Work in progress

by

Alfred Alocias Mariadason

THESIS

for the degree of

MASTER OF SCIENCE

Faculty of Mathematics and Natural Sciences
University of Oslo

May 2018

TODO: write creative commons licence

Abstract

Work in progress

Acknowledgements

I would like to firstly thank my awesome supervisor Morten Hjort-Jensen. Over the course of both my Master's thesis and throughout the bachelor you have always been enthusiastic and supportive of literaly anything I would throw at you. The support I got for my ideas and the immense freedom you gave me in my thesis truly made it enjoyable. Thank you for believing in me and keeping me motivated throughout the thesis.

I would also like to thank Håkon Kristensen for helping me with everything in the thesis. Whitout your help I most certainly wouldn't have come through with the results presented. The help you gave me with the double-well problem and the ideas you have constantly given me have been the greatest help I could possible have had.

The awesome figures made were all thanks to Anders Johansson who introduced me to Asymptote and helped me setup the initial script.

I couldn't possibly finish without mentioning the rest of Computational Physics group with whom I've now spent 4 years with and enjoyed every bit of it. Everything from the game nights, the late night programming(and cursing...) to the endless houres spent eating lunch and playing Mario Cart and Super Smash has without doubt made my days as a student all the better.

CONTENTS

Contents **5**

1 Introduction **11**
1.1 Many-Body Methods 12
1.2 Structure and Goals 13

2 Many-Body Quantum Theory **15**
2.1 The Hamiltonian and the Born-Oppenheimer Approximation 15
2.1.1 Quantum Dot System 16
2.2 Slater Determinant and Permanent 18
2.3 Variational Principle 19
2.4 Energy Functional 20
2.5 Hartree-Fock Theory 20
2.5.5 Assumptions 21
2.5.5 The $\mathscr{J}$ and $\mathscr{K}$ Operators 21
2.5.5 Hartree-Fock Equations 22
2.6 Restricted Hartree-Fock and Roothan-Hall-Equations 23
2.7 Unrestricted Hartree-Fock and Poople-Nesbet-Equations 24
2.8 Convergence, Mixing and the Hartree-Fock Limit 25
2.8.8 DIIS Procedure 25
2.8.8 Hartree-Fock Limit 27
2.9 Quantum Monte Carlo 27
2.9.9 The Variational Principle and Expectation Value of Energy 27
2.9.9 Metropolis-Hastings Algorithm 29
2.9.9 Diffusion Theory and the PDF 30
2.9.9 Importance Sampling 30
2.9.9 The Trial Wavefunction: One-Body 32
2.9.9 The Trial Wavefunction: Splitting the Slater Determinant 32
2.9.9 The Trial Wavefunction: Jastrow Factor 33

2.9.9 Connect the Jastrows . . . 35
2.10 Statistics and Blocking . . . 36
2.11 Density Matrices . . . 36

3 Basis Functions **39**
3.1 Hermite Functions . . . 40
3.2 Gaussian Type Orbitals . . . 40
3.2.2 Hermite-Gaussian Functions . . . 41
3.3 Integral Elements . . . 42
3.3.3 Overlap Distribution . . . 43
3.3.3 Overlap Integral . . . 45
3.3.3 Potential Integral . . . 46
3.3.3 Laplacian Integral . . . 46
3.3.3 Coulomb Potential Integral . . . 47
3.3.3 Coulomb Interaction Integral . . . 49
3.3.3 Recurrence Relation . . . 51
3.3.3 Coulomb Distribution Integral . . . 52
3.4 Double-Well Functions . . . 53
3.4.4 The Eigenvalue problem . . . 53
3.4.4 Choosing the Basis Functions . . . 54
3.4.4 Degeneracy . . . 55
3.5 Summary . . . 55
3.6 Further Work . . . 57

4 Numerical Optimization **59**
4.1 The Optimization Problem . . . 59
4.2 Gradient Descent . . . 60
4.3 Adaptive Stochastic Gradient Descent . . . 64
4.4 Newtons-Method and Quasi-Newton Methods . . . 64
4.5 BFGS Method . . . 67
4.6 Line Search methods . . . 68
4.7 Stochastic-Adaptive-BFGS . . . 69
4.8 Simulated Annealing . . . 70

5 Implementation **75**
5.1 Cartesian Basis . . . 78
5.2 Hartree-Fock . . . 78
5.2.2 Recurrence Relation and Coefficients . . . 79

5.2.2 Parallelization of Two-Body Matrix . . . 80
5.2.2 Tabulation of Two-Body Matrix . . . 82
5.3 Variational Monte Carlo . . . 83
5.3.3 Statistics . . . 84
5.3.3 Thermalisation . . . 84
5.3.3 Slater Optimizations . . . 85
5.3.3 Jastrow Optimizations . . . 85
5.3.3 Optimization For Tabulation . . . 87
5.3.3 Tabulating Hermite Polynomials . . . 87
5.4 Minimization . . . 88
5.5 Auto-Generation . . . 89
5.5.5 Calculation of Hermite Polynomials . . . 89
5.5.5 Double-Well Basis . . . 90
5.6 Verification . . . 90
5.6.6 Hartree-Fock . . . 90
5.6.6 Variational Monte-Carlo . . . 91

6 Results **93**
6.1 Tweaks and Experimentation . . . 93
6.1.1 Hartree-Fock . . . 94
6.1.1 Variational Monte-Carlo . . . 94
6.2 Hartree-Fock . . . 95
6.2.2 Harmonic Oscillator . . . 95
6.2.2 Double-Well . . . 98
6.3 VMC . . . 99
6.3.3 Two-Dimensional Harmonic Oscillator . . . 100
6.3.3 Three-Dimensional Harmonic Oscillator . . . 101
6.3.3 Two-Dimensional Double-Well . . . 102
6.3.3 Three-Dimensional Double-Well . . . 103
6.3.3 Densities . . . 103

A Theory **109**
A.1 Atomic Unit . . . 109
A.2 Interaction-Term in Fock-Operator . . . 110
A.3 Multi-Index Notation . . . 111

B **113**
B.1 Derivatives of Hermite Functions . . . 113

B.2 Derivatives of Padé-Jastrow Function 114
B.3 Derivatives of NQS-Wavefunction 114

C Double-Well **117**

Bibliography **123**

Symbols List

Work in progress

Symbol	Meaning
ψ	Spin-orbital
ϕ	Spacial part of spin-orbital
χ	Basis functions used in basis-expansion
Ψ	Trial wavefunction
Φ	Slater determinant
g	Hermite-Gaussian
G	Gaussian-Type-Orbital (GTO)
Ω	Overlap Distribution
ξ	Recurrence relation for Coulomb integral
$\boldsymbol{x}$	Bold lower case symbol is a *vector*.
$\boldsymbol{X}$	Bold upper case symbol is a *matrix*.
X_{ij}	Element (i, j) of matrix $\boldsymbol{X}$.
$\boldsymbol{x}_{ij}$	Vector difference $\boldsymbol{x}_i - \boldsymbol{x}_j$.
x_{ij}	Length of vector difference $\boldsymbol{x}_{ij}$.
∇_x	Gradient with derivatives of components in $\boldsymbol{x}$
∇_x^2	Laplacian with second derivatives of components in $\boldsymbol{x}$
$\mathscr{H}$	Caligraphed symbol is an operator.
$\{X\}_{k=0}^{N}$	A set with elements $X_0, \ldots, X_N$

Table 1: List of symbols used with explaination.

Source Code

Work in progress

CHAPTER 1

INTRODUCTION

At present time we live in an era of flourishing technological advances and groundbreaking scientific discoveries. A time in which discoveries are made across fields with large cooperative work. With the scale of the problems, so to does the demand for computations in science grow.

The modern computer was introduced to the world somewhere in the middle of the last century and has become a staple tool in scientific work ever since. Within all big breakthroughs in science in the modern days, some kind of computer simulations were made, a numerical experiment was made prior to the actual physical experiments. The reason for this lies in the heart of quantum mechanics, the *Schrödinger equation*. The time dependent version if this equation is

$$i\hbar\frac{\partial}{\partial t}\Psi(\boldsymbol{r},t)=\mathscr{H}\Psi(\boldsymbol{r},t), \tag{1.1}$$

with initial conditions which can be *stationary states*. These are then solved by the *time-independent Schrödinger equation*

$$\mathscr{H}\Psi(\boldsymbol{r})=E\Psi(\boldsymbol{r}) \tag{1.2}$$

Analytical solutions to this equation does exist. However in the case the number of particles increases, meaning the number of parameters within $\boldsymbol{r}$ increases, analytical solutions become hard or even impossible to find and the good-old pen-and-paper approach dwindles in usefulness. The only possibility is then to resort to numerical solutions and the modern computer, although a beast to be tamed, is the best candidate tool.

1.1 Many-Body Methods

In order to tackle the numerical problem that is solving the many-body Schrödinger equation, some of these are the first-order methods Hartree-Fock theory and Density Functional Theory, hierarchical methods such as Configuration Interaction and Coupled Cluster theory and statistical methods like Variational Monte Carlo and Diffusion Monte Carlo.

In case with Hartree-Fock and Density Functional Theory, the accuracy is not desirably high, however they are efficient and are most useful as inputs to the more accurate hierarchical methods[1] or the variational method.

For the configuration interaction methods one either truncates the defined hierarchical structure[2] or uses all contributions in which case the method is known as Full Configuration Interaction. These methods suffer from *exponential scaling* and are therefore much heavier in computational strain than the less accurate methods mentioned. The truncated version however is not size-consistent. The truncated Coupled-Cluster method does not have any size-inconsistencies and achieves polynomial scaling making the Coupled-Cluster method a standard method for when high accuracy is in question.

The statistical methods mentioned employes a different approach by utilizing the statistical nature of the wavefunction in the Schrödinger equation and modeling the entire problem as a stochastic diffusion process. Although the accuracy is on par with the hierarchical methods, the strenuous effort involved in the minimization within the variational method can utterly obliterate ones spirit.

One great property which statistical methods is that they readily, without much effort, gives the beasts known as *density-matrices* or N-body densities[3]. These can give some insight into desirable physics of the system. It is also worth to mention that the variational method is in many approaches an input to the more accurate diffusion method since the latter requires a good initial starting point.

A certain system which techniques from many-body quantum theory can be used is with the *artificial atoms* known as *quantum dots*. Quantum dots were first proposed in the 50's and have since played a big part in some big breakthroughs in science and technology. They are essentially really small semiconductor systems and they have played a central role in nanophysics because of their nice electrical and optical properties. They tend to exist in solid state meaning they can be more easily cooled. The result is that experiments have been conducted with lasers, LEDS, the new generation of transistors in the modern computer and as the ultimate goal; to use quantum dots in logic gates in *quantum computers*. With this quantum-dots have become a

[1] They are hierarchical in the sense that increasingly accurate approximations can be systematically constructed.
[2] In which case it is actually known as Configurations Interaction.
[3] Technically possible to obtain from the hierarchical methods as well, but it is harder.

popular and interesting quantum system to study and a possible route of tackling the problem is with a computational approach. The quantity of interest in the calculations is the *ground-state energy* which the lowest possible eigenvalue of the Hamiltonian.

1.2 Structure and Goals

In this thesis we go into details with the popular Hartree-Fock method and the Variational Monte-Carlo method with the *artificial atoms* known as quantum dots by using single- and double-well potentials as primary systems of study with this chapter being a brief introduction to the structure and goals of the thesis.

The main goal was to make a code-base for large-scale quantum many-body calculations from scratch. Of course there exists many such code-bases and competing with those in terms of scalability and computational efficiency is beyond the scope of this thesis, the choice for a from-scratch approach was made in order to gain some insight into the methods and a better understanding of the programming aspects which would follow. With this in mind, we did however get much inspiration from previous theses and implementations from other sources as well.
The main goals for the thesis were

1. Make a general C++ code for the *Hartree-Fock method* and the *Variational Monte Carlo method* which could take any basis into play with minimal effort.

2. Use the C++ code on different *quantum mechanical many-body systems* and use this as validation of the code and benchmark the code.

3. Build an optimal one-body basis with the Hartree-Fock method and make improvements on this basis with the variational Monte-Carlo method with a *Slater-Jastrow* wavefunction.

With this in mind a general open-source code for the restricted Hartree-Fock method was developed and is open for use in https://github.com/Oo1Insane1oO/HartreeFock. The variational Monte-Carlo method is also open-source and for use in https://github.com/Oo1Insane1oO/VMC. Both repositories each have a directory with tests implemented, see section 5.6 for more information on these tests.
Extending the code to other systems is made easier with python scripts given in the mentioned repositories. A better description of these are given in the repositories themselves and involve auto-generation of abstract wavefunction classes which only need to be filled in with analytical expressions and are otherwise already integrated with the existing code.
The thesis itself is built in four parts

- Quantum Theory: Theory for the implementation.

- Choice of Basis: The form of the wavefunction.
- Implementation: The code with optimizations.
- Results: The resulting data from the simulations.

We basically start of with the background in quantum many-body theory, make a specialization into a set of basis functions, implement these in a general C++ -code with auto-generation of expressions and template classes in Python and present the results from the simulations run with the implementation.

CHAPTER 2

MANY-BODY QUANTUM THEORY

We cannot make any implementations without a solid foundation in theory and as such we will in this chapter reviews the theory regarding basics of *many-body quantum theory* and further deepen into the Hartree-Fock method and the Variational Monte Carlo method.

The reader is referred to [19] for an introductory text on quantum mechanics(for single particles) and also the so-called *Dirac-notation* used throughout the entire chapter. We also write all equations with atomic units, see section A.1 for detail.

The material presented is based on lecture notes [26, 27] written by Morten Hjort-Jensen for the course FYS4411/9411.

2.1 The Hamiltonian and the Born-Oppenheimer Approximation

The task at hand is a given solve the many-body system described by *Schrödinger's* equation

$$\mathscr{H}|\Psi_i\rangle = E_i|\Psi_i\rangle, \tag{2.1}$$

for some state $|\Psi_i\rangle$ with energy E_i. Usually, the desired state is the ground-state energy E_0 of the system, meaning we are primarily interested in the *ground-state* $|\Psi_0\rangle$.
With the goal determined we can define the system to consist of N identical particles[1] with positions $\{\boldsymbol{r}_i\}_{i=0}^{N-1}$ and A nuclei with positions $\{\boldsymbol{R}_k\}_{k=0}^{A-1}$. The Hamiltonian H is then

$$\mathscr{H} = -\frac{1}{2}\sum_i \nabla_i^2 + \sum_{i<j} f\left(\boldsymbol{r}_j, \boldsymbol{r}_j\right) - \frac{1}{2}\sum_k \frac{\nabla_k^2}{M_k} + \sum_{k<l} g\left(\boldsymbol{R}_k, \boldsymbol{R}_l\right) + V\left(\boldsymbol{R}, \boldsymbol{r}\right). \tag{2.2}$$

[1] These are in both atomic physics and in the quantum dot case always fermions or bosons.

The first and second terms represent the kinetic- and inter-particle interaction terms[2] for the N identical particles while the latter three represent kinetic- and interaction terms for the nuclei(with the last one being the nuclei-particle interaction). The constant M_k is the mass of nucleon k and Z_k is the corresponding atomic number.

We assume the nuclei to be much heavier than the identical particles, meaning they move much slower than the electrons, at which the system can be viewed as electrons moving around the vicinity of stationary nuclei. Meaning the kinetic term for the nuclei vanish and the nuclei-nuclei interaction becomes a constant[3]. The approximation we end up with is the so-called *Born-Oppenheimer approximation* and the Hamiltonian is now

$$\mathscr{H} = \mathscr{H}_0 + \mathscr{H}_I, \tag{2.3}$$

where we have split the Hamiltonian in a *one-body* part and a *two-body* or *interaction* parts defined as

$$\mathscr{H}_0 \equiv -\frac{1}{2}\sum_i \nabla_i^2 + V(\boldsymbol{R}, \boldsymbol{r}) \tag{2.4}$$

and

$$\mathscr{H}_I \equiv \sum_{i<j} f\left(\boldsymbol{r}_i, \boldsymbol{r}_j\right). \tag{2.5}$$

2.1.1 Quantum Dot System

For the quantum-dot system we, along with the Born-Oppenheimer approximation, replace the nuclei-particle interaction(also known as a confinement potential) by a different confinement potential such as a harmonic oscillator or a double-well potential. This neat change to the existing atomic Hamiltonian is the reason for why quantum-dots are called artificial atoms.

Confinement Potential

The idea is to first take the *parabolic-dot* which is to use the classical Hamiltonian of a charged electron in an electromagnetic field [30]

$$H = \frac{1}{2m}\left(\boldsymbol{p} - e\boldsymbol{A}\right)^2 + e\phi, \tag{2.6}$$

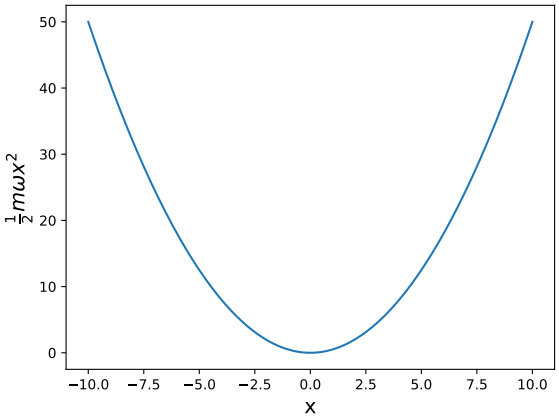
and add the confinement potential and the inter-particle interaction with the Born-Oppenheimer approximation still in effect. Using the derivations in the theses of Patrick Merlot [40] and Yang Min Wang [58] the added energy introduced by the magnetic moment is a constant factor in the inter-particle interaction term which can in effect be ignored in the calculations.

[2] This is usually the well-known Coulomb interaction.

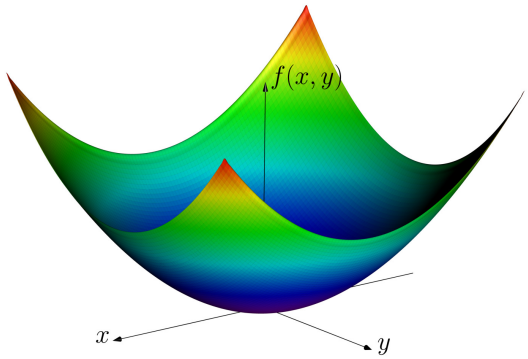
[3] Adding a constant to an operator does not alter the eigenvector, only the eigenvalues by the constant factor[35].

Taking our inspiration from the theses of Merlot and Ming and an article by Simen Kvaal[34], one model for the *confinement* potential is the harmonic oscillator potential, a so-called *parabolic dot.*(see figures 2.1a and 2.1b)

$$V(\boldsymbol{R},\boldsymbol{r}) = \frac{1}{2} m r^2. \tag{2.7}$$



(a) One-dimensional harmonic oscillator

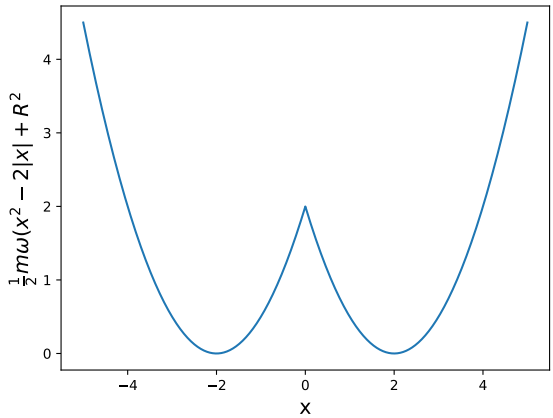


(b) Two-dimensional harmonic oscillator

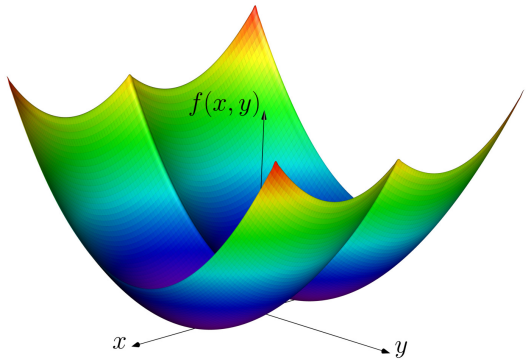
Figure 2.1: Visualization of the harmonic oscillator potential.

A second approach is to use a *double-dot* which is basically just two displaced harmonic oscillators [51, 58]. For simplicity we shift it in x-direction giving the same form used in [28, 40, 58] and [8].(see figures 2.2a and 2.2b)

$$V(\boldsymbol{R},\boldsymbol{r}) = \frac{1}{2} m\omega^2 \left(r^2 - \delta R|x| + R^2\right). \tag{2.8}$$



(a) One-dimensional double-well



(b) Two-dimensional double-well

Figure 2.2: Visualization of the double-well potential.

Inter-particle Interaction Potential

For the inter-particle interaction potential $\mathscr{H}_I$ many approaches exists. For pure bosonic systems for instance a so-called *hard-sphere potential* has been studied[31]. For fermionic systems however the most popular one is the Coulomb repulsion

$$f(\boldsymbol{r}_i, \boldsymbol{r}_j) = \frac{1}{|\boldsymbol{r}_i - \boldsymbol{r}_j|}, \tag{2.9}$$

which is the one used here.

2.2 Slater Determinant and Permanent

Throughout section 2.1 we only referred to the wavefunction Ψ as a state, a function closely connected to the probabilistic nature of the quantum particles. However, we have not given it a form. One possible solution is the *Hartree product* Ψ_{H} defined as

$$\Psi_{\mathrm{H}} = \prod_i \psi_i(\boldsymbol{r}_i). \tag{2.10}$$

With $\{\psi\}_{i=0}^{N}$ being the orbitals which solve the single-particle Schrödinger equation for H_0. The Hartree product is unfortunately a poor choice since it does not solve the H_I part meaning it is not a physically valid solution. This comes from the fact that the product does not take into account the fact that the particles in question are *identical and indistinguishable particles*. Since the particles are identical, switching the labels on the particles shouldn't change the expectation value of some observable. If we run this remark through we end up with the conclusion that the state $|\Psi\rangle$ must be either symmetric or antisymmetric with the symmetric part being the *bosonic state* and antisymmetric being the *fermionic state*. The connection between antisymmetric states and fermions is called the *Pauli exclusion principle*.

The problem with the Hartree product is, with the above sentiment, that it is not symmetric nor antisymmetric. However we can transform it with an operator

$$\mathscr{B} \equiv \frac{1}{N!} \sum_P \sigma_b \mathscr{P} \tag{2.11}$$

where σ_b is defined as

$$\sigma_b \equiv \begin{cases} 1 & b \text{ represents bosonic system} \\ (-1)^p & b \text{ represents fermionic system} \end{cases} \tag{2.12}$$

$\mathscr{P}$ is a permutation operator that switches the labels on particles [4] and p is the parity of permutations. The operator $\mathscr{B}$ has the following properties

- Applying $\mathscr{B}$ to itself doesn't change the operator meaning $\mathscr{B}^2 = \mathscr{B}$.

[4] $P_{ij}\Psi(\boldsymbol{r}_1, \ldots, \boldsymbol{r}_i, \ldots, \boldsymbol{r}_j, \ldots, \boldsymbol{r}_N) = \Psi(\boldsymbol{r}_1, \ldots, \boldsymbol{r}_j, \ldots, \boldsymbol{r}_i, \ldots, \boldsymbol{r}_N)$ [56].

- The Hamiltonian $\mathscr{H}$ and $\mathscr{B}$ *commute*, that is $[\mathscr{B},\mathscr{H}]=[\mathscr{H},\mathscr{B}]=0$.

- $\mathscr{B}$ is *unitary*, which means $\mathscr{B}^\dagger\mathscr{B}=\mathscr{I}$.

The solution Ψ_T to the Schrödinger equation can now be written as

$$\Psi_T(\boldsymbol{r})=\sqrt{N!}\mathscr{B}\Psi_{\mathrm{H}}(\boldsymbol{r}). \tag{2.13}$$

The antisymmetric case of $\mathscr{B}$ results in a *Slater determinant*

$$\Psi_T^{\mathrm{AS}}=\frac{1}{\sqrt{N!}}\sum_P(-1)^p\mathscr{P}_P\prod_i\psi_i, \tag{2.14}$$

while the symmetric case gives the so-called *permanent*[5].

$$\Psi_T^{\mathrm{S}}=\sqrt{\frac{\prod_{i=1}^N n_i!}{N!}}\sum_P\mathscr{P}_P\prod_i\psi_i. \tag{2.15}$$

The extra factor in the symmetric case is to preserve the normalization due to number of particles, the n_i factor is determined by the quantum number for state i.

2.3 Variational Principle

One important remark is that the Slater determinant and the permanent do not solve the interaction part, but only serves as a so-called *ansatz* or guess on the true ground-state wavefunction. This is quite useful due to the *variational principle*.
The Variational principle states that for any normalized function Ψ in Hilbert Space [19] with a Hermitian operator $\mathscr{H}$ the minimum eigenvalue E_0 for $\boldsymbol{H}$ has an upper-bound given by the expectation value of $\mathscr{H}$ in the function Ψ. That is

$$E_0\leq\langle\mathscr{H}\rangle=\frac{\langle\Psi|\mathscr{H}|\Psi\rangle}{\langle\Psi|\Psi\rangle}. \tag{2.16}$$

See [19] for details.

The mentioned ansatz is thereby guaranteed to give energies larger than or equal the true ground state energy meaning a minimization method is sufficient in order to get closer to this minimum.

[5]The permanent is basically just a determinant with all the negative signs replaced by positive ones.

2.4 Energy Functional

We can find a more convenient expression for this energy by using equation (2.13) and equation (2.16). This gives us

$$E[\Psi] = N! \langle \Psi_{\mathrm{H}} | \mathscr{H} \mathscr{B} | \Psi_{\mathrm{H}} \rangle, \tag{2.17}$$

where the hermitian and unitary property of $\mathscr{B}$ as well as the fact that $\mathscr{B}$ and H commute have been used. This energy functional. Applying the $\mathscr{B}$ operator to the Hartree-product, pulling the sum out of the integrals and relabeling with the definitions

$$\begin{aligned}
\langle p | h | q \rangle &\equiv \langle \psi_p(\boldsymbol{r}) | h(\boldsymbol{r}) | \psi_q(\boldsymbol{r}) \rangle = \int \psi_p^*(x) h(\boldsymbol{r}) \psi_q(\boldsymbol{r}) \mathrm{d}r, \\
\langle pq | f | rs \rangle &\equiv \langle \psi_p(\boldsymbol{r}_1)\psi_q(\boldsymbol{r}_2) | f(\boldsymbol{r}_1, \boldsymbol{r}_2) | \psi_r(\boldsymbol{r}_1)\psi_s(\boldsymbol{r}_2) \rangle, \\
&= \int \psi_p(\boldsymbol{r}_1)\psi_q(\boldsymbol{r}_2) f(\boldsymbol{r}_1, \boldsymbol{r}_2) \psi_r(\boldsymbol{r}_1)\psi_s(\boldsymbol{r}_2) \mathrm{d}r,
\end{aligned}$$

yields in[6]

$$E[\Psi] = \langle p | H_0 | p \rangle + \frac{1}{2} \sum_{p,q} [\langle pq | f_{12} | pq \rangle \pm \langle pq | f_{12} | qp \rangle]. \tag{2.18}$$

The first part is written with the assumption that the single-particle wave functions $\{\psi\}$ are orthogonal and the $1/2$ factor in front of the so-called *direct* and *exchange* terms[7] is due to the fact that we count the permutations twice in the sum when applying the $\mathscr{B}$ operator. The sign in the interaction term are chosen as positive for bosonic systems and negative for fermionic systems.

The expression given in equation (2.18) is the functional form we will use to derive the Hartree-fock equations in the following section.

2.5 Hartree-Fock Theory

The Hartree-Fock method is a many-body method for approximating the wavefunction of a stationary many-body quantum state and thereby also obtain an estimate for the energy in this state.

The main idea is to represent the system as a *closed-shell* system and then variationally optimize the Slater determinant [55] and then iteratively solve the arising non-linear eigenvalue problem. This approach is not generally feasable as the size of the system increases, however for a closed-shell system in the quantum dot case it should be enough to build a basis, which is the

[6] Notice also that f_{12} is to imply integrals over two labels r_1 and r_2.

[7] The direct term is just due to inherent behaviour of the charge of the particles (known as the Coulomb repulsion). The exchange term is a direct consequence of the probabilistic nature of the identical particles.

goal here. For more details around the motivation and usages of the Hartree-Fock method. See [55].

In this section we will derive the Hartree-Fock equations, following closely literature [56].

2.5.5 Assumptions

The Hartree-Fock method makes the following assumptions of the system

- *The Born-Oppenheimer approximation* holds.
- All relativistic effects are negligible.
- The wavefunction can be described by a single *Slater determinant*.
- The *Mean Field Approximation* holds.

With these inherent approximations the last one is the most important to take into account as it can cause large deviations from test solutions (analytic solutions, experimental data etc.) since the electron correlations are in reality, for many cases, not negligible. There exists many methods that try to fix this problem, but the *Variational Monte Carlo* (or VMC) is the method for deeper explorations in this Thesis, see section 2.9 for more details.

2.5.5 The $\mathscr{J}$ and $\mathscr{K}$ Operators

Before we begin with the Hartree-Fock equations it is desirable to rewrite the energy function obtained in section 2.4 (form given in equation (2.18)) with two operators $\mathscr{J}$ and $\mathscr{K}$ defined as

$$\mathscr{J} \equiv \sum_k \left\langle \psi_k^* \right| f_{12} \left| \psi_k \right\rangle = \int \psi_k^*(\boldsymbol{r}) f_{12} \psi_k(\boldsymbol{r}) \mathrm{d}r, \tag{2.19}$$

and

$$\mathscr{K} \equiv \sum_k \left\langle \psi_k^* \right| f_{12} \left| \psi \right\rangle = \int \psi_k^*(\boldsymbol{r}) f_{12} \psi(\boldsymbol{r}) \mathrm{d}r. \tag{2.20}$$

The $\mathscr{J}$ operator just gives the simple interaction-term while the $\mathscr{K}$ operator gives the exchange term with the arbitrary (notice no index) $\psi(\boldsymbol{r})$. The energy functional is thus rewritten to

$$E[\Psi] = \sum_i \left\langle \psi_i \left| h + \frac{1}{2}(\mathscr{J} \pm \mathscr{K}) \right| \psi_i \right\rangle, \tag{2.21}$$

where the one-body Hamiltonian is split into a sum of single particle functions as $H_0 = \sum_i h(\boldsymbol{r}_i)$.

