TOWARDS MORE ACCURATE AND EFFICIENT QUANTUM MONTE CARLO CALCULATIONS: TWO IMPROVEMENTS

A Thesis

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Master of Science

by

Mingtong Han

August 2017

ABSTRACT

A pair-product projector, which projects onto an intrinsically Fermionic ground state, is implemented as part of a newly proposed Projection Monte Carlo method aimed at overcoming the Fermionic Sign Problem without employing the Fixed-Node approximation. Evaluating the fully anti-symmetrized pair-product projector requires factorial time. Three polynomial-time approximations are implemented and shown to be very close to the fully anti-symmetrized projector.

A Fortran program is developed to generate configuration state functions for atomic systems using the projection method, with the goal of building all symmetries of a system into the trial wave functions used in Quantum Monte Carlo calculations. This brings an additional advantage of a significantly reduced number of variational parameters in trial wave functions. The program makes use of a bit-packed representation of Slater determinants and various algorithms to cut down run time and memory cost.

BIOGRAPHICAL SKETCH

Mingtong Han graduated from Peking University with a Bachelor of Science in Physics in 2014. Same year, she joined the Master of Science program in Applied Physics at Cornell University. At Cornell, she found a great interest in computational physics and mathematical modeling which led to this research topic. After completing her MS, Mingtong will be joining Goldman Sachs as an analyst.

ACKNOWLEDGEMENTS

I would like to offer Professor Cyrus Umrigar my deepest gratitude for the invaluable guidance and support throughout the two research projects. Our group have been immensely helpful to the work in this thesis, who were always ready to share their knowledge and expertise. The first project was a work of corporation with Junhao Li who implemented the interpolation part of the code and the fully anti-symmetrized projector. Adam Holmes and Matt Otten both generously shared their meaningful opinions on the second project. Special thanks to Professor David Muller and Professor Eva Tardos for serving on my special committee.

I'd like to thank my parents for their love, encouragement and financial support, who have always been there for me through all my endeavors. Special thanks to Heidi, Gargi, Yunsheng, Alex, Steven, Emily, Di, Matheus, Al and Reet for making the cloudy and snowy days in Ithaca bright and warm.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Quantum Monte Carlo (QMC) methods are among the most accurate methods for solving the Schrodinger equation for many-body systems. The ability to employ a 3N-dimensional trial wave function with explicit electron-electron, electron-nucleus and electron-electron-nucleus correlations is one of the main advantages of QMC methods over Density Functional Theory (DFT) which reduces the complexity of a problem to much lower dimensional functions. QMC methods scale better with system size than high-order Quantum Chemistry methods, which makes larger and more complicated many-body systems accessible with the same computational power. The two most commonly used classes of QMC methods are Variational Monte Carlo (VMC), and various Projector Monte Carlo (PMC) methods.

1.1 Variational Monte Carlo

The main idea of VMC is to calculate the expectation values of quantum mechanical observables with Monte Carlo integration by using a trial wave function Ψ_T. The quantity of most interest is the ground state energy of a system, which can be estimated as follows:

$$E_V = \frac{\langle \Psi_T | \hat{H} | \Psi_T \rangle}{\langle \Psi_T | \Psi_T \rangle} \geqslant E_0 \tag{1.1}$$

The variational energy E_V provides an upper bound of the exact ground state energy E_0.

In VMC calculations, $3N-$dimensional points, called walkers, are sampled

from the probability density function $\frac{|\Psi_T(\mathbf{R})|^2}{\int |\Psi_T(\mathbf{R})|^2}$ using the Metropolis-Hastings algorithm [13, 8]. Equation 1.1 can be written as follows:

$$E_V = \frac{\int \left|\Psi_T(\mathbf{R})\right|^2 \left[\Psi_T(\mathbf{R})^{-1} \hat{H} \Psi_T(\mathbf{R})\right] d\mathbf{R}}{\int \left|\Psi_T(\mathbf{R})\right|^2 d\mathbf{R}} \tag{1.2}$$

where the local energy is defined as:

$$E_L(\mathbf{R}) = \Psi_T(\mathbf{R})^{-1} \hat{H} \Psi_T(\mathbf{R}) \tag{1.3}$$

In the Metropolis-Hastings algorithm, moves are proposed by sampling from certain probability distributions such as shifted Gaussian (though CHAMP makes more efficient choices), and the moves are accepted or rejected with a probability that is derived from the detailed-balance condition. The local energy is then evaluated at the updated location of each walker. A more efficient way to evaluate the local energy is to use the local energies at the initial and final points of the move, weighted by the rejection and acceptance probabilities, respectively. Finally, the variational energy is evaluated by averaging the local energies:

$$E_V = \frac{1}{M} \sum_{m=1}^{M} E_L(\mathbf{R}_m) \tag{1.4}$$

where M is the number of walkers.

E_V approaches the exact energy E_0 as Ψ_T becomes closer the exact ground state wave function Ψ_0. In practice, since the variational wave function has only limited flexibility, E_V goes down but does not become exact when the parameters in the wave function are optimized. Projector Monte Carlo (PMC) methods can be used to obtain a yet more accurate energy. Because the accuracy and efficiency of PMC methods also depend on the trial wave function, it is common practice to optimize trial wave functions by minimizing the VMC energy before running the more sophisticated PMC calculations, as VMC is the simplest of QMC methods.

1.2 Projector Monte Carlo

Projector Monte Carlo (PMC) methods provide a means for obtaining the exact energy (at least for Bosonic systems) using the mixed expectation value of energy

$$E_0 = \frac{\langle \Psi_0 | \hat{H} | \Psi_T \rangle}{\langle \Psi_0 | \Psi_T \rangle} = \frac{\int f(\mathbf{R}) E_L(\mathbf{R}) d\mathbf{R}}{\int f(\mathbf{R}) d\mathbf{R}} = \langle E_L \rangle_{\Psi_0 \Psi_T} \tag{1.5}$$

$$E_L(\mathbf{R}) = \frac{\hat{H}\Psi_T(\mathbf{R})}{\Psi_T(\mathbf{R})} \tag{1.6}$$

$$f(\mathbf{R}) = \Psi_0(\mathbf{R})\Psi_T(\mathbf{R}) \tag{1.7}$$

where E_L is the local energy, and $\Psi_0(\mathbf{R})$ is the true ground state wave function. Now, instead of sampling Ψ_T^2, $f(\mathbf{R})$ is sampled instead. Note that Ψ_T here isn't necessarily the same as the trial wave function used in VMC. Sometimes, this function is intentionally different in which case it is called the guiding function Ψ_G, in which case, weight factors of Ψ_T/Ψ_G need to be included in some of the terms (See Ref. [22] for detailed expressions.)

The question arises: how can one sample from $\Psi_0(\mathbf{R})\Psi_T(\mathbf{R})$ when $\Psi_0(\mathbf{R})$ is unknown? This is done using a stochastic implementation of the power method. The idea is to start from an initial guess Ψ, and approach Ψ_0 by repeated application of a projector $\hat{P}$ whose dominant eigenstate corresponds to the ground state of the Hamiltonian. There are various PMC methods, some formulated in real-space and others in orbital space. In this thesis, only real-space PMC is discussed. For example, in Diffusion Monte Carlo (DMC), an exponential projector is used

$$\hat{P}_{DMC}(\tau) = e^{-(\hat{H}-E_T)\tau} \tag{1.8}$$

where $\tau = it$ is imaginary time and E_T is an undetermined trial energy. As $\tau \to \infty$, all the excited states, namely eigenstates of $\hat{H}$ with $E_i > E_0$ will decay

exponentially faster and Ψ will eventually be dominated by Ψ_0:

$$\lim_{\tau\to\infty} \hat{P}_{DMC}(\tau)\,|\Psi\rangle = \lim_{\tau\to\infty} \sum_i e^{-(E_i-E_T)\tau}\,|\Psi_i\rangle\,\langle\Psi_i|\Psi\rangle \tag{1.9}$$

$$= \lim_{\tau\to\infty} e^{-(E_0-E_T)\tau}\,|\Psi_0\rangle\,\langle\Psi_0|\Psi\rangle \tag{1.10}$$

The real-space projector, often called a Green function, is written as

$$G(\mathbf{R}'|\mathbf{R};\tau) = \langle\mathbf{R}'|\hat{P(\tau)}|\mathbf{R}\rangle \tag{1.11}$$

$$\Psi(\mathbf{R}';\tau) = \int G(\mathbf{R}'|\mathbf{R};\tau)\Psi(\mathbf{R})d\mathbf{R} \tag{1.12}$$

where $\mathbf{R}'$ is the proposed move for the walker at $\mathbf{R}$. The importance sampled version of the Green function is:

$$\widetilde{G}(\mathbf{R}'|\mathbf{R};\tau) = \Psi_T(\mathbf{R}')G(\mathbf{R}'|\mathbf{R};\tau)\frac{1}{\Psi_T(\mathbf{R})} \tag{1.13}$$

$$f(\mathbf{R}';\tau) = \int \widetilde{G}(\mathbf{R}'|\mathbf{R};\tau)f(\mathbf{R})d\mathbf{R} \tag{1.14}$$

For example, the DMC importance sampled Green function is:

$$\widetilde{G}_{DMC}(\mathbf{R}'|\mathbf{R};\tau) = \frac{1}{(2\pi\tau)^{3N/2}} e^{-\frac{(\mathbf{R}'-\mathbf{R}-\nu(\mathbf{R})\tau)^2}{2\tau}} e^{-\left(\frac{E_L(\mathbf{R}')+E_L(\mathbf{R})}{2}-E_T\right)\tau} \tag{1.15}$$

$$\nu(\mathbf{R}) = \nabla\Psi(\mathbf{R})/\Psi(\mathbf{R}) \tag{1.16}$$

where $\nu(\mathbf{R})$ is called the drift velocity. Since $\widetilde{G}_{DMC}(\mathbf{R}'|\mathbf{R};\tau)$ is not normalized it is necessary to employ weighted walkers.

