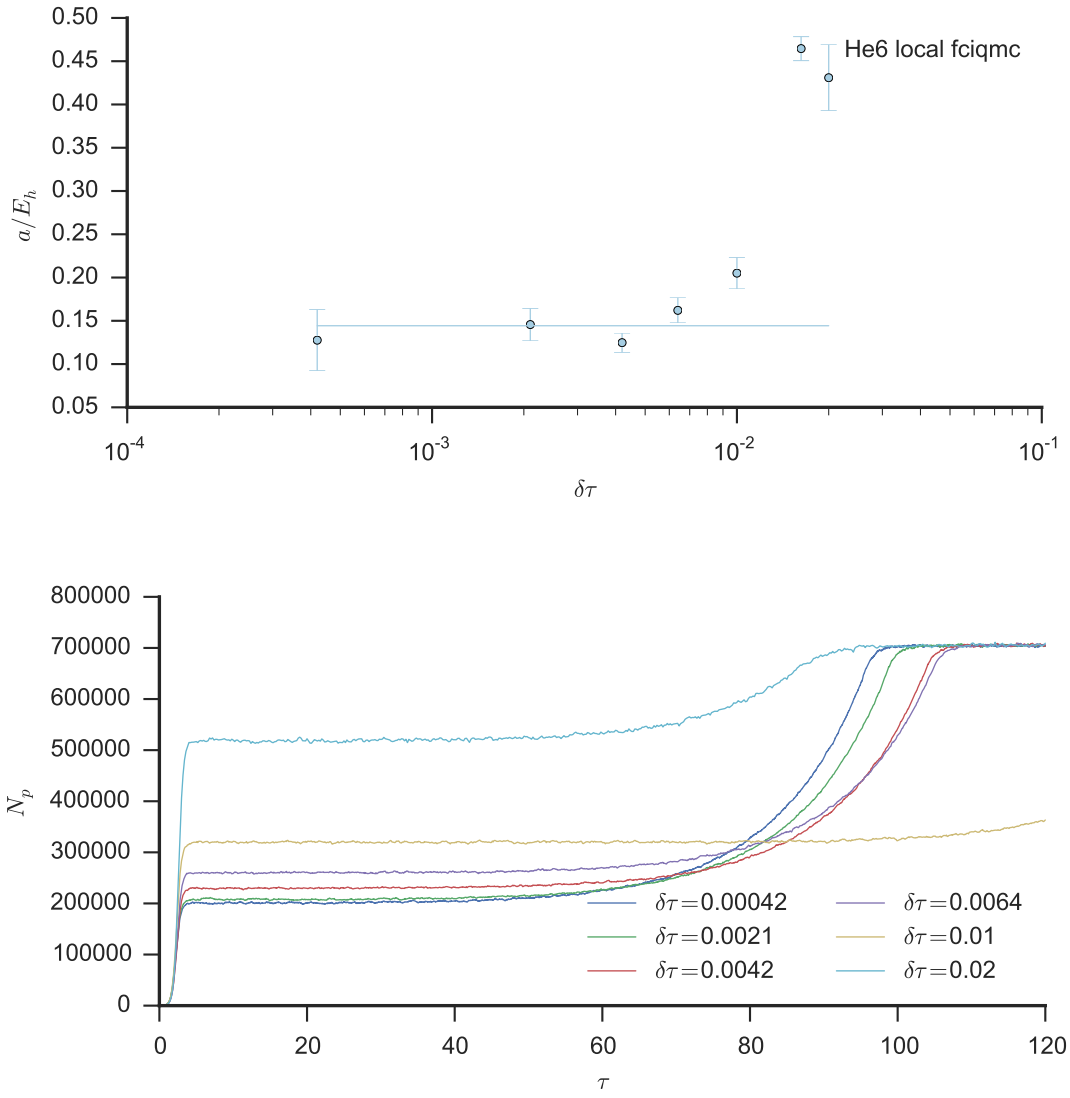



Figure A.8.: Top: The inefficiency $a$ remains constant until $\delta\tau \approx 0.0021$. Bottom: The plateau height can be seen to increase after $\delta\tau \approx 0.0021$.