2.5.5 Hartree-Fock Equations

As a reminder. The wavefunctions $\{\psi\}$ in equation (2.21) are spin-orbitals with both a spacial part and a spin part. In order to obtain the Hartree-Fock equations we try to minimize the energy functional which in turn gives the ground-state energy for a many-body system. This is done by a variational method.

The first observation to notice is the fact that variations in the spin-orbitals $\{\psi\}$ need to respect the spin-orthogonality relation

$$\langle \psi_i | \psi_j \rangle = \delta_{ij}, \tag{2.22}$$

with δ_{ij} being the well-known Kronecker-delta. This property is essentially a constraint to the minimization problem and the method to be used is the *Lagrange multiplier method* [35], with the *Lagrangian*

$$\mathscr{L} = \delta E[\Psi] - \sum_{ij} \Lambda_{ij} \left[\langle \psi_i | \psi_j \rangle - \delta_{ij} \right]. \tag{2.23}$$

We know then that the minimum is reached when a displacement on the spin-orbitals $\psi_i \rightarrow \psi_i + \delta\psi_i$ results in an energy variation of zero meaning $\delta E[\Psi] = 0$ in the minimum. Which gives the variational problem

$$\sum_i \langle \delta\psi_i | h + \mathscr{J} \pm \mathscr{K} | \psi_i \rangle - \sum_{ij} \Lambda_{ij} \langle \delta\psi_i | \psi_j \rangle + \text{c.c} = 0, \tag{2.24}$$

where c.c is a notation for the complex conjugate of the inner-products on its left-hand side.

The shift in the spin orbitals $\{\delta\psi\}$ is arbitrary and the constraints are symmetric[8] meaning we can with the *Fock-operator*

$$\mathscr{F} \equiv h + \mathscr{J} \pm \mathscr{K}, \tag{2.25}$$

define the following eigenvalue problem

$$\mathscr{F}\psi_i = \sum_j \Lambda_{ij} \psi_j. \tag{2.26}$$

Choosing the Lagrange parameter Λ_{ij} such that $\{\psi\}_{k=1}^N$ forms an orthonormal set for $\mathscr{F}$ with eigenvalues $\{\varepsilon\}_{k=1}^N$. This reduces the eigenvalue equation to

$$\mathscr{F} | \psi \rangle = \boldsymbol{\varepsilon} | \psi \rangle \tag{2.27}$$

with $\boldsymbol{\varepsilon} = (\varepsilon_0, \ldots, \varepsilon_N)$ being the set of eigenvalues of $\mathscr{F}$ meaning we have $N+1$ equations to be solved.

[8] $\langle \psi_i | \psi_j \rangle = \langle \psi_j | \psi_i \rangle^* \Rightarrow \Lambda_{ij} = \Lambda_{ji}^*$

If we only take the N lowest eigenfunctions into the Slater the corresponding eigenenergy is referred to as the *Hartree-Fock energy* and is the estimated ground-state energy which the Hartree-Fock method gives. We can rewrite the energy functional with the eigenenergies to

$$E[\Psi] = \sum_i \left\langle \psi_i \middle| \varepsilon_i - \frac{1}{2}(\mathscr{J} \pm \mathscr{K}) \middle| \psi_i \right\rangle. \tag{2.28}$$

In the derivation of the Hartree-Fock equations we only worked with spin-orbital functions $\{\psi\}$. However it is much more convenient to rewrite these in terms of spatial orbitals $\{\phi\}$ and integrate the spin-dependent part out. There are two ways of doing this and the two different approaches give the so-called *restricted Hartree-Fock* and *unrestricted Hartree-Fock* methods.

2.6 Restricted Hartree-Fock and Roothan-Hall-Equations

The restricted spin-orbitals are paired as[9]

$$\{\psi_{2l-1}, \psi_{2l}\} = \{\phi_l(\boldsymbol{r})\alpha(s), \phi_l(\boldsymbol{r})\beta(s)\} \tag{2.29}$$

with $\alpha(s)$ and $\beta(s)$ being different spin-states (up and down). This pairing of spin-states with the same and same spin orbitals means we can pull the spin degrees of freedom out from the $\mathscr{J}$ and $\mathscr{K}$ operators, reduce the sum to only run over half the states and multiply the entire sum by 2. The result is that the restricted energy-functional reads

$$E[\Psi] = \sum_{i=1}^{N} \varepsilon_i - \sum_{i=1}^{\frac{N}{2}} \langle i \,|\, 2\mathscr{J} \pm \mathscr{K} \,|\, i \rangle. \tag{2.30}$$

Notice that the $\mathscr{K}$ operators sum only runs up to half the number of states.

As the title suggests we are going to end up with a set of equations referred to as the *Roothan-Hall-equations*. We start by first expanding the spacial part $\{\phi\}$ of the spin orbitals $\{\psi\}$ in some known orthonormal basis $\{\chi\}_{i=1}^{L}$

$$\phi_i(\boldsymbol{r}) = \sum_{p=1}^{L} C_{pi}\chi_p(\boldsymbol{r}), \tag{2.31}$$

and introduce the *Fock-matrix* F (associated with the Fock-operator) with elements

$$F_{pq} = h_{pq} + \sum_{pq} \rho_{pq}\left(2D_{prqs} \pm D_{prsq}\right). \tag{2.32}$$

We have here introduced a one-body matrix defined as

$$h_{pq} \equiv \left\langle p \middle| h \middle| q \right\rangle, \tag{2.33}$$

[9]This is specialised for a two-spin system. For a system with more spin-states one needs to either choose different spacial-orbitals or add more such orbitals which effectively changes the energy-levels.

a *density matrix* defined as[10]

$$\rho_{pq} \equiv \sum_{i=1}^{\frac{N}{2}} C_{pi} C_{qi}^{*}, \tag{2.35}$$

and an interaction-matrix D with elements

$$D_{pqrs} \equiv \langle pq \,|\, f_{12} \,|\, rs \rangle \tag{2.36}$$

for convenience. The implicit relabeling of $\chi_p(\boldsymbol{r}) \rightarrow p$ is also present in the above expression for the Fock-matrix. The Hartree-Fock equations (equation (2.27)) are then for the restricted case written as

$$\boldsymbol{F}\boldsymbol{C}_i = \boldsymbol{\varepsilon} S \boldsymbol{C}_i \tag{2.37}$$

with S being the overlap matrix with elements

$$S_{pq} \equiv \langle p \,|\, q \rangle. \tag{2.38}$$

This concludes the essential parts of the derivations of the Hartree-Fock equations. We also present the unrestricted case in section 2.7.

2.7 Unrestricted Hartree-Fock and Poople-Nesbet-Equations

Here we delve into equations for the unrestricted case. The equations are in this case called the *Poople-Nesbet equations*. The derivations are exactly the same as for the restricted case, but without the spin-pairing in equation (2.29). We get two equations, one for the spin-up states and one for the spin-down [56]. The equations are[11]

$$\begin{aligned} \boldsymbol{F}^{+}\boldsymbol{C}^{+} &= \boldsymbol{\varepsilon}\, \boldsymbol{S}\, \boldsymbol{C}^{+} \\ \boldsymbol{F}^{-}\boldsymbol{C}^{-} &= \boldsymbol{\varepsilon}^{-} \boldsymbol{S}\, \boldsymbol{C}^{-} \end{aligned} \tag{2.39}$$

The elements of the Fock-matrices for spin-up and spin-down are

$$F_{pq}^{\pm} = h_{pq} + \sum_{k_{\pm}} \sum_{rs} C_{rk_{\pm}}^{\pm\dagger} C_{sk_{\pm}}^{\pm\dagger} \left[D_{prqs} - D_{prsq} \right] + \sum_{k_{\mp}} \sum_{rs} C_{rk_{\mp}}^{\mp\dagger} C_{sk_{\mp}}^{\mp\dagger} D_{prqs}. \tag{2.40}$$

The Hartree-Fock algorithm thus involves two eigenvalue problems at each iteration. Notice also that the summation index $k_{\pm}$ runs over the spin-up or spin-down states respectively and r and s run over all the spacial basis functions.

[10] This is just the matrix formed by

$$\sum_i |\phi_i\rangle\langle\psi_i| \tag{2.34}$$

which is in quantum mechanics defined as the so-called *density matrix*.

[11] These are just two Roothan-Hall equations.

2.8 Convergence, Mixing and the Hartree-Fock Limit

The Hartree-Fock equations themselves are what we call a system of non-linear equations meaning that they need to be solved iteratively[12]. A simple brute-force way of setting a convergence threshold is to say

$$\left|\boldsymbol{\varepsilon}^{\mathrm{HF}}_{\mathrm{new}} - \boldsymbol{\varepsilon}^{\mathrm{HF}}_{\mathrm{old}}\right| < \text{threshold}. \tag{2.41}$$

The factors determining the achievement of correct convergence to a ground-state is the basis set of choice, but even with a good set the divergence can occur. There are two methods in particular which are made to account for this. The first one is a simple *mixing* [56] of the current density matrix and the one obtained at the previous iteration

$$\boldsymbol{\rho}_{\mathrm{new}} = \alpha\boldsymbol{\rho}_{\mathrm{new}} + (1-\alpha)\boldsymbol{\rho}_{\mathrm{old}}, \qquad 0 < \alpha \leq 1. \tag{2.42}$$

This method stems from the observation that for some systems the Hartree-Fock energies start to oscillate and the above mixing seems to be quite efficient at reducing the oscillations.

2.8.8 DIIS Procedure

The more popular approach to avoid oscillations and achieve(possibly faster) convergence is the DIIS procedure, also known as Pulay mixing [48, 49]. The idea is to define an *extrapolated* error term dependent on the previous M iterations

$$\boldsymbol{Y} = \sum_{m=1}^{M} c_m \boldsymbol{y}_i. \tag{2.43}$$

This extrapolated error is then minimized in a least-squares sense under the constraint that

$$\sum_{m=1}^{M} c_m = 1, \tag{2.44}$$

with the Lagrange multiplier method. This then ends up with the following $m+1$ system of linear equations to be solved for c_i^m and the Lagrange multiplier λ

$$\begin{pmatrix} B_{11} & \dots & B_{1m} & -1 \\ \vdots & \ddots & \vdots & \vdots \\ B_{m1} & \dots & B_{mm} & -1 \\ -1 & \dots & -1 & 0 \end{pmatrix} \cdot \begin{pmatrix} c_1 \\ \vdots \\ c_m \\ \lambda \end{pmatrix} = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ -1 \end{pmatrix} \tag{2.45}$$

The elements of matrix $\boldsymbol{B}$ are

$$B_{ij} = \boldsymbol{e}_i \cdot \boldsymbol{e}_j. \tag{2.46}$$

[12] and pray for convergence...

Following directly the calculations of articles [48, 49] by Pulay, P, the error term can be defined by the Fock matrix as

$$\boldsymbol{e}_i = \boldsymbol{F}_i\boldsymbol{\rho}_i\boldsymbol{S} - \boldsymbol{S}\boldsymbol{\rho}_i\boldsymbol{F}_i, \tag{2.47}$$

where the density matrix $\boldsymbol{\rho}_i$ and Fock-matrix $\boldsymbol{F}_i$ are calculated without any mixing in the given iteration i. The Fock matrix is then calculated as usual in every iteration and then the last m error terms are used to mix an extrapolated Fock-matrix as

$$\boldsymbol{F}_{\text{mix}} = \sum_{m=1}^{M} c_i \boldsymbol{F}_i. \tag{2.48}$$

Here is a figure which describes the algorithm.

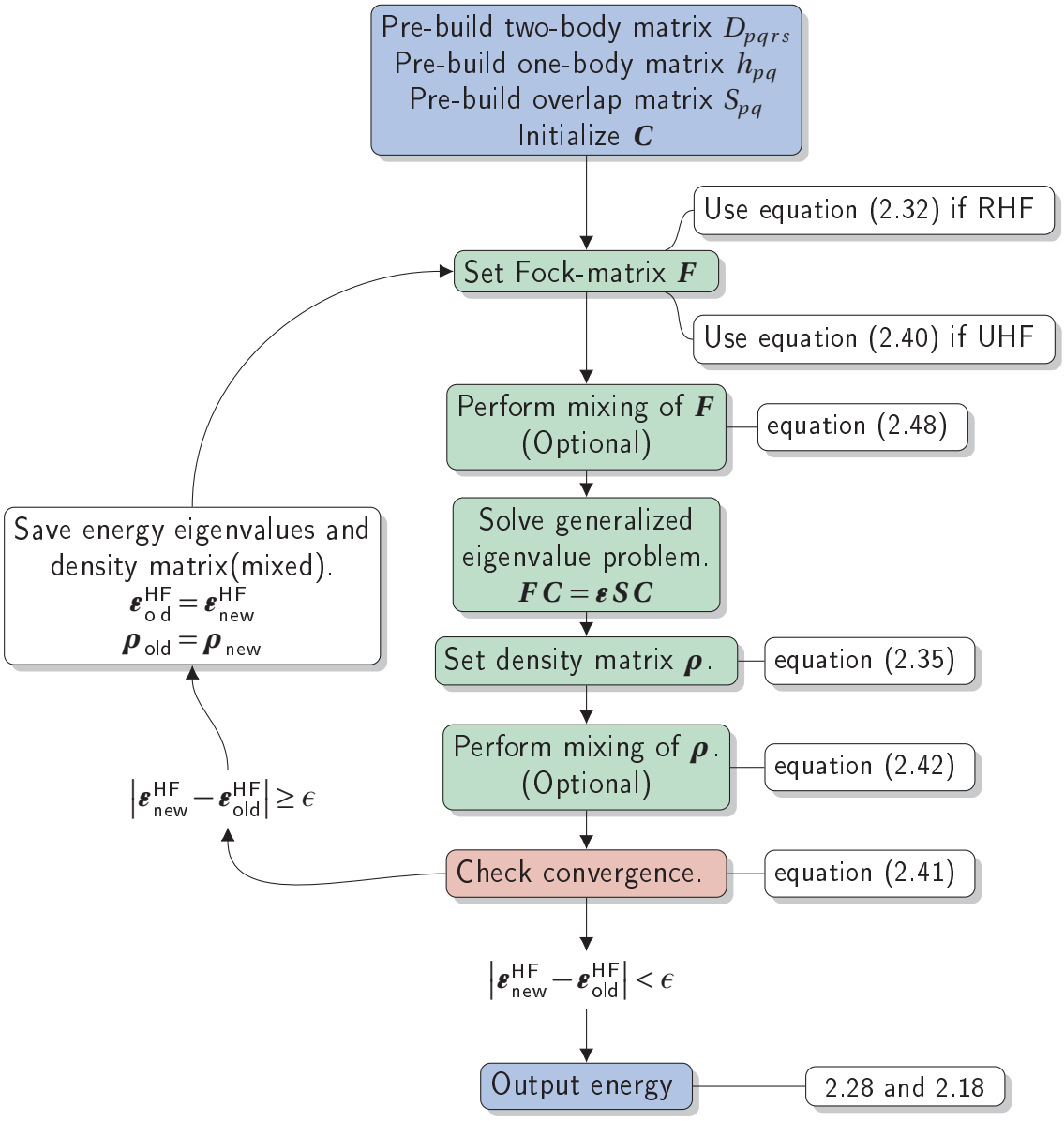


Figure 2.3: Hartree-Fock Algorithm.

2.8.8 Hartree-Fock Limit

From the form of the Hartree-Fock equations and the basis expansion used at the beginning, we see that adding more basis functions gives a better estimate for the ground-state energy. However from chapter 6 we can see that the convergence seems to reach a limit (since the energies with the variational method is lower). This limit is known as the *Hartree-Fock limit* [22] and is the lowest possible energy that can be obtained with a single-determinant wavefunction. At this limit we have the following names for the different quantities involved

- Hartree-Fock energy: Obtained energy
- Hartree-Fock orbitals: Basisfunction, equation (2.31)
- Hartree-Fock wavefunction: Slater determinant of the orbitals

This particular limit is of interest because when it is reached we can be confident that the obtained basis from the Hartree-Fock iteration is the best possible one and further desirable optimizations requires different methods like Coupled-Cluster or Variational Monte Carlo as mentioned.

2.9 Quantum Monte Carlo

Quantum Monte Carlo, or QMC is a method for solving Schrödinger's equation by a statistical approach using so-called *Markov Chain* simulations (also called random walk). The nature of the wave function at hand is fundamentally a statistical model defined on a large configuration space with small areas of non-zero values. The Monte Carlo methods are perfect for solving such a system because of the non-homogeneous distributions of calculation across the space. A standard approach with equal distributions of calculation would then be a waste of computation time.

We will in this chapter address the Metropolis algorithm which is used to create a Markov chain and derive the equations used in the variational method.

The chapter will use *Dirac Notation* [19] and all equations stated assume *atomic units* [2] ($\hbar = m_e = e = 4\pi\varepsilon_0$), see section A.1.

2.9.9 The Variational Principle and Expectation Value of Energy

Given a Hamiltonian $\hat{H}$ and a trial wave function $\Psi_T(\boldsymbol{R};\boldsymbol{\alpha})$, the variational principle [19, 43] states that the expectation value of $\hat{H}$

$$E[\psi_T] = \left\langle \hat{H} \right\rangle = \frac{\left\langle \psi_T \middle| \hat{H} \middle| \psi_T \right\rangle}{\langle \Psi_T | \Psi_T \rangle}, \tag{2.49}$$

is an upper bound to the ground state energy

$$E_0 \leq \left\langle \hat{H} \right\rangle. \tag{2.50}$$

Now we can define our probability distribution as(see section 2.9.9 for a more detailed reasoning)

$$P(\boldsymbol{R}) \equiv \frac{\left|\psi_T\right|^2}{\langle \Psi_T | \Psi_T \rangle}, \tag{2.51}$$

and with a new quantity

$$E_L(\boldsymbol{R};\boldsymbol{\alpha}) \equiv \frac{1}{\Psi_T(\boldsymbol{R};\boldsymbol{\alpha})} \hat{H}\Psi_T(\boldsymbol{R};\boldsymbol{\alpha}), \tag{2.52}$$

the so-called local energy, we can rewrite equation (2.49) as

$$E[\Psi_T(\boldsymbol{R};\boldsymbol{\alpha})] = \langle E_L \rangle. \tag{2.53}$$

The idea now is to find the lowest possible energy by varying a set of parameters $\boldsymbol{\alpha}$. This is done by numerical minimization(see chapter 5). We essentially minimize the expectation value of the energy, see section 2.9.9.

An important property of the local energy is when we differentiate it with respect to one of the variational parameters $\{\alpha\}$ within the context of an expectation value. The result in this case would be zero. This is easily seen by direct calculation in equation (2.54).

$$\begin{aligned}
\left\langle \frac{\partial E_L}{\partial \alpha} \right\rangle &= \int \frac{\left|\psi\right|^2 \frac{\partial}{\partial \alpha}\left[\frac{1}{\psi} H\psi\right]}{\int \left|\psi\right|^2 \mathrm{d}r} \mathrm{d}r \\
&= \int \frac{\left|\psi\right|^2 \frac{\psi^* \frac{\partial}{\partial \alpha}(H\psi) - (H\psi^*)\frac{\partial \psi}{\partial \alpha}}{\left|\psi\right|^2}}{\int \left|\psi\right|^2 \mathrm{d}r} \mathrm{d}r \\
&= \int \frac{\psi^* H \frac{\partial \psi}{\partial \alpha} - \psi^* H \frac{\partial \psi}{\partial \alpha}}{\int \left|\psi\right|^2 \mathrm{d}r} \mathrm{d}r \\
&= 0
\end{aligned} \tag{2.54}$$

We have used the fact that H is not dependent on any variational parameter and used the hermitian properties [19] of H.

This neat result presented in equation (2.54) will show its usefulness in the minimization when derivatives of the expectation value come into play since finding the derivative of the local energy would be much more of a hassle.

And for reference we write

$$\frac{\partial \langle E \rangle}{\partial \alpha} = 2\left(\left\langle \frac{E_L}{\psi} \frac{\partial \psi}{\partial \alpha} \right\rangle - \langle E \rangle \left\langle \frac{1}{\psi} \frac{\partial \psi}{\partial \alpha} \right\rangle \right), \tag{2.55}$$

the derivative with respect to a variational parameters α of the expectation value $\langle E \rangle$. We have also applied equation (2.54) here.

2.9.9 Metropolis-Hastings Algorithm

The Metropolis algorithm bases itself on moves (also called transitions) as given in a Markov process[14, 43]. This process is given by

$$w_i(t+\varepsilon) = \sum_j w_{i\to j} w_j(t) \tag{2.56}$$

where $w(j \to i)$ is just a transition from state j to state i. For the transition chain to reach a desired convergence while reversibility is kept, the well known condition for detailed balance must be fulfilled [52]. If detailed balance is true, then the following relation is true

$$w_i T_{i\to j} A_{i\to j} = w_j T_{j\to i} A_{j\to i} \Rightarrow \frac{w_i}{w_j} = \frac{T_{j\to i} A_{j\to i}}{T_{i\to j} A_{i\to j}}. \tag{2.57}$$

We have here introduced two scenarios, the transition from configuration i to configuration j and the reverse process j to i. Solving the acceptance A for the two cases where the ratio in 2.57 is either 1(in which case the proposed state j is accepted and transitions is made) and when the ratio is less then 1. The Metropolis algorithm would in this case not automatically reject the latter case, but rather reject it with a proposed uniform probability. Introducing now a probability distribution function(PDF) P the acceptance A can be expressed as

$$A_{i\to j} = \min\left(\frac{P_{i\to j}}{P_{j\to i}}\frac{T_{i\to j}}{T_{j\to i}}, 1\right). \tag{2.58}$$

The so-called selection probability T is defined specifically for each problem. For our case the PDF in question is the absolute square of the wave function and the selection T is a Green's function derived in section 2.9.9. The algorithm itself is described in figure 2.4.

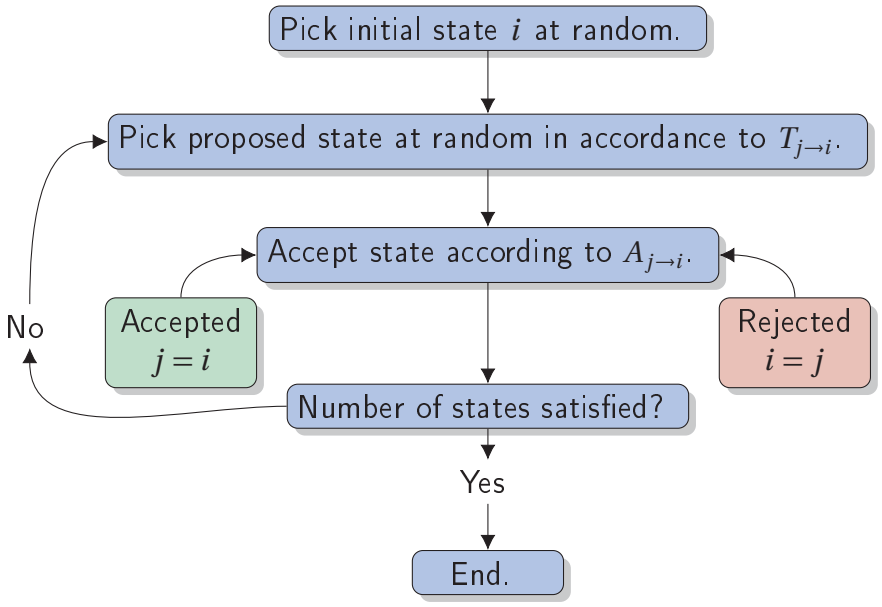


Figure 2.4: Metropolis algorithm.

2.9.9 Diffusion Theory and the PDF

The motivation for the use of diffusion theory is described very well in the Masters thesis of Jørgen Høgberget[28]. The essential results were a Green's function propagator arising from the *short time approximation* and the observation of interest here which has to do with the statistics describing the expectation value, it states that *any* distribution may be applied in calculation, however if we take a close look at the local energy(equation (2.52)) we see that the local energy is not defined at the zeros of $\Psi_T(\boldsymbol{R};\boldsymbol{\alpha})$ for all distributions. This means that an arbitrary PDF does not guarantee generation of points which makes $\psi_T = 0$. This can be overcome by introducing the square of the wave function to be defined as the distribution function as given in equation (2.51). This basically means that when using Quantum Monte Carlo methods, the incorporation of the fact that when an energy is more undefined(meaning $\psi_T \to 0$) the less probable that point is actually means the generation of states in which ψ_T is removed. In the next chapter we will derive the method which includes this observation into play, the so-called *importance sampling*.

2.9.9 Importance Sampling

Using the selection probability mentioned in section 2.9.9, the Metropolis-Hastings algorithm is called an *Importance sampling* because it essentially makes the sampling more concentrated around areas where the PDF has large values.

Because of the intrinsic statistical property of the wave function, quantum mechanical systems can be modelled as a diffusion process, or more specifically, an *Isotropic Diffusion Process* which is essentially just a random walk model. Such a process is described by the Langevin equation with the corresponding Fokker-Planck equation describing the motion of the walkers(particles). See [27, 47, 56] for details. The full derivations presented here follows closely the descriptions in [27, 56]. Let us start off with the Langevin equation

$$\frac{\partial r}{\partial t} = DF(r(t)) + \eta \tag{2.59}$$

and apply Euler's method and obtain the new positions

$$r^{\text{new}} = r^{\text{old}} + DF^{\text{old}}\Delta t + \xi, \tag{2.60}$$

with the r's being the new and old positions in the Markov chain respectively and $F^{\text{old}} = F(r^{\text{old}})$. The quantity D is a diffusion therm equal to $1/2$ due to the kinetic energy(remind of natural units) and ξ is a Gaussian distributed random number with 0 mean and $\sqrt{\Delta t}$ variance.

As mentioned a particle is described by the Fokker-Planck equation

$$\frac{\partial P}{\partial t} = \sum_i D\frac{\partial}{\partial x_i}\left(\frac{\partial}{\partial x_i} - \boldsymbol{F_i}\right)P. \tag{2.61}$$

With P being the PDF(in current case the selection probability) and F being the drift therm. In order to achieve convergence, that is a stationary probability density, we need the left hand side to be zero in equation (2.61) giving the following equation

$$\frac{\partial^2 P}{\partial {x_i}^2} = P\frac{\partial \boldsymbol{F_i}}{\partial x_i} + \boldsymbol{F_i}\frac{\partial P}{\partial x_i}. \tag{2.62}$$

It is apparent that the drift term must be on form

$$\boldsymbol{F} = g(x)\frac{\partial P}{\partial x}. \tag{2.63}$$

Which finally gives

$$\boldsymbol{F} = \frac{2}{\psi_T}\nabla\psi_T. \tag{2.64}$$

This is the so-called *quantum force* responsible for pushing the walkers towards regions where the wave function is large.

The missing part now is to model the selection probability in equation (2.58). Inserting the quantum force into the Focker-Planck equation(equation (2.61)), the following diffusion equation appears

$$\frac{\partial P}{\partial t} = -D\nabla^2 P. \tag{2.65}$$

Applying the *Fourier Transform* to spatial coordinate r in equation (2.65), the equation is transformed to

$$\frac{\partial P(\boldsymbol{s},t)}{\partial t} = -D\boldsymbol{s}^2 P(\boldsymbol{s},t), \tag{2.66}$$

with solution

$$P(\boldsymbol{s},\Delta t) = P(\boldsymbol{s},0)\mathrm{e}^{D\boldsymbol{s}^2\Delta t}. \tag{2.67}$$

Now we need to find the constant $P(\boldsymbol{s},0)$, and as is apparent with $t=0$, we will make use of an initial condition. The initial positions are spread out from origin, that is $D\Delta t\boldsymbol{F}_j$. We can express this with a *Dirac-delta function*[4, 19] giving

$$P(s,0) = \delta\left(\boldsymbol{r}_i - D\Delta t\boldsymbol{F}_j\right). \tag{2.68}$$

Inserting this into equation (2.67) and making the inverse Fourier transform yields the following Green's function as solution

$$P(a,b,\Delta t) = \frac{1}{\sqrt{4\pi D\Delta t}}\exp\left(-\frac{(\boldsymbol{r}_a - \boldsymbol{r}_b - D\Delta t\boldsymbol{F}_b)^2}{4D\Delta t}\right). \tag{2.69}$$

This expression is precisely the selection probability T, Notice also that the indices a and b label a state transition $a \to b$ and not particle indices. The full transition probability needs to be summed over for all particles since we only solved the Focker-Planck equation for 1 particle(since the other solutions are found in the exact same manner). For clarity the full selection probability ratio is

$$\frac{T(b,a,\Delta t)}{T(a,b,\Delta t)} = \sum_i \exp\left(-\frac{\left(\boldsymbol{r}_i^{(b)} - \boldsymbol{r}_i^{(a)} - D\Delta t\boldsymbol{F}_i^{(a)}\right)^2}{4D\Delta t} + \frac{\left(\boldsymbol{r}_i^{(a)} - \boldsymbol{r}_i^{(b)} - D\Delta t\boldsymbol{F}_i^{(b)}\right)^2}{4D\Delta t}\right). \tag{2.70}$$

2.9.9 The Trial Wavefunction: One-Body

The trial wave function is generally an arbitrary choice specific for the problem at hand, however it is in most cases favorable to expand the wave function in the eigenbasis (eigenstates) of the Hamiltonian since they form a complete set. This can be expressed as

$$\Psi_T(\boldsymbol{R};\boldsymbol{\alpha}) = \sum_k C_i \psi_k(\boldsymbol{R};\boldsymbol{\alpha}), \tag{2.71}$$

where the ψ_i's are the eigenstates of the Hamiltonian. The coefficients can be found by any method preferable and is the usual procedure is to use a set of basis functions and then minimize to find the coefficients $\{C\}_{k=1}^{L}$. We use the Hartree-Fock method to minimize in this thesis. The trial wavefunction is also generally expressed as a *Slater determinant* for the fermionic case and a general product for bosonic systems[11, 14, 19, 43] We will explain the fermionic case shortly since it is the main focus here and since the bosonic wavefunction is simple to express. The Slater determinant is expressed as

$$\Phi_T(\boldsymbol{R};\boldsymbol{\alpha}) = \det(\boldsymbol{\Phi}(\boldsymbol{R};\boldsymbol{\alpha}))\xi(s) \tag{2.72}$$

where the *Slater matrix* $\boldsymbol{\Phi}$ has elements

$$\Phi_{ij} = \phi_{n_j}(\boldsymbol{r}_i;\boldsymbol{\alpha}) \tag{2.73}$$

such that each row is evaluated for particle i and each column is for a quantum number n_j dependent on the basis used. The $\xi(s)$ is the spin-dependent part. Notice also that we switched the labeling from Ψ_T to Φ_T. This is to make a distinction between *one-body* and *correlation* terms. The latter will be introduced in section 2.9.9. In this case the *single-particle* functions $\phi_j(r)$ are expanded in some basis(in most cases a Hartree-Fock basis).