The PMC energy is the equal to the exact ground state energy E_0 independent of Ψ_T if no other approximations are made, except for statistical error which is unavoidable but can be reduced with a larger number of Monte Carlo samples. The finite time-step error and population-control error in actual implementation can both be eliminated (See Refs. [3, 17, 23, 16, 14]).

The thesis presents two different projects for improving the accuracy and efficiency of Quantum Monte Carlo calculations for eventual inclusion in the Cornell-Holland Ab-initio Materials Package (CHAMP). The pair-product projector described in Chapter 2 is part of a newly proposed PMC algorithm [22] that attempts to overcome the Fermion Sign Problem without resorting to the commonly used fixed-node approximation by means of an anti-symmetrized projector. Chapter 3 describes a new program to construct configuration state functions (CSFs) for atoms that builds full symmetry into the trial functions used in any Quantum Monte Carlo method, with an extra benefit of speeding up the optimization of trial wave functions using VMC. Each chapter presents the motivation of the project, a general mathematical description of the algorithm and implementation details including algorithms for sub-problems in the project.

CHAPTER 2

PAIR-PRODUCT PROJECTOR

2.1 The Fermion Sign Problem

PMC methods can give exact energies (aside from statistical error) in certain special situations, but in general they suffer from a sign problem which manifests itself differently depending on the particular PMC method used. The Sign Problem refers to the fact that an undesired state grows in magnitude relative to the state of interest as the system evolves under repeated application of the projector.

For example, the DMC projector in Eq. 1.15 assumes the wave function $\Psi(\mathbf{R})$ is always of the same sign. This is true for ground state wave functions of Bosonic systems and a few other simple electronic systems [22]. However, for fermions such as electrons, the state of interest is actually the Fermionic ground state whose wave function has nodes and both positive and negative signs. The Sign Problem arises because there is at least one Bosonic state with lower energy than the Fermionic ground state, which means the desired Fermionic ground state is different from what is mathematically the ground state of the Hamiltonian, or in other words the dominant state of the projector. Another aspect of the Sign Problem in DMC is that, even if both positive and negative weights are allowed, walkers representing Ψ_0 and $-\Psi_0$ will both build up. Because these are equally good solutions of the Schrodinger equation, both will be sampled with about equal probability resulting in a severe signal-to-noise issue due to cancellation of contributions from positive and negative weight walkers [22].

The Fixed-Node approximation [2, 15, 20] is the most common solution to the Sign Problem. The idea is to use a trial wave function Ψ_T^{FN} with nodal surfaces approximating those of the true fermionic ground state wave function, and force walkers to stay within their initial nodal pockets. This method produces the exact fermionic ground state energy if and only if the initial guess of the nodal surfaces coincides with the nodal surfaces of the true ground state wave function, which is almost impossible because the $(3N-1)$−dimensional nodal surfaces can be very complicated while the trial wave functions have limited flexibility even if they have a few thousand variational parameters. This means the resulting ground state energy has a systematic positive bias.

Methods have been developed in effort to ameliorate the bias caused by the Fixed-Node assumption, but none have been very successful. Therefore, we try approaching the problem from a completely new angle and discard the Fixed-Node assumption entirely.

The root of the Sign Problem is that the projector in use has Bosonic states with energy lower than the true Fermionic ground state of the system. Therefore, instead of forcing the wave function towards an approximate nodal surface to obtain a fermionic state, we build anti-symmetry of the wave function into a Pair-Product projector so that its intrinsic ground state is a fermionic state.

The Pair-Product projector resolves the first aspect of the Sign Problem. For the second aspect of the Sign Problem in DMC, there are methods being developed such as stochastic reconfiguration [18, 19].

2.2 Pair-Product Projector

The main idea of this new approach is to first construct a projector that does not make the Fixed-Node assumption, and then anti-symmetrize the projector so that its instrinsic ground state is fermionic.

We use a pair-product projector G_1 by assuming that the projector for a multi-electron system is equal to the product of all pair projectors representing electron-electron and electron-nucleus interactions in the system:

$$G_1(\mathbf{R}'|\mathbf{R};\tau) = e^{\tau E_T} \prod_{\alpha=1}^{N_{nuc}} \prod_{k=1}^{N} p_{en}(\mathbf{r}'_k - \mathbf{r}_\alpha|\mathbf{r}_k - \mathbf{r}_\alpha;\tau) \prod_{1\le i<j\le N} p_{ee}(\mathbf{r}'_j - \mathbf{r}'_i|\mathbf{r}_j - \mathbf{r}_i;\tau) \quad (2.1)$$

where N is the number of electrons, N_{nuc} is the number of nuclei, and the pair projectors are defined by exponential of pair actions (also called pair potentials)[5]:

$$p_{en}(\mathbf{r}'_k - \mathbf{r}_\alpha|\mathbf{r}_k - \mathbf{r}_\alpha;\tau) = e^{-u_{en}(\mathbf{r}'_k-\mathbf{r}_\alpha|\mathbf{r}_k-\mathbf{r}_\alpha;\tau)} \quad (2.2)$$

$$p_{ee}(\mathbf{r}'_j - \mathbf{r}'_i|\mathbf{r}_j - \mathbf{r}_i;\tau) = e^{-u_{ee}(\mathbf{r}'_j-\mathbf{r}'_i|\mathbf{r}_j-\mathbf{r}_i;\tau)} \quad (2.3)$$

The pair actions can be obtained by a pre-existent Path Integral Monte Carlo (PIMC) program using the matrix squaring technique [11].

The projector can be anti-symmetrized by summing the contributions from all possible permutations of electrons at the proposed new 3N-dimentional coordinate $\mathbf{R}'$:

$$G_2(\mathbf{R}'|\mathbf{R};\tau) = \sum_{\sigma} sgn(\sigma) G_1(\sigma(\mathbf{R}')|\mathbf{R};\tau) \quad (2.4)$$

where σ is a permutation of N electrons, and $sgn(\sigma)$ is +1 if σ is an even permutation and $sgn(\sigma)$ is -1 if σ is an odd permutation.

The computational cost of the fully anti-symmetrized pair-product projector is $O(N!)$. Note that this cannot be improved for the exact projector because the

electron-electron pair action terms forbids us from trivially packing all the terms into one determinant which can be evaluated in $O(n^3)$ time by Gaussian Elimination. The non-reducibility of the problem has been mathematically proved [12].

2.3 Approximations

To improve the time complexity of evaluating the projector, we propose three different approximations to the full anti-symmetrization of the projector which are close to the exact sum of the all permutations but take only polynomial time to evaluate.