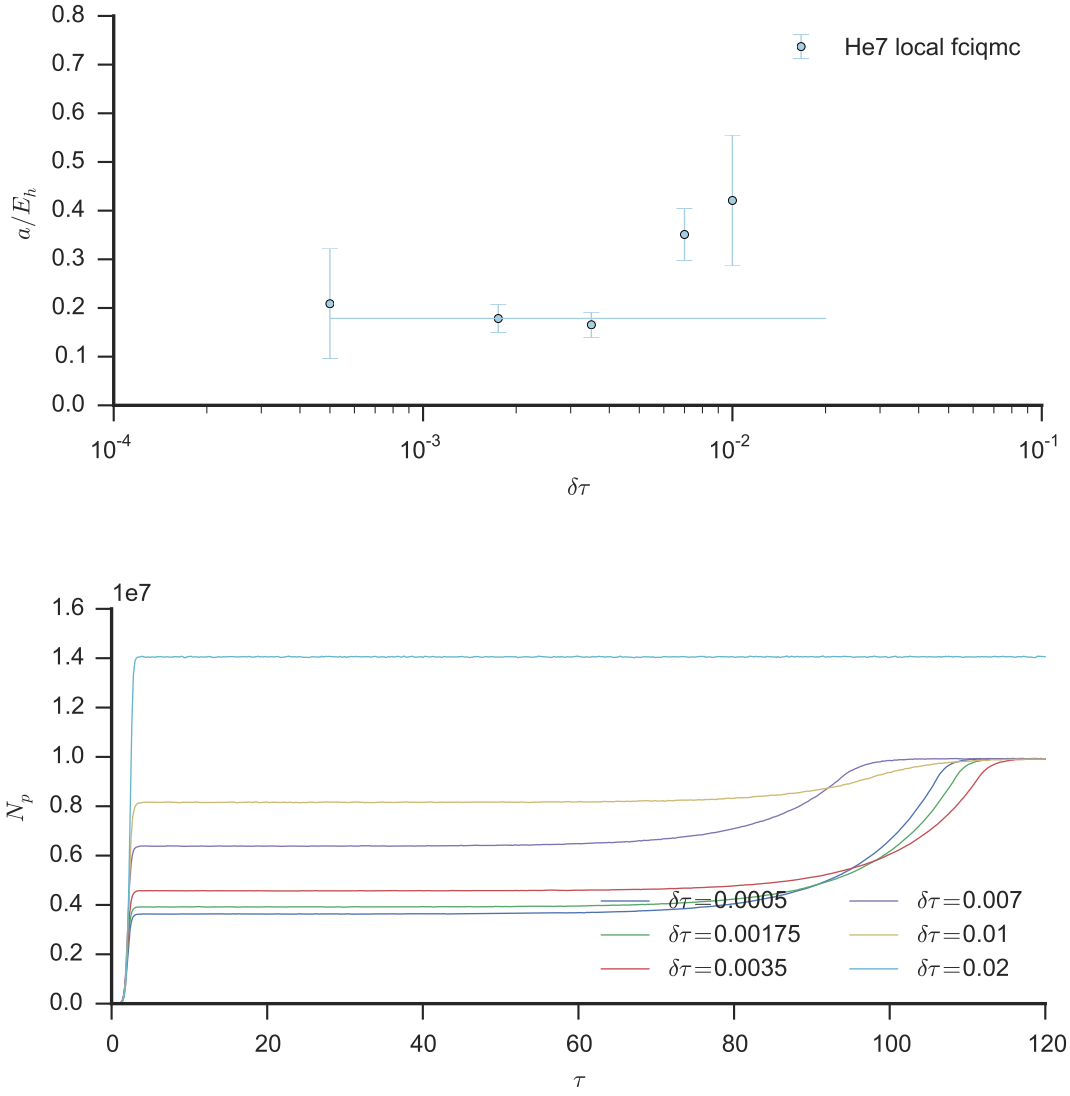



Figure A.9.: Top: The inefficiency $a$ remains constant until $\delta\tau \approx 0.00175$. Bottom: The plateau height can be seen to increase after $\delta\tau \approx 0.00175$.

# Appendix B.

# DFE Annihilation

In this Appendix we describe the algorithm which merges two input psip arrays into a third output psip array. This algorithm is used to perform annihilation between the newly spawned walkers and the updated main walker list on the DFE. Each element of a psip array contains the bit string of the determinant and the number of psips residing on that determinant. We denote an element of psip array $a$ as $(d_a, n_a)$ where $n_a$ is the number of positive or negative psips on determinant $d_a$.

The algorithm is designed such that if both input psip arrays are in the same determinant order[1], then the resulting output psip array has the same order. The same determinant cannot appear twice on each individual input psip array but may appear once on both the input psip lists.

$$d_a[j] > d_a[i] \; \forall \; i > j \tag{B.1}$$

The DFE contains two inputs which convert the two input psip arrays into two streams using controlled inputs. On each cycle the input logic may either read the next element of the psip array or repeat the last element of the psip array (Fig. 6.2 shows how a controlled input works). The first stage in the DFE annihilation algorithm controls these inputs such that the input psip arrays do not get too far unsynchronised in terms of determinant order. This is done by reading the input whose determinant was smallest on the last cycle (if they were both the same then we read both) with a few exceptions (see Fig. B.1). If all of the elements of one stream have been read then we only read from the other input.