2.9.9 The Trial Wavefunction: Splitting the Slater Determinant

An important part of the trial-wavefunction presented here is that the one-body term is *independent* of spin, meaning the Hamiltonian is not explicitly dependent on the spin degrees of freedom. For the case of a Hamiltonian with an inherent spin part, the following splitting of the Slater determinant is not valid. In that case the expectation value(presented in the variational principal in equation (2.16)) would be a product of the expectation value over the spin-independent part of the Hamiltonian and the expectation value over the remaining spin-dependent parts[46]. The results presented here is however for systems of spin-independent systems, and in those cases the spin-part is essentially just another label which can be integrated out(similar to the procedure with the restricted Hartree-Fock method). For the splitting with a *spin-dependent* Hamiltonian see [42] and [26].

The procedure is simply to arrange the basis functions in the following Slater determinant

$$\frac{1}{N!}\begin{pmatrix} \phi_1(\boldsymbol{r}_1)\xi_\uparrow & \dots & \phi_{N/2}(\boldsymbol{r}_1)\xi_\uparrow & \phi_1(\boldsymbol{r}_1)\xi_\downarrow & \dots & \phi_{N/2}(\boldsymbol{r}_1)\xi_\downarrow \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \phi_1(\boldsymbol{r}_N)\xi_\uparrow & \dots & \phi_{N/2}(\boldsymbol{r}_N)\xi_\uparrow & \phi_1(\boldsymbol{r}_N)\xi_\downarrow & \dots & \phi_{N/2}(\boldsymbol{r}_N)\xi_\downarrow \end{pmatrix}. \tag{2.74}$$

This restructuring of the single-particle states implies that

$$\det(\boldsymbol{\Phi}) \propto \det\big(\boldsymbol{\Phi}_\uparrow\big)\det\big(\boldsymbol{\Phi}_\downarrow\big), \tag{2.75}$$

where we have defined

$$\boldsymbol{\Phi}_\uparrow = \begin{pmatrix} \phi_1(\boldsymbol{r}_1) & \dots & \phi_{N/2}(\boldsymbol{r}_1) \\ \vdots & \ddots & \vdots \\ \phi_1(\boldsymbol{r}_{N/2}) & \dots & \phi_{N/2}(\boldsymbol{r}_{N/2}) \end{pmatrix} \tag{2.76}$$

and

$$\boldsymbol{\Phi}_\downarrow = \begin{pmatrix} \phi_1(\boldsymbol{r}_{N/2+1}) & \dots & \phi_{N/2}(\boldsymbol{r}_{N/2+1}) \\ \vdots & \ddots & \vdots \\ \phi_1(\boldsymbol{r}_N) & \dots & \phi_N(\boldsymbol{r}_N) \end{pmatrix}. \tag{2.77}$$

This essentially says that we put the first $N/2$ particle labels in spin-up configurations and the remaining in spin-down configurations and use the same single-particle functions. On a technical note, this rewriting is an approximation. However it can be shown(see [42]) that the expectation value is still the same. The Slater determinant term is now rewritten as

$$\det(\boldsymbol{\Phi}) \approx \det\big(\boldsymbol{\Phi}_\uparrow\big)\det\big(\boldsymbol{\Phi}_\downarrow\big). \tag{2.78}$$

2.9.9 The Trial Wavefunction: Jastrow Factor

As mentioned, we model the wavefunction as a product of the one-body Slater determinant and a correlation part known as a *Jastrow factor*. The Jastrow can have many forms, but is build to exert some key features. It should [31].

- Be dependent on the inter-particle distances.
- Approach unity at large distances.
- Vanish when particles are close to one another.

In this thesis we explore two forms, the popular *Padé-function* and a more recent one built with a specific neural network known as a *Boltzmann Machine*.

Cusp Condition

The derivations for the cusp conditions is described well in the master thesis of Lars Eivind Lervåg [37]. We will here present the result of his for both two and three dimensions. Assuming the Jastrow factor J has the form

$$J = \prod_{i<j} \mathrm{e}^{f\left(r_{ij}\right)}, \tag{2.79}$$

and considering the local energy as two particles i and j approach, the final resulting conditions are known as the cusp conditions. The conditions are given in equation (2.80)

$$\begin{aligned} \left.\frac{\partial f}{\partial r_{ij}}\right|_{r_{ij}=0} &= \frac{1}{D-1}, \qquad \text{anti-parallel spin} \\ \left.\frac{\partial f}{\partial r_{ij}}\right|_{r_{ij}=0} &= \frac{1}{D+1}, \qquad \text{parallel spin.} \end{aligned} \tag{2.80}$$

Padé Function

A popular form of a Jastrow function is the *Padé*, function[13] it is defined as [21, 26]

$$J_{\text{Padé}} = \exp\left(\sum_{i<j} \frac{\sum_l a_{ij}^{(l)} r_{ij}^l}{1+\sum_l \beta_l r_{ij}^l}\right)., \tag{2.81}$$

with the β_l's as variational parameters. This function follows the key features mentioned above for a correlation factor and the restrictions are

$$a_{ij}^{\text{2D}} = \begin{cases} \dfrac{1}{3}, & \text{parallel spin} \\ 1, & \text{anti-parallel spin} \end{cases} \tag{2.82}$$

and

$$a_{ij}^{\text{3D}} = \begin{cases} \dfrac{1}{4}, & \text{parallel spin} \\ \dfrac{1}{2}, & \text{anti-parallel spin} \end{cases} \tag{2.83}$$

on the factor a along with the fact that $a_{kj}^{(l)} = 0$ for $l > 1$, means we may relabel $a_{ij}^{(l)} \to a_{ij}$ and $\beta_l \to \beta$ and reduce the Padé-Jastrow factor to

$$J_{\text{Padé}} = \exp\left(\sum_{i<j} \frac{a_{ij} r_{ij}}{1+\beta r_{ij}}\right). \tag{2.84}$$

For expressions concerning the gradient and Laplacian, see Appendix B.

[13] In which case it is know as the Padé-Jastrow function.

Simple Exponential

With the desirable features for a correlation function in mind, the simplest form for a correlation factor is a scaled exponential evaluated with the inter-particle distance, [14]

$$J = \exp\left(\sum_{i<j} a_{ij} r_{ij}\right). \tag{2.85}$$

The factor a_{ij} is defined the same way as in Padé function. In itself this function is fairly useless as it doesn't model correlations very well for different systems or even the same system with different parameters. However we will present a new type of function in the next section which does not give raise to any cusp conditions, however it has a more flexible form and may in connection with this simple exponential give us all the necessary properties desired in a correlation function.

The Trial Wavefunction: NQS Wavefunction

A more recent and completely different approach is to model the wavefunction with a *neural network* as presented by Carleo and Troyer[7]. The approach used here is based on the *Restricted Boltzmann Machine* as described by Hinton[25]. The form is

$$J_{\text{NQS}} = \exp\left(-\sum_{i=1}^{N}\frac{(\boldsymbol{r}_i - \boldsymbol{a}_i)^2}{2\sigma^2}\right)\prod_{j}^{M}\left(1+\exp\left(b_j + \sum_{i=1}^{N}\sum_{d=1}^{D}\frac{x_i^{(d)} w_{i+d,j}}{\sigma^2}\right)\right). \tag{2.86}$$

The derivatives of this are given in Appendix B.

2.9.9 Connect the Jastrows

In the previous section we presented four functions that can be used to model correlations in quantum systems, however they all had some limitations as well as advantageous properties. The question then remains, can we create a function which exhibits all the nice properties mentioned and none of the limitations? The answer is a bit ambivalent. We don't really know for sure, but a good start is to multiply the functions having the cusp conditions(Padé and exponential) with the NQS wavefunction and use this product as the new jastrow. The motivations for this is simple, we want the cusp conditions introduced by the old popular functions, but also want a more flexible type of function which can take care of other(possible) correlations in the system. How well this actually performed is presented in » REF RESULTS WHEN YOU HAVE SOME (GO GET THEM!) «. We will write out the exact form for clarity.

$$J = \exp\left(\sum_{i<j} a_{ij} r_{ij}\right)\exp\left(-\sum_{i=1}^{N}\frac{(\boldsymbol{r}_i - \boldsymbol{a}_i)^2}{2\sigma^2}\right)\prod_{j}^{M}\left(1+\exp\left(b_j + \sum_{i=1}^{N}\sum_{d=1}^{D}\frac{x_i^{(d)} w_{i+d,j}}{\sigma^2}\right)\right). \tag{2.87}$$

[14] The a's can also be variational parameters.

and

$$J = \exp\left(\sum_{i<j} \frac{a_{ij} r_{ij}}{1+\beta r_{ij}}\right) \exp\left(-\sum_{i=1}^{N} \frac{(\boldsymbol{r}_i - \boldsymbol{a}_i)^2}{2\sigma^2}\right) \prod_j^M \left(1 + \exp\left(b_j + \sum_{i=1}^{N}\sum_{d=1}^{D} \frac{x_i^{(d)} w_{i+d,j}}{\sigma^2}\right)\right). \tag{2.88}$$

2.10 Statistics and Blocking

Since we are dealing with physical systems of probabilistic origin, there is an inherent variance which is dependent on the covariance within the calculated expectation values. Calculating these for all the sample values is out of question as it would take too much time to compute, however the error-estimation given by the variance is not really necessary for the sampling itself[15]. This gives us the possibility of estimating the error post-simulation(as long as the individual sample values are stored).

A nice and quite intuitive method is the resampling method known as *blocking*. The details are explained with article [16] by Flyvbjerg and Petersen. Te idea is to take a set of M samples $\{X\}_{k=1}^{M}$(from i.e a Monte-Carlo simulation) and then dividing that data-set into n blocks such that we have M/n sub-samples. We denote the mean of block n by μ_n and then calculate the variance in each block

$$\Omega_n^2 = \frac{1}{n_b} \sum_{k=n}^{2n} (X_k - \mu_n)^2, \tag{2.89}$$

which gives a set $\{\Omega^2\}_{i=1}^{M/n}$. The estimate for the total variance of the entire sample is then approximated by

$$\sigma^2[X] \approx \frac{1}{n} \sum_i (\Omega_i - \nu)^2, \tag{2.90}$$

where ν is the mean of the set Ω^2.

The observation is then that as we increase the block-size the variance estimate should converge towards the true variance of the system(which includes the covariance). The exact size of the block is undetermined and the simplest approach is to just experiment with the size until convergence is reached.

However the thesis of Marius Jonsson »REF« the block-size is estimated with a neat algorithm which we also use.

2.11 Density Matrices

The Slater determinant presented is quite complicated and the solution-space for the Schrödinger equation with said Slater is too large to extract any useful physical properties directly. Luckily we

[15] It is a nice indicator for the quality of the sample, but one can get away by calculating the simple variance $\sigma^2[X] = \langle X^2 \rangle - \langle X \rangle^2$.

have one part of the physical system we may extract from a variational calculation, the so-called *one-body density*. From quantum mechanics the one-body density for a normalized wavefunction is defined as

$$\Lambda(\boldsymbol{r},\boldsymbol{r}') = N\int \psi^*(\boldsymbol{r},\boldsymbol{r}_2,\dots,\boldsymbol{r}_N)\psi(\boldsymbol{r}',\boldsymbol{r}_2,\dots,\boldsymbol{r}_N)\mathrm{d}\boldsymbol{r}_2,\dots,\mathrm{d}\boldsymbol{r}_N. \tag{2.91}$$

This is just to integrate the wavefunction over all spacial coordinates except $\boldsymbol{r}$ and $\boldsymbol{r}'$. The question now is what exactly does this mean, what information does this integral give us? The one-body density tells us that the element

$$N\int \psi^*(\boldsymbol{r})\psi(\boldsymbol{r})\mathrm{d}\boldsymbol{r}_2,\dots,\mathrm{d}\boldsymbol{r}_N = N\times P\left(\text{Finding a particle within volume d}\boldsymbol{r}\text{ around point }\boldsymbol{r}\right), \tag{2.92}$$

with P denoting a probability. This might not give any direction at first when it comes to extraction of physical properties of the system, however one has to keep in mind that the actual *configuration* of particles within the spacial solution-grid gives a direct insight into the actual physics of the system. A good example of this is the distribution [39]

$$N\int \psi^*(\boldsymbol{r}_1)\psi(\boldsymbol{r}_1)\mathrm{d}\boldsymbol{r}_{12} = 0. \tag{2.93}$$

This integral is a *two-body density* and essentially what we have asked ourselves now is, what is the probability of finding two particles in the same exact state? The answer is apparently *zero* and we see that the all-time famous Pauli-exclusion principle practically appears straight out of the density definition.

With this result as motivation in mind, it is not unreasonable to question if density-matrices might possess more information about the physical nature of the system in question.

Before we tackle the procedure of calculating the densities with the Metropolis algorithm, let us express the full N-body density. The expressions are directly copied from this insightful article by Per-Olov Löwdin [39]. The density matrix of order p is defined as

$$\Lambda^{(p)}\left(\boldsymbol{r}'_1,\dots,\boldsymbol{r}'_p|\boldsymbol{r}_1,\dots,\boldsymbol{r}_p\right) = \binom{N}{p}\int \psi^*(1',2',\dots,p',\dots,N)\psi(1,2,\dots,p,\dots,N)\mathrm{d}\boldsymbol{r}'_{12},\dots,\mathrm{d}\boldsymbol{r}'_p. \tag{2.94}$$

The integrals are over all permutations of $\boldsymbol{r}'_{12}$.

In order to actually find this density with the Metropolis algorithm we need to rewrite it in the same manner as with the local energy, which is to introduce $\left|\psi\right|^2$. For the fermionic case this rewriting gives us the following matrix elements

$$\Lambda_{ij} = N\int \phi_i^*(\boldsymbol{r}_1)\phi_j(\boldsymbol{r}')\frac{\psi(\boldsymbol{r}',\boldsymbol{r}_2,\dots,\boldsymbol{r}_N)}{\psi(\boldsymbol{r}_1,\dots,\boldsymbol{r}_N)}\left|\psi(\boldsymbol{r}_1,\dots,\boldsymbol{r}_N)\right|^2\mathrm{d}\boldsymbol{r}'\mathrm{d}\boldsymbol{r}_1\dots\mathrm{d}\boldsymbol{r}_N, \tag{2.95}$$

where the ϕ's are the single-particle basis functions within the Slater determinant ψ.

The elements of the $\boldsymbol{\Lambda}$ matrix can then be calculated by simple counting. The basic premise is to

essentially create a histogram of particle counts with the bins being various radial distances up to some cutoff $r_{\max}$. Then at every Metropolis step (after the test) we increment each value in the bins array with the number of particles currently within each respective bin. The array is then normalized by the usual total sum (sum of all values in the bins array), but also the radial distance to the power D to the right-most edge of each bin. The reason for this additional normalization is to account for the radial contributions to the configurations, a larger radial distance means that the sparsity of the particle-density increases. We are however only interested in the non-dimensional count within each vicinity meaning we need to divide away the dimensional volume element of each bin. The element for a bin n is found to be[16]

$$V_n^{(D)} = r_{n+1}^D - r_n^D = (r_n + \Delta r)^D - r_n^D. \tag{2.96}$$

To keep this stable we also notice that if the number of bins is satisfyingly large, higher orders of Δr vanish and only the terms with linear factors of Δr give a significant contribution. Notice also that the highest order r_n^D gets canceled out. All this together gives

$$V_n^{(D)} = D\, r_n^{D-1} \Delta r. \tag{2.97}$$

The factor D in front can be dropped as the histogram normalization cancels it anyways.
The resulting one-body densities are presented visually in » REF THESE FIGURES (GO GET EM!) «

[16] Constant proportional factors are dropped due to them being canceled in the histogram normalization(division of the sum of whole bin array).

CHAPTER 3

BASIS FUNCTIONS

Basis sets, the fundamental objects used in describing pretty much everything, anything from a coordinate system to an abstract vector-space or function-space are all part of the same set which we refer to as a basis. It is a set of objects applied to systems in order to essentially change our view of them, to extract desirable properties and to describe them in a rigorous way. In quantum mechanics the abstract vector-space models are used and the functions in mind are the wavefunctions thrown into the Schrödinger equation and the space at hand is the *Hilbert Space*. These functions are the central part of the particle-description in quantum theory, therefore the choice of a basis is of monumental importance when solving quantum mechanical systems. Often[1] the choice of basis functions determine the efficiency and degree of usefulness that a specific method actually holds. This is usually to such degree that a poor choice of basis renders the method in question less useful.

The choice of basis is most definitely true for both the Hartree-Fock and the Variational Monte Carlo methods, and great care has to be taken when introducing such choices. For the variational method, the basis is taken directly from the Hartree-Fock method meaning much of the physical properties desired within the trial wavefunction are within this basis. In which case all except electron-electron correlations[2] are modeled quite well. For the basis in Hartree-Fock the choice needs to be in correspondence with the system of interest.

We will in this chapter mention some popular basis sets used in atomic physics and deepen into a particular set of functions called *Gaussian Type orbitals* and use them with the well known *Hermite functions* and make a detailed calculation of the integral elements used in the Hartree-

[1] Read always

[2] Technically also nucleon-electron correlation as well, but quantum-dot systems do not have any atomic cores present.

Fock method. These integrals have been calculated in polar coordinates directly before[1], the motivation for calculating the integrals elements in Cartesian is due to the double-well potential. Because of the higher order monomials in which the double-well potentials can be defined by the symmetry in polar coordinates is broken making the polar approach less desirable. The procedure in building the basis for the double-well system is also explained in detail. A second motivation for taking the Cartesian route is due to the complex part introduced in the spherical-harmonic part of the polar wavefunction. This complex part has to be evaluated in the Monte-Carlo sampling introducing a whole new level of undesired computational complexity.

3.1 Hermite Functions

The Hermite functions are

$$\phi_n^a(\boldsymbol{r}) \equiv \prod_d N_d H_{n_d}(\sqrt{a}\, x_d) \exp\left(-\frac{a}{2} x_d^2\right) \tag{3.1}$$

with $\boldsymbol{r} = \sum_d \mathbf{e}_d\, x_d$ and the sum over d being the sum over the number of dimensions and H_n is the Hermite polynomial of order n. The integer n_d is the order of the function[3] while the parameter a is a scaling factor and N_d is a normalization factor. These functions show up as eigenfunctions for the *quantum harmonic oscillator system* [19] with the scaling parameter a equal to the oscillator frequency (ω) of the system.

The Hermite functions are orthogonal and give a good ansatz for the VMC method, see section 2.9, with the scaling parameter transformed with an additional variational parameter. The problem with these are however that the matrix-elements introduced in the Hartree-Fock method (section 2.5) are not solvable directly with the Hermite functions as basis functions. However, we can write the Hermite functions in terms of *Hermite-Gaussians*. See section 3.2.2.

3.2 Gaussian Type Orbitals

Gaussian Type Orbitals or GTO's are functions of the form [55]

$$G_n(\boldsymbol{\alpha}; \boldsymbol{r}, \boldsymbol{A}) \equiv \prod_d (x_d - A_d)^{n_d} \mathrm{e}^{-\alpha_d (x_d - A_d)^2}. \tag{3.2}$$

We call α for the scaling parameter and i for the order of the GTO. The variable A is where the function is centered. These are the literature referred to as *primitive Gaussians* and they alone make a poor approximation to the true wave function.

[3] In quantum mechanics the number n is referred to as the principal quantum number and is associated with the energy of a given orbital(energy-level) of the system.

In atomic physics these functions are used directly as a linear combination referred to as *contracted Gaussian functions*. These are written as

$$G_k(x,A) \equiv \sum_{a_k=0}^{P} C_{a_k} G_{a_k}(\alpha_{a_k}; x, A) \tag{3.3}$$

and are fitted[4] to *Slater-type orbitals*, which are functions with decaying properties, or found by some variational method before hand.

These functions are unfortunately not orthogonal, but they behave nicely in integrals and actually give an analytic expression for the interaction-elements mentioned in section 2.5. For this reason we will go forth and use the Gaussian contracted functions and write the Hermite-functions in terms of them, since Hermite-Functions are essentially just a polynomial expansion with the constituents in the expansion being Gaussian functions.

3.2.2 Hermite-Gaussian Functions

The GTO's described can be explicitly expressed in terms of so-called *Hermite-Gaussian functions*[5] defined as

$$g_n(\boldsymbol{\alpha}; \boldsymbol{r}, \boldsymbol{A}) = \prod_d \left(\frac{\partial}{\partial A_d} \right)^{n_d} \mathrm{e}^{-\alpha_d (x_d - A_d)^2}. \tag{3.4}$$

This means also that

$$G_n(\boldsymbol{\alpha}; \boldsymbol{r}, \boldsymbol{A}) = \prod_d (2\alpha_d)^{-n_d} \left(\frac{\partial}{\partial A_d} \right)^{n_d} \mathrm{e}^{-\alpha_d (x_d - A_d)^2}. \tag{3.5}$$

Some properties of the one-dimensional Hermite-Gaussians are given in equation (3.6).

$$\begin{aligned} \frac{\partial g_t}{\partial A_x} &= g_{t+1} \\ g_{t+1} &= \left(\frac{\partial}{\partial A_x} \right)^t \frac{\partial g_0}{\partial A_x} = 2\alpha (x - A_x) \left(\frac{\partial}{\partial A_x} \right)^t g_0 \\ g_{t+1} &= 2\alpha ((x - A_x) g_t - t\, g_{t-1}) \\ (x - A_x) g_t &= \frac{1}{2\alpha} g_{t+1} + t\, g_{t-1} \end{aligned} \tag{3.6}$$

The mentioned rewriting of the Hermite functions in terms of the Hermite-Gaussians is

$$\phi_n^a(\boldsymbol{r}) = \prod_d N_d \sum_{l=0}^{n_d} C_{n_d l}\, g_l \left(\frac{\boldsymbol{\alpha}}{2}, \boldsymbol{r}, \boldsymbol{A} \right) \tag{3.7}$$

[4] Meaning we find the parameters C_{a_k}. and α_{a_k}

[5] The reason for the name is that the polynomial factors generated by the differentiation are precisely the Hermite polynomials.

with $C_{n_d l}$ being the $l'th$ Hermite-coefficient for the Hermite polynomial of order n_d. This means that the matrix-elements in Hartree-Fock theory is just a linear combination over integrals over Hermite-Gaussians. The following section will tackle this in detail for the two-dimensional case. The three-dimensional case is given by [24], see also [23].

3.3 Integral Elements

In the Hartree-Fock scheme we need to calculate the integrals which define the different matrix elements. The integrals to be evaluated are

$$\begin{aligned}
\left\langle i \,\middle|\, j \right\rangle &= \int_{-\infty}^{\infty} g_i(\alpha_i; r, A) g_j(\alpha_j; r, B) \mathrm{d}\boldsymbol{r}, \\
\left\langle i \,\middle|\, x_d^k \,\middle|\, j \right\rangle &= \int_{-\infty}^{\infty} g_i(\alpha_i; r, A) r^k g_j(\alpha_j; r, B) \mathrm{d}\boldsymbol{r}, \\
\left\langle i \,\middle|\, \nabla^2 \,\middle|\, j \right\rangle &= \int_{-\infty}^{\infty} g_i(\alpha_i; r, A) \nabla^2 g_j(\alpha_j; r, B) \mathrm{d}\boldsymbol{r}, \\
\left\langle ij \,\middle|\, \frac{1}{r} \,\middle|\, kl \right\rangle &= \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} g_i(\alpha_i; r_1, A) g_j(\alpha_j; r_2, B) \frac{1}{r_{12}} g_k(\alpha_k; r_1, C) g_l(\alpha_l; r_2, D) \mathrm{d}\boldsymbol{r}_1 \mathrm{d}\boldsymbol{r}_2.
\end{aligned} \tag{3.8}$$

where $\mathrm{d}\boldsymbol{r}$ means integration over all dimensions and with the g's being the usual *Hermite-Gaussians* defined as

$$g_n(\alpha; \boldsymbol{r}, \boldsymbol{A}) = \prod_d (x_d - A_d)^{n_d} \mathrm{e}^{-\alpha(x_d - A_d)^2}. \tag{3.9}$$

We will in this chapter limit ourselves to work with isotropic Gaussians (meaning α_d is the same for all dimensions) as this will yield a simpler closed-form solution to the integrals. For a calculation of integral elements using non-isotropic Gaussian functions see [8]. This article gives a detailed explanation of the calculations of integrals over s-type non-isotropic gaussians. The extension to general Gauss-Hermites can be done in a similar way as presented here, however the proportionality factors involved in the recursive relation are different.

Before we calculate the integrals, let us first express the Hermite-Gaussians in a more convenient way(again see [24] and [23]),

$$g_n(\alpha; \boldsymbol{r}, \boldsymbol{A}) = \prod_d \left(\frac{\partial}{\partial A_{x_d}} \right)^{n_d} \mathrm{e}^{-\alpha(x_d - A_d)^2} = \prod_d \left(\frac{\partial}{\partial A_{x_d}} \right)^{n_d} g_0(\alpha; \boldsymbol{r}, \boldsymbol{A}). \tag{3.10}$$

Since the derivatives are with respect to the center variables we may pull them out of the integration meaning the integrals will only be over s-type Gaussians,

$$g_0(\boldsymbol{\alpha};\boldsymbol{r},\boldsymbol{A}) = \prod_d \mathrm{e}^{-\alpha_d(x_d - A_d)^2}, \tag{3.11}$$

greatly simplifying the calculations. With the mentioned simplification in mind, the problem is to find a closed-form expression for the integrals over s-type Gaussians.

We also introduce the *Gaussian product rule*[6] which basically states that the product of two Gaussian functions is just a third Gaussian centered between the center of the two. The expressions are

$$g_0(\alpha;\boldsymbol{r},\boldsymbol{A})g_0(\beta;\boldsymbol{r},\boldsymbol{B}) = K_{AB}\exp\big(-(\alpha+\beta)\boldsymbol{r}_s^2\big), \tag{3.12}$$

with

$$\begin{aligned} K_{AB} &\equiv \exp\left(-\frac{\alpha\beta}{\alpha+\beta}R_{AB}^2\right), \\ R_{AB} &= |\boldsymbol{A}-\boldsymbol{B}|, \\ \boldsymbol{r_s} &= \boldsymbol{r}-\boldsymbol{P}, \\ \boldsymbol{P} &= \frac{\alpha\boldsymbol{A}+\beta\boldsymbol{B}}{\alpha+\beta}. \end{aligned} \tag{3.13}$$

The vector $\boldsymbol{r}_S$ is just somewhere between $\boldsymbol{A}$ and $\boldsymbol{B}$(We will see that r_S disappears when the integration is done). The Gaussian product rule greatly simplifies the integral over two Gaussian functions since we can just pull K_{AB} out of the integration since it is a constant.

3.3.3 Overlap Distribution

An *overlap distribution* is defined as the product between two Hermite-Gaussian functions,

$$\Omega_{ij} = \prod_d g_{i_d}(x_d,\alpha,A_d)g_{j_d}(x_d,\beta,B_d) = K_{A_dB_d}x_A^{i_d}x_B^{j_d}\mathrm{e}^{-(\alpha+\beta)x_P^2}, \tag{3.14}$$

with the Gaussian product rule. This is just another Gaussian function centered in P, but with the extra monomial factors in $\boldsymbol{r}-\boldsymbol{A}$ and $\boldsymbol{r}-\boldsymbol{B}$. These factors are troublesome when integrating, but with the motivation that Hermite-Gaussians make life simpler, we expand the overlap distribution in a Hermite-Gaussian basis. Following Helgaker [24] and working in one dimension(since Hermite-Gaussians can be split in each respective dimension) we have[7]

$$\Omega_{ij}(\alpha,\beta,\boldsymbol{r},\boldsymbol{A},\boldsymbol{B}) = \sum_{t=0}^{i+j} E_t^{ij} g_t(\alpha,\beta,\boldsymbol{r},\boldsymbol{P}). \tag{3.15}$$

[6]Still in the isotropic case.

[7]The indices i and j are now in 1 dimension!

We stress again that the indices in equation (3.15) and the calculations further are in 1 dimension. Explicit expressions for the coefficients E_t^{ij} are difficult to derive, however a set of recurrence relations are possible to find using the properties of the Hermite-Gaussian functions. Consider firstly the incremented distribution in equation (3.16).

$$\begin{aligned}
\Omega_{i+1,j} &= \sum_{t=0}^{i+1+j} E_t^{i+1,j} g_t \\
&= \left(x_P - \frac{\beta}{\alpha+\beta}(A_x - B_x) \right) \Omega_{ij} \\
&= \sum_{t=0}^{i+j} E_t^{ij} \left(x_P - \frac{\beta}{\alpha+\beta}(A_x - B_x) \right) g_t \\
&= \sum_{t=0}^{i+j} E_t^{ij} \left(\left(t g_{t-1} + \frac{1}{2(\alpha+\beta)} g_{t+1} \right) - \frac{\beta}{\alpha+\beta}(A_x - B_x) g_t \right) \\
&= \sum_{t=0}^{i+j} \left((t+1) E_{t+1}^{ij} + \frac{1}{2(\alpha+\beta)} E_{j-1}^{ij} - \frac{\beta}{\alpha+\beta}(A_x - B_x) \right) g_t,
\end{aligned} \tag{3.16}$$

where we used the properties listed in equation (3.6) (mainly the recurrence) and the expansion in equation (3.15). The incrementation of j follows the exact same derivation and starting coefficient is

$$E_0^{00} = K_{AB}. \tag{3.17}$$

This is found by inserting in $i = j = 0$ into equation (3.16), realizing the exponental is the same for all i and j and using the orthogonality between the Hermite-Gaussians[8]. The recurrent coupled relations for the E's given in here

$$\begin{aligned}
E_t^{i+1,j} &= \frac{1}{2(\alpha+\beta)} E_{t-1}^{ij} - \frac{\beta}{\alpha+\beta}(A_x - B_x) E_t^{ij} + (t+1) E_{t+1}^{ij} \\
E_t^{i,j+1} &= \frac{1}{2(\alpha+\beta)} E_{t-1}^{ij} - \frac{\alpha}{\alpha+\beta}(A_x - B_x) E_t^{ij} + (t+1) E_{t+1}^{ij}.
\end{aligned} \tag{3.18}$$

The overlap distribution can with this be expanded in Hermite-Gaussian functions.

As mentioned, the whole point of using Hermite-Gaussian functions is because of the inherent definition with the derivative with respect to the center-point. This means that for attaining the final expression we must in the end differentiate the expansion coefficients. We state in equation

[8] Another way of expressing this statement is to say that each index t in the sum corresponds to an equation for E_t^{ij}.