One idea is to use an averaged U_{ee} for all electron-electron interactions and pack all the electron-nucleus interactions into one determinant which can be evaluated in $O(n^3)$. We can approximate U_{ee} using either the end-point pair product action of electrons interacting with each other [5]:

$$U_{ee}(\mathbf{R}'|\mathbf{R}) = \sum_{i<j} \frac{u_{ee}(\mathbf{r}'_{ij}|\mathbf{r}'_{ij};\tau) + u_{ee}(\mathbf{r}_{ij}|\mathbf{r}_{ij};\tau)}{2} \tag{2.5}$$

or the pair product action of the electrons interacting with each other for the identity permutation [5]:

$$U_{ee}(\mathbf{R}'|\mathbf{R}) = \sum_{i<j} u_{ee}(\mathbf{r}'_{ij}|\mathbf{r}_{ij};\tau) \tag{2.6}$$

The overall projector becomes an approximate anti-symmetric projector:

$$G_2(\mathbf{R}'|\mathbf{R};\tau) = e^{-\left(U_{ee}(\mathbf{R}'|\mathbf{R})-E_T\right)\tau} \times \begin{vmatrix} g(\mathbf{r}'_1|\mathbf{r}_1) & g(\mathbf{r}'_1|\mathbf{r}_2) & \dots & g(\mathbf{r}'_1|\mathbf{r}_N) \\ g(\mathbf{r}'_2|\mathbf{r}_1) & g(\mathbf{r}'_2|\mathbf{r}_2) & \dots & g(\mathbf{r}'_2|\mathbf{r}_N) \\ \vdots & & & \\ g(\mathbf{r}'_N|\mathbf{r}_1) & g(\mathbf{r}'_N|\mathbf{r}_2) & \dots & g(\mathbf{r}'_N|\mathbf{r}_N) \end{vmatrix} \tag{2.7}$$

where $g(\mathbf{r'}_j|\mathbf{r}_i)$ is the product of the pair projectors of an electron moving from $\mathbf{r}_i$ to $\mathbf{r}_j^{'}$ interacting with all the nuclei:

$$g(\mathbf{r'}_j|\mathbf{r}_i) = \prod_{\alpha=1}^{N_{nuc}} p_{en}(\mathbf{r}'_k - \mathbf{r}_\alpha|\mathbf{r}_k - \mathbf{r}_\alpha;\tau) \tag{2.8}$$

These two approximations will tend to be too small at large τ because as τ increases the $g(\mathbf{r'}_j|\mathbf{r}_i)$ elements in the determinant become equal more quickly than the u_{ee} factors in U_{ee}.

A possible better approximation is to also incorporate the electron-electron interactions into the determinant. As the electron-electron interactions can vary significantly due to permutation, packing them into the determinant is likely to get closer to the exact anti-symmetrization. We use the following $g(\mathbf{r'}_j|\mathbf{r}_i)$ elements for the determinant:

$$g(\mathbf{r'}_j|\mathbf{r}_i) = \Big(\frac{\mu}{2\pi\tau}\Big)^{3/2} e^{-\frac{\mu(\mathbf{r'}_j-\mathbf{r}_i)^2}{2} - W_{ee}(\mathbf{r'}_j|\mathbf{r}_i;\tau)} \prod_{\alpha=1}^{N_{nuc}} p_{en}(\mathbf{r}'_k - \mathbf{r}_\alpha|\mathbf{r}_k - \mathbf{r}_\alpha;\tau) \tag{2.9}$$

where $W_{ee}(\mathbf{r'}_j|\mathbf{r}_i;\tau)$ is defined as:

$$W_{ee}(\mathbf{r'}_j|\mathbf{r}_i;\tau) = \sum_{k\neq 1,i,j} u_{ee}(\mathbf{r}'_j - \mathbf{r'}_k|\mathbf{r}_i - \mathbf{r}_k;\tau) \; + u_{ee}(\mathbf{r}'_j - \mathbf{r'}_i|\mathbf{r}_i - \mathbf{r}_1;\tau) + u_{ee}(\mathbf{r}'_j - \mathbf{r'}_1|\mathbf{r}_i - \mathbf{r}_j;\tau) \tag{2.10}$$

The overall approximated projector becomes:

$$G_2(\mathbf{R'}|\mathbf{R};\tau) = e^{\tau E_T} \times \begin{vmatrix} g(\mathbf{r'}_1|\mathbf{r}_1) & g(\mathbf{r'}_1|\mathbf{r}_2) & \dots & g(\mathbf{r'}_1|\mathbf{r}_N) \\ g(\mathbf{r'}_2|\mathbf{r}_1) & g(\mathbf{r'}_2|\mathbf{r}_2) & \dots & g(\mathbf{r'}_2|\mathbf{r}_N) \\ \vdots & & & \\ g(\mathbf{r'}_N|\mathbf{r}_1) & g(\mathbf{r'}_N|\mathbf{r}_2) & \dots & g(\mathbf{r'}_N|\mathbf{r}_N) \end{vmatrix} \tag{2.11}$$

2.4 Implementation

The exact pair product projector and the three approximation methods are implemented in Fortran for CHAMP in coorporation with Junhao Li and Cyrus

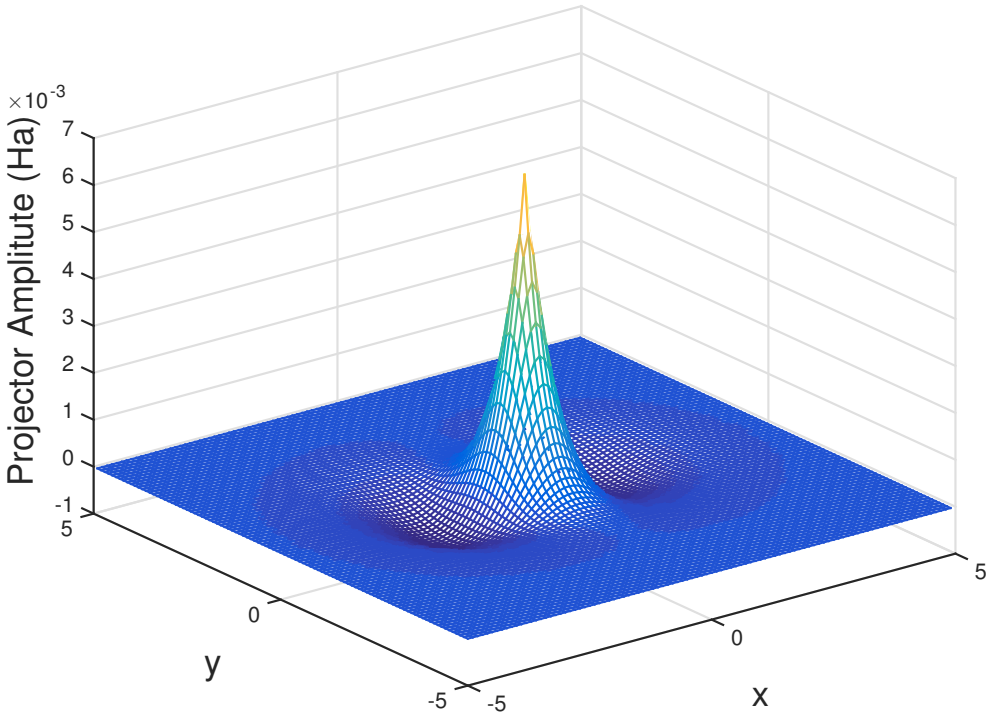

Figure 2.1: The fully anti-symmetrized pair-product projector of a three-electron system calculated at $\tau = 1$.

Umrigar. The code makes use of pair actions u_{ee} and u_{en} pre-caculated with the PIMC program [11], and interpolates them with a 2D cubic spline. The rest are implemented according to equations 2.1 - 2.11.

The exact projector of a three-electron system is shown in Figure 2.2, and compared with the three approximations in Figure 2.2, Figure 2.3 and Figure 2.4. The three approximations are very close in shape to the exact projector, but differ in amplitudes near the peaks and dips. It is clear in Figure 2.3 and Figure 2.4 that the third approximation outperforms the first two.

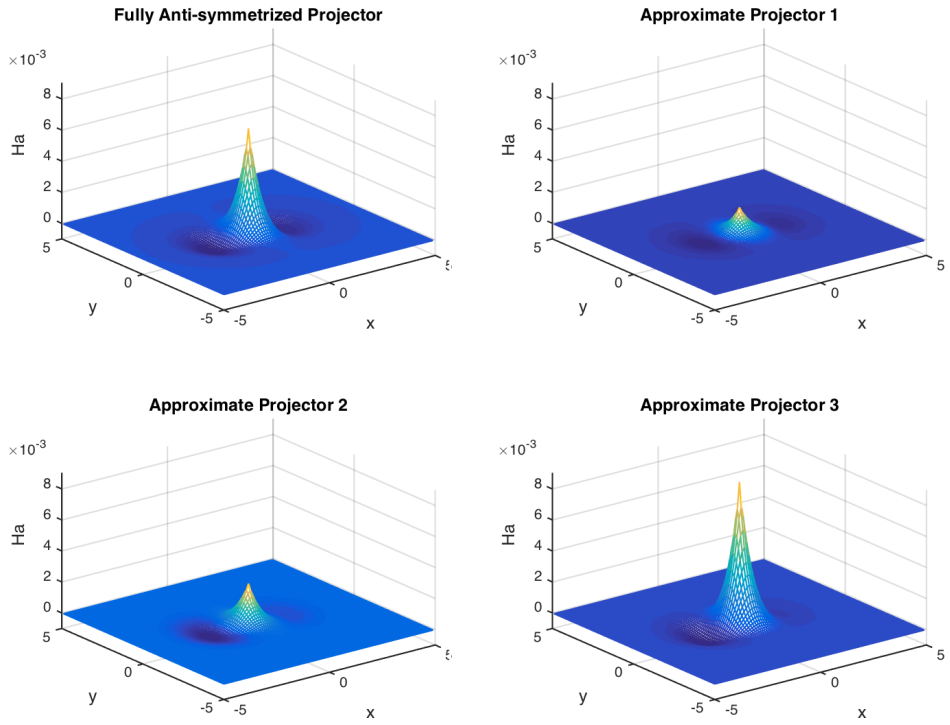

Figure 2.2: The fully anti-symmetrized pair-product projector and three approximated projectors of a three-electron system calculated at $\tau = 1$.