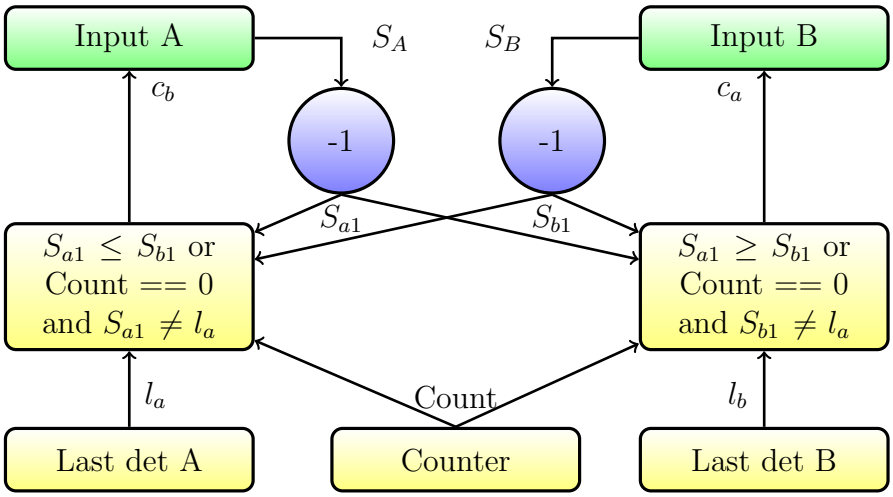
After this the determinant which is smallest is merged into the output stream, again there are a few caveats (see Fig. B.2). If both streams have the same deter-

[1]To do this we have to define an order for the determinants. As the determinant is stored as a bit string with one representing an occupied spin orbital and zero representing an unoccupied spin orbital once can simply use an integer presentation of this string. e.g 1100 = 12

minant then annihilation is performed by setting the number of psips on the output stream to the sum of the number on the input streams. If one stream is finished then the other is outputted.

If all psips on the same determinant annihilate then a cycle is wasted and the probability of spawning and diagonal death is set to zero. The determinant is removed from the main psip list by turning off the output at the bottom of the kernel.

Without knowledge of the number of distinct determinants on both the input psip arrays it is impossible to know 'a priori' how many cycles it will take to read all the data. An upper bound is the sum of the total lengths each of the input psip arrays. If we assume the total number of determinants is much smaller than the size of the determinant space, then the actual length of the merged psip array will be close to this limit.




| $S_a$ | $S_{a1}$ | $S_b$ | $S_{b1}$ | Count | $c_a$ | $c_b$ | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | ↓ | | | | ↓ | | | | |
| 1 | ? | 2 | ? | 1 | T | T | 2 | 3 | 4 | 5 | 2 | 4 | 5 | 7 | 8 |
| | | | | | | | | ↓ | | | ↓ | | | | |
| 3 | 1 | 2 | 2 | 2 | T | F | 2 | 3 | 4 | 5 | 2 | 4 | 5 | 7 | 8 |
| | | | | | | | | ↓ | | | | ↓ | | | |
| 3 | 3 | 4 | 2 | 3 | F | T | 2 | 3 | 4 | 5 | 2 | 4 | 5 | 7 | 8 |
| | | | | | | | | | ↓ | | | ↓ | | | |
| 4 | 3 | 4 | 4 | 4 | T | F | 2 | 3 | 4 | 5 | 2 | 4 | 5 | 7 | 8 |
| | | | | | | | | | | ↓ | | | ↓ | | |
| 5 | 4 | 5 | 4 | 5 | T | T | 2 | 3 | 4 | 5 | 2 | 4 | 5 | 7 | 8 |
| | | | | | | | | | | ↓ | | | | ↓ | |
| ? | 5 | 7 | 5 | 6 | T | T | 2 | 3 | 4 | 5 | 2 | 4 | 5 | 7 | 8 |
| | | | | | | | | | | ↓ | | | | | ↓ |
| ? | ? | 8 | ? | 7 | F | T | 2 | 3 | 4 | 5 | 2 | 4 | 5 | 7 | 8 |

Figure B.1.: Top: The input control stage of the DFE annihilation algorithm which merges two input psip arrays into a third output psip array. Stream offsets are represented in blue. The input with the smallest determinant on the last cycle is read or if both inputs are the determinants are the same. On the first cycle we read both inputs and we turn of the input if all the data has been read. An input which is not read simply repeats the value on the last cycle. In this way the two inputs remain reasonably in sync and so can provide an input to the merge algorithm. Bottom: An example of the input control for annihilation of sorted spawned psips and the surviving parent psips see Fig. B.1. $d_a = (1, 3, 4, 5)$ and $d_b = (2, 4, 5, 7, 8)$ where 1, 2,etc. represents the first,second,etc. possible determinant in our arbitrary defined determinant order. Input A reads $d_a$ and Input B reads $d_b$. T represents true and F represents false.