(3.19) the coefficients differentiated with respect to the difference variable $Q_x = A_x - B_x$

$$\begin{aligned}
E_0^{00;n+1} &= -\frac{2\alpha\beta}{\alpha+\beta}\left(Q_x E_0^{00;n} + n E_0^{00;n-1}\right) \\
E_t^{i+1,j;n} &= \frac{1}{2(\alpha+\beta)} E_{t-1}^{ij;n} - \frac{\beta}{\alpha+\beta}\left(Q_x E_t^{ij;n} + n E_t^{ij;n-1}\right) + (t+1) E_{t+1}^{ij;n} \\
E_t^{i,j+1;n} &= \frac{1}{2(\alpha+\beta)} E_{t-1}^{ij;n} - \frac{\alpha}{\alpha+\beta}\left(Q_x E_t^{ij;n} + n E_t^{ij;n-1}\right) + (t+1) E_{t+1}^{ij;n} \\
E_t^{ij;n} &\equiv \frac{\partial^n E_t^{ij}}{\partial Q_x^n}
\end{aligned} \tag{3.19}$$

3.3.3 Overlap Integral

With the simplification to s-types and the product rule, the integration may begin. Starting with the overlap integral and using equation (3.15)[9], the results are presented in equation (3.20) below.

$$\begin{aligned}
\left\langle i \,\middle|\, j \right\rangle &= \int_{-\infty}^{\infty} \Omega_{ij}(\alpha_p, \beta_p, \boldsymbol{r}, \boldsymbol{A}, \boldsymbol{B}) \mathrm{d}\boldsymbol{r} \\
&= \sum_p^{i+j} E_p^{ij} \int_{-\infty}^{\infty} g_p(\alpha, \beta, \boldsymbol{r}, \boldsymbol{P}) \mathrm{d}\boldsymbol{r} \\
&= \sum_p^{i+j} E_p^{ij} \int_{-\infty}^{\infty} (\boldsymbol{r} - \boldsymbol{P})^p \mathrm{e}^{-(\alpha_p+\beta_p)(\boldsymbol{r}-\boldsymbol{P})^2} \mathrm{d}\boldsymbol{r} \\
&= \sum_p^{i+j} E_p^{ij} \left(\frac{((-1)^p - 1)\Gamma\left(\frac{p+1}{2}\right)}{2(\alpha_p + \beta_p)^{\frac{p+1}{2}}} \right)^d .
\end{aligned} \tag{3.20}$$

The power d in equation (3.20) comes from splitting the integral into the d dimensions. We are also using the *multi-index notation*[10] (section A.3) and expanding

$$E_n^{ab} = \prod_d E_{n_d}^{a_d b_d}, \tag{3.21}$$

such that the coefficients are all just products over coefficients in each dimension. A substitution in each dimension(i.e $u = x - P_x$) is also used. Notice in addition that the scaling factors α and β are specific for each p because of the overlap expansion.

[9] Also using the following integral $\int_{-\infty}^{\infty} \mathrm{e}^{-\lambda x^2} = \sqrt{\frac{\pi}{\lambda}}, \quad \lambda > 0$. See [32].

[10] The power d also means that with the multi-index notation the entire expression in the paranthesis are to be calculated for each dimension in p and then multiplied together.

3.3.3 Potential Integral

The second integral with the x_d^k part shows up in the external potential part of the Hamiltonian and the expression is(again with the Gaussian product rule) given here.

$$\begin{aligned}
\left\langle i \left| x_d^k \right| j \right\rangle &= \int_{-\infty}^{\infty} x_d^k \Omega_{ij}(\alpha_p, \beta_p, \boldsymbol{r}, \boldsymbol{A}, \boldsymbol{B}) \mathrm{d}\boldsymbol{r} \\
&= \sum_p^{i+j} E_p^{ij} \int_{-\infty}^{\infty} x_d^k (\boldsymbol{r} - \boldsymbol{P})^p \mathrm{e}^{-(\alpha_p+\beta_p)(\boldsymbol{r}-\boldsymbol{P}_p)^2} \mathrm{d}\boldsymbol{r} \\
&= \sum_p^{i+j} E_p^{ij} \left(\frac{((-1)^p - 1)\Gamma\left(\frac{p+1}{2}\right)}{2(\alpha_p+\beta_p)^{\frac{p+1}{2}}} \right)^{D-1} \int_{-\infty}^{\infty} (u + P_d)^k \exp\left(-(\alpha_p+\beta_p)u^2\right) \mathrm{d}u \\
&= \sum_p^{i+j} E_p^{ij} \left(\frac{((-1)^p - 1)\Gamma\left(\frac{p+1}{2}\right)}{2(\alpha_p+\beta_p)^{\frac{p+1}{2}}} \right)^{D-1} \sum_{l=0}^{k} \binom{k}{l} P_d^{k-l} \int_{-\infty}^{\infty} u^l \exp\left(-(\alpha_p+\beta_p)u^2\right) \mathrm{d}u \\
&= \sum_p^{i+j} E_p^{ij} \left(\frac{((-1)^p - 1)\Gamma\left(\frac{p+1}{2}\right)}{2(\alpha_p+\beta_p)^{\frac{p+1}{2}}} \right)^{D-1} \sum_{l=0}^{k} \binom{k}{l} \frac{P_d^{k-l}}{2(\alpha_p+\beta_p)^{\frac{l}{2}}} \left((-1)^l + 1\right) \Gamma\left(\frac{l+1}{2}\right). \qquad (3.22)
\end{aligned}$$

The integrals are split in each dimension and the dimensions not equal to d(in x_d^k) are pulled out and the approach in equation (3.20) is applied. The integral over dimension d is then substituted with $u = x_d + P_d$. In line four $(u + P_d)^k$ is rewritten with the *binomial expansion*[11].

3.3.3 Laplacian Integral

The third integral with the Laplacian operator arises in the kinetic part of the Hamiltonian. This integral can be expressed in terms of equation (3.20), the overlap integral. However the Laplacian

[11] The integral $\int_{-\infty}^{\infty} x^n \mathrm{e}^{-ax^2} \mathrm{d}x = \frac{1}{2} a^{-\frac{n}{2}} \Gamma\left(\frac{k+1}{2}\right), \quad n > -1, n$ even, see [32].

applied to a Hermite-Gaussian has to be calculated first. This is done here

$$\begin{aligned}
\nabla^2 g_i(\alpha;\boldsymbol{r},\boldsymbol{A}) &= \sum_d \frac{\partial^2}{\partial {x_d}^2}\left(\prod_{d'}(x-A_{d'})_{d'}^{i_{d'}}\exp\left(-\alpha(x_{d'}-A_{d'})^2\right)\right)\\
&= \sum_d \prod_{d'\neq d} g_{i,d'}\frac{\partial^2}{\partial {x_d}^2}\left((x_d-A_d)^{i_d}\exp\left(-\alpha(x_d-A_d)^2\right)\right)\\
&= \sum_d \prod_{d'\neq d} g_{i,d'}g_{i,d}\Big(4\alpha^2(x_d-A_d)^{i_d+2}-2\alpha(2i_d+1)(x_d-A_d)^{i_d}\\
&\qquad\qquad + i_d(i_d-1)(x_d-A_d)^{i_d-2}\Big)\\
&= g_i\sum_d\Big(4\alpha^2(x_d-A_d)^{i_d+2}-2\alpha(2i_d+1)(x_d-A_d)^{i_d}\\
&\qquad + i_d(i_d-1)(x_d-A_d)^{i_d-2}\Big).
\end{aligned} \tag{3.23}$$

Now for the integral we have

$$\begin{aligned}
\left\langle i\,\middle|\,\nabla^2\,\middle|\,j\right\rangle &= \int_{-\infty}^{\infty} g_i(\alpha;\boldsymbol{r},\boldsymbol{A})\nabla^2 g_j(\beta;\boldsymbol{r},\boldsymbol{B})\mathrm{d}\boldsymbol{r}\\
&= \sum_d\prod_{d'\neq d}\left\langle i_{d'}\,\middle|\,\sigma_{d'}(S_d(\beta;x-B_d))\,\middle|\,j_{d'}\right\rangle,
\end{aligned} \tag{3.24}$$

with

$$\begin{aligned}
S_d(\alpha;x_d-A_d) &\equiv \left(4\alpha^2(x_d-A_d)^{i_d+2}-2\alpha(2i_d+1)(x_d-A_d)^{i_d}+i_d(i_d-1)(x_d-A_d)^{i_d-2}\right)\\
\sigma_d(S_d) &\equiv \begin{cases}1, & d'\neq d\\ S_d, & d'=d\end{cases}
\end{aligned} \tag{3.25}$$

meaning the Laplacian integral can be expressed in terms of the overlap integrals $\left\langle i\,\middle|\,j+2\right\rangle$, $\left\langle i\,\middle|\,j\right\rangle$ and $\langle i\,|\,i-2\rangle$[12].

3.3.3 Coulomb Potential Integral

Lastly, the troublesome Coulomb integral needs to be calculated. Due to the $1/r$ term we cannot split the integral in each respective dimension as previously. Before we approach the full Coulomb integral, let us calculate a simpler integral over a so-called *Coulomb Potential distribution*

$$\int_{-\infty}^{\infty}\mathrm{e}^{-\alpha(\boldsymbol{r}-\boldsymbol{A})^2}\frac{1}{|\boldsymbol{r}-\boldsymbol{B}|}\mathrm{d}\boldsymbol{r}. \tag{3.26}$$

[12]Since $xg_i = g_{i+1}$ and $\frac{g_i}{x} = g_{i-1}$.

The calculation of this integral will be beneficial for the calculation of the Coulomb integral as we can reuse most of the tricks applied to it. With equation (3.12), the Gaussian product rule, in mind. We rewrite the inverse term with

$$\int_{-\infty}^{\infty} \mathrm{e}^{r_B^2 t^2}\,\mathrm{d}t = \frac{\sqrt{\pi}}{r_B} \Rightarrow \frac{1}{r_B} = \frac{1}{\sqrt{\pi}} \int_{-\infty}^{\infty} \mathrm{e}^{r_B^2 t^2}\,\mathrm{d}t. \tag{3.27}$$

The Coulomb potential integral is thus, with equation (3.12)(again the product rule)

$$\begin{aligned}
\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \mathrm{e}^{-\alpha(\boldsymbol{r}-\boldsymbol{A})^2} \frac{1}{|\boldsymbol{r}-\boldsymbol{B}|}\mathrm{d}\boldsymbol{r} &= \frac{1}{\sqrt{\pi}} \int_{-\infty}^{\infty} \mathrm{e}^{-\alpha(\boldsymbol{r}-\boldsymbol{A})^2} \mathrm{e}^{t^2(\boldsymbol{r}-\boldsymbol{B})^2}\,\mathrm{d}\boldsymbol{r}\,\mathrm{d}t \\
&= \frac{1}{\sqrt{\pi}} \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \mathrm{e}^{-\frac{\alpha t^2}{\alpha+t^2}(\boldsymbol{A}-\boldsymbol{B})^2} \mathrm{e}^{-(\alpha+t^2)r_S^2}\,\mathrm{d}\boldsymbol{r}\,\mathrm{d}t \\
&= \frac{1}{\sqrt{\pi}} \int_{-\infty}^{\infty} \left(\frac{\pi}{\alpha+t^2}\right)^{\frac{d}{2}} \mathrm{e}^{-\frac{\alpha t^2}{\alpha+t^2}(\boldsymbol{A}-\boldsymbol{B})^2}\,\mathrm{d}t.
\end{aligned} \tag{3.28}$$

The integral over t has to be addressed separately for two- and three dimensions. For the three-dimensional case the reader is referred to [24]. Here we will derive a closed-form expression for the two-dimensional case. First let us use the substitution presented in equation (3.29).

$$\begin{aligned}
u &= \frac{t}{\sqrt{\alpha+t^2}} & \lim_{t\to-\infty} u(t) &= -1 \\
t &= u\sqrt{\frac{\alpha}{1-u^2}} & \lim_{t\to\infty} u(t) &= 1 \\
\frac{\mathrm{d}u}{\mathrm{d}t} &= \frac{\alpha}{(\alpha+t^2)^{\frac{3}{2}}}
\end{aligned} \tag{3.29}$$

The integrand (ignoring the exponential part) is then

$$\begin{aligned}
\frac{\mathrm{d}t}{\alpha+t^2} &= \frac{1}{\alpha+t^2}\frac{(\alpha+t^2)^{\frac{3}{2}}}{\alpha}\mathrm{d}u \\
&= \frac{\sqrt{\alpha+t^2}}{\alpha}\mathrm{d}u \\
&= \frac{t}{\alpha u}\mathrm{d}u \\
&= \frac{1}{\alpha u} u\sqrt{\frac{\alpha}{1-u^2}}\mathrm{d}u \\
&= \frac{1}{\sqrt{\alpha}}\sqrt{\frac{1}{1-u^2}}\mathrm{d}u,
\end{aligned} \tag{3.30}$$

giving

$$I_{2\mathrm{D}} = \sqrt{\frac{\pi}{\alpha}} \int_{-1}^{1} \frac{1}{\sqrt{1-u^2}} \mathrm{e}^{-\alpha u^2 |\boldsymbol{A}-\boldsymbol{B}|^2}\,\mathrm{d}u. \tag{3.31}$$

For the three-dimensional case we have a simpler form(easily seen with the same substitution)

$$I_{3D} = \pi \int_{-1}^{1} \mathrm{e}^{-\alpha u^2 |\boldsymbol{A}-\boldsymbol{B}|^2} \mathrm{d}u. \tag{3.32}$$

These integrals must be solved numerically using *Chebyshev-Gauss Quadrature*[59]. One can also rewrite the 2D-integral in terms of the *Modified Bessel function of the first kind* by using $u^2 = 1/2(1-\cos(\theta)$[60]. A take on this resulted nowhere as the closed form expanded itself in an ever-increasing order of polynomial factors with the first and second order of the modified Bessel function of first kind. The 3D-integral can be rewritten with an *incomplete Gamma function* From equation (3.10), the integrals have to be differentiated in order to get the final expression, see section 3.3.3.

3.3.3 Coulomb Interaction Integral

In the previous section an expression for the integral over the Coulomb potential was derived. Before we embark into handling the full Coulomb interaction integral, another exercise with a simpler interaction integral is worthwhile. The integral to study is

$$I' = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \mathrm{e}^{-\alpha(\boldsymbol{r'}-\boldsymbol{A})^2} \mathrm{e}^{-\beta(\boldsymbol{r}-\boldsymbol{B})^2} \frac{1}{|\boldsymbol{r'}-\boldsymbol{r}|} \mathrm{d}\boldsymbol{r}\,\mathrm{d}\boldsymbol{r'}. \tag{3.33}$$

This is an interaction between two distributions. Firstly, notice that we can rewrite the distribution centered in $\boldsymbol{A}$ and the Coulomb interaction with the previously calculated Coulomb potential integral given in equation (3.28). Using I as a general label for equation (3.31) and equation (3.32) we have

$$\begin{aligned} I' &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \mathrm{e}^{-\alpha(\boldsymbol{r'}-\boldsymbol{A})^2} \mathrm{e}^{-\beta(\boldsymbol{r}-\boldsymbol{B})^2} \frac{1}{|\boldsymbol{r'}-\boldsymbol{r}|} \mathrm{d}\boldsymbol{r}\,\mathrm{d}\boldsymbol{r'} \\ &= \int_{-\infty}^{\infty} I_D(\alpha;|\boldsymbol{r}-\boldsymbol{A}|)\mathrm{e}^{-\beta(\boldsymbol{r}-\boldsymbol{B})^2} \mathrm{d}\boldsymbol{r}. \end{aligned} \tag{3.34}$$

Inserting in the definition for u(the substitution in equation (3.29)) and using the extremely useful Gaussian product rule for the product between the distribution centered in $\boldsymbol{B}$ and the exponential factor in I(which is labelled the same for both the two- and three dimensional case) is

$$\mathrm{e}^{-\alpha u^2(\boldsymbol{r}-\boldsymbol{A})^2} \mathrm{e}^{-\beta(\boldsymbol{r}-\boldsymbol{B})^2} = \mathrm{e}^{-(\alpha u^2+\beta)\boldsymbol{r}_S{}^2} \mathrm{e}^{-\frac{\alpha u^2\beta}{\alpha u^2+\beta}(\boldsymbol{A}-\boldsymbol{B})^2}. \tag{3.35}$$

Inserting this into equation (3.34) with equation (3.36)

$$\upsilon \equiv \begin{cases} \sqrt{\frac{\pi}{\alpha}}\sqrt{\frac{1}{1-u^2}}, & \text{2D} \\ \pi, & \text{3D} \end{cases} \tag{3.36}$$

we have

$$
\begin{aligned}
I' &= \int_{-\infty}^{\infty}\int_{-1}^{1} \upsilon \mathrm{e}^{-(\alpha u^2+\beta)\boldsymbol{r}\boldsymbol{s}^2} \mathrm{e}^{-\frac{\alpha u^2\beta}{\alpha u^2+\beta}(\boldsymbol{A}-\boldsymbol{B})^2} \mathrm{d}\boldsymbol{r}\mathrm{d}u \\
&= \int_{-1}^{1} \upsilon \left(\frac{\pi}{\alpha u^2+\beta}\right)^{\frac{d}{2}} \mathrm{e}^{-\frac{\alpha u^2\beta}{\alpha u^2+\beta}(\boldsymbol{A}-\boldsymbol{B})^2} \mathrm{d}u.
\end{aligned} \tag{3.37}
$$

Specializing to the two-dimensional case and using the substitution in equation (3.38),

$$
\begin{aligned}
\upsilon &= u\sqrt{\frac{\alpha+\beta}{\alpha u^2+\beta}} \\
\frac{\mathrm{d}\upsilon}{\mathrm{d}u} &= \frac{\beta\sqrt{\alpha+\beta}}{(\alpha u^2+\beta)^{3/2}} \\
u &= \upsilon\sqrt{\frac{\beta}{\alpha+\beta-\alpha\upsilon^2}} \\
\upsilon(-1) &= -1 \\
\upsilon(1) &= 1
\end{aligned} \tag{3.38}
$$

the integrand is

$$
\begin{aligned}
\frac{1}{\sqrt{1-u^2}}\frac{1}{\alpha u^2+\beta}\mathrm{d}u &= \frac{1}{\sqrt{1-u^2}}\frac{1}{\alpha u^2+\beta}\frac{(\alpha u^2+\beta)^{3/2}}{\beta\sqrt{\alpha+\beta}}\mathrm{d}\upsilon \\
&= \frac{1}{\sqrt{1-u^2}}\frac{u}{\beta\upsilon}\mathrm{d}\upsilon \\
&= \sqrt{\frac{\alpha+\beta-\alpha\upsilon^2}{(\alpha+\beta)(1-\upsilon^2)}}\frac{1}{\beta\upsilon}\upsilon\sqrt{\frac{\beta}{\alpha+\beta-\alpha\upsilon^2}}\mathrm{d}\upsilon \\
&= \frac{1}{\sqrt{\beta(\alpha+\beta)}}\frac{1}{\sqrt{1-\upsilon^2}}\mathrm{d}\upsilon.
\end{aligned} \tag{3.39}
$$

Meaning we finally have

$$
I'_{2\mathrm{D}} = \frac{\pi^{\frac{3}{2}}}{\sqrt{\alpha\beta(\alpha+\beta)}}\int_{-1}^{1}\frac{1}{\sqrt{1-\upsilon^2}}\mathrm{e}^{-\frac{\alpha\beta}{(\alpha+\beta)}\upsilon^2(\boldsymbol{A}-\boldsymbol{B})^2}\mathrm{d}\upsilon. \tag{3.40}
$$

This expression will be of great use when calculating the final full interaction integral over the Coulomb distribution. The next section will derive the mentioned recurrence relation before the full Coulomb integral is calculated

3.3.3 Recurrence Relation

Following directly from [24], we proceed with finding a similar recurrence relation for the derivatives. We define a function containing the integral which needs to be solved numerically, namely

$$\zeta_n(x) \equiv \int_{-1}^{1} \frac{u^{2n}}{\sqrt{1-u^2}} \mathrm{e}^{-u^2 x} \mathrm{d}u. \tag{3.41}$$

This function also has the relation

$$\frac{\partial \zeta_n}{\partial x} = -\zeta_{n+1}. \tag{3.42}$$

The Coulomb potential integral is then, in terms of $\zeta_n(x)$ as

$$I_{2\mathrm{D}} = \sqrt{\frac{\pi}{\alpha}} \zeta_0\left(\alpha R_{AB}^2\right) \tag{3.43}$$

and the first derivative with respect to A_x is

$$\begin{aligned} \frac{\partial I_{2\mathrm{D}}}{\partial A_x} &= \sqrt{\frac{\pi}{\alpha}} \frac{\partial}{\partial A_x} \zeta_0\left(\alpha R_{AB}^2\right) \\ &= -2\sqrt{\alpha\pi} X_{AB} \zeta_1\left(\alpha R_{AB}^2\right). \end{aligned} \tag{3.44}$$

With this we define an auxiliary function given in equation (3.45)

$$\begin{aligned} \xi_{tu}^n &= \left(\frac{\partial}{\partial A_x}\right)^t \left(\frac{\partial}{\partial A_y}\right)^u \xi_{00}^n \\ \xi_{00}^n &= (-2)^n \alpha^{n-\frac{1}{2}} \zeta_n\left(\alpha R_{AB}^2\right) \end{aligned} \tag{3.45}$$

and take a look at the incrementation of t

$$\begin{aligned} \xi_{t+1,u}^n &= \left(\frac{\partial}{\partial A_x}\right)^t \left(\frac{\partial}{\partial A_y}\right)^u \frac{\partial \xi_{00}^n}{\partial A_x} \\ &= \left(\frac{\partial}{\partial A_x}\right)^t X_{AB} \xi_{0u}^{n+1}. \end{aligned} \tag{3.46}$$

Using the commutator between $\partial_{A_x}^t$ [13]

$$\begin{aligned} \frac{\partial^t}{\partial {A_x}^t} X_{AB} &= \left[\frac{\partial^t}{\partial {A_x}^t}, X_{AB}\right] + X_{AB} \frac{\partial^t}{\partial {A_x}^t} \\ &= t \frac{\partial^{t-1}}{\partial {A_x}^{t-1}} + X_{AB} \frac{\partial^t}{\partial {A_x}^t}, \end{aligned} \tag{3.47}$$

the final form of equation (3.46) is[14] given in equation (3.48).

$$\begin{aligned} \xi_{t+1,u}^n &= t\,\xi_{t-1,u}^{n+1} + X_{AB} \xi_{t,u}^{n+1} \\ \xi_{t,u+1}^n &= u \xi_{t,u-1}^{n+1} + Y_{AB} \xi_{t,u}^{n+1} \end{aligned} \tag{3.48}$$

[13] $\partial_x^t = \frac{\partial^t}{\partial x^t}$

[14] The incrementation of u is derived in the same way as we did with t.

With this all Hermite integrals of order $t+u \leq N$ can be calculated from ζ of order $n \leq N$, the only difference being X_{AB} and Y_{AB}. The Coulomb interaction integral (equation (3.40)) follows this exact recurrence relation, but with a different proportionality factor $\alpha\beta/(\alpha+\beta)$. We will write it out in equation (3.49) for the sake of clarity.

$$\begin{aligned} \frac{\partial I'_{2\mathrm{D}}}{\partial A_x} &= -\frac{2\alpha\beta}{\alpha+\beta} X_{AB} \zeta_1 \left(\frac{\alpha\beta}{\alpha+\beta} R_{AB}^2 \right) \\ \xi_{00}^n &= \left(\frac{-2\alpha\beta}{\alpha+\beta} \right)^n \zeta_n \left(\frac{\alpha\beta}{\alpha+\beta} R_{AB}^2 \right). \end{aligned} \tag{3.49}$$

Notice that the only difference between the obtained recurrence relations and the ones obtained by Helgaker[23] is in equation (3.45) and equation (3.49). Other than this the increment of ζ_n gives the same X_{AB}(and similar for the other directions) as with the incomplete gamma function.

3.3.3 Coulomb Distribution Integral

With the derived expressions for the Coulomb potential integral the full two-body distribution can be treated. The expression with the simplification in equation (3.10) gives

$$\begin{aligned} \left\langle ij \left| \frac{1}{r_{12}} \right| kl \right\rangle &= \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \Omega_{ik}(\boldsymbol{\alpha}, \boldsymbol{\gamma}, \boldsymbol{r_1}, \boldsymbol{A}, \boldsymbol{C}) \frac{1}{|\boldsymbol{r_1} - \boldsymbol{r_2}|} \Omega_{jl}(\boldsymbol{\beta}, \boldsymbol{\delta}, \boldsymbol{r_2}, \boldsymbol{B}, \boldsymbol{D}) \mathrm{d}\boldsymbol{r_1} \mathrm{d}\boldsymbol{r_2} \\ &= \sum_{pq}^{i+k,j+l} E_p^{ik} E_q^{jl} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \frac{g_p(\alpha+\gamma, \boldsymbol{r_1}, \boldsymbol{P}) g_q(\beta+\delta, \boldsymbol{r_2}, \boldsymbol{Q})}{|\boldsymbol{r_1} - \boldsymbol{r_2}|} \mathrm{d}\boldsymbol{r_1} \mathrm{d}\boldsymbol{r_2} \\ &= \frac{a}{\sqrt{(\alpha+\gamma+\beta+\delta)}} \sum_{pq}^{i+k,j+l} E_p^{ik} E_q^{jl} (-1)^q \xi_{p+q} \left(\frac{(\alpha+\gamma)(\beta+\delta)}{\alpha+\gamma+\beta+\delta}, \boldsymbol{R}_{\boldsymbol{S_1 S_2}} \right), \end{aligned} \tag{3.50}$$

where we have used the multi-index[15] notation for p, q, i, k, j, and l equation (3.21) and used equation (3.40) to arrive at the final step. An additional simplification due to the fact that ζ_n is only dependent on the relative distance of the centers is also used, for the x-coordinate it is stated as

$$\left(\frac{\partial}{\partial P_x} \right)^{p_x} \left(\frac{\partial}{\partial Q_x} \right)^{q_x} = (-1)^{p_x+q_x} \left(\frac{\partial}{\partial P_x} \right)^{p_x+q_x} \tag{3.51}$$

and the same for the other directions. The factor a is given in equation (3.52).

$$a \equiv \begin{cases} \dfrac{\pi^{\frac{3}{2}}}{\sqrt{(\alpha+\gamma)(\beta+\delta)}}, & 2\mathrm{D} \\ \dfrac{\pi^{\frac{5}{2}}}{(\alpha+\gamma)(\beta+\delta)}, & 3\mathrm{D} \end{cases} \tag{3.52}$$

[15] Essentially just expanding an index in each dimension, for instance $i = (i_x, i_y, i_z)$ with corresponding $p = (p_x, p_y, p_z)$ with each index inside the tuple running to each respective index, meaning for instance $p_x = 0$ to i_x and so on.

3.4 Double-Well Functions

This section will explain the building of a basis for the *double-well potential*. We will expand them in a linear combination of harmonic oscillator functions and find the coefficients of this expansion by solving the arising eigenvalue-problem. Let us first express the potential as

$$U^{\mathrm{DW}}(r) = V^{\mathrm{HO}}(r) + V_n^{\mathrm{DW}}(r). \tag{3.53}$$

A double well potential is essentially just a perturbation of the usual harmonic oscillator potential (which is a single-well). The V_n^{DW} part is assumed to be a polynomial of *even* degree n. This means that the integral over such a potential can be calculated using equation (3.22). Notice also that the n degree polynomial only needs to be symmetric meaning $|x|$ is also a valid polynomial to integrate over with equation (3.22).

3.4.4 The Eigenvalue problem

The mentioned eigenvalue problem comes from the basis expansion of the spacial part and from the trick of projecting with a single function from the left. We will explain this briefly. Firstly let us write out the expansion

$$\left|\psi_p^{\mathrm{DW}}\right\rangle = \sum_l C_{lp}^{\mathrm{DW}} \left|\psi_l^{\mathrm{HO}}\right\rangle \tag{3.54}$$

and then project from left the bra state $\left\langle\psi_k^{\mathrm{HO}}\right|$ in the inner-product space of h^{DW}

$$\left\langle\psi_k^{\mathrm{HO}}\middle|h^{\mathrm{DW}}\middle|\psi_p^{\mathrm{DW}}\right\rangle = \sum_l C_{pl}\left\langle\psi_k^{\mathrm{HO}}\middle|h^{\mathrm{DW}}\middle|\psi_l^{\mathrm{HO}}\right\rangle = \sum_l C_{pl}\varepsilon_l^{\mathrm{DW}}. \tag{3.55}$$

This gives us an eigenvalue equation

$$\boldsymbol{H}^{\mathrm{DW}}\boldsymbol{C}^{\mathrm{DW}} = \boldsymbol{\varepsilon}^{\mathrm{DW}}\boldsymbol{C}^{\mathrm{DW}} \tag{3.56}$$

with

$$H_{ij}^{\mathrm{DW}} = \left\langle\psi_i^{\mathrm{HO}}\middle|h^{\mathrm{DW}}\middle|\psi_j^{\mathrm{HO}}\right\rangle. \tag{3.57}$$

Using equation (3.53) we can write H_{ij} as

$$H_{ij}^{\mathrm{DW}} = \varepsilon_i^{\mathrm{HO}}\delta_{ij} + \left\langle\psi_i^{\mathrm{HO}}\middle|V_n^{\mathrm{DW}}\middle|\psi_j^{\mathrm{HO}}\right\rangle, \tag{3.58}$$

by using the solution to Schrödinger's equation for the harmonic oscillator system.

We are now in a position to build a basis for the double-well system by reusing all the results and expressions concerning the single-well system. The only difference is the extra integral over V_n^{DW} where n would be larger than 2.