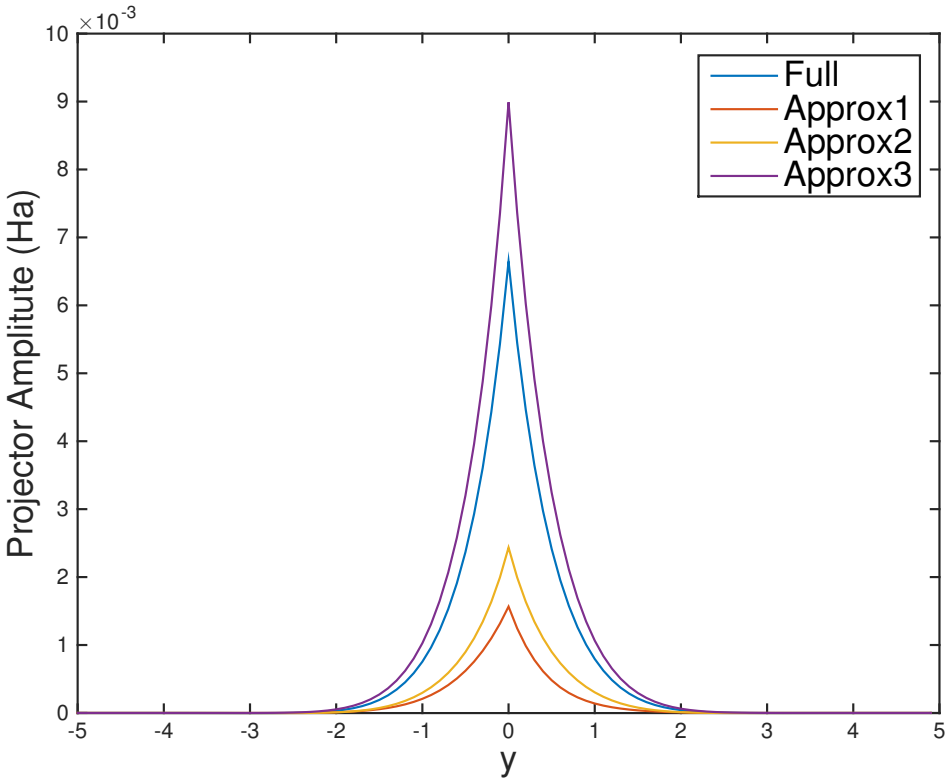

Figure 2.3: 2D cut of fully anti-symmetrized pair-product projector and three approximated projectors of a three-electron system on plane $x = 0$ calculated at $\tau = 1$.

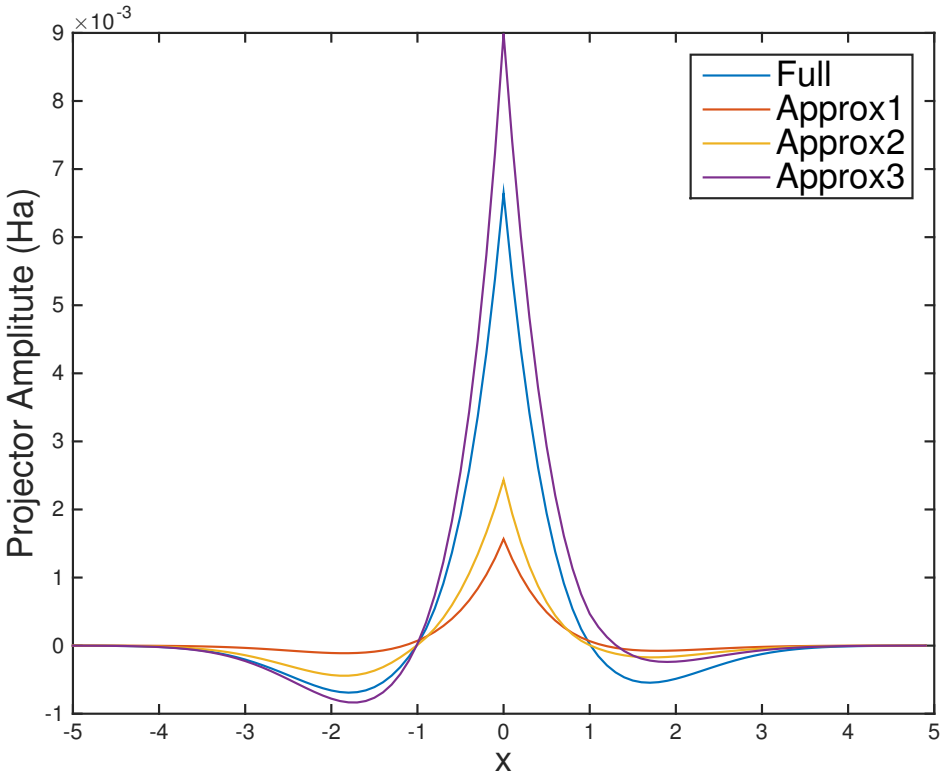

Figure 2.4: 2D cut of fully anti-symmetrized pair-product projector and three approximated projectors of a three-electron system on plane $y = 0$ calculated at $\tau = 1$.

CHAPTER 3

CONSTRUCTING CONFIGURATION STATE FUNCTIONS FOR ATOMS

3.1 Motivation

The main purpose of the program for this part of the thesis is to build all symmetries of atomic systems into QMC wave functions. Symmetries can be used in various computational physics problems in order to save computational effort. It is feasible to take advantage of symmetries in QMC calculations, because one advantageous characteristic of QMC methods is that one can incorporate prior knowledge of a physical system, such as symmetries, into the specification of its trial wave function. QMC calculations generally use starting trial wave functions obtained from quantum chemistry programs such as GAMESS or Gaussian. However, these programs typically use only abelian symmetries instead of all the symmetries that exist [21]. Hence, it is very desirable to have a program in CHAMP to build all symmetries of atomic systems into the trial wave functions, which is why this program is developed.

In addition, making use of symmetries in QMC calculations provides the extra advantage of reducing the number of variational parameters in trial wave functions. The typical form of trial wave functions in QMC is the Slater-Jastrow type [9]. The single determinant Slater-Jastrow wave function is written as

$$\Psi_T(\mathbf{R}) = D\Big(\{\phi_i(\mathbf{r_j})\}\Big)e^{J(\mathbf{R})} \tag{3.1}$$

where the first factor D is a determinant of single-particle orbitals that makes the wave function anti-symmetric, and the Jastrow factor $e^{J(\mathbf{R})}$ provides explicit electron-electron and electron-electron-nucleus correlation terms. More accu-

rate trial wave functions can be obtained by using a linear combination of determinants [7, 23, 4]. Accurate trial wave functions reduce both the statistical and the systematic errors (particularly those due to the Fixed-Node approximation) in QMC calculations. Hence it is common practice to optimize trial wave functions by optimizing all the variational parameters in the wave functionswith VMC calculations before running more sophisticated QMC algorithms. There are two major benefits with a reduced number of variational parameters. First, the optimization of trial wave function with VMC is faster with fewer parameters to tune, and second, in a stochastic approach such as QMC, the noise in the remaining parameters can be greatly reduced [21].

In quantum chemistry, a configuration state function (CSF) is a symmetry adapted linear combination of Slater determinants that transforms as an irreducible representation of the spatial and spin symmetry groups of an atom or molecule. In order to build symmetries into wave functions for atomic or molecular systems, one can expand the approximate eigenstates of $\hat{H}$ in terms of CSFs [21]. CSFs for a few cases were worked out very laboriously by hand by Claudia Filippi [6], and then programmed by Silvio a Beccara [1] in Pascal.

A new program is developed implementing the projection method to construct CSFs for atomic systems. The program makes substantial improvements upon Silvio's program in terms of run time and memory consumption by adopting a bit-packed representation and using different algorithms for generating all possible Slater determinants for electron configurations and for determining electron permutations when applying operators to determinants.