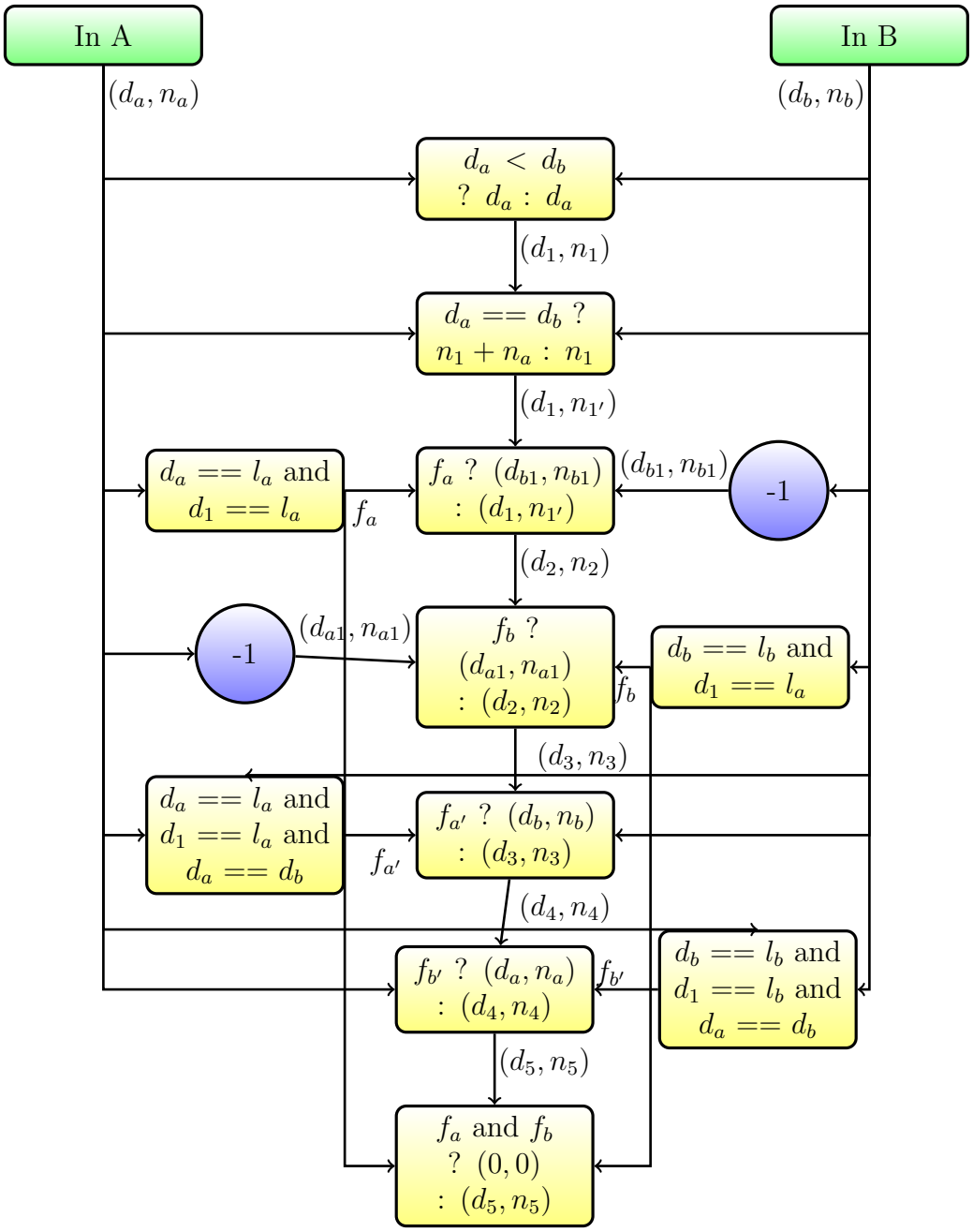



Figure B.2.: The algorithm which merges the two controlled input streams into a single stream of determinants and psips. The stream with the smallest determinant is merged into the output list and if the two determinants are the same the psips are annihilated. Subject to the following boundary conditions: if one stream has finished on this cycle we output the other stream with a stream offset of minus one. Unless there was an annihilation event, then we output the other stream without a stream offset. If both streams have finished we output a pad $(0, 0)$.