For clarity let us also write out the expression for the resulting Hartree-Fock basis to be used with the Variational Monte-Carlo method. The expression is simply

$$\psi_p^{\mathrm{HF}} = \sum_{kl} C_{pk}^{\mathrm{HF}} C_{kl}^{\mathrm{DW}} \psi_l^{\mathrm{HO}}. \tag{3.59}$$

The procedure of diagonalizing H_{ij} also gives an additional set of energies we can use. The full form of the integral-elements involved in Hartree-Fock are written in equation (3.60).

$$\begin{aligned} \left\langle \psi_p^{\mathrm{DW}} \middle| \psi_q^{\mathrm{DW}} \right\rangle &= \delta_{pq} \varepsilon_p^{\mathrm{DW}} \delta_{pq} \\ \left\langle \psi_p^{\mathrm{DW}} \middle| h^{\mathrm{DW}} \middle| \psi_q^{\mathrm{DW}} \right\rangle &= \varepsilon_p^{\mathrm{DW}} \delta_{pq} \\ \left\langle \psi_p^{\mathrm{DW}} \psi_q^{\mathrm{DW}} \middle| \frac{1}{r_{12}} \middle| \psi_r^{\mathrm{DW}} \psi_s^{\mathrm{DW}} \right\rangle &= \sum_{tuvw}^{ijkl} C_{tp}^{\mathrm{DW}} C_{uq}^{\mathrm{DW}} C_{vr}^{\mathrm{DW}} C_{ws}^{\mathrm{DW}} \left\langle \psi_t^{\mathrm{HO}} \psi_u^{\mathrm{HO}} \middle| \frac{1}{r_{12}} \middle| \psi_v^{\mathrm{HO}} \psi_w^{\mathrm{HO}} \right\rangle \end{aligned} \tag{3.60}$$

The two-body elements over the harmonic oscillator functions can be calculated by expansion in s-type Gaussian constituents and then using equation (3.50).

With this eigenvalue problem in mind, one might ask why go through the trouble? The reason lies in the form of the double-well potential. Since it is a simple shift of the single-well(harmonic oscillator) it is reasonable to assume that the energies(the eigenvalues $\varepsilon^{\mathrm{DW}}$) are only shifted slightly off from the single-well energies. It is then also reasonable to believe that a basis set expansion in the single-well functions(harmonic oscillator functions) gives a nice set of basis-functions for building the double-well basis.

3.4.4 Choosing the Basis Functions

In order to actually solve the eigenvalue equation we use Python and the NumPy package, however we still need to choose the ψ^{HO}'s first. This will be experimented with and we will choose enough basis functions to reach to *Hartree-Fock limit*. The choice will also follow the harmonic oscillator levels in terms of degeneracy(see figure 5.3b and figure 5.3a). This means for instance that if we choose to only use 1 spacial function we only need the ground-state function, but there is no reason to believe that the electron-configuration would prefer any of the functions in the second level over one another. This means that we choose the basis functions according to the *magic numbers* of the harmonic oscillator system basis.

For the actual Hartree-Fock calculations it is worthwhile to mentioned that the expansion sum in equation (3.54) has to be truncated. The hope is that as we add more and more HO-functions to the eigenvalue problem (with the magic-number in mind), equation (3.58), the eignvalues will converge. We would then let the index l run up to the number of eigenvalues presented, but truncate the number of columns used at the number of eigenvalues which have converged. For an illustration of the eigenvalues for $R = 2.0$ and $\omega = 1.0$ in two dimensions see table 1. See also section 6.2.2 for details on the specific implementation.

3.4.4 Degeneracy

When we tackled the single-well problem the energy-levels were degenerate and followed the magic numbers. For the double-well only the degeneracy due to spin is present in two-dimensions. This is due to the breaking of symmetry by the displacement. For the three-dimensional case a similar break occurs, but since the displacement is only in the x-direction the two-dimensional harmonic oscillator symmetry is still present for the y- and z-directions. This means that the states which are directly dependent on the n_x quantum number have lower energy(since the double-well has lower energy than the harmonic oscillator) and we have new levels.
Since we are limiting ourselves to full-shell systems, this degeneracy has to be respected in the calculations. More details are given in chapter 5.

3.5 Summary

This chapter tackled the one- and two-body integrals over s-type Gaussian functions in two dimensions. We will here rewrite the expressions found and write out the full expression for integrals over harmonic oscillator functions.

Firstly the integrals over Hermite-Gaussians (monomials multiplied by exponential). The Hermite-Gaussian was expressed as

$$g_n(\boldsymbol{\alpha};\boldsymbol{r},\boldsymbol{A}) = \prod_d \left(\frac{\partial}{\partial A_d}\right)^{n_d} \mathrm{e}^{-\alpha_d (x_d - A_d)^2}. \tag{3.61}$$

An expansion of these in terms of Hermite-polynomials was then made to arrive at an overlap distribution, with the Gaussian product rule for the product of two Hermite-Gaussians, with a recursive relation for the expansion coefficients, see equation (3.63) below.

$$\begin{aligned}
\Omega_{ij}(\alpha,\beta,\boldsymbol{r},\boldsymbol{A},\boldsymbol{B}) &= \sum_{t=0}^{i+j} E_t^{ij} g_t(\alpha,\beta,\boldsymbol{r},\boldsymbol{P}) \\
E_t^{i+1,j} &= \frac{1}{2(\alpha+\beta)} E_{t-1}^{ij} - \frac{\beta}{\alpha+\beta}(A_x - B_x) E_t^{ij} + (t+1) E_{t+1}^{ij} \\
E_t^{i,j+1} &= \frac{1}{2(\alpha+\beta)} E_{t-1}^{ij} - \frac{\alpha}{\alpha+\beta}(A_x - B_x) E_t^{ij} + (t+1) E_{t+1}^{ij} \\
E_0^{00} &= K_{AB}
\end{aligned} \tag{3.62}$$

The overlap integral was then found to be

$$\left\langle g_i(\boldsymbol{\alpha};\boldsymbol{r},\boldsymbol{A}) \middle| g_j(\boldsymbol{\beta};\boldsymbol{r},\boldsymbol{B}) \right\rangle = \sum_p^{i+j} E_p^{ij} \left(\frac{((-1)^p - 1)\Gamma\left(\frac{p+1}{2}\right)}{2(\alpha_p + \beta_p)^{\frac{p+1}{2}}} \right)^d. \tag{3.63}$$

The integral over a potential x_d^k was found with the binomial expansion to be

$$\left\langle g_i(\boldsymbol{\alpha};\boldsymbol{r},\boldsymbol{A})\middle|\, x_d^k \,\middle| g_j(\boldsymbol{\beta};\boldsymbol{r},\boldsymbol{B})\right\rangle = \sum_{p=\text{even}}^{i+j} E_p^{i+j}\left(\frac{\Gamma\left(\frac{p+1}{2}\right)}{2(\alpha_p+\beta_p)^{\frac{p+1}{2}}}\right)^{D-1} \times \sum_{\substack{l=0\\ \text{even}}}^{k}\binom{k}{l}\frac{P_d^{k-l}}{2(\alpha_p+\beta_p)^{\frac{l}{2}}}\Gamma\left(\frac{l+1}{2}\right). \tag{3.64}$$

The integral with the Laplacian was

$$\left\langle g_i(\boldsymbol{\alpha};\boldsymbol{r},\boldsymbol{A})\middle|\nabla^2\middle| g_j(\boldsymbol{\beta};\boldsymbol{r},\boldsymbol{B})\right\rangle = \sum_d \prod_{d'\neq d}\left\langle g_{i_{d'}}(\alpha_{d'};x_{d'},A_{d'})\middle|\,\sigma_{d'}(S_d(\beta;x_d-B_d))\,\middle| g_{j_{d'}}(\beta_{d'};x_{d'},B_{d'})\right\rangle, \tag{3.65}$$

with

$$\begin{aligned} S_d(\alpha;x_d-A_d) &\equiv \left(4\alpha^2(x_d-A_d)^{i_d+2} - 2\alpha(2i_d+1)(x_d-A_d)^{i_d} + i_d(i_d-1)(x_d-A_d)^{i_d-2}\right) \\ \sigma_d(S_d) &\equiv \begin{cases} 1, & d'\neq d \\ S_d, & d'=d \end{cases} \end{aligned} \tag{3.66}$$

Finally the integral over the Coulomb interactions was calculated using a recursive relation for the one-dimensional integral it was defined by. The expressions are

$$\left\langle g_{iA}^{\boldsymbol{\alpha}} g_{jB}^{\boldsymbol{\beta}}\middle|\frac{1}{r_{12}}\middle| g_{kC}^{\boldsymbol{\delta}} g_{lD}^{\boldsymbol{\gamma}}\right\rangle = \frac{a}{\sqrt{(\alpha+\gamma+\beta+\delta)}}\sum_{pq}^{i+k,j+l} E_p^{ik} E_q^{jl}(-1)^q \xi_{p+q}\left(\frac{(\alpha+\gamma)(\beta+\delta)}{\alpha+\gamma+\beta+\delta}, \boldsymbol{R}_{\boldsymbol{S_1 S_2}}\right). \tag{3.67}$$

with

$$a \equiv \begin{cases} \dfrac{\pi^{\frac{3}{2}}}{\sqrt{(\alpha+\gamma)(\beta+\delta)}}, & \text{2D} \\ \dfrac{\pi^{\frac{5}{2}}}{(\alpha+\gamma)(\beta+\delta)}, & \text{3D} \end{cases} \tag{3.68}$$

And the recursive relation in two dimensions(left) and three dimensions (right)

$$\begin{aligned} \xi^n_{t+1,u} &= t\,\xi^{n+1}_{t-1,u} + X_{AB}\xi^{n+1}_{t,u} \\ \xi^n_{t,u+1} &= u\,\xi^{n+1}_{t,u-1} + Y_{AB}\xi^{n+1}_{t,u} \\ \xi^n_{00} &= \left(\frac{-2\alpha\beta}{\alpha+\beta}\right)^n \zeta_n\left(\frac{\alpha\beta}{\alpha+\beta}R^2_{AB}\right) \\ \zeta_n(x) &= \int_{-1}^{1}\frac{u^{2n}}{\sqrt{1-u^2}}\mathrm{e}^{-u^2x}\,\mathrm{d}u \end{aligned} \qquad \begin{aligned} \xi^n_{t+1,u,v} &= t\,\xi^{n+1}_{t-1,u,v} + X_{AB}\xi^{n+1}_{t,u,v} \\ \xi^n_{t,u+1,v} &= u\,\xi^{n+1}_{t,u-1,v} + Y_{AB}\xi^{n+1}_{t,u,v} \\ \xi^n_{t,u,v+1} &= u\,\xi^{n+1}_{t,u,v-1} + Y_{AB}\xi^{n+1}_{t,u,v} \\ \xi^n_{000} &= \left(-2\alpha\beta\right)^n \zeta_n\left(\frac{\alpha\beta}{\alpha+\beta}R^2_{AB}\right) \\ \zeta_n(x) &= \int_{-1}^{1} u^{2n}\mathrm{e}^{-u^2x}\,\mathrm{d}u \end{aligned} \tag{3.69}$$

We also give a reminder again that these expressions are only valid for *isotropic* Gaussian functions, Gaussians whose scaling factor(i.e α) is the same in all dimensions. For the non-isotropic

case the see [8].

And as promised, here are the full expressions for the integral elements with harmonic oscillator functions using equation (3.67) and the orthogonality of the harmonic oscillator functions

$$\left\langle \psi_i^{\mathrm{HO}} \middle| \psi_j^{\mathrm{HO}} \right\rangle = N_i \delta_{ij} \tag{3.70}$$

$$\left\langle \psi_i^{\mathrm{HO}} \middle| h^{\mathrm{HO}} \middle| \psi_j^{\mathrm{HO}} \right\rangle = N_i \varepsilon_i^{\mathrm{HO}} \delta_{ij} \tag{3.71}$$

$$\left\langle \psi_i^{\mathrm{HO}} \psi_j^{\mathrm{HO}} \middle| \frac{1}{r_{12}} \middle| \psi_k^{\mathrm{HO}} \psi_l^{\mathrm{HO}} \right\rangle = N_{ijkl} \frac{a}{\sqrt{2\omega}} \sum_{tuvw}^{ijkl} H_{tuvw}^{ijkl} \sum_{pq}^{t+v,u+w} E_p^{tv} E_q^{uw} (-1)^q \xi_{p+q}\left(\frac{\omega}{2}, \mathbf{0}\right) \tag{3.72}$$

$$a \equiv \begin{cases} \dfrac{\pi^{\frac{3}{2}}}{\sqrt{\omega}}, & \text{2D} \\ \dfrac{\pi^{\frac{5}{2}}}{\omega}, & \text{3D} \end{cases} \tag{3.73}$$

with the notations

$$\begin{aligned} N_{ijkl} &= X_i X_j X_k X_l \\ \sum_{tuvw}^{ijkl} &= \sum_{p}^{t} \sum_{q}^{u} \sum_{r}^{v} \sum_{s}^{w} \\ \sum_{pq}^{t+v,u+w} &= \sum_{p}^{t+v} \sum_{q}^{u+w} \\ H_{tuvw}^{ijkl} &= H_{i,t} H_{j,u} H_{k,v} H_{l,w} \\ E_p^{tv} &= E_{p,t+v} \end{aligned} \tag{3.74}$$

And using the multi-index notation for the dimensions in p and q indices. The H are the Hermite coefficients.

3.6 Further Work

This concludes this chapter on basis functions. We have found an expression for the integral elements involving Hermite-Gaussians and used these to express the integral over harmonic oscillator functions which in turn gave an expression for the integrals over double-well functions as they were just expanded in harmonic oscillator functions. The next chapter will present optimizations methods used in the variational method and the implementations of these integrals are presented in chapter 6.

As a note for further work the expressions found here are for isotropic gaussians only. For certain systems, like the one presented in article [8], more flexibility in the Gaussian functions can

be desired, such as finding an expression for the integrals of Gaussian functions with higher order monomial factors and extending the existing code to calculate those can be useful.

Experimentation with different centered gaussians for the double-well case can be interesting since it is not unreasonable to believe that functions centered on the nodes of the double-well might be a better model for the system. In which case the number of basis functions needed might be reduced and the computational strain would more relaxing.

CHAPTER 4

NUMERICAL OPTIMIZATION

In the Variational Monte Carlo method in section 2.9 the essential point was to vary a set of *variational parameters* in order to reach an eigenbasis which gives the ground-state energy of the Hamiltonian in question. There are many ways one could approach this. One way could be to wildly guess random parameters and hope for the best. Obviously this is a poor approach. The more sound approach would be to optimize(minimization in the VMC case) the wavefunction using methods from (as the title suggests) *numerical optimization*. The methods used in this thesis are the *Conjugate Gradient method*, a version of the *Adaptive Stochastic Gradient Descent*, the well known *BFGS* method and a more recent *Stochastic-Adaptive-BFGS* and also *Simulated Annealing*. The description of the approaches for numerical optimization is only made briefly. For a better mathematical explanation see the various references in the text.

4.1 The Optimization Problem

We will explain the general approach for minimizing a multi-variate function and set the terminology in this section.

The problem in question is the following. Given a continuously differentiable function $f : \mathbb{R}^n \to \mathbb{R}$, for what set of parameters $\{x\}_{k=1}^n$ is

$$\nabla_x f = \mathbf{0} \tag{4.1}$$

fulfilled[1]. This means we seek a point $\boldsymbol{x}_m$ in real space were the variation of the value of f is zero.

[1] $\nabla_x = \sum_k \boldsymbol{e}_k \frac{\partial}{\partial x_k}$ with $\boldsymbol{e}_k \in \mathbb{R}^n$ a unit vector along direction k.

In reality the condition in equation (4.1) is only approximate, that is we terminate the search for a minimum if we reached a point where the absolute value of f is within a threshold ϵ

$$\left|\nabla_x f\right| \leq \epsilon. \tag{4.2}$$

One might at this point ask the question, wouldn't the condition presented in equation (4.2) (and equation (4.1) be valid for a maximum as well? The answer is yes, it would. The simple fix to this is to define the *search direction*, more precisely the sign of the search direction. The next section explains this in better detail.

4.2 Gradient Descent

We defined the optimization problem and defined a simple condition for the extremal and mentioned a search direction in the previous section. A search direction in our context is a direction $\boldsymbol{p} \in \mathbb{R}^n$ which points towards $\boldsymbol{x}_m$. To find $\boldsymbol{p}$ we use the well known *second derivative test* to determine the curvature of f. This, mentioned qualitatively, means that the gradient of f at any point $\boldsymbol{x}_i$ points towards an extremal and that the negative gradient(negative sign) points towards the minimum and the positive gradient points towards the maximum. This observation gives a simple rule for finding $\boldsymbol{x}_m$. Start out with blindly guessing a point $\boldsymbol{x}_0$ and keep updating the parameters according to the recursive rule

$$\boldsymbol{x}_n = \boldsymbol{x}_{\boldsymbol{n-1}} - \gamma \nabla_x f \tag{4.3}$$

and terminate the search when equation (4.2) is fulfilled[2] [44]. An algorithmic view is seen in figure 4.1.

[2]Change the negative sign in front of the gradient if a maximum is desired.

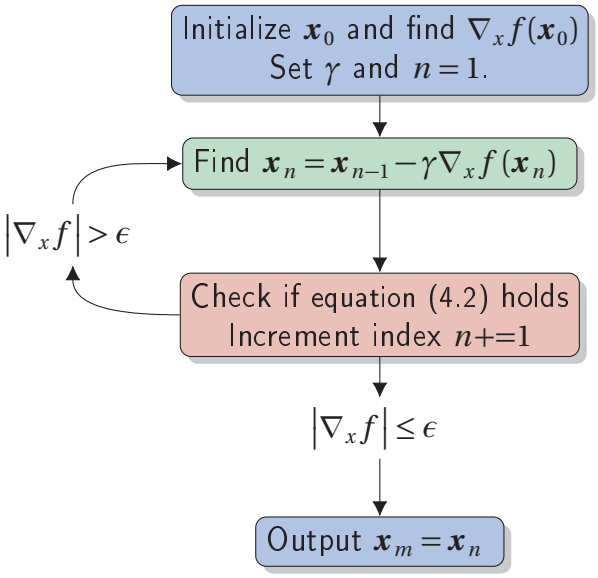


Figure 4.1: Gradient Descent algorithm.

This method of finding the minimum is known as the method of *Gradient Descent* and its power lies in its simplicity. The problem however is stability, the termination condition is firstly not optimal and the step-size γ is a constant which can give a lot of oscillations around the minimum as the algorithm might get close to the minimum and then *over-shoot* and go past the minimum point, turn around (because the sign changes) and over-shoot again and then keep going. Many methods have been devised to account for these problems and other. We will contain ourselves with the methods we presented in the introduction of this chapter.

As an illustration of the method here are a couple of figures of the method applied to two test-functions(figures 4.2 and 4.3). The first function is sphere-function

$$f(\boldsymbol{x}) = \sum_{d=1}^{D} x_d^2, \tag{4.4}$$

and the second is the so-called Rosenbrock function

$$f(\boldsymbol{x}) = \sum_{d=1}^{D-1} 100\left(x_{d+1} - x_d^2\right)^2 + (x_d - 1)^2. \tag{4.5}$$

Sphere Function

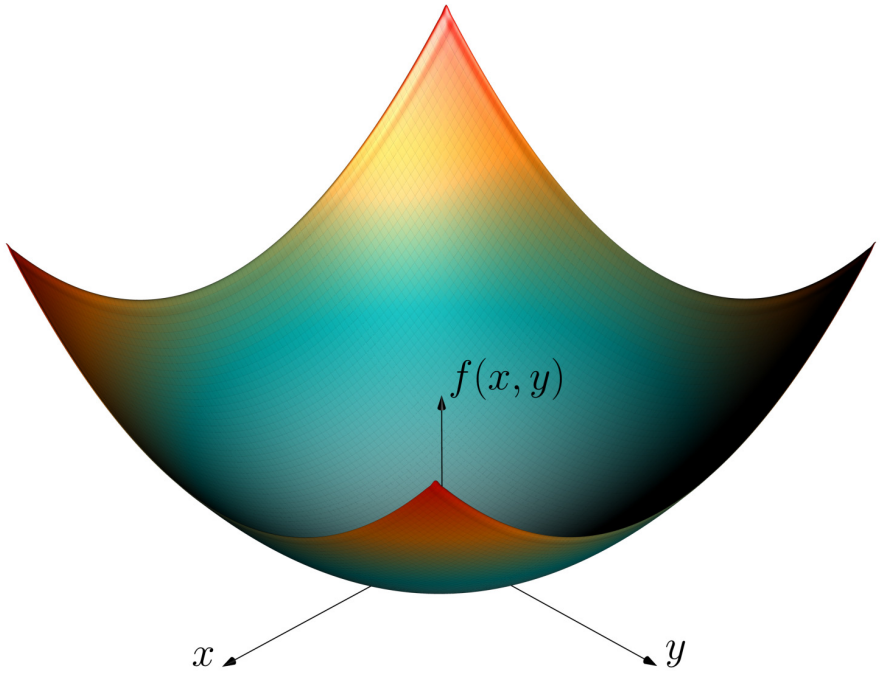


Figure 4.2: Illustration of sphere function[54] with minimum $\boldsymbol{x}_m = (0,0)$ with value $f(\boldsymbol{x}_m) = 0$.

Rosenbrock Function

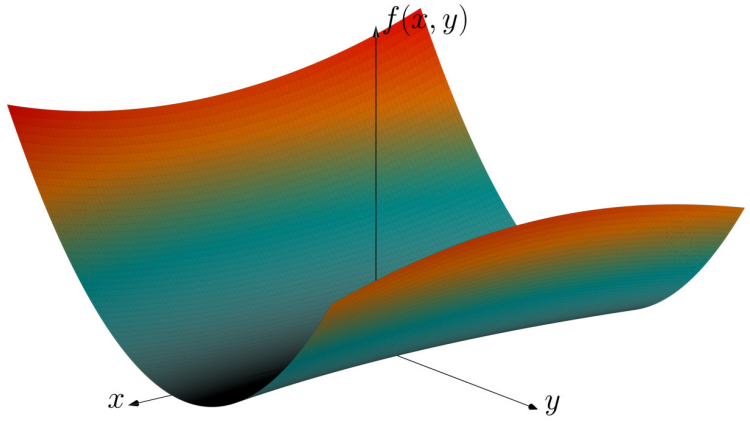


Figure 4.3: Illustration of Rosenbrock function[54] with minimum $\boldsymbol{x}_m = (1,1)$ with value $f(\boldsymbol{x}_m) = 0$.

As for the convergence we give a table with number of iterations.

Table 1: Table showing convergence of the gradient descent method with the spherical function. $\boldsymbol{x}_0$ is the initial starting point, γ is the step-size, $\boldsymbol{x}_m$ is the minimum after the given iterations and $f(\boldsymbol{x}_m)$ is the function value at said minimum point.

$\boldsymbol{x}_0$	γ	Iterations	$\boldsymbol{x}_m$	$f(\boldsymbol{x}_m)$
$(5,5)$	0.9	20	$(-0.072,-0.072)$	0.010
$(5,5)$	0.9	50	$(-8.920\times10^{-5},-8.920\times10^{-5})$	1.591×10^{-8}
$(5,5)$	0.9	100	$(-1.273\times10^{-9},-1.273\times10^{-9})$	3.242×10^{-18}
$(5,5)$	0.5	20	$(0.0,0.0)$	0.0
$(5,5)$	0.5	50	$(0.0,0.0)$	0.0
$(5,5)$	0.5	100	$(0.0,0.0)$	0.0
$(5,5)$	0.1	20	$(0.072,0.072)$	0.010
$(5,5)$	0.1	50	$(8.920\times10^{-5},8.920\times10^{-5})$	1.591×10^{-8}
$(5,5)$	0.1	100	$(1.273\times10^{-9},1.273\times10^{-9})$	3.242×10^{-18}

Table 2: Table showing convergence of the gradient descent method with the Rosenbrock function. $\boldsymbol{x}_0$ is the initial starting point, γ is the step-size, $\boldsymbol{x}_m$ is the minimum after the given iterations and $f(\boldsymbol{x}_m)$ is the function value at said minimum point.

$\boldsymbol{x}_0$	γ	Iterations	$\boldsymbol{x}_m$	$f(\boldsymbol{x}_m)$
$(0,0.5)$	0.001	100	$(0.181,0.030)$	0.034
$(0,0.5)$	0.001	500	$(0.512,0.258)$	0.327
$(0,0.5)$	0.001	1000	$(0.675,0.454)$	0.106
$(0,0.5)$	0.001	100000	$(1.000,1.000)$	0.0
$(0,0.5)$	0.0001	100	$(0.027,0.068)$	1.399
$(0,0.5)$	0.0001	500	$(0.105,0.009)$	0.801
$(0,0.5)$	0.0001	1000	$(0.184,0.031)$	0.666
$(0,0.5)$	0.0001	100000	$(0.994,0.989)$	3.131×10^{-5}

From tables 1 and 2 it is apparent that with the gradient descent method the step-size is of great importance. With the spherical function one needs a sweet-spot value to reach the minimum while for the more complex Rosenbrock function the number of iterations needed is very high and with a lower step-size the minimum was actually not even reached with 100000 iterations. In conclusion, the gradient descent method is great for its simplicity, but it does not incorporate the curvature of the function when minimizing meaning the choice for the step-size greatly determines the outcome of the minimization, and for functions where the minimium lies in a fairly flat valley, as with the Rosenbrock function, the computational cost increases since the number of iterations needed for convergence is high.

4.3 Adaptive Stochastic Gradient Descent

Along with the limitations of the method of gradient descent, the *Adaptive Stochastic Gradient Descent* tries to account for those, but also takes into account the variance introduced by the stochastic nature of the probability distribution. As such, many variations of the method have been proven to be popular among problems in which the function to be minimized is an expectation value. The method used in this thesis is the one described in [57]. We will give a summary of the method here, for a more detailed outline and description see [57].

Like the gradient descent method the adaptive stochastic gradient descent method updates the parameters in the same manner as in equation (4.3), the difference however is that the step γ is changed for each iteration as follows

$$\begin{aligned} \gamma_{n+1} &= \frac{a}{t_{n+1}+A} \\ t_{n+1} &= \max(t_n + g(X_n), 0) \\ X_n &= -\nabla f_n \cdot \nabla f_{n+1} \\ g(x) &= g_{\min} + \frac{g_{\max} - g_{\min}}{1 - \frac{g_{\max}}{g_{\min}} \mathrm{e}^{-\frac{x}{\omega}}} \end{aligned} \tag{4.6}$$

The whole idea of the method is that the form of g and the accumulative combination of gradient estimations for each step, the total error would tend quickly to zero, meaning the central element (namely the gradient) in the minimization is well behaving.

The main concern with the method is convergence, although the error in the gradient estimations tend towards zero, the step-sizes themselves will also be quite small after some iterations. For this reason we use the adaptive method with a quasi-Newton method. This means that we start with a random guess at the parameters and keep iterating with the quasi-Newton method until the norm of the gradient is below some threshold, at which the adaptive method is applied from that point and onward till convergence is reached.

4.4 Newtons-Method and Quasi-Newton Methods

We will here explain briefly *Newton's method* and *Quasi-Newton* methods as the ideas presented will be used in the next section.

Newton's method [35] (or Newton-Raphson method) is originally a method for finding the zeros of a function. The rule states that given a real-valued function $f : \mathbb{R} \to \mathbb{R}$ and an initial guess

$x \in \mathbb{R}$ for the zero-point, recursively find better approximations for the zero by setting

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}. \tag{4.7}$$

This method would then within a number iterations find the zero that is closest to x_0.

For the optimization problem the condition for a point to be an extremal is equation (4.1) meaning, again, that one needs to find the zero of the derivative. Newton's method in this case would be

$$x_{n+1} = x_n - \frac{f'(x_n)}{f''(x_n)}, \qquad n \geq 0. \tag{4.8}$$

Of course in the real world one might work with multi-variate function, not to worry as Newton's method for optimization problems in the multi-variate case with $f : \mathbb{R}^n \to \mathbb{R}$ (still real-valued) is

$$\boldsymbol{x}_{n+1} = \boldsymbol{x}_n - \left|\boldsymbol{H} f(\boldsymbol{x}_n)\right|^{-1} \dot{\nabla} f(\boldsymbol{x}_n), \qquad n \geq 0, \tag{4.9}$$

where $\boldsymbol{H}$ is the Hessian matrix. One might also introduce a step-length multiplied to the Hessian part in order to induce conditions [44] which ensure some stability of the method.

Newton's method, in most cases, converges faster(less iterations) towards the minimum than gradient descent making it favorable, however the full Hessian has to be known. This matrix(or its inverse) is in many cases too expensive to compute or difficult to express in closed-form. In these cases the class of methods knows as Quasi-Newton methods can be utilized.

Quasi-Newton methods give an estimate of the inverse Hessian by using the first derivatives. Introduce the Taylor approximation of f around an iteration point $\boldsymbol{x}_n$

$$f(\boldsymbol{x}_k + \boldsymbol{s}) \approx f(\boldsymbol{x}_k) + \left(\nabla f(\boldsymbol{x}_k)\right)^T \boldsymbol{s} + \frac{1}{2}\boldsymbol{s}^T \boldsymbol{H}\boldsymbol{s}, \tag{4.10}$$

differentiate with respect to the change $\boldsymbol{s}$

$$\nabla_s f(\boldsymbol{x}_k + \boldsymbol{s}) \approx \nabla f(\boldsymbol{x}_k) + \boldsymbol{H}\boldsymbol{s} \tag{4.11}$$

and introduce the condition in equation (4.1) and set this gradient to zero to find the change $\boldsymbol{s}$

$$\boldsymbol{s} = -\boldsymbol{H}^{-1} \nabla f(\boldsymbol{x}_k). \tag{4.12}$$

Another way to determine this particular form for $\boldsymbol{s}$ is to say that the approximation to the Hessian must satisfy the *secant equation* which is equation (4.11). The updating rule for $\boldsymbol{x}_n$ is then given by

$$\boldsymbol{x}_{n+1} = \boldsymbol{x}_n - \gamma_k \boldsymbol{H}_n^{-1} \nabla f(\boldsymbol{x}_n). \tag{4.13}$$

The factor γ_k is again introduced to give some stability conditions. The important part of this equation is the index on the inverse Hessian. This is essentially just a relabeling at the change $\boldsymbol{s}$

is technically applied for each iterate $\boldsymbol{x}_n$. Note also that $\boldsymbol{s}$ takes the role of the search direction in this case. The algorithm is then to make an initial guess on the Hessian(usually just the identity matrix) and then use a type of updating formula that finds a new approximation for the Hessian at each step n. There are a number of these updating formulas, just to mention some we have DFP, SR1, McCormick, Broyden, BFGS and more. The one we will mention in more detail is the BFGS method, but a main formula that shows up in all of the updating methods is the *Sherman-Morrison formula* for the inverse. This basically means that the need for calculation of the inverse matrix is completely removed.

With the mentioned expression we can devise an algorithm similar to Newton's method for finding the minimum $\boldsymbol{x}_m$. Starting with an initial guess for the inverse Hessian $\boldsymbol{H}_0^{-1}$ and minimum $\boldsymbol{x}_0$ with the condition that $\boldsymbol{H}_0^{-1}$ is positive-definite (identity matrix is a nice start if nothing else is known) proceed with the algorithm outlined in figure 4.4.