3.2 Constructing CSFs by Projection

The projection method is used to construct CSFs in this program, because it is conceptually the simplest among the various quantum chemistry methods [21]. For atoms, CSFs are simultaneous eigenstates of $\hat{L}^2$, $\hat{L}_z$, $\hat{S}^2$, $\hat{S}_z$, $\hat{I}$ where the operators in capital letters stand for multi-electron operators on all electrons in an atom. For example, $\hat{L}$ and $\hat{L}_z$ satisfy the following relations:

$$\hat{L} = \sum_i \hat{l}_i \tag{3.2}$$

$$\hat{L}_z = \sum_i \hat{l}_{zi} \tag{3.3}$$

where $\hat{l}_i$ and $\hat{l}_{zi}$ stand for operators on single electron of index i. The idea of projection method is to start with desired eigenstates of $\hat{L}_z$, $\hat{S}_z$, $\hat{I}$, since single determinants are eigenstates of these operators, and apply projection operators to project away contributions from unwanted $\hat{L}^2$, $\hat{S}^2$ states in order to retain only the eigenstates of desired $\hat{L}^2$, $\hat{S}^2$.

The electron configurations to be considered will serve as a major part of the input for the program. The electron configurations are defined by principal quantum numbers n and angular momentum quantum numbers l of each active electron, leaving out closed-shell electrons. For example, for Be atom, the configurations of interest could be $2s^2$, $2p^2$, $2s3s$, $3d^2$, omitting the core $1s^2$.

For starting states, only $L_z = 0$ and $S_z = S$ (or $S_z = -S$) states are deployed. The main reason for doing so is because the former gives real wave functions which will be very convenient to handle, and the latter yields the smallest number of Slater determinants. It is justifiable to do so because the states with the same L^2 and S^2 values but different L_z and S_z values are degenerate, and thus it is correct to choose whichever eigenstates of $\hat{L}^2$ and $\hat{S}^2$ that are convenient.

With starting states of desired L_z, S_z and I values, the CSFs of an electron configuration are constructed by applying the projection operators $\hat{P}_L\hat{P}_S$ on each of these states. The unormalized projector $\hat{P}_L$ is defined as

$$\hat{P}_L = \prod_{L' \neq L} (\hat{L}^2 - L'(L'+1)) \tag{3.4}$$

where the product is over all possible combined angular momentum quantum numbers except for desired L. Each subtraction in the projector removes the contribution from an unwanted eigenstate of $\hat{L}^2$ whose eigenvalue is $L'(L'+1)$. The $\hat{P}_S$ projector is defined in the same way. Since an atomic system consisting of a finite number of electrons can only be in a finite number of angular momentum or spin states, there are only a finite number of operations in the projection operator to perform. Note that projection operators $\hat{P}_L\hat{P}_S$ don't change the L_z or S_z of the determinants they are acted upon because eigenstates of $\hat{L}^2$ are also eigenstates of of $\hat{L}_z$.

After applying the projection operators, each starting Slater determinant will yield one CSF which is a linear combination of Slater determinants. Some of these CSFs can be linearly dependent, so it is necessary to orthogonalize the CSFs and keep only the linearly independent ones.

In summary, to generate CSFs for a specific electron configuration with the projection method, the program starts from states of desired L_z, S_z and I values, projects with operators $\hat{P}_L\hat{P}_S$, and then orthogonalizes the CSFs in order to keep only the linearly independent ones.

3.3 Implementation of the Projection Method

3.3.1 Input Verification

The electron configurations and the desired ground state, namely desired values of quantum numbers L, S and I are specified in the input file. For example, consider the Carbon atom with four active electrons. For the ground state, which is ${}^3P$ with even parity, a few eligible configurations are $2s^2 2p^2$, $2p^4$, $2s2p^2 3d$, $2s^2 3d^2$. As the electron configurations are manually entered, it is necessary to check that each configuration is valid for the desired values of L, S, I before proceeding.

To check if the configurations can satisfy the desired combined angular momentum quantum number L, the program determines L_{max} and L_{min} for each configuration as follows:

$$L_{max} = \sum_i l_i \tag{3.5}$$

$$L_{min} = \min_{all\ combinations} \left| \sum_i \pm l_i \right| \tag{3.6}$$

where L_{min} is the smallest absolute sum of the angular quantum numbers with all possible sign combinations. Equations 3.5, 3.6 are effectively the same as finding L_{max} and L_{min} recursively in pairs using the fact that possible values of combined L of two angular momentum quantum numbers l_1 and l_2 are $|l_1 - l_2|$, $|l_1 - l_2|+1$, ..., $l_1 + l_2$, and a third angular momentum quantum number is combined by making each of the possible L values as the new l_1, and making l_3 as the new l_2. For example, consider the configuration $2s^1 2p^2 3d^1$. There are four electrons with $l_1 = 0$, $l_2 = 1$, $l_3 = 1$, $l_4 = 2$, then the method above yields $L_{max} = 4$, $L_{min} = 0$.

To verify the configurations can produce states of the desired S value, S_{max} and S_{min} can be determined in the same way as L_{max} and L_{min}, only that the spin

quantum number can take two values $\frac{1}{2}$ and $-\frac{1}{2}$ only. The calculation of S_{min} can be simplified as:

$$S_{min} = \begin{cases} \frac{1}{2}, \text{ for odd number of electrons} \\ 0, \text{ for even number of electrons} \end{cases} \tag{3.7}$$

For the same example $2s^1 2p^2 3d^1$, the correct spin quantum number range is between $S_{max} = 2$ and $S_{min} = 0$.

To check parity or inversion symmetry I, simply sum up the angular quantum number l of all electrons in a configuration, as parity of the sum is the same as the total parity of an atom. For example, both $3p^3$ and $1s2s3s$ can lead to the ^4S excite states of Li, but they will not mix because the former is symmetric under inversion and the latter is anti-symmetric.

3.3.2 Bit-packed Representation of Slater Determinants

After verifying all the input configurations and before generating Slater determinants for each configuration, one major decision to make is how to represent the Slater determinants in the program, as different representations will lead to very different implementations of operators. In Silvio's notes and program, an explicit method[1] is used where a Slater determinant is simply represented by four quantum numbers of electrons, (n, l, l_z, s_z). This method requires $4N$ integers to store a Slater determint, where N is the number of electrons.

In this program, a bit packing method is used where a determinant is represented by the bits of two integers, one for up-spin electrons and one for down-spin. There are two major advantages of using the bit-packed representation.

First, it is very compact, as it requires only 2 integers to store a Slater determinant as opposed to $4N$ integers in the explicit method. Second and more importantly, it makes sorting determinants much more efficient as each determinant is essentially two integers. In comparison, sorting determinants with the explicit representation requires sorting the electrons within each single determinant first, and then sorting the determinants based on $4N$ integers in each determinant. The bit packed representation will bring substantial improvement to speed and memory consumption, because thousands even millions of temporary determinants can be produced during projection in Section 3.3.4. Efficient sorting of determinants is also important in Section 3.3.5, because it is necessary to list all determinants of a CSF in a standard order in order to keep only the linearly independent CSFs.

The bit packing scheme is described as follows. Each bit represents a corresponding state, i.e. a unique combination of (n, l, l_z) values. . A 0-bit stands for unoccupied states and 1-bit stands for occupied ones. A hierarchical encoding is adopted by the program such that:

1. Bits of the same principal quantum number n are grouped together, and bits of smaller n values are always on the right.
2. Bits of the same n value are subsequently grouped by angular momentum quantum number l, and bits of smaller l values are always on the right.
3. Bits of the same n and l values are ordered by magnetic quantum number l_z, with smaller l_z values always on the right.

A segment of the complete encoding scheme is shown in Table 3.1. It can be easily observed that a subshell of angular momentum quantum number l takes

...	13	12	11	10	9	8	7	6	5	4	3	2	1	0	position
	3									2				1	n
	2					1			0	1			0	0	l
	3d					3p			3s	2p			2s	1s	Subshell
	2	1	0	-1	-2	1	0	-1	0	1	0	-1	0	0	l_z
Determinant 1															
		1	0	1	0	0	0	0	0	0	0	0	1	0	↑
													1	0	↓
Determinant 2															
	1	0	0	0	1	0	0	0	0	0	0	0	1	0	↑
													1	0	↓

Table 3.1: Bit-packing scheme and two determinants for ^1P

$2l + 1$ bits, and a shell of principle quantum number n takes n^2 bits. The first five full shells will take 55 bits. As there are 64 bits in an integer on a 64-bit system, this encoding scheme will cover up to $6d$, which is quite enough for ground states of atoms.

With the encoding scheme above, it is easy to obtain the quantum numbers (n, l, l_z) given a bit in a determinant, and easy to obtain a bit-packed determinant given the (n, l, l_z) values of electrons. Examples of a Slater determinants represented with this method are given in Table 3.1.

Any representation of Slater determinants needs to satisfy the two following properties:

1. Any determinant with two identical orbitals equals zero.
2. An odd permutation of orbitals in a determinant produces a minus sign due to standard ordering of orbitals.

These requirements are incorporated in all operations on determinants, and implementation details are discussed in Section 3.3.4.

3.3.3 Generating All Possible Determinants

To generate all $L_z = 0$ and $S_z = S$ states for each configuration, the program needs to identify all possible orbital arrangements for each configuration. To be more specific, the task in this section is to find all valid combinations of (n, l, l_z, s_z) values for each electron that form $L_z = 0$ and $S_z = S$ states given the number of electrons of each subshell (n, l), .

Silvio's program [1] adopted a simple algorithm, which is to list all possible determinants for a given configuration, and then remove states with L_z and S_z values different from the desired ones. This method is feasible, but it involves generating possible determinants of for all L_z and S_z values. As there are many more possible L_z and S_z values than the desired ones $L_z = 0$ and $S_z = S$, and in particular, there are more determinants for other S_z values than $S_z = S$ and $S_z = -S$, the number of determinants generated to be removed will be many more times the number of determinants to keep. This means there is significant and unnecessary time and space consumption to generate the unwanted states using this method.

This problem can be avoided by using a recursive search tree to explore partial candidate determinants by assigning l_z and s_z values to electrons subshell by subshell. A candidate is abandoned if the assignment violates L_z or S_z quotas, otherwise expanded with a subsequent search tree.

The algorithm to construct the search tree is outlined as follows. Each parent node in the tree holds a valid partial determinant. For every subsequent subshell, a child node is created for every possible extension to the partial determinant of the parent node, or in other words assignment of l_z and s_z values

$2s^2$	↑	10		
	↓	10		
		$L_z^{remain} = 0$		
		$S_z^{remain} = 1$		
	$s_z = \frac{1}{2}, l_z = \pm 1$		$s_z = \frac{1}{2}, l_z = \pm 2$	
$2s^2\mathbf{3}d^2$	↑	**1010000000**10	↑	**10001000000**10
	↓	10	↓	10

Table 3.2: Illustration of constructing the search tree.

for electrons of the subshell in consideration. An assignment will be abandoned if it leaves a remaining L_z or S_z quota (L_z^{remain} or S_z^{remain}) that is outside of the $[L_z^{min}, L_z^{max}]$ and $[S_z^{min}, S_z^{max}]$ constraints for the electrons still unassigned, where L_z^{remain} is equal to desired L_z value minus the L_z value of the extended partial determinant, and the same for S_z^{remain}. Note that the $[L_z^{min}, L_z^{max}]$ and $[S_z^{min}, S_z^{max}]$ ranges are calculated for only the subshells whose electrons have not yet been assigned.

For example, given the configuration $2s^2 3d^2$ and desired $L = 1$ and $S = 1$, there are two determinants that satisfy $L_z = 0$, $S_z = 1$ and the Pauli Principle, as shown in Table 3.1. To obtain these two determinants, the algorithm starts with the two s electrons of the first subshell. As there is only one way to arrange them, one node is created with up-spin determinant 10 and down-spin determinant 10, and remaining quotas of $L_z = 0$, $S_z = 1$ for the rest of the electrons. For the rest of the electrons, which happens to be the last subshell, it can be calculated that $L_z^{max} = 3$, $L_z^{min} = -3$, $S_z^{max} = 1$, $S_z^{min} = -1$. L_z^{remain} are S_z^{remain} are within the allowed range, which means the first node can be kept. In the next recursion, two nodes are created for two valid electron assignments for the $3d$ subshell extending the partial determinant, as shown in Table 3.2.

At the end of the algorithm, each leaf of the tree produces a valid determi-

nant that follows the Pauli principle and has $L_z = 0$ and $S_z = S$ for a given configuration.

3.3.4 Projection Operators

In order to construct CSFs for only the desired $\hat{L}^2$, $\hat{S}^2$ states, the main task in this section is to implement projection operators and apply them on the determinants obtained from section 3.3.3 to project away contributions from unwanted $\hat{L}^2$, $\hat{S}^2$ states.

To facilitate distributivity of operators, it is convenient to implement the operators such that the determinants resulting from the operators acting on determinants are appended to an existing linear combination of determinants. A linear combination of determinants are stored as a list of determinants and a list of corresponding coefficients. An operator acting on such a linear combination of determinants is equal to the determinants and corresponding coefficients produced by its operation on each individual determinant appended to the same determinants list and coefficients list. This is necessary because most of the time during the projection process, operators operate on more than one determinant.

It is reasonable to build various operators in a hierarchical way such that complicated multi-particle operators can be calculated by arithmetic operations of simple multi-particle operators, and simple multi-particle operators is subsequently arithmetic operations of single-particle operators. Addition of operators is equivalent to summing the determinants resulting from applying each operator on the original determinants. The product of multiple operators is equivalent to applying the rightmost operator to the original determinants, and

then applying the next adjacent operator on the outcome of the first operation and so on.

The smallest operator components in this program are the single-particle operators including $\hat{l}_z$, $\hat{s}_z$, and raising and lowering operators $\hat{l}_\pm$, $\hat{s}_\pm$.

$$\hat{l}_z = l_z |l, l_z\rangle \tag{3.8}$$

$$\hat{l}_\pm |l, l_z\rangle = \sqrt{l(l+1) - l_z(l_z \pm 1)}\, |l, l_z \pm 1\rangle \tag{3.9}$$

The relations between different operators for angular momentum are listed as follows:

$$\hat{P}_L = \prod_{L' \neq L} (\hat{L}^2 - L'(L'+1)) \tag{3.10}$$

$$\hat{L}^2 = \hat{L}_- \hat{L}_+ + \hat{L}_z^2 + \hat{L}_z \tag{3.11}$$

$$\hat{L}_z = \sum_i \hat{l}_{zi} \tag{3.12}$$

$$\hat{L}_\pm = \sum_i \hat{l}_{\pm i} \tag{3.13}$$

The relations for spin momentum operators are defined the same way.

Each single-electron operator acting on one determinant will generate a multiple of itself (diagonal operator) or a new determinant (off-diagonal operator). It can also generate nothing. Since the original determinants and CSFs are eigenstates of $\hat{l}_z$, operating with $\hat{l}_z$ results in the original list of determinants multiplied by l_z. The same is true for $\hat{s}_z$. One operation of $\hat{l}_\pm$ produces either a zero or a new determinant with a coefficient. The $\hat{l}_+$ operator on a specific electron will shift the corresponding bit in the determinant to the left by one bit if the bit to its left is not set (i.e. occupied), and provided $l_z \neq l$, otherwise the operator will yield a zero. Similarly the $\hat{l}_-$ operator will shift a specific bit to the right by one bit if the bit to its right is not set and $l_z \neq -l$, otherwise it will yield a

zero. Also, similarly, the $\hat{s}_+$ operator moves a 1-bit from down-spin determinant to the same position of the up-spin determinant if the up-spin bit is not already set, otherwise the operation yields a zero. $\hat{s}_-$ is just the other way around.

As the raising and lowering operators in effect move bits around, it is necessary to keep track of permutations when applying more than one single-particle operator. Note that, with the spin-orbital ordering that we have chosen, applying only a series of $\hat{l}_\pm$ operators will never result in a permutation, but applying $\hat{s}_\pm$ operators can. A series of $\hat{l}_\pm$ operators won't change the spin of the bits the operators act upon, but only move bits to their adjacent unoccupied positions within the determinant of the same spin. Any attempt to move a 1-bit to somewhere with another 1-bit in between will result in a zero, which means a bit will never hop over any other 1-bit, and thus no permutations will be created. The case is different for $\hat{s}_\pm$ operators due to the choice of using two integers to store up-spin and down-spin determinants. When a down-spin bit is raised to up-spin, this movement is equivalent to the bit hopping over all 1-bits to the left of its original position in the down-spin determinant and all 1-bits to the right of its destination in the up-spin determinant, if viewing the determinants with the up-spin determinant on the left and the down-spin determinant on the right (see examples in Table 3.3). Note that how the up-spin and down-spin determinants are viewed in relation to each other doesn't make a difference as long as it is consistent throughout the program. From this reasoning, it is obvious that applying the projection operators $\hat{P}_L\hat{P}_S$ may produce odd permutations.