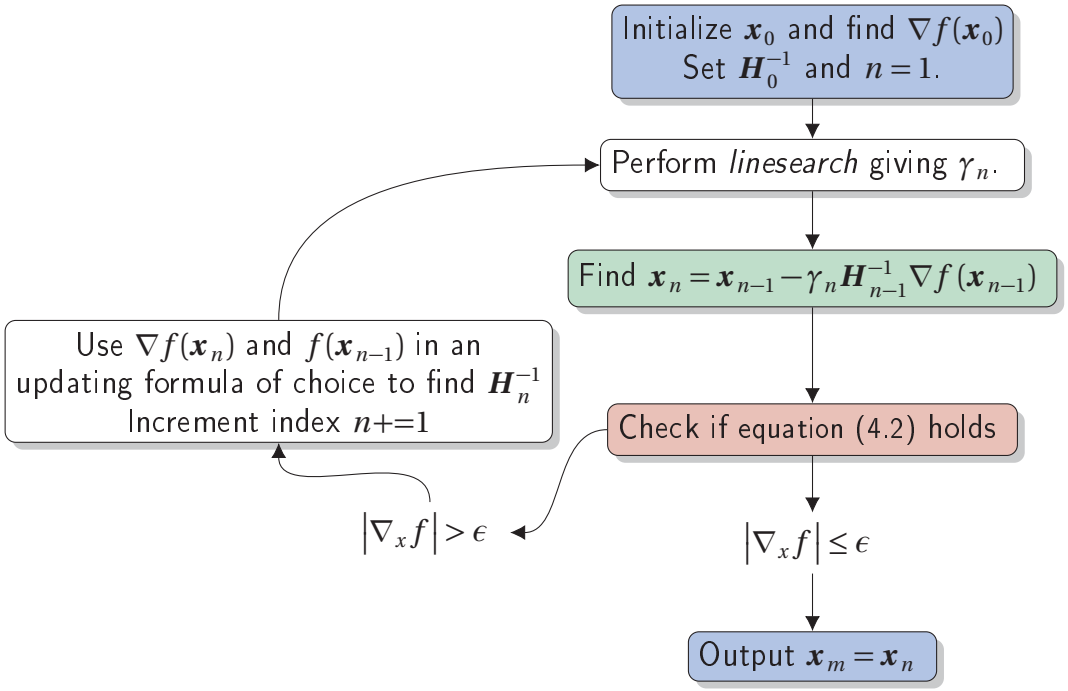


Figure 4.4: Quasi-Newton algorithm.

To illustrate the power of the method we again apply it to the sphere function and the Rosenbrock function using the BFGS scheme as the updating method and the Moré-Thuente line search method to find the step-size. The table of convergence is as follows

Table 3: Table showing convergence of a Quasi-Newton method with BFGS method with the spherical function. $\boldsymbol{x}_0$ is the initial starting point, $\boldsymbol{x}_m$ is the minimum after the given iterations and $f(\boldsymbol{x}_m)$ is the function value at said minimum point.

$\boldsymbol{x}_0$	Iterations	$\boldsymbol{x}_m$	$f(\boldsymbol{x}_m)$
$(1,1)$	1	$(-0.071,-0.071)$	1.000
$(-1,2)$	1	$(0.447,-0.894)$	1.000
$(1,1)$	2	$(0.000,0.000)$	0.000
$(-1,2)$	2	$(0.000,0.000)$	0.000
$(10,10)$	1	$(-0.071,-0.071)$	1.000
$(10,10)$	2	$(0.000,0.000)$	0.000
$(100,100)$	1	$(-0.071,-0.071)$	1.000
$(100,100)$	2	$(0.000,0.000)$	0.000

Table 4: Table showing convergence of a Quasi-Newton method with BFGS method with the Rosenbrock function. $\boldsymbol{x}_0$ is the initial starting point, $\boldsymbol{x}_m$ is the minimum after the given iterations and $f(\boldsymbol{x}_m)$ is the function value at said minimum point.

$\boldsymbol{x}_0$	Iterations	$\boldsymbol{x}_m$	$f(\boldsymbol{x}_m)$
$(-0.5,2.0)$	1	$(-0.706,0.708)$	7.280
$(-0.5,2.0)$	2	$(-0.780,0.649)$	3.342
$(-0.5,2.0)$	10	$(0.238,0.051)$	0.584
$(-0.5,2.0)$	30	$(1.000,1,000)$	0.000
$(5.5,-10.0)$	1	$(-0.996,0.091)$	85.214
$(5.5,-10.0)$	2	$(-0.908,1.087)$	10.549
$(5.5,-10.0)$	10	$(0.027,0.012)$	0.9613
$(5.5,-10.0)$	30	$(1.000,1,000)$	0.000

Comparing tables 3 and 4 with tables 1 and 3 we can see that the BFGS scheme outperforms the gradient descent method by a stupendous and almost comical amount with the number of iterations in mind. However one still has to keep in mind that each iterations of the BFGS method is far more computationally extensive meaning the gradient descent method can be more favorable in the case where the function to be minimized is expensive to compute.

For our case, the latter sentiment is true. The expectation value to the energy still has a large complexity, however with the optimizations mentioned in sections 5.3.3 and 5.3.3 the time it takes to calculate the expectation value is not too large and the BFGS scheme can still be used.

4.5 BFGS Method

In the previous section(also mentioned in figure 4.4) we gave an outline for Newton's method and the class known as Quasi-Newton methods. The latter used an approximation for the inverse of

the Hessian matrix, which was updated at each step in the algorithm. For the sake of brevity only conditions employed to arrive at the expression for the updating formula and the formula itself is given here, for more see [5, 15, 17, 44, 53]. The conditions enforced is

- Secant condition: $\boldsymbol{H}_{n+1}\boldsymbol{s}_n = \nabla f(\boldsymbol{x}_{n+1}) - \nabla f(\boldsymbol{x}_n)$
- Strong curvature: $\boldsymbol{s}_k^T \cdot (f(\boldsymbol{x}_{n+1}) - \nabla f(\boldsymbol{x}_n)) > 0$

and the resulting formula states with $\boldsymbol{y}_k = f(\boldsymbol{x}_{n+1}) - \nabla f(\boldsymbol{x}_n)$

$$\boldsymbol{H}_{n+1} = \boldsymbol{H}_n + \frac{\boldsymbol{y}_n \boldsymbol{y}_n^T}{\boldsymbol{y}_n^T \boldsymbol{s}_n} - \frac{\boldsymbol{H}_n \boldsymbol{s}_n \boldsymbol{s}_n^T \boldsymbol{H}_n}{\boldsymbol{s}_n^T \boldsymbol{H}_k \boldsymbol{s}_n}. \tag{4.14}$$

With the Sherman-Morrison formula[50] the inverse is updated with

$$\boldsymbol{H}_{n+1}^{-1} = \boldsymbol{H}_n^{-1} + \frac{\left(\boldsymbol{s}_n^T \boldsymbol{y}_n + \boldsymbol{y}_n^T \boldsymbol{H}_n^{-1} \boldsymbol{y}_n\right)\left(\boldsymbol{s}_n \boldsymbol{s}_n^T\right)}{\left(\boldsymbol{s}_n^T \boldsymbol{y}_n\right)^2} - \frac{\boldsymbol{H}_n^{-1} \boldsymbol{y}_n \boldsymbol{s}_n^T + \boldsymbol{s}_n \boldsymbol{y}_n^T \boldsymbol{H}_n^{-1}}{\boldsymbol{s}_n^T \boldsymbol{y}_n}. \tag{4.15}$$

4.6 Line Search methods

In the optimization methods described in section 4.4 there was one important part neglected, namely how to find the step-length γ_n introduced in the updating formula. As it is, one can choose it in any manner desired, however a class of one-dimensional minimization methods knows as *lineasearch methods* are often used to get an (usually rough) estimate for the step length at each iteration in the optimization. These methods all have some conditions for stability and convergence as an innate property, meaning the validity of the step length is better[3]. Some popular line search methods are backtracking line search, Hager-Zhang method, Strong Wolfe conditions and the More-Thuente line search method. The one used here is the latter. For an exact derivation and explanation of line search methods in general see [44]. See also the article by Jorge J. Moré and David J. Thuente [41].

The basic idea of line search methods is to solve a one-dimensional problem of minimizing

$$\phi(\alpha) = f(\alpha \boldsymbol{p}_k + \boldsymbol{x}_k), \tag{4.16}$$

with $f : \mathbb{R}^n \to \mathbb{R}$ and $\boldsymbol{p}_k$ is a search direction as described with the quasi-Newton methods and $\boldsymbol{x}_k$ is the current iterate(point) in the minimization. Notice also that

$$\frac{\partial \phi}{\partial \alpha} = \boldsymbol{p}_k \cdot \nabla f(\alpha \boldsymbol{p}_k + \boldsymbol{x}_k) \tag{4.17}$$

by the chain-rule and the gradient on the right hand side is over the parameters $\boldsymbol{x}_k$. One usually perform this line search loosely since the search direction is not necessarily directly pointing

[3] It's actually present...

towards the minimum, meaning we only search for a step length that gives a *sufficient decrease* in the function value f. The basic procedure is then to use one of these line search methods to find γ_n at each iteration in the minimization and then use the step-length outputted by the line search algorithm to update the parameters.

4.7 Stochastic-Adaptive-BFGS

In section 4.5 we mentioned the popular BFGS method for updating the Hessian matrix and its inverse. A more resent method which uses that method with the stochastic nature of a functional expectation value is a method called SABFGS [18] described by Zhou C., Gao W. and Goldfarb D. This method uses the BFGS update for the Hessian, but uses an adaptive step instead of the deterministic line search for the step-size.
There one problem however, the method itself is only valid for *self-concordant* functions. The energy-functional is by-far not within this criteria. This problem can be accounted for by using the Wolfe conditions. That is to check that the step-size satisfies the Wolfe conditions at each iteration before actually making an update. The algorithm presented takes this into account as well.

Algorithm 1 SA-BFGS

Input: $\boldsymbol{x}_0$, $\boldsymbol{H}_0$, $\boldsymbol{G}_0$, $\beta < 1$
for $k = 0$ to $M_{\max}$ **do**
  $\boldsymbol{g}_k = \nabla F_k(\boldsymbol{x}_k)$ ▷ Gradient in current step
  $\boldsymbol{d}_k = -\boldsymbol{H}_k \boldsymbol{g}_k$ ▷ Search direction
  $\delta_k = \sqrt{\boldsymbol{d}_k^T \boldsymbol{G}_k(\boldsymbol{x}_k)\boldsymbol{d}_k}$
  $\alpha_k = \frac{\boldsymbol{g}_k^T \boldsymbol{H}_k \boldsymbol{g}_k}{\delta_k^2}$
  $t_k = \frac{\alpha_k}{1+\alpha_k \delta_k}$ ▷ Step size
  $\boldsymbol{g}_{k+1} = \nabla F_k(\boldsymbol{x}_k + t_k \boldsymbol{d}_k)$ ▷ Propose new set of parameters
  if $\boldsymbol{g}_{k+1}^T \boldsymbol{d}_k < \beta\, \boldsymbol{g}_k^T \boldsymbol{d}_k$ **then** ▷ Wolfe-Conditions
    Set $\boldsymbol{d}_k = -\boldsymbol{g}_k$
    Recompute δ_k, α_k and t_k
    Set $\boldsymbol{H}_{k+1} = \boldsymbol{H}_k$ and $\boldsymbol{G}_{k+1} = \boldsymbol{G}_k$
  else
    Set $\boldsymbol{H}_{k+1}$ with BFGS inverse update ▷ equation (4.15)
    Set $\boldsymbol{G}_{k+1}$ with BFGS update ▷ equation (4.14)
  end if
  $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + t_k \boldsymbol{d}_k$ ▷ Update parameters
end for

4.8 Simulated Annealing

A huge problem with the mentioned methods is the fact that they only converge towards a *local minimum* which is not necessarily the *global minimum* which of desire. Many methods already exists to account for this, the one used in this thesis and to be described in this section is the method known as *simulated annealing*. Simulated annealing follows a simple algorithm, see algorithm 2.

Algorithm 2 Simulated Annealing

Initialize a solution $s = s_0$. ▷ I.e a set of parameters $\{\alpha\}_{k=1}^{N}$
for $j = 1$ to $M_{\max}$ **do**
  Set temperature T with specific function for $\frac{j}{M_{\max}}$
  Pick a new state s_{new} within some neighbour of s
  if $P(f(s), f(s_{\text{new}}), T) \geq \xi$ **then** ▷ Metropolis-Test.
    $s = s_{\text{new}}$
  end if
end for

The idea is to start with searching a large part of the solution space, since a high temperature increases the search-range, and hope that as j reaches $M_{\max}$ the probability function P is such that the solution is trapped within the down-hill of the global minimum.
The specific form of P, T and how to choose a neighbour is specific from problem to problem however an effective and simple way to define these is by using the Metropolis-algorithm with

$$P = \exp\left(-\frac{f(s_{\text{new}}) - f(s)}{T}\right), \tag{4.18}$$

and define the temperature as

$$T_j = \frac{T_{\max}}{j}. \tag{4.19}$$

And $T_{\max}$ is the initial temperature. One then chooses new neighbours within some min/max range from the current s [29]. After the annealing is done a Quasi-Newton method is used and then one of the adaptive methods is used to converge to the minimum. In order to test this scheme we apply it to more complex functions with many local minima. Two such functions are the *Ackley function* [54] defined as

$$f(\boldsymbol{x}) = -a\exp\left(-b\sqrt{\frac{1}{D}\sum_{d=1}^{D}x_d^2}\right) - \exp\left(-\frac{1}{D}\sum_{d=1}^{D}\cos(c\,x_d)\right) + a + \exp(1) \tag{4.20}$$

and the *Rastrigin function* [54]

$$f(\boldsymbol{x}) = 10D + \sum_{d=1}^{D}\left(x_d^2 - 10\cos(2\pi x_i)\right). \tag{4.21}$$

Where D is the number of parameters and a, b and c being parameters to be tweaked. Figures are shown below in figures 4.5 and 4.6

Ackley Function

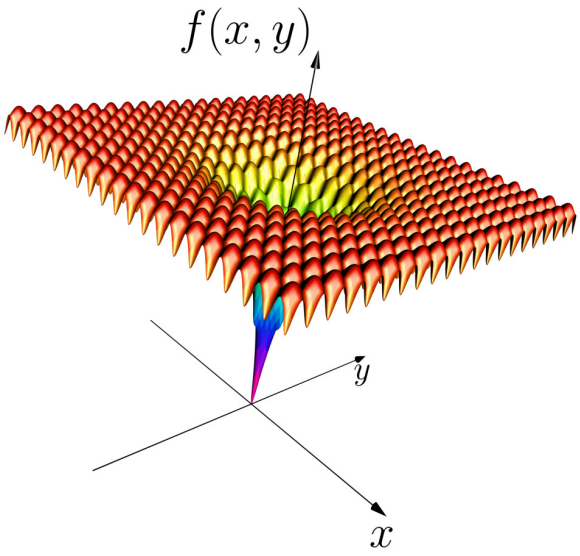


Figure 4.5: Illustration of Ackley function [54] with global minimum $\boldsymbol{x}_m = (0,0)$ of value $f(\boldsymbol{x}_m) = 0$.

Rastrigin Function

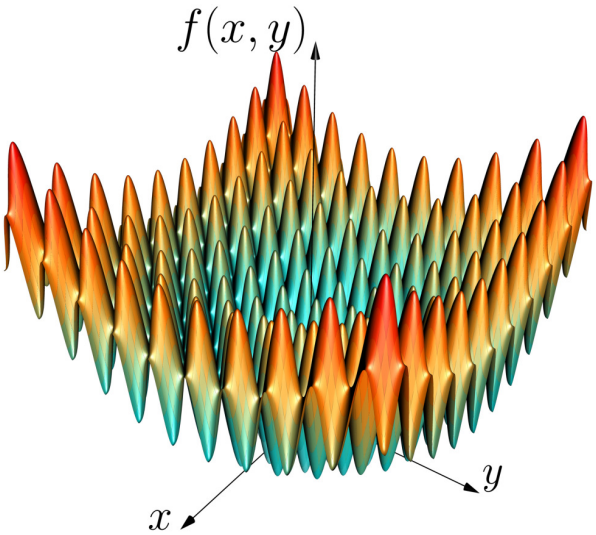


Figure 4.6: Illustration of Rastrigin function [54] with global minimum $\boldsymbol{x}_m = (0,0)$ of value $f(\boldsymbol{x}_m) = 0$.

As for convergence, we again employ the same procedure as with the spherical and Rosenbrock functions as previously, but with the Ackelyn and Rastrigin functions and present the tables here.

Table 5: Table showing convergence of the simulated annealing method with the Ackelyn function. $\boldsymbol{x}_0$ is the initial starting point, $T_{\max}$ is the initial temperature, $\boldsymbol{x}_m$ is the minimum after the given iterations and $f(\boldsymbol{x}_m)$ is the function value at said minimum point. The neighbouring function used is a simple gaussian applied to all parameters with mean being the current value of the parameter and variance 1.0.

$\boldsymbol{x}_0$	$T_{\max}$	Iterations	$\boldsymbol{x}_m$	$f(\boldsymbol{x}_m)$
$(-10.0, 0.40)$	100.0	10^2	$(0.130, 0.096)$	1.046
$(-10.0, 0.40)$	100.0	10^3	$(0.007, -0.026)$	0.097
$(-10.0, 0.40)$	100.0	10^4	$(0.002, -0.004)$	0.011
$(-10.0, 0.40)$	100.0	10^5	$(-0.003, -0.003)$	0.013
$(-0.01, 0.01)$	100.0	10^2	$(-0.101, 0.036)$	0.584
$(-0.01, 0.01)$	100.0	10^3	$(-0.012, -0.006)$	0.044
$(-0.01, 0.01)$	100.0	10^4	$(-0.004, 0.001)$	0.013
$(-0.01, 0.01)$	100.0	10^5	$(-0.001, -0.001)$	0.003
$(-10.0, 0.40)$	50.0	10^2	$(0.012, -0.122)$	0.702
$(-10.0, 0.40)$	50.0	10^3	$(-0.021, 0.023)$	0.113
$(-10.0, 0.40)$	50.0	10^4	$(0.008, -0.016)$	0.061
$(-10.0, 0.40)$	50.0	10^5	$(0.002, -0.003)$	0.009
$(-0.01, 0.01)$	50.0	10^2	$(0.141, 0.059)$	0.959
$(-0.01, 0.01)$	50.0	10^3	$(-0.007, -0.041)$	0.162
$(-0.01, 0.01)$	50.0	10^4	$(-0.013, 0.003)$	0.043
$(-0.01, 0.01)$	50.0	10^5	$(0.001, 0.002)$	0.005

Table 6: Table showing convergence of the simulated annealing method with the Rastrigin function. $\boldsymbol{x}_0$ is the initial starting point, $T_{\max}$ is the initial temperature, $\boldsymbol{x}_m$ is the minimum after the given iterations and $f(\boldsymbol{x}_m)$ is the function value at said minimum point. The neighbouring function used is a simple gaussian applied to all parameters with mean being the current value of the parameter and variance 1.0.

$\boldsymbol{x}_0$	$T_{\max}$	Iterations	$\boldsymbol{x}_m$	$f(\boldsymbol{x}_m)$
$(-5.0, 1.0)$	100.0	10^2	$(-0.988, 0.967)$	2.152
$(-5.0, 1.0)$	100.0	10^3	$(0.045, -0.038)$	0.684
$(-5.0, 1.0)$	100.0	10^4	$(0.007, -0.002)$	0.009
$(-5.0, 1.0)$	100.0	10^5	$(0.001, -0.002)$	0.001
$(-0.01, 0.001)$	100.0	10^2	$(-0.873, 0.016)$	3.844
$(-0.01, 0.001)$	100.0	10^3	$(-0.011, -0.010)$	0.043
$(-0.01, 0.001)$	100.0	10^4	$(0.001, 0.010)$	0.021
$(-0.01, 0.001)$	100.0	10^5	$(-0.002, -0.002)$	0.001
$(-5.0, 1.0)$	50.0	10^2	$(-0.936, -0.016)$	1.731
$(-5.0, 1.0)$	50.0	10^3	$(0.063, 0.020)$	0.864
$(-5.0, 1.0)$	50.0	10^4	$(0.014, -0.004)$	0.045
$(-5.0, 1.0)$	50.0	10^5	$(0.002, 0.000)$	0.001
$(-0.01, 0.001)$	50.0	10^2	$(0.977, -0.024)$	1.168
$(-0.01, 0.001)$	50.0	10^3	$(-0.012, -0.017)$	0.085
$(-0.01, 0.001)$	50.0	10^4	$(0.002, 0.004)$	0.004
$(-0.01, 0.001)$	50.0	10^5	$(-0.001, -0.003)$	0.001

We can see from tables 5 and 6 that the simulated annealing method does get close to the actual minimum, however a great number of iterations is needed and even as we reach 10^5 iterations the method still doesn't converge towards the minimum. This seems quite dissappointing, but one has to remember that the method is to be used as a way to reach an area were deterministic methods such as gradient descent method or the BFGS scheme is guaranteed to converge towards the minimum. As long as we control the temperature properly such a point can be found with simulated annealing.

CHAPTER 5

IMPLEMENTATION

With almost every expression given in chapter 3 being some kind of sum it is not hard to imagine that solving them by hand would be quite the challenge and most likely impossible, but with the modern computer at our disposal they can be tackled numerically. However in order to get the data from the numerical simulations the actual implementation, the code, has to be written, tested, verified and run.

This chapter will explain the code used in the thesis in detail. The main code base is given in https://github.com/Oo1Insane1oO/HartreeFock and https://github.com/Oo1Insane1oO/VMC. General information of the usage of packages are given in appendix » REF APPENDIX « while the structure and workflow of the code is given here. We will first present a workflow-chart of the two main code-bases used, namely the Hartree-Fock implementation and the Variational Monte-Carlo implementation.

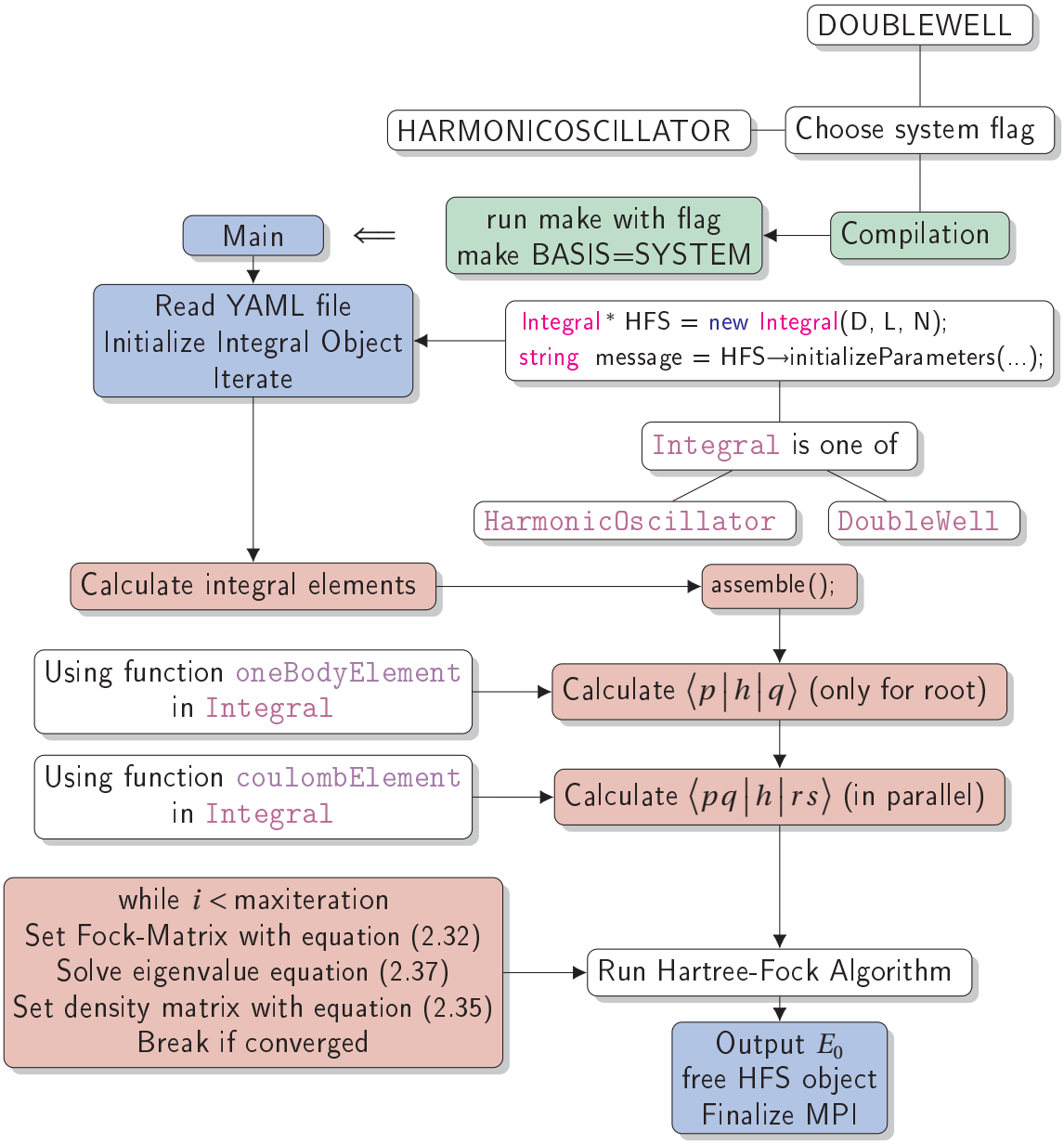


Figure 5.1: Flow chart of Hartree-Fock implementation.

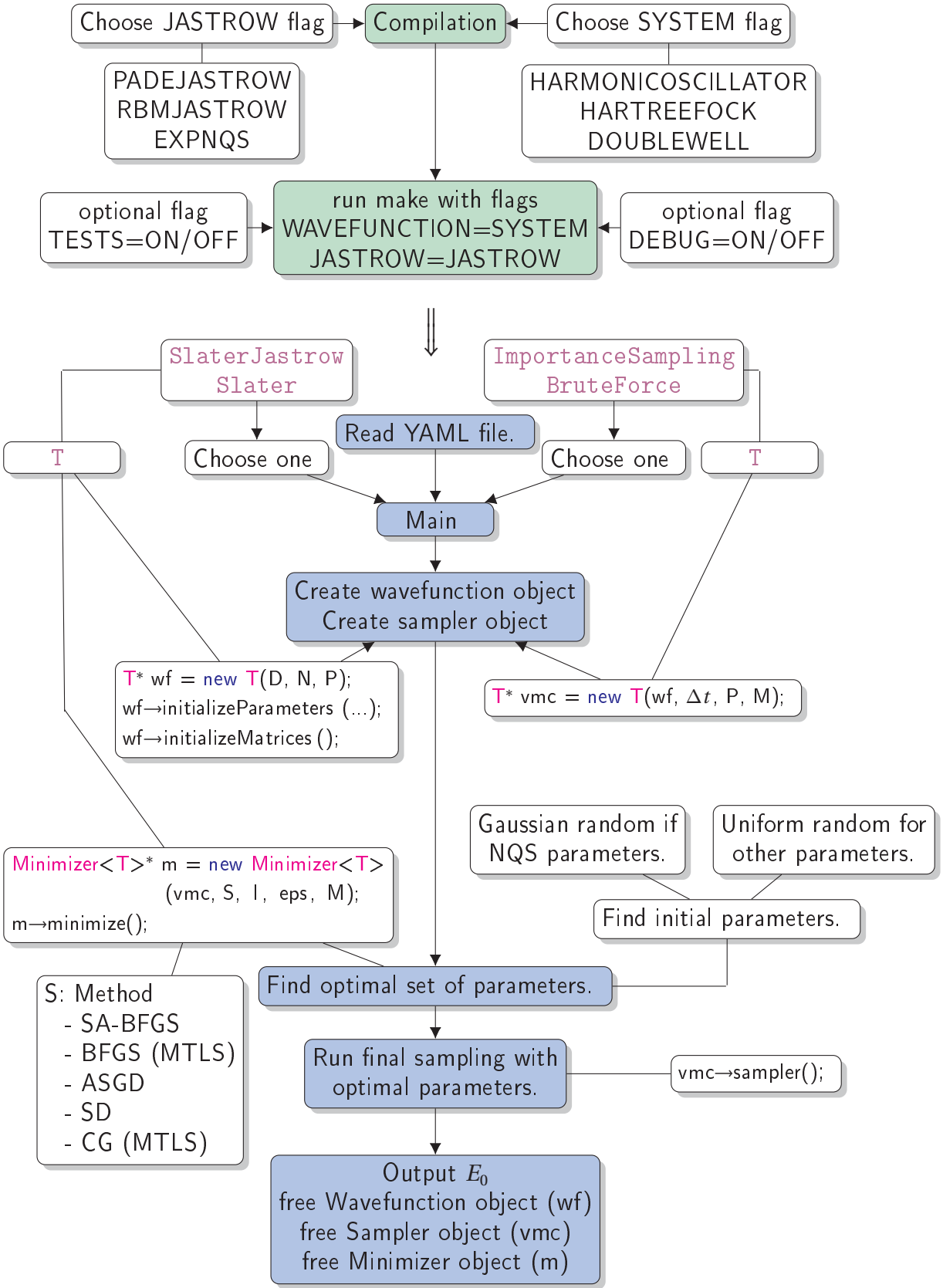
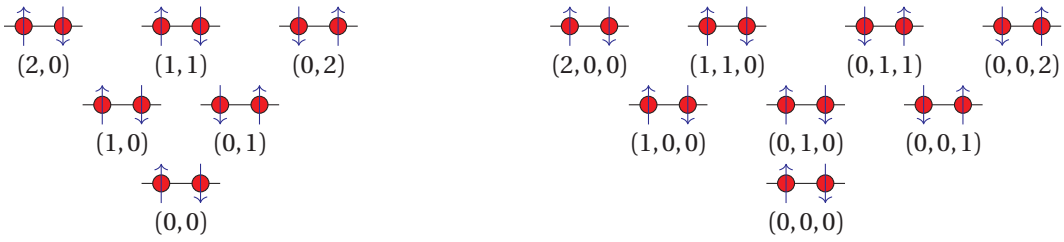


Figure 5.2: Flow chart of Variational Monte-Carlo implementation.

5.1 Cartesian Basis

In chapter 4 we mentioned the use of basis functions the different Many-Body methods. These can be pre-built using nifty intuition. One such observation is in the way harmonic oscillator functions station themselves on energy-levels(in the full-shell case). The following image[1] describes this for the first few levels



(a) 2D with (n_x, n_y) (b) 3D with (n_x, n_y, n_z)

Figure 5.3: Harmonic Oscillator Levels

This specific arrangement of basis-functions is implemented in class `Cartesian` and is used in both the Hartree-Fock and VMC implementations. It essentially builds a matrix of states with the rows being the specific state and the columns containing the quantum numbers(in cartesian), the spin-value(as an integer), magic number and energy(in natural units proportional to the oscillator frequency). The essential form is

$$\begin{pmatrix} n_x & n_y & s & m_s & E & M \end{pmatrix} \quad \begin{pmatrix} n_x & n_y & n_z & s & m_s & E & M \end{pmatrix} \tag{5.1}$$

with the n's being the principal numbers, s the spin value m_s the spin projection(up or down in our case), E the energy and M the magic number. All the numbers above are integers, meaning the actual energy E need to be converted if the actual energy of the state is desired, the same is applied to the spin projection(which is to multiply by $1/2$). The `Cartesian` class builds the states with alternating spin (the spacial parts are doubled with spin), but also has a function for restructuring by setting the states with spin down in acending order first and the same states with spin up after.

5.2 Hartree-Fock

Only the restricted case is implemented and is present as the class `HartreeFockSolver`. The matrix-elements(integrals) are implemented in `HermiteIntegrals` class. This class

[1] As the old idiom goes; "A picture is worth a thousand words"

also uses an auto-generated header for the Hermite-coefficients, see section 5.5 below. The `HartreeFockSolver` is implemented in a general way such that an abstract class for the integral elements is all that is needed. The `HartreeFockSolver` can then be inherited and used. An example of how to create a solver object with the Double-Well system called HFS with number of dimensions D, number of basis functions L and number of particles N.

```
DoubleWell* HFS = new DoubleWell(D, L, N);
string message = HFS->initializeParameters(...);
```

With the (...) meaning one initializes it with however manner the function was made. The initialization function must also return a message determined by the success of the initialization. If it succeeds it returns an empty message while if not it returns a pre-defined message.
Here is a simple example code-snippet which initializes and runs the Hartree-Fock algorithm

```
DoubleWell* HFS = new DoubleWell(D, L, N);

string message = HFS->initializeParameters(...);
if (message.compare("")) {
    if (myRank == 0) {
        std::cout << message << std::endl;
    }
    delete HFS;
    finalize();
}

double E = HFS->iterate(M, 1e-8, true);
```

The `iterate` function takes in M as the maximum number of iterations, the convergence tolerance (when to break the iteration) and a boolean for showing progress or not. It calculates the integral-elements and runs the Hartree-Fock algorithm and returns the estimation of the ground-state energy.

5.2.2 Recurrence Relation and Coefficients

The one-dimensional integrals ξ (recurrence relation in equation (3.49))

$$\begin{aligned} \frac{\partial I'_{2D}}{\partial A_x} &= -\frac{2\alpha\beta}{\alpha+\beta} X_{AB} \zeta_1 \left(\frac{\alpha\beta}{\alpha+\beta} R_{AB}^2 \right) \\ \xi_{00}^n &= \left(\frac{-2\alpha\beta}{\alpha+\beta} \right)^n \zeta_n \left(\frac{\alpha\beta}{\alpha+\beta} R_{AB}^2 \right) \end{aligned} \tag{5.2}$$

involved in the two-body elements and the coefficients equation (3.18)

$$\begin{aligned} E_t^{i+1,j} &= \frac{1}{2(\alpha+\beta)} E_{t-1}^{ij} - \frac{\beta}{\alpha+\beta}(A_x - B_x)E_t^{ij} + (t+1)E_{t+1}^{ij} \\ E_t^{i,j+1} &= \frac{1}{2(\alpha+\beta)} E_{t-1}^{ij} - \frac{\alpha}{\alpha+\beta}(A_x - B_x)E_t^{ij} + (t+1)E_{t+1}^{ij} \end{aligned} \tag{5.3}$$

involved in the overlap-distributions in the Hartree-Fock calculations are calculated using the class `Hexpander`. The coefficients can be calculated and tabulated once with the `setCoefficients` function. It is preferable to set all the needed coefficients once before calculating the elements[2]. The `Integral` classes all set these in the `initializeParameters` function based on the number of basis-functions used. For recurrence ξ we implemented a modified version of the three-dimensional version by Dragly[12] while the ζ integrals are calculated using Gauss-Chebyshev-quadrature[59]. These elements can also be tabulated in the same manner as with the coefficients using the function `setAuxiliary2D` and `setAuxiliary3D` respectively for two- and three dimensions.

Both the coefficients and integral elements can be obtained with the functions `coeff` and `auxiliary2D` and `auxiliary3D`.

In our setup the `Hexpander` class itself is inherited from the `GaussianIntegrals` class and used within.

5.2.2 Parallelization of Two-Body Matrix

The most time-consuming part of the Hartree-Fock procedure is the calculation of the two-body matrix-elements giving the interaction terms. This is parallelized in the `assemble` function in `HartreeFockSolver`. The basic premise is to represent the N^4 elements in $\left\langle pq \left| r^{-1} \right| rs \right\rangle$ as a one-dimensional array with the mapping

$$(p,q,r,s) \rightarrow p + N(q + N(r + Ns)). \tag{5.4}$$

Which is to say that the element (p,q,r,s) is stored in position $(p+N(q+N(r+Ns))$ in the one-dimensional array. The symmetry

$$(p,q,r,s) = (q,p,r,s) \tag{5.5}$$

reduces the number of elements down to $N(N+1)/2$. Notice also that the number of (r,s) elements each process needs to calculate is also this same size meaning the total size is actually

$$\text{totalsize} = \frac{N^2(N+1)^2}{4}. \tag{5.6}$$

[2]Such that precious computation hours are not wasted.

All the symmetries imply that the following elements are the same

$$\begin{gathered}(p,q,r,s)\\(r,q,p,s)\\(r,s,p,q)\\(p,s,r,q)\\(q,p,s,r)\\(s,p,q,r)\\(s,r,q,p)\\(q,r,s,p)\end{gathered} \tag{5.7}$$

A matrix pqMap of size $N(N+1)/2 \times 2$ is then created with elements

$$\text{pqMap}_{pq} = (p,q). \tag{5.8}$$

This is essentially just a matrix with each row being a tuple with p and q value.
The rows are then distributed evenly among P processes according to this equation (5.9),

$$\text{rows}_p = \begin{cases} \left\lceil \frac{\frac{N}{2}(N+1)}{P} \right\rceil & \text{rank} < \left(\frac{N}{2}(N+1) \mod P \right) \\ \left\lfloor \frac{\frac{N}{2}(N+1)}{P} \right\rfloor & \text{else} \end{cases} \tag{5.9}$$

The problem now however is that processes of higher and higher ranks my end up with calculating more since larger indices involve computationally more heavy functions to be evaluated. We can account for this by weighting the number of rows each process gets by the product[3] of the principal quantum numbers for the state which the indices represent, that is

$$S_i = \sum_{j=0}^{P_i} \prod_d (n_{j_d} + 1). \tag{5.10}$$

The sum with index j runs over the sub-chunk for process i where the size of each chunk is defined by equation (5.9). The algorithm is given in algorithm 3.

[3]The product is used since the loops are nested and run up to the given quantum number.

Algorithm 3 Even Weighting

Make an array sizes of size P with the number of elements for each process.
Make an array displ of size P with the displacement (index).
Make array S of size P with elements as sepcified in equation (5.10).
Set elements of sizes array to zero.
$O = \text{Floor}(\text{Mean}(S))$
$p = 0$ ▷ Index for process
$j_S = 0$ ▷ Total sum for each p
$k = 0$ ▷ Index for displacement in sizes array
$V_{\max} = O + 3n_{\max}^2$ ▷ $n_{\max}$ is the largest n-quantum number in the system.
for $l = 0$ to $l < \frac{N(N+1)}{2}$ **do** ▷ Iterate over rows in pqMap
  $j_S = j_S + \prod_d (n_{p_d}^{(l)} + 1)(n_{q_d}^{(l)} + 1)$
  $k \mathrel{+}= 1$
  if $j_S \geq V_{\max}$ **then** ▷ Make sure not to overshoot
    sizes$[p] = k - 1$ ▷ Discard the last element
    $l = l - 1$ ▷ Re-evaluate for next p
    $k = 0$
    $j_S = 0$
    $p \mathrel{+}= 1$
  else if $j_S > O$ **then** ▷ Chunk for process p is good.
    sizes$[p]$ = k ▷ update sizes
    $k = 0$
    $j_S = 0$
    $p \mathrel{+}= 1$
  else if $l = \frac{N(N+1)}{2} - 1$ and $j_S \leq O$ **then** ▷ Throw remaining rows at last
    sizes$[p] = k$
  end if
end for

Each process then gets its respective chunk of `pqMap` array such that the each process calculates its own chunk according to the size set. Each process then calculates the two-body elements with the (p, q) elements received. The total size for each process also needs to take the (r, s) elements into account as they are also of size $N(N+1)/2$. Each sub-chunk is then sent to root process and concatenated to a large one-dimensional array and the actual two-body matrix of size N^4 is assembled and antisymmetrized. The Hartree-Fock algorithm is then run only on one process.

5.2.2 Tabulation of Two-Body Matrix

The `HartreeFockSolver` class uses an input file for the two-body matrix if given and calculates and writes one out if not given. As of now this is done in a brute-force fashion, that is the entire matrix, including all zero-values, are written to file. This entire structure can be improved upon by introducing a *Sparse-Matrix* structure, which is a matrix in which only the non-zero elements

are stored and a displacement array for the given indices of the non-zero elements is created. This would decrease the every-increasing memory usage and reduce the calculation time as look-up time in the array.
The most promising road for implementing a Sparse structure is to use the SparseMatrix module in Eigen since the entire code-base already builds heavily upon Eigen from before.

5.3 Variational Monte Carlo

The Variational Monte-Carlo implementation is mainly in three classes, namely `VMC`, `BruteForce` and `ImportanceSampling`. The structure is set with `BruteForce` and `ImportanceSampling` both inheriting from `VMC`. This structure essentially gives room for splitting specific parts of the Brute-Force algorithm from the Metropolis-Hastings algorithm, but still using the same code for minimization. We will explain the minimization parts in the next section.
The main input which `VMC` needs is a wavefunction. An abstract class-template is implemented and can be generated using a python script. The template is built in such a way that one only needs to fill in specific analytic expressions for the gradient, Laplacian and gradient for the variational parameters. The latter part is optional.
The wavefunction itself is built using the `Slater` or `SlaterJastrow` class. However, in order to use the `SlaterJastrow` one has to specify which Jastrow function to use at compile time. A simple example illustrates this better

```
SlaterJastrow* wf = new SlaterJastrow(dim, numParticles, parameters);
wf->initializeParameters(omega);
wf->initializeMatrices();

ImportanceSampling<SlaterJastrow>* vmc = new
        ImportanceSampling<SlaterJastrow>(wf, stepmc, parameters,
                                          maxIterations, rank, numProcs);

double E = vmc->sampler();

delete vmc;
delete wf;
```

The first chunk initializes the SlaterJastrow class with the pre-specified system which needs to be given at compile time as a flag i.e WAVEFUNCTION=HARMONICOSCILLATOR. The second part sets the sampling to use Metropolis-Hastings algorithm. The parameters variable must be an Eigen[20] vector or array and contain the variational parameters. The representation of each element in this vector(or array) is specific to each system. This is just an example of how to build

a simple run, however we have also built a run file which uses YAML[10]. The basic template is as follows

```
omega: 1.0
numparticles: 6
maxitermc: 100000
stepmc: 0.01
parameters: [0.99, 0.47] #optional
numparameters: 2
jastrow: true #optional
importance: true #optional
```

This is a template used for the `HarmonicOscillator` with the Padé-function as Jastrow factor. If the NQS-function is used an additional parameter 'numhiddenbias' giving the number of hidden biases used, is needed.
This form of input makes it fairly simple to actually run the code for different systems with ease. One needs to compile with the specific flag for the system(wavefunction and Jastrow) and then supply an YAML input file at runtime.

5.3.3 Statistics

Due to the statistical procedure of using a stochastic diffusion process as our model there is an inherent variance introduced int the Metropolis sampling. In order to get a good estimate for this variance we used the method of blocking[16]. The implementation is within the class `Resampler` in which we have two methods, one called `blocking` which effectively reduced the block-sizes and saves the variance for each block-size and `autoblocking` which estimates the optimal block-size. The latter is taken from the thesis of Marius Jonsson[33] and rewritten in C++ using the Eigen library.

5.3.3 Thermalisation

When running the sampling the initial distribution of positions is not necessarily a good configuration and might corrupt the sample and give an unreasonably high variance. To avoid this we *thermalize* the system before calculating the local energies and the other local quantities. The procedure is as simple as to run the sampling, but not calculate the local energy or any other quantity until a certain number of iterations has been exhausted. This gives the particles enough movement to find a good configuration from which a nice sample can be obtained.

5.3.3 Slater Optimizations

In the Metropolis sampling we need access to the ratio of two determinants, namely the Slater wavefunction at the current state and the one at previous state. These are quite expensive to calculate[4]. This can be overcome by using the fact that moving only *one* particle at each iteration also constitutes to a state-transition. Following [27] and given row i as the index for the row that is changed, the following expressions are valid

$$\begin{aligned}\frac{\Psi}{\widetilde{\Psi}} &= \sum_j \psi_{ij}\widetilde{\psi}^{-1}_{ji} \\ \frac{\nabla_i\Psi}{\Psi} &= \sum_j \psi_{ij}\nabla_i\psi^{-1}_{ji} \\ \frac{\nabla_i^2\Psi}{\Psi} &= \sum_j \psi_{ij}\nabla_i^2\psi^{-1}_{ji}\end{aligned} \tag{5.11}$$

with $\widetilde{\Psi}$ being the wavefunction at previous state. The derivatives all use the wavefunction at current state. One might ask now, but isn't this just worse? We have gone from needing two determinants to needing two inverses[5]. Fret not, the Sherman-Morrison formula[50] for updating an inverse matrix if only one row has changed in the original matrix comes to the rescue. The elements of the inverse can by this be expressed as(with i being the row that changed)

$$\psi^{-1}_{kj} = \begin{cases} \widetilde{\psi}^{-1}_{kj} - \dfrac{\Psi}{\widetilde{\Psi}}\widetilde{\psi}^{-1}_{ki}\displaystyle\sum_{l=1}^{N}\psi_{il}\widetilde{\psi}^{-1}_{lj}, & j \neq i \\ \dfrac{\Psi}{\widetilde{\Psi}}\widetilde{\psi}^{-1}_{ki}\widetilde{\psi}^{-1}_{ki}\displaystyle\sum_{l=1}^{N}\widetilde{\psi}_{il}\widetilde{\psi}^{-1}_{lj}, & j = i \end{cases} \tag{5.12}$$

This means that the inverses and the determinant ratios can be calculated fully once before the sampling and then updated using the above formulas. This procedure is implemented in the `Slater` class.

5.3.3 Jastrow Optimizations

The Jastrow factors presented in section 2.9.9, section 2.9.9 and section 2.9.9 also give rise to optimizations in the case of moving only one particle at a time. In particular, the Padé function and the simple exponential can both be represented by a matrix of size $N \times D$ like this

$$\boldsymbol{J} = \Big(\boldsymbol{J}(\boldsymbol{r}_1)\ldots\boldsymbol{J}(\boldsymbol{r}_N).\Big) \tag{5.13}$$

[4] Actual complexity of a determinant is $\mathcal{O}(n \times n)$, with n being the size of the Slater matrix.
[5] !?.

Moving one particle at a time means only one column in $\boldsymbol{J}$ is changed meaning only that column needs to be updated. Additionally, the $\boldsymbol{J}$ matrix doesn't actually need to be present in the code since we are only interested in ratios $J/\widetilde{J}$. If only one index i changes we have

$$\begin{aligned}\frac{J}{\widetilde{J}} &= \exp\left(\sum_{i<j}\left(f_{ij}-\widetilde{f}_{ij}\right)\right)\\ &= \exp\left(\sum_{i\neq j} f_{ij}\right).\end{aligned} \tag{5.14}$$

Notice that i is fixed in the last step.
The gradient of J can be represented as an $N\times N\times D$ matrix

$$\nabla \boldsymbol{J} = \begin{pmatrix} 0 & \nabla \boldsymbol{J}(\boldsymbol{r}_{1,2}) & \dots & \nabla \boldsymbol{J}(\boldsymbol{r}_{1,N-1}) \\ \nabla \boldsymbol{J}(\boldsymbol{r}_{2,1}) & 0 & \dots & \nabla \boldsymbol{J}(\boldsymbol{r}_{2,N-1}) \\ \vdots & \vdots & \ddots & \vdots \\ \nabla \boldsymbol{J}(\boldsymbol{r}_{N,1}) & \dots & \dots & 0 \end{pmatrix} \tag{5.15}$$

When only one particle is moved at a time only one row and column changes in this matrix. In addition the matrix is symmetric the exception of a sign flip

$$\nabla \boldsymbol{J}(\boldsymbol{r}_{ij}) = -\nabla \boldsymbol{J}(\boldsymbol{r}_{ji}). \tag{5.16}$$

Both of these optimizations are implemented in the `PadeJastrow` and `ExpNQS` classes.

For the NQS-Jastrow we notice that the sum in the exponential involving the weights can be represented as a matrix $\boldsymbol{W}$ of size $N\times H$ with H being the number of hidden biases with elements

$$W_{i,j} = \sum_d \frac{x_i^{(d)} w_{i+d,j}}{\sigma^2}. \tag{5.17}$$

The entire exponential with the hidden biases is represented as a vector $\boldsymbol{B}$ of size H with elements

$$B_j = \exp\left(b_j + \sum_i W_{i,j}\right). \tag{5.18}$$

So for each iteration only one row in $\boldsymbol{W}$ is updated and then the entire $\boldsymbol{B}$ vector is recalculated and reused.
The part involving only the visible biases can be optimized in the same manner as with the Padé function and simple exponential, meaning one only needs to calculate

$$\frac{J_a}{J'_a} = \exp\left(\frac{(r_i-a_i)^2}{\sigma^2}\right). \tag{5.19}$$

Again with i being the index of the moved particle. These optimizations are implemented in the `RBMJastrow` class.

5.3.3 Optimization For Tabulation

The `Slater` class checks on compile time the specific wavefunction class for the functions

- `set`: Called during initialization (before each sampling)
- `reSetAll`: Sets all matrices to zero (used in testing)
- `initializeMatrices`: Allocate memory
- `update`: Update positions and wavefunction
- `reset`: Revert to previous positions and wavefunction
- `resetGradient`: Revert to previous gradient
- `acceptState`: Update previous positions and wavefunction to current
- `acceptGradient`: Update previous gradient to current one

If these functions are implemented in the wavefunction class they will be called in by the `VMC` class during the Metropolis sampling. The whole purpose for this is so that calculations can be tabulated and then only the parts which are changed be updated. Detailed explanation of what the functions can, and need to, do is given in the GitHub repository.

5.3.3 Tabulating Hermite Polynomials

In the `HarmonicOscillator` and `HartreeFock` classes used by `VMC` we calculate the Hermite polynomials

$$\begin{aligned} H_n(\sqrt{\omega}r) &= \prod_d H_{n_d}\left(\sqrt{\omega}x_d\right) \\ H_n(\sqrt{\omega}r) &= \prod_d H_{n_d-1}\left(\sqrt{\omega}x_d\right) \\ H_n(\sqrt{\omega}r) &= \prod_d H_{n_d-2}\left(\sqrt{\omega}x_d\right) \end{aligned} \tag{5.20}$$

for $n = 0, \ldots L$ where L is the number of basis functions in the Hartree-Fock basis for `HartreeFock` and highest order of single-particle function for `HarmonicOscillator`. These are tabulated in a matrix of size $N \times D \times L$ with N being the number of particles and D. The mapping goes by

$$\boldsymbol{H}_{ijk} = H_{n_j^{(d)}}\left(\sqrt{\omega}x_i^{(d)}\right). \tag{5.21}$$

For each iteration in the Metropolis sampling, if we only move one particle at a time, only one row changes in $\boldsymbol{H}_{ijk}$. This update is reflected in the functions mentioned in section 5.3.3. `initializeMatrices` firstly allocates space for two matrices which represent the current and

old versions of $\boldsymbol{H}_{ijk}$, the `set` function set calculates $\boldsymbol{H}_{ijk}$ using the auto-generated header explained in section 5.5, `update` then updates the row which has changed and then `acceptState` or `reset` is called depending on the Metropolis-Test.

5.4 Minimization

The central part of the Variational Monte-Carlo method is the actual *variation* of parameters. This is, as explained in chapter 3, a minimization problem. We will in this chapter explain the implementation of the methods presented in chapter 5 and also show some examples of how to use `Minimizer` class, in which the mentioned methods are implemented.
The main class for minimizations is the `Minimizer` and is built as a *friend class*. This is a C++ declaration[6] which goes into the header of the class who is to befriended (in our case `VMC`) with the class who is to be the friend (`Minimizer`). In this way `Minimizer` has full access to all the variables and functions defined inside `VMC` even if said variables and functions are declared as *private*[7]. The reason for this choice is just to split the whole minimization part of the variational method from the actual sampling.
The method are implemented within the following functions

- `minimizeSD` Gradient Descent.
- `minimizeBFGS` Quasi-Newton with BFGS and Moré-Thuente linesearch.
- `minimizeCG` Quasi-Newton with Polak-Ribière and Moré-Thuente linesearch.
- `minimizeASGD` Stochastic-Adaptive-Gradient-Descent
- `minimizeSABFGS` Stochastic-Adaptive method with BFGS.
- `minimizeSIAN` Simulated Annealing.

The specific function is called based on the initial string given to `Minimizer` and the means to to make a `Minimizer` object is to initialize it as

```
Minimizer<T>* m = new Minimizer<T>(vmc, "method", numIterations, eps);
m->minimize();
```

[6] A C++ declaration is a phrase which goes in front of variables, functions and other objects and is a pure compiler specific rule.

[7] On an equal note, this is truly not a good idea in general and great care, inner reflection and utmost scepticism must be within the procedural motion of ones thought process before even the idea of running forth with such a choice is grasped. It is up for discussion if mentioned procedures were completed before the calamity that is `Minimizer` was implemented.

where T is a sampler type (either BruteForce or ImportanceSampling) and vmc is the object with said type initializes. The minimize function is the function which actually does the minimization and follows the procedure given in figure 5.4.

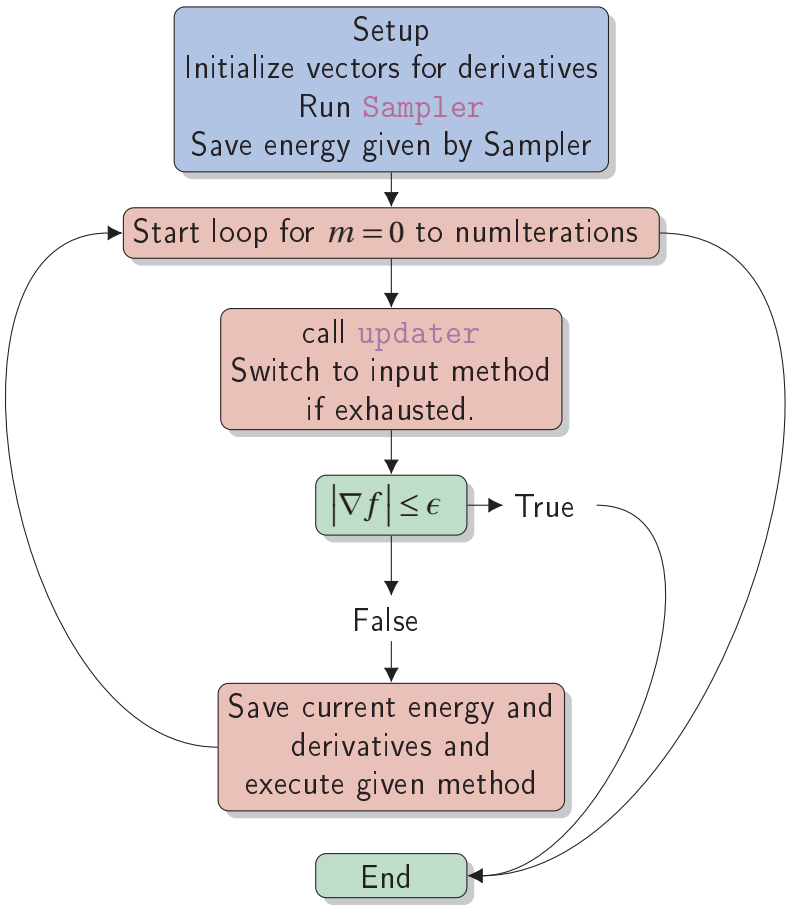


Figure 5.4: Flow chart of procedure done by minimize function.

The energy after the last cycle (or after threshold is met) is the energy currently saved within the VMC object.

5.5 Auto-Generation

5.5.5 Calculation of Hermite Polynomials

The analytic expressions involved in the quantum-dot systems are dependent on Hermite polynomials and their coefficients. These are calculated symbolically using the recurrence relation for Hermites with the SymPy package in python. These expressions are then written to C++ code and written to a C++ -header file. The script can be found in https://github.com/Oo1Insane1oO/

Hermite. The generated header file is then included in the integral class in `HartreeFockSolver` and in the wavefunction classes(namely `HartreeFock` and `HarmonicOscillator`) in `VMC`.

5.5.5 Double-Well Basis

The basis built by the procedure described in section 3.4 is done in python using the eigenvalue solver in the SciPy package. The script itself can be found in https://github.com/Oo1Insane1oO/doubleWellFit, but is also included in the Hartree-Fock repository.
The script only builds(calculates the elements of) the H matrix

$$H_{ij} = \varepsilon_i^{\text{HO}}\delta_{ij} + \left\langle \psi_i^{\text{HO}} \middle| V_n^{\text{DW}} \middle| \psi_j^{\text{HO}} \right\rangle, \tag{5.22}$$

from equation (3.58) and then finds its eigenvalues and eigenvectors. These are then written to a C++ header file[8] as a class with a static function[9] to get coefficients and eigenvalues. This file is then included in the `DoubleWell` integral class and inherited from.

5.6 Verification

In order to be confident in the results presented for the double-well potential we, along with *unit-tests*, benchmarked our results with the harmonic oscillator potential. The tests are set-up with the UnitTest++ [9] library and one only needs to pass a flag $TESTS = ON$ at compile time to enable them. The benchmark results are given in tables 1, 2, 5, 6, 7, 8, 9 and 10.

5.6.6 Hartree-Fock

The first of the test for the Hartree-Fock method was to see if it reproduced the exact energies in the case of no interaction(neglect Coulomb-term in Hamiltonian). These tests are given in tests directory in the GitHub repository.

The benchmarks using the Hartree-Fock method with harmonic oscillator potential is given in table 1. These results are directly compared with the master's thesis [38] and article [45].

For the double-well system the only benchmark was the result from master's thesis of Høgberget[28] with 2 particles. However it is expected that the energies lie somewhere between the single-well results and the non-interaction case due to the potential itself being a perturbation of the single-well system. The results are given in tables 3, 4, 11, 13, 12 and 14.

[8] Hard-coding the smart way
[9] Static means the function does not need an object of the container class to be called.

5.6.6 Variational Monte-Carlo

The first tests here were also to find the exact energies, which are also given as tests in a similar fashion as with the Hartree-Fock. The second benchmark was to check if the energies were comparable with the results of [28] using basis functions of form

$$\phi_j(\boldsymbol{r}) = \psi^{\mathrm{HO}}_{n_j}\left(\sqrt{\alpha\omega}\, r\right), \tag{5.23}$$

with the Padé function given in equation (2.84).

For the real deal with the Hartree-Fock basis we checked that the VMC method could reproduce the Hartree-Fock energies when no Jastrow factor was used, that is only a single-particle Slater-determinant *with no variational parameters* as the wavefunction ansatz. The expression is

$$\phi^{\mathrm{HF}}_{ij} = \sum_l C_{jl}\psi^{\mathrm{HO}}_{n_l}\left(\sqrt{\omega} r_i\right). \tag{5.24}$$

These results are also implemented as a bunch of tests.

For the benchmarks, we compared our energies again with the thesis of Høgberget[28].

For the double-well system we only used a Hartree-Fock basis and checked that the Hartree-Fock energies were reproduced (in case with no Jastrow). The only benchmark was again the system with two particles.

CHAPTER 6

RESULTS

No amount of physics theory will ever be interesting if it was not backed up by some proof or at least some results.
In this chapter we present the results from the simulations of the quantum dot system in the harmonic oscillator system and the double-well system. The harmonic oscillator was used mainly as a benchmark for verification for which is performed well. We also present some results regarding the minimization scheme in the variational method. All the simulations with the variational method was done with importance sampling as well.

6.1 Tweaks and Experimentation

With the minimization methods clarified and the function to be minimized outlined the actual minimization could start[1].
Within the minimization methods presented there are quite the number of constant parameters which needed to be set pre-hand. Often one finds these some-what manually with a close eye on the function to be minimized. This approach is the one we used in the VMC method, with the most memorable one being the parameters set in the simulated annealing method and the notorious step-size in the Metropolis sampling.

[1] With the methods at disposal the *real challenge* was yet to come...

6.1.1 Hartree-Fock

With the Hartree-Fock method the only part tweaked was the number of previous quantities to use in the DIIS procedure. Essentially just changing M in (equation (2.43))

$$\boldsymbol{Y} = \sum_{m=1}^{M} c_m \boldsymbol{y}_i. \tag{6.1}$$

The exact number was changed to get convergence, however somewhere between $M = 2$ to $M = 8$ gave good results.

6.1.1 Variational Monte-Carlo

For the VMC runs the only parameter to be tweaked was the step-size Δt which had to be tweaked depending on ω and the number of particles. The general rule was to increase Δt when ω was lowered, this has to do with the form of the potential, see figure 2.1a. For N, the number of particles, the step had to be slightly increased as N increased, this is due to the repulsion part being stronger with more particles.
We also mention that the only criteria for convergence was the absolute value of the derivative

$$|\nabla_{\boldsymbol{\alpha}} \langle E[\boldsymbol{\alpha}; \boldsymbol{R}] \rangle| < \epsilon \tag{6.2}$$

With $\epsilon \in (0.1, 0.001)$ depending on the size of the system. The general idea was to look for sign oscillations of the gradient with respect to the variational parameters. If it started to oscillate the maximum step-size in the quasi-newton scheme was reduced(the maximum allowed in the case with the Moré-Thuente linesearch) and looked to see if any improvement was made.