To keep track of permutations, it is crucial to record the order in which the electrons appear in their diagonal in Slater determinants in the original state and in states after applying operators. A very intuitive idea is to have all elec-

		13	12	11	10	9	8	7	6	5	4	3	2	1	0	
		3d					3p			3s	2p			2s	1s	
	Coef	2	1	0	-1	-2	1	0	-1	0	2	0	-1	0	0	
Φ	1		1	0	1	0	0	0	0	0	0	0	0	1	0	↑
			3		*2*									*1*		labels
														1	0	↓
														4		labels
$\hat{S}^+\Phi$	0															
$\hat{S}^-\Phi$	1		1	0	**0**	0	0	0	0	0	0	0	0	1	0	↑
			3											*1*		*labels*
					1	0	0	0	0	0	0	0	0	1	0	↓
					2									*4*		labels
	1				1	0	0	0	0	0	0	0	0	1	0	↑
					2									*1*		*labels*
			1	0	0	0	0	0	0	0	0	0	0	1	0	↓
			3											*4*		*labels*
$\hat{S}^-\Phi$	1		1	0	0	0	0	0	0	0	0	0	0	1	0	↑
					1	0	0	0	0	0	0	0	0	1	0	↓
	-1				1	0	0	0	0	0	0	0	0	1	0	↑
			1	0	0	0	0	0	0	0	0	0	0	1	0	↓

Table 3.3: Illustration of $\hat{S}^+$ and $\hat{S}^-$ operators and bit position labels.

trons, namely 1-bits in the determinant, labeled as 1 through N from right to left, up-spin determinant first. These shall be refereed to as bit position labels for the rest of the thesis. When applying operators, not only are the two-integer bit-packed determinants altered, the relevant bit position labels will also be moved to corresponding positions. The number of permutations after performing the operators will be the same as permutations required to sort the bit position labels. An odd number of permutations will produce a minus sign and can be absorbed to be part of the coefficient of the determinant.

For example, there are four 1-bits in the determinant Φ in Table 3.3. They are labeled 1 to 4 up-spin first and down-spin second from lower orbitals to higher. To apply $\hat{S}^+$ on Φ, each of the four $\hat{s}^+$ results in a zero, because the three 1-bits

in the up-spin determinant are already in $s_z = \frac{1}{2}$, and the 2s bit in the down-spin determinant cannot be raised when 2s in up-spin determinant is already occupied. To apply $\hat{S}^-$ on Φ, the up-spin 3d bits labelled 2 and 3 can be lowered and produces two new determinants as shown in Table 3.3. The labels are moved together with the 1-bits. To absorb the permutations into the sign of coefficients, the bit position labels of first determinant of $\hat{S}^-\Phi$ are (1, 3, 4, 2) which takes two permutations to sort and gives a plus sign to the coefficient 1. The labels of the second determinant are (1, 2, 4, 3) which takes one permutation to sort. An odd permutation produces a minus sign, so the new coefficient becomes -1 as shown in the last block of Table 3.3.

If the number of electrons N is small, a N-length list is a good data structure to implement the bit position labels. The list only needs to be updated when there are spin operators $\hat{s}_\pm$, because as argued above only $\hat{l}_\pm$ operators won't result in any permutation. An update will require position information of the adjacent 1-bit of the destination position in order to move the label to the appropriate position in relation to the other labels, which will take $O(N)$ time.

If the number of electrons N is large, a 2×64 array can be used to implement the bit position labels efficiently by holding positions in the array for 0-bits as well. Each element in the array stores label of the corresponding bit in the determinant, with 1-bits labeled as 1 through N and 0-bits labeled as 0. In this case, the bit position labels records true position of bits instead of merely the relational positions of bits. When an operator is applied to the determinant, a label will be moved in the array. Such an update to the bit position labels will only take $O(1)$ time. This is essentially a tradeoff between time and space. The program written for this part of the thesis adopts the second way of implemen-

tation as described above.

Another ordered vector method is proposed in Silvio's notes [1], which encodes quantum numbers of each electron into an index with a formula

$$\text{index} = -10^4 n - 10^3 l - 10^2(l_z + 2 * l) + 10^5 * (s_z + 1), \tag{3.14}$$

and permutations are counted by sorting the vector of such indices. The bit position labeling method is more convenient for bit-packed representation because the bit positions are directly accessible from the two-integer determinants, while the latter works better for Silvio's non-bit representation where the quantum numbers of electrons are directly accessible.

It is important to count permutations and collect identical determinants after *all* operators have been applied to the determinants. If labels are sorted and minus signs are absorbed by coefficients before all operators are performed, information of movements of the bits will be partially lost because it is very likely a series of operators required to create certain permutations are interrupted. In fact, in intermediate steps, two determinants can only be viewed as *truly* identical if and only if their two-integer determinants as well as their bit position labels are identical. Checking if two determinants are truly identical will be very time consuming, so the better option is to do so after all operations have already been performed.

The final result of the whole projection process will be a d-length list of distinct determinants and a $d \times n_{csf}$ coefficient table, where d is the number of determinants left after sorting the determinants and removing duplicates, and n_{csf} is the number of CSFs, i.e. the number of determinants generated in Section 3.3.3.

3.3.5 Orthogonalization

To retain only the linearly independent CSFs, orthogonalization is required after normalizing the coefficients of the CSFs. The classical Gram-Schmidt algorithm should be sufficient in most cases. Due to numerical round-off errors, coefficients within a tolerance are considered zero. The modified Gram-Schmidt method and the House Holder method can be better options for numerical stability [10] (a House Holder subroutine is provided by Ref. [10]). The program runs with the classical Gram-Schmidt, but a modified Gram-Schmidt subroutine is also ready to use in case of special need.

3.3.6 Checking the CSFs

One internal way to check the correctness of the CSFs generated by this method is to apply $\hat{L}^2$ and $\hat{S}^2$ operators to the CSFs. If these CSFs are indeed eigenstates of $\hat{L}^2$ and $\hat{S}^2$, each CSF should result in the original CSF times $L(L+1)$ and $S(S+1)$ respectively. Comparison between two CSFs should be done after the determinants in each CSF are sorted and duplicates removed, which is also why convenient sorting of determinants provided by bit-packed representation is important, as discussed in Section 3.3.2. This section is used as a quick debugging subroutine for the program and is skipped in real calculations.

3.4 Converting to Real Spherical Harmonics

QMC programs often use real orbitals rather than complex ones. The capibility to convert CSFs in complex spherical harmonics to real ones is being developed as a useful extension to the program.

The spherical harmonics, $Y_{l,l_z}(\theta,\phi)$, are complex functions. They can be converted to real spherical harmonics Z_{l,l_z} using the following transformation:

$$Y_{l,|l_z|} = \frac{1}{\sqrt{2}}(Z_{l,|l_z|} + iZ_{l,-|l_z|}) \tag{3.15}$$

$$Y_{l,-|l_z|} = \frac{1}{\sqrt{2}}(Z_{l,|l_z|} - iZ_{l,-|l_z|}) \tag{3.16}$$

There are different sign conventions for spherical harmonics, and only one is listed here.

For determinants in real spherical harmonics, the same bit-packed representation is adopted as discussed in Section 3.3.2, but for the multiplying coefficient of each determinant, two double precision numbers are needed for the real and imaginary parts. The bit representation will retain the two properties in Section 3.3.2 for determinants of real orbitals. For complex coefficients, it is also reasonable to use the *COMPLEX_NUMBERS* type in Fortran, but it requires extra caution because some functions of Fortran such as *cmplx*() will produce a result with both real and imaginary parts in single precision. Instead one should use dcmplx().

The conversion for determinants from complex spherical harmonics to real can be achieved by substituting each orbital in a determinant as in equations 3.15, 3.16 and expanding the product of all orbitals in the determinant. For

each orbital in the complex harmonics representation, two new determinants in real harmonics representation will be created when expanding the whole product except for two cases. For the case $lz = 0$, the complex spherical harmonic $Y_{l,0}(\theta, \phi)$ is a real function because the phase factor part $e^{il_z\phi}$ becomes 1. For the case where two orbitals with the same n and l quantum numbers, same spin s_z, but opposite l_z are occupied, no new determinants are generated because there is only one valid form of determinant with these two orbitals under the standard ordering, hence any other terms will either be 0 using the first property of determinants, or be transformed to the same determinant using the second property of determinants. For example, consider a determinant $\left[Y_{l,1}Y_{l,-1}\right]$ with two electrons of the same spin s_z, using equations 3.15, 3.16 the expansion goes as follows:

$$\left[Y_{l,1}Y_{l,-1}\right] = \frac{1}{2}\left[\left(Z_{l,1} + iZ_{l,-1}\right)\left(Z_{l,1} - iZ_{l,-1}\right)\right] \tag{3.17}$$

$$= \frac{1}{2}\left(\left[Z_{l,1}Z_{l,1}\right] - i\left[Z_{l,1}Z_{l,-1}\right] + i\left[Z_{l,1}Z_{l,1}\right] + \left[Z_{l,-1}Z_{l,-1}\right]\right) \tag{3.18}$$

$$= \frac{1}{2}\left(0 - i\left[Z_{l,1}Z_{l,-1}\right] - i\left[Z_{l,1}Z_{l,-1}\right] + 0\right) \tag{3.19}$$

$$= -i\left[Z_{l,1}Z_{l,-1}\right] \tag{3.20}$$

where [...] denotes a determinant, and here the writing order within the brackets matters. Going from step 3.19 to 3.20 uses the two properties of determinants. The final result consists of only one determinant $\left[Z_{l,1}Z_{l,-1}\right]$. In more general cases, suppose there are N electrons in the complex determinant but only n electrons whose l_z value is not 0 or if there is no other electron that has the same l value, same spin s_z but opposite magnetic quantum number $-l_z$, then there will be 2^n determinants in the final result in real spherical harmonics.

The above can be derived exactly in the first quantized form by expanding

the actual determinant. The reason why the conversion in second quantized form is a exactly a product of the expansion of each electron is because of the multiplicativity of determinants.

In actual implementation, the conversion from complex harmonics to real can be done by processing each 1-bit in a determinant, namely each orbital that is occupied, one at a time as follows. At the beginning of each iteration, we have an intermediate result which is a linear combination of determinants in real harmonics with complex coefficients, where the number of bits set in each determinant is the number of orbitals already converted. To convert the next orbital, start with the bit position that corresponds to the orbital. If the bit has $l_z = 0$, directly set the same bit in each determinant of the intermediate results, while counting permutations needed to maintain the standard ordering of orbitals in each determinant, and change the sign of the corresponding coefficients if there is an odd permutation. The number of determinants in the intermediate result doesn't change after this iteration. On the other hand, if l_z of the orbital is not zero, multiply the intermediate result with an expansion of this orbital. To do so, append the intermediate result with a copy of itself, then multiply the first half of determinants with $Z_{l,|l_z|}$ and the second half with $\pm iZ_{l,-|l_z|}$. This involves setting an unset bit of each determinant and changing sign of a coefficient should there be an odd permutation. In this case, the number of determinants of real harmonics in the intermediate result doubles. A special case is when the orbital of $(n, l, -l_z)$ is previously converted, going through the same doubling and multiplying process will result in turning half of the determinants to 0, because half of determinants in the intermediate result have the (n,l, l_z) bit set, and the other half have the $(n, l, -l_z)$ bit set. The more efficient way is not to double the determinants, but instead test which bit corresponding to orbitals $(n, l, \pm l_z)$

is set, and only multiply it with the counterpart term in equations 3.15, 3.16 to avoid the extra work of adding and deleting them. In this case, the number of determinants in the intermediate result doesn't change.

For a CSF, the program does the same conversion for each of the determinants (in real harmonics) in the CSF multiplied by corresponding coefficients, and sums up coefficients of identical determinants to obtain a linear combination of unique determinants in real harmonics. As only $L_z = 0$ states are employed in this program, we will obtain only real functions, possibly with a common complex phase that that can be set to 1. In the end, the number of determinants in real harmonics in a CSF could be different from the number of determinants in complex harmonics before conversion. The work for Section 3.4 is still on-going.

CHAPTER 4

CONCLUSION

The thesis presents two projects aimed at improving the accuracy and efficiency of QMC calculations. In the first part of the thesis, a pair-product projector is implemented for a newly proposed PMC method which attempts to ameliorate the Fermionic Sign Problem without making the Fixed-Node approximation. The new projector takes the form of an anti-symmetrized product of pair-projectors of all electron-electron and electron-nucleus pairs in a system. It ameliorates the Fermionic Sign Problem by projecting to an intrinsic fermionic ground state. The fully anti-symmetrized pair-product projector is evaluated in factorial time. Three polynomial-time approximations are implemented and shown to be very close to the fully anti-symmetrized projector.

In the second part of the thesis, a standalone program is developed to generate configuration state functions for atomic systems using the projection method, with the goal of building all symmetries of the system into the trial wave functions used in QMC. The program makes use of a bit-packed representation of Slater determinants, which cuts down the memory cost of a determinant from $4N$ to 2. A search tree algorithm is employed in the program which greatly lowers computational cost to generate Slater determinants for electron configurations. In addition, the program significantly reduces the number of variational parameters in trial wave functions, which will speed up VMC optimization of trial wavefunctions and reduce noise in the remaining parameters.

One extension to the second project, which is still in progress, is to convert the CSFs in complex spherical harmonics to real spherical harmonics in order to align with the input format employed by many QMC calculations.

Another possible future add-on to the program is to provide a convenient way to select important Slater determinants for the trial wave functions, which none of the standard quantum chemistry programs do. This capability is particularly important for strongly correlated systems where it is crucial to include some highly excited determinants in order to have a sufficiently accurate wave function, but it is not viable to include all determinants of that high excitation order because the number of such determinants is enormous [21]. Currently, it is only possible to include the important low-order excitations using the standard quantum chemistry programs.

BIBLIOGRAPHY

[1] Silvio a Beccara. Silvio's unpublisted notes. 2014.

[2] James B Anderson. Quantum chemistry by random walk. h 2 p, h+ 3 d 3 h 1 a 1, h2 3σ+ u, h4 1σ+ g, be 1 s. *The Journal of Chemical Physics*, 65(10):4121–4127, 1976.

[3] JB Anderson and DR Garmer. Validity of random walk methods in the limit of small time steps. *The Journal of chemical physics*, 87(3):1903–1904, 1987.

[4] MICHEL CAFFAREL and ANDREAS SAVIN. Heinz-jurgen flad. *Recent Advances in Quantum Monte Carlo Methods*, 2:73, 1997.

[5] D. M. Ceperley. *Rev. Mod. Phys.*, 67(279), 1995.

[6] Claudia Filippi. Unpublished notes.

[7] RJ Harrison and NC Handy. Quantum monte carlo calculations on be and lih. *Chemical physics letters*, 113(3):257–263, 1985.

[8] W Keith Hastings. Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57(1):97–109, 1970.

[9] Robert Jastrow. Many-body problem with strong forces. *Phys. Rev.*, 98:1479–1484, Jun 1955.

[10] David Ronald Kincaid and Elliott Ward Cheney. *Numerical analysis: mathematics of scientific computing*, volume 2. American Mathematical Soc., 2002.

[11] I. Kylnp. Application of the path integral monte carlo method for solving the quantum statistics of electron-proton systems. 2006.

[12] Junhao Li. Unpublished notes. 2016.

[13] Nicholas Metropolis, Arianna W Rosenbluth, Marshall N Rosenbluth, Augusta H Teller, and Edward Teller. Equation of state calculations by fast computing machines. *The journal of chemical physics*, 21(6):1087–1092, 1953.

[14] MP Nightingale and HWJ Blöte. Gap of the linear spin-1 heisenberg antiferromagnet: A monte carlo calculation. *Physical Review B*, 33(1):659, 1986.

[15] Peter J Reynolds, David M Ceperley, Berni J Alder, and William A Lester Jr. Fixed-node quantum monte carlo for molecules. *The Journal of Chemical Physics*, 77(11):5593–5603, 1982.

[16] PJ Reynolds, RK Owen, and WA Lester Jr. Is there a zeroth order time-step error in diffusion quantum monte carlo? *The Journal of chemical physics*, 87(3):1905–1906, 1987.

[17] Stuart M Rothstein and Jan Vrbik. A greens function used in diffusion monte carlo. *The Journal of chemical physics*, 87(3):1902, 1987.

[18] Sandro Sorella. Green function monte carlo with stochastic reconfiguration. *Physical review letters*, 80(20):4558, 1998.

[19] Sandro Sorella and Luca Capriotti. Green function monte carlo with stochastic reconfiguration: An effective remedy for the sign problem. *Physical Review B*, 61(4):2599, 2000.

[20] DFB Ten Haaf, HJM Van Bemmel, JMJ Van Leeuwen, W Van Saarloos, and DM Ceperley. Proof for an upper bound in fixed-node monte carlo for lattice fermions. *Physical Review B*, 51(19):13039, 1995.

[21] C. J. Umrigar. Notes on constructing configuration state functions by projection. 2017.

[22] CJ Umrigar. Observations on variational and projector monte carlo methods. *The Journal of chemical physics*, 143(16):164105, 2015.

[23] CJ Umrigar, MP Nightingale, and KJ Runge. A diffusion monte carlo algorithm with very small time-step errors. *The Journal of chemical physics*, 99(4):2865–2890, 1993.