Simulated Annealing

With the simulated annealing the idea was to use it as a sort of "thermalization", that is to run it initially in hope that it finds the valley within the function-mesh in which the global minima lies. The approach was to run a great number of iterations starting with a temperature of 100. In order to finish those runs in time they were run in parallel and the parameters that gave the lower energy and lowest where then used as a starting point for the quasi-newton method which was then run with a close eye on the energies and variance[2].

RBM

For the RBM-Jastrow we did not tweak as much manually, but we initialized the parameters in the RBM-wavefunction with a Gaussian distribution with mean 0 and variance between $0.01-1.0$ and

[2]Coined as the VBS method, VMC-babysitting.

settled with a value that gave a fairly good acceptance rate and variance in the sampling without the energies being way off the results with the Padé-function.

6.2 Hartree-Fock

6.2.2 Harmonic Oscillator

Table 1: RHF with the 2D HO-system. L is the number of spacial orbitals used and N is the number of particles. The "—" indicates that the system cannot be run with such few basis functions.

		N						
ω	L	2	6	12	20	30	42	56
	1	0.596333	—	—	—	—	—	—
	3	0.596333	4.864244	—	—	—	—	—
	6	0.526903	4.435740	17.272337	—	—	—	—
	10	0.526903	4.019787	15.358377	43.303270	—	—	—
	15	0.525666	3.963148	14.098239	38.031297	89.280189	—	—
	21	0.525666	3.870617	13.700447	35.572157	78.990503	162.260603	—
	28	0.525635	3.863135	13.270861	34.076983	71.159148	144.850714	269.962677
0.10	36	0.525635	3.852880	13.151070	32.907610	70.171143	135.399113	242.763519
	45	0.525635	3.852591	13.000151	32.379047	67.961359	128.642142	227.093062
	55	0.525635	3.852393	12.969872	31.823087	66.272349	124.116636	216.406591
	66	0.525635	3.852391	12.933578	31.606556	65.024272	120.477371	207.859610
	78	0.525635	3.852382	12.929215	31.359747	64.244714	92.494105	162.039924
	91	0.525635	3.852381	12.924947	31.280274	63.531909	90.361587	153.361100
	105	0.525635	3.852381	12.924756	31.190173	63.167691	89.034638	148.084050
	120	0.525635	3.852381	12.924659	26.316806	62.778269	88.267263	144.999311
	1	1.223192	—	—	—	—	—	—
	3	1.223192	9.266117	—	—	—	—	—
	6	1.141775	8.725018	32.056852	—	—	—	—
	10	1.141775	8.139719	29.582103	79.220310	—	—	—
	15	1.141741	8.095876	27.596111	72.011644	161.787811	—	—
	21	1.141741	8.021956	27.194900	67.907357	146.805026	292.019496	—
	28	1.141717	8.020571	26.722532	66.336613	139.073038	265.770363	483.281000
0.28	36	1.141717	8.019625	26.651149	64.754792	134.380347	252.164864	441.883026
	45	1.141713	8.019611	26.559697	64.309088	131.153627	242.160894	418.703745

	55	1.141713	8.019571	26.554432	63.805612	129.589656	236.417148	402.296590
	66	1.141712	8.019571	26.550045	63.695333	128.063933	231.961536	391.319694
	78	1.141712	8.019570	26.550035	63.567267	127.516181	229.033490	383.057251
	91	1.141712	8.019570	26.550031	63.552773	126.918914	227.252510	377.466395
	105	1.141712	8.019569	26.550027	63.539438	126.748572	225.641452	373.080210
	120	1.141712	8.019569	26.550025	63.539007	126.559672	224.937324	303.460780
	1	1.886227	—	—	—	—	—	—
	3	1.886227	13.640713	—	—	—	—	—
	6	1.799856	13.051620	46.361130	—	—	—	—
	10	1.799856	12.357471	43.663267	113.412648	—	—	—
	15	1.799748	12.325128	41.108851	105.288766	230.039825	—	—
	21	1.799748	12.271499	40.750512	99.754600	212.145519	413.129302	—
	28	1.799745	12.271375	40.302719	98.193478	202.100349	380.824889	681.044994
0.50	36	1.799745	12.271361	40.263752	96.553216	197.568022	363.766695	631.885017
	45	1.799743	12.271337	40.216688	96.223206	193.554142	352.630536	601.088912
	55	1.799743	12.271326	40.216252	95.833317	192.225626	345.721160	581.182893
	66	1.799742	12.271324	40.216195	95.785792	190.810227	341.838273	568.944986
	78	1.799742	12.271320	40.216165	95.734582	190.462448	338.512355	559.966285
	91	1.799742	12.271320	40.216144	95.733305	190.069482	337.206237	554.039857
	105	1.799742	12.271320	40.216143	95.732781	190.007169	335.822963	550.259738
	120	1.799742	12.271320	40.216139	95.732764	189.942874	335.425547	547.164438
	1	3.253314	—	—	—	—	—	—
	3	3.253314	22.219813	—	—	—	—	—
	6	3.162691	21.593198	73.765549	—	—	—	—
	10	3.162691	20.766919	70.673849	177.963297	—	—	—
	15	3.161921	20.748402	67.569930	167.899811	357.543694	—	—
	21	3.161921	20.720257	67.296869	161.339721	336.270048	637.559628	—
	28	3.161909	20.720132	66.934745	159.958722	322.684662	597.572501	1045.153168
1.00	36	3.161909	20.719248	66.923094	158.400172	318.435444	574.794727	979.581293
	45	3.161909	20.719248	66.912244	158.226030	314.080027	563.971458	943.145991
	55	3.161909	20.719217	66.912035	158.017667	313.170733	555.393205	920.685095
	66	3.161909	20.719215	66.911365	158.010276	312.139011	552.472493	906.165846
	78	3.161909	20.719215	66.911364	158.004951	312.010418	549.388384	898.709307
	91	3.161908	20.719215	66.911323	158.004757	311.869364	548.688114	892.082465
	105	3.161908	20.719215	66.911322	158.004317	311.863887	547.907494	889.765470
	120	3.161908	20.719215	66.911321	158.004315	311.860399	547.801022	887.301998

Table 2: RHF with the 3D HO-system. L is the number of spacial orbitals used and N is the number of particles. The "—" indicates that the system cannot be run with such few basis functions.

		N			
ω	L	2	8	20	40
0.10	1	0.552355	—	—	—
	4	0.552355	7.035821	—	—
	10	0.529067	6.630816	37.165887	—
	20	0.694713	6.005329	33.059287	130.747321
	35	0.529066	5.960234	30.287665	112.917323
	56	0.529066	5.872465	27.977530	104.463835
	84	0.529065	5.868615	28.424272	92.416262
	120	0.529065	5.862549	28.165123	85.209104
0.28	1	1.262270	—	—	—
	4	1.262270	13.801205	—	—
	10	1.237384	13.307332	68.950504	—
	20	1.237384	12.483549	63.643967	235.682314
	35	1.237229	12.455162	59.237015	210.630043
	56	1.237229	12.399375	58.417453	197.754907
	84	1.237225	12.398982	57.328784	190.564893
	120	1.237225	12.398687	57.201823	186.064771
0.50	1	2.064282	—	—	—
	4	2.064282	20.707652	—	—
	10	2.038858	20.174413	99.689043	—
	20	2.038858	19.243279	93.747893	333.818879
	35	2.038523	19.225643	88.237201	304.701805
	56	2.038523	19.191660	87.541531	287.355799
	84	2.038509	19.191654	86.506890	280.356812
	120	2.038509	19.191626	86.439181	274.994258
1.00	1	3.798016	—	—	—
	4	3.798016	34.557120	—	—
	10	3.772173	33.996867	158.555190	—
	20	3.772173	32.945301	152.252978	516.025169
	35	3.771606	32.937947	145.038663	481.710199
	56	3.771606	32.925002	144.518526	458.308755
	84	3.771574	32.924935	143.704615	452.880282
	120	3.771574	32.924628	143.684577	446.315149

6.2.2 Double-Well

For the double-well we used the larges basis-set from the Hartree-Fock calculations, which was 120 spacial functions. Meaning the sum over l in (equation (3.54))

$$\left|\psi_p^{\mathrm{DW}}\right\rangle = \sum_l C_{lp}^{\mathrm{DW}} \left|\psi_l^{\mathrm{HO}}\right\rangle, \tag{6.3}$$

runs up to 120 while the number of basisfunctions used is chosen just as usual[3]. Keep in mind that the degeneration is lifted in the double-well system. The resulting energies for are given in tables 3 and 4.

Table 3: Energies for the 2D double-well system with $R = 2.0$ and 120 spacial HO-functions.

		N					
ω	L	2	4	6	8	10	12
1.00	2	2.614045	—	—	—	—	—
	4	2.572112	8.449791	—	—	—	—
	6	2.572112	8.343256	17.448309	—	—	—
	8	2.551560	8.266869	17.255396	29.038820	—	—
	10	2.550609	8.186828	17.123528	28.460691	42.176859	—
	12	2.550609	8.186828	16.953490	28.160636	41.610074	58.282067
	14	2.550565	8.185889	16.948999	28.073802	41.188306	57.685346
	16	2.550565	8.176024	16.906547	28.009987	41.065445	56.993033
	18	2.550282	8.175819	16.902102	27.988350	41.032690	56.876960
	20	2.550147	8.175781	16.898007	27.922423	40.909824	56.692874
	22	2.550147	8.175730	16.897306	27.919717	40.863587	56.603646
	24	2.550127	8.175667	16.897284	27.910955	40.836082	56.547895
	26	2.550127	8.175135	16.895412	27.901747	40.813443	56.374309
	28	2.550126	8.175118	16.895407	27.900143	40.807591	56.366349
	30	2.550126	8.175034	16.895385	27.900003	40.790316	56.316550
	32	2.550126	8.175026	16.895354	27.899753	40.788743	56.311534
	34	2.550107	8.175015	16.895257	27.898457	40.784378	56.298971
	36	2.550107	8.175015	16.895217	27.898456	40.783153	56.297067
	38	2.550102	8.175015	16.895162	27.898368	40.782215	56.291868
	40	2.550101	8.175015	16.895156	27.898343	40.781392	56.289293
	42	2.550101	8.174999	16.895150	27.898275	40.781311	56.279636
	44	2.550101	8.174999	16.894978	27.898136	40.781241	56.279533
	46	2.550101	8.174994	16.894976	27.898086	40.781199	56.277412
	48	2.550101	8.174994	16.894971	27.898061	40.781090	56.277407
	50	2.550101	8.174989	16.894969	27.897886	40.780997	56.277117
	52	2.550101	8.174989	16.894967	27.897871	40.780950	56.277028
	54	2.550101	8.174988	16.894966	27.897828	40.780876	56.276991
	56	2.550100	8.174987	16.894963	27.897827	40.780821	56.276945

[3]Keep increasing until Hatree-Fock limit is reached.

Table 4: Energies for the 3D double-well system with $R = 2.0$ and 120 spacial HO-functions.

		N				
ω	L	2	4	8	10	14
	2	3.298116	—	—	—	—
	4	3.298116	9.398891	—	—	—
	5	3.285820	9.331379	—	—	—
	7	3.285820	9.331379	—	—	—
	8	3.285820	9.289217	29.378674	—	—
	11	3.276743	9.252427	29.211193	42.323259	—
1.00	13	3.276743	9.252427	29.040386	42.097755	—
	16	3.276743	9.219465	28.922935	41.928041	73.165457
	17	3.276685	9.217823	28.905024	41.605395	72.629768
	19	3.276685	9.217823	28.905024	41.605395	72.226197
	23	3.276685	9.217823	28.593576	41.113311	71.455889
	26	3.276609	9.217740	28.589614	41.023458	71.271105
	30	3.276609	9.217740	28.589614	41.023458	70.646966

6.3 VMC

Here are the results for the quantum-dot simulations with the variational method both with and without a Hartre-Fock basis. We only used a basis as large as the one needed to reach the Hartree-Fock limit[4] or up to the largest one run in case the limit was not reached. The exact numbers are seen in tables 1 and 2. The first test of the code was to reproduce the results of [28, 45] using

$$\Psi_T = \det(\mathbf{\Phi}) J_{\text{Padé}} \tag{6.4}$$

as the trial wavefunction where

$$\Phi_{ij} = \prod_d H_{n_{jd}}\left(\sqrt{\alpha\omega}\, x_{id}\right) \exp\left(-\frac{\alpha\omega}{2} x_{id}^2\right) \tag{6.5}$$

and

$$J_{\text{Padé}} = \exp\left(\sum_{i<j} \frac{a_{ij} r_{ij}}{1+\beta\, r_{ij}}\right). \tag{6.6}$$

Results are presented in table 5

[4] See section 2.8.8

6.3.3 Two-Dimensional Harmonic Oscillator

Table 5: Energies of VMC calculation of two dimensional harmonic oscillator using equation (6.4) as the trial wavefunction. Number of Monte-Carlo samples used is $2^{20} = 1048576$. Refs. F. Pederiva [45].

ω[a.u]	N			
	2	6	12	20
0.1	0.4407(4)	3.5650(4)	12.3164(4)	30.0480(4)
0.28	1.0020(4)	7.6198(4)	25.5948(3)	61.8090(3)
0.5	1.6650(4)	11.8017(4)	39.3166(3)	93.9240(2)
1.0	3.0000(5)	20.2863(3)	68.1465(3)	156.2778(2)

Table 6: Energies of two dimensional harmonic oscillator using basis built with Hartree-Fock. Number of Monte-Carlo samples used is $2^{20} = 1048576$. The numbers inside the curly brackets indicate the number at which the basis expansion was truncated(the Hartree-Fock limit).

ω[a.u]	N			
	2	6	12	20
0.1	0.46552(5){15}	3.70137(4){36}	12.64342(4){91}	-
0.28	1.04939(4){6}	7.89627(4){36}	26.21301(4){66}	62.93503(5){120}
0.5	1.70130(4){6}	12.02776(4){21}	39.76442(3){45}	95.21976(3){91}
1.0	3.05625(4){6}	20.45876(3){36}	66.37115(3){45}	157.41119(3){78}

The energies are consistently higher than with the wavefunction in equation (6.4). This only shows that the single-particle wavefunctions constructed from Hartre-Fock are actually not as good of a guess on the trial-wavefunction. Introducing a similar α parameter in the Hartree-Fock basis as

$$\psi_p^{\mathrm{HF}}\left(\sqrt{\alpha\omega}\boldsymbol{r}\right) = \sum_l C_{lp}\psi_l^{\mathrm{HO}}\left(\sqrt{\alpha\omega}\boldsymbol{r}\right) \tag{6.7}$$

reduces the energies further. The results are presented in table 7. This is basically taking the constructed basis from the Hartree-Fock simulations and evaluation the function with a variational parameter inspired from the same approach as with the results in table 5.

Table 7: Energies of two dimensional harmonic oscillator using basis built with Hartree-Fock along with variational parameter α as in equation (6.7). Number of Monte-Carlo samples used is $2^{20} = 1048576$. The numbers inside the curly brackets indicate the number at which the basis expansion was truncated(the Hartree-Fock limit).

ω[a.u]	N			
	2	6	12	20
0.10	0.44473(5){15}	3.63897(4){36}	12.46408(4){91}	—
0.28	1.04978(4){6}	7.72929(4){36}	25.96595(4){66}	62.65652(3){120}
0.50	1.66418(4){6}	11.97781(4){21}	39.57182(3){45}	94.76303(3){91}
1.00	3.00624(4){6}	20.38811(3){36}	66.28996(3){45}	157.46167(3){78}

As mentioned, the energies are consistently lower, however they are not as low as with the results with equation (6.4)(table 5). This again means that for the harmonic oscillator case the optimal trial wavefunction is indeed not the Hartree-Fock basis, but the harmonic oscillator functions. The energies are however very close giving a good foundation for believing the code works properly as intended.

6.3.3 Three-Dimensional Harmonic Oscillator

Here are the same results for the three-dimensional case.

Table 8: Energies of VMC calculation of three dimensional harmonic oscillator using equation (6.4) as the trial wavefunction. Number of Monte-Carlo samples used is $2^{20} = 1048576$.

ω[a.u]	N	
	2	8
0.1	0.50006(5)	5.80479(4)
0.28	1.20156(5)	12.48178(4)
0.5	2.00027(5)	19.33356(4)
1.0	3.72985(5)	33.30958(4)

Table 9: Energies of three dimensional harmonic oscillator using basis built with Hartree-Fock. Number of Monte-Carlo samples used is $2^{20} = 1048576$. The numbers inside the curly brackets indicate the number at which the basis expansion was truncated(the Hartree-Fock limit).

ω	N	
	2	8
0.1	0.51122(5){70}	5.87372(4){120}
0.28	1.21844(5){70}	12.36177(4){168}
0.5	2.02030(4){20}	19.15006(4){112}
1.0	3.72918(5){20}	33.58046(4){168}

Table 10: Energies of three dimensional harmonic oscillator using basis built with Hartree-Fock along with variational parameter α as in equation (6.7). Number of Monte-Carlo samples used is $2^{20} = 1048576$. The numbers inside the curly brackets indicate the number at which the basis expansion was truncated(the Hartree-Fock limit).

ω	N	
	2	8
0.1	0.50751(5){70}	5.84082(4){240}
0.28	1.20320(5){20}	12.37435(4){168}
0.5	2.01439(4){20}	19.09917(4){112}
1.0	3.72959(5){70}	33.04162(4){168}

Again the energies are higher than with equation (6.4), meaning the same conclusion for the optimal wavefunction as with the two-dimensional case is still the case for the three-dimensional case as well.

6.3.3 Two-Dimensional Double-Well

Table 11: Energies for the 2D double-well system with a Hartree-Fock basis and $R = 2.0$ using 120 basis function in the eigenvalue problem and the Padé Jastrow as correlation function. The numbers inside the curly braces indicate the number of shells used in the Hartree-Fock. Number of Monte-Carlo samples used was $2^{20} = 1048576$.

ω	N			
	2	4	6	8
1.0	2.42238(4){10}	7.95247(4){42}	16.61419(4){44}	27.54453(3){50}

Table 12: Energies for the 2D double-well system with a Hartree-Fock basis and $R = 2.0$ using 120 basisfunctions in the eigenvalue problem and an α and Padé-function as correlation. The numbers inside the curly brackets indicat ethe number at which the basis expansion was truncated(the Hartree-Fock limit). Number of Monte-Carlo samples used is $2^{20} = 1048576$.

ω	N			
	2	4	6	8
1.0	2.36618(4){10}	7.90232(4){42}	16.55609(4){44}	27.58524(4){50}

6.3.3 Three-Dimensional Double-Well

Table 13: Energies for the 3D double-well system with a Hartree-Fock basis and $R = 2.0$ using 120 basis function in the eigenvalue problem and the Padé Jastrow as correlation function. The numbers inside the curly braces indicate the number of shells used in the Hartree-Fock. Number of Monte-Carlo samples used was $2^{20} = 1048576$.

ω	N		
	2	4	8
1.0	3.25118(4){11}	9.17489(4){17}	28.49671(4){26}

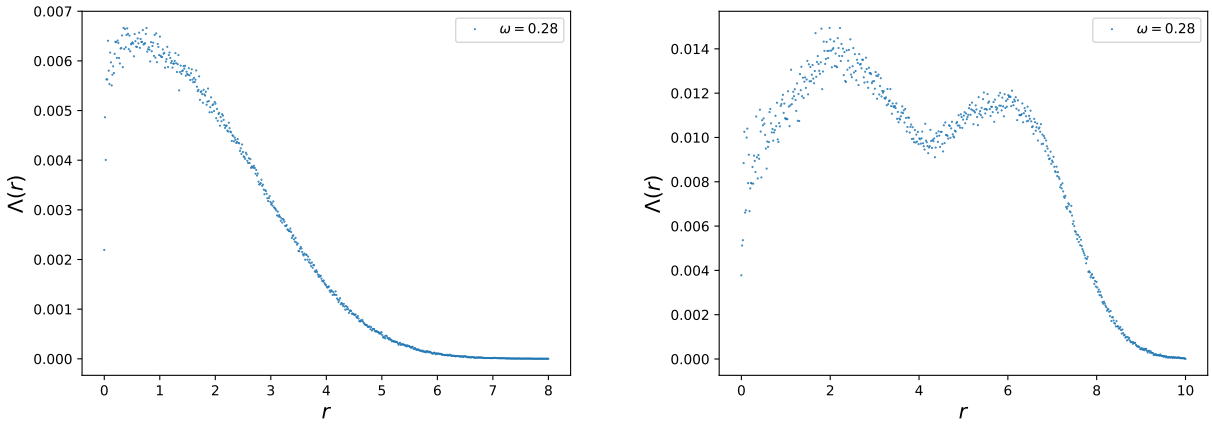
Table 14: Energies for the 3D double-well system with a Hartree-Fock basis and $R = 2.0$ using 120 basis function in the eigenvalue problem and and α parameter and the Padé Jastrow as correlation function. The numbers inside the curly braces indicate the number of shells used in the Hartree-Fock. Number of Monte-Carlo samples used was $2^{20} = 1048576$.

ω	N		
	2	4	8
1.0	3.22226(4){11}	9.17013(4){17}	28.62826(4){26}

6.3.3 Densities

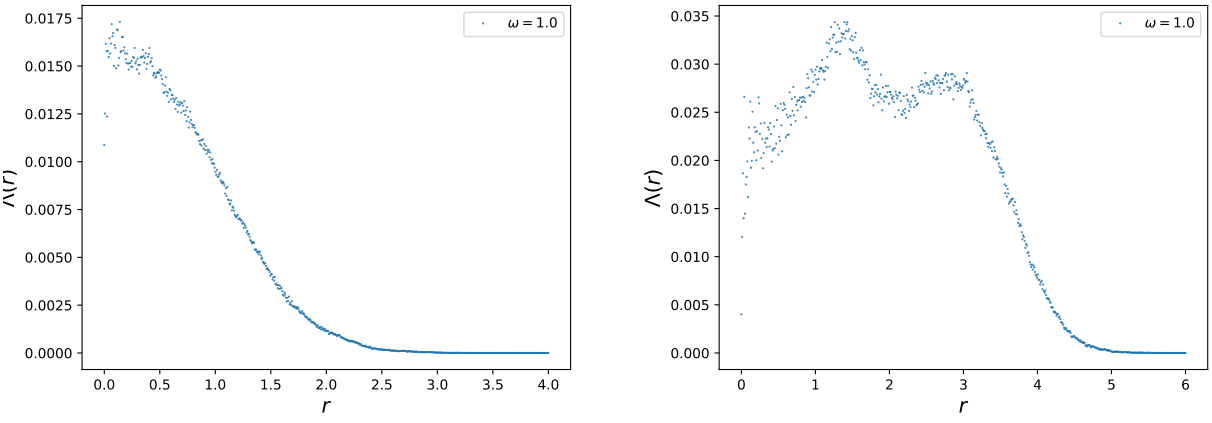
The resulting one-body radial densities described in section 2.11 are presented here.

Harmonic Oscillator



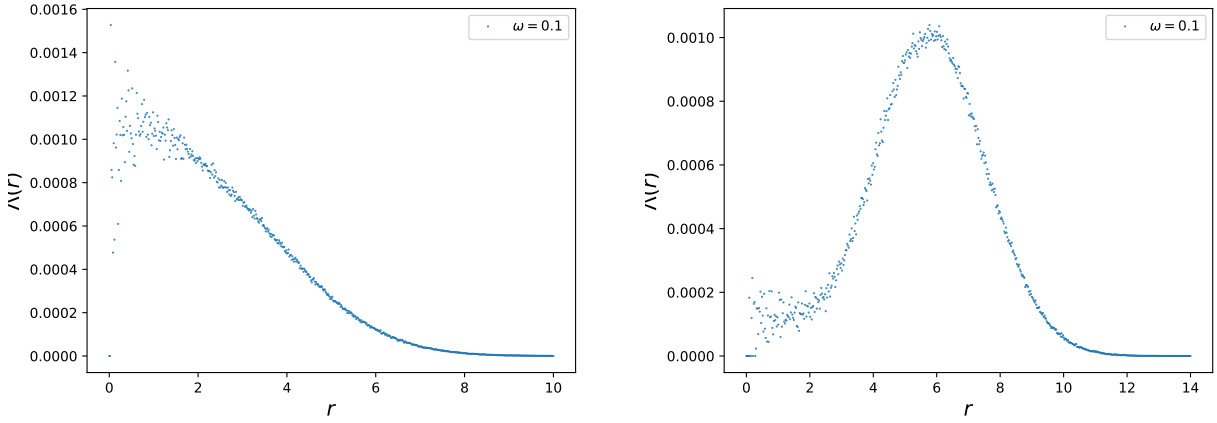
(a) $N = 2$, $\omega = 0.28$. (b) $N = 20$, $\omega = 0.28$.

Figure 6.1: One-Body density for the two dimensional harmonic oscillator potential using a Hartree-Fock basis with Padé function and α parameter.



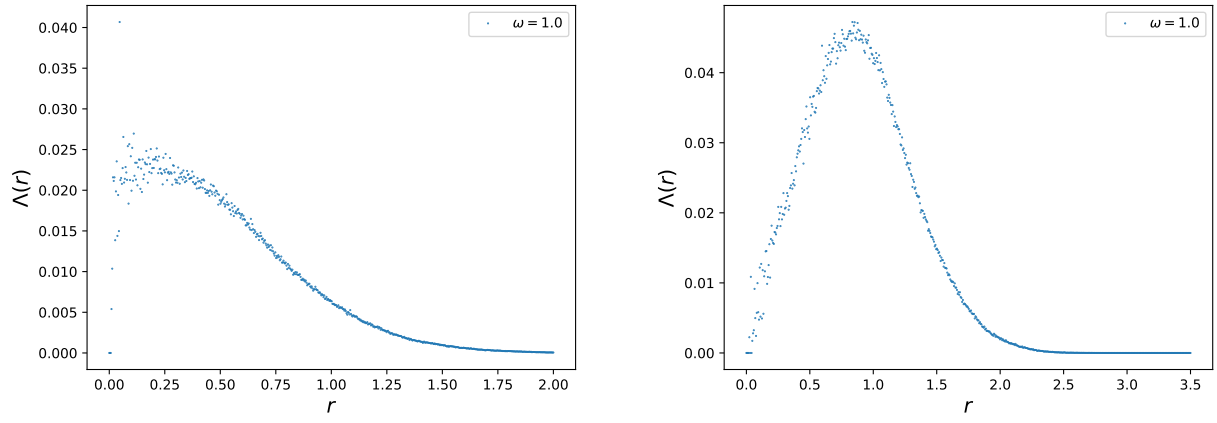
(a) $N = 2$, $\omega = 1.0$. (b) $N = 20$, $\omega = 1.0$.

Figure 6.2: One-Body density for the two dimensional harmonic oscillator potential using a Hartree-Fock basis with Padé function and α parameter.



(a) $N = 2$, $\omega = 0.1$. (b) $N = 20$, $\omega = 0.1$.

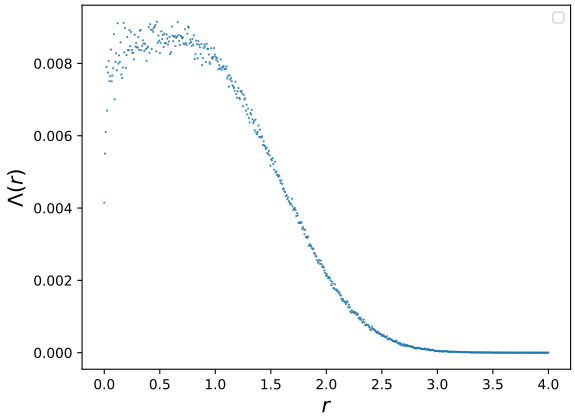
Figure 6.3: One-Body density for the three dimensional harmonic oscillator potential using a Hartree-Fock basis with Padé function and α parameter.



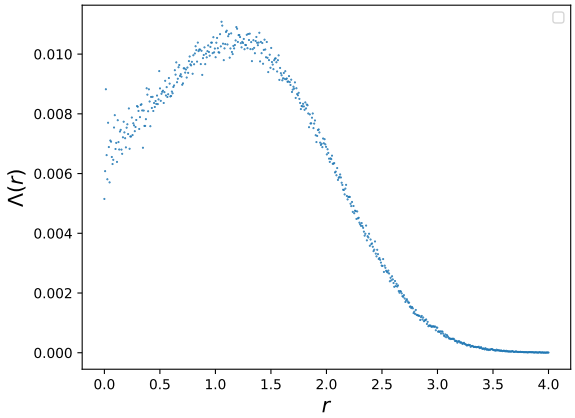
(a) $N = 2$, $\omega = 1.0$. (b) $N = 20$, $\omega = 1.0$.

Figure 6.4: One-Body density for the three dimensional harmonic oscillator potential using a Hartree-Fock basis with Padé function and α parameter.

Double-Well

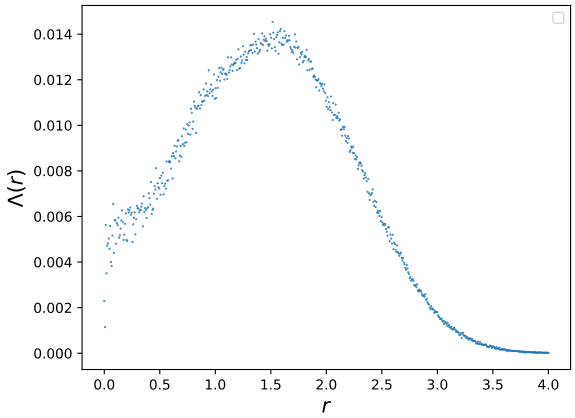


(a) $N = 2$, $\omega = 1.0$.

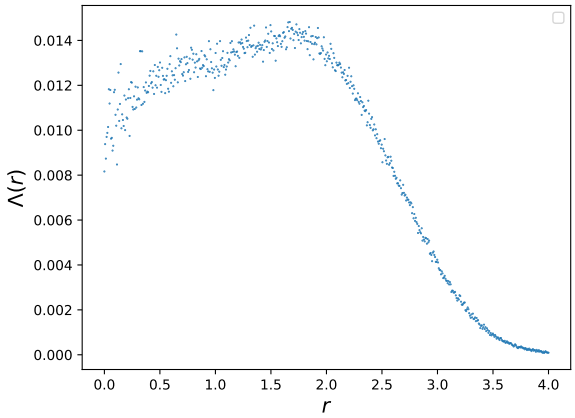


(b) $N = 4$, $\omega = 1.0$.

Figure 6.5: One-Body density for the two dimensional double-well potential using a Hartree-Fock basis with Padé function and α parameter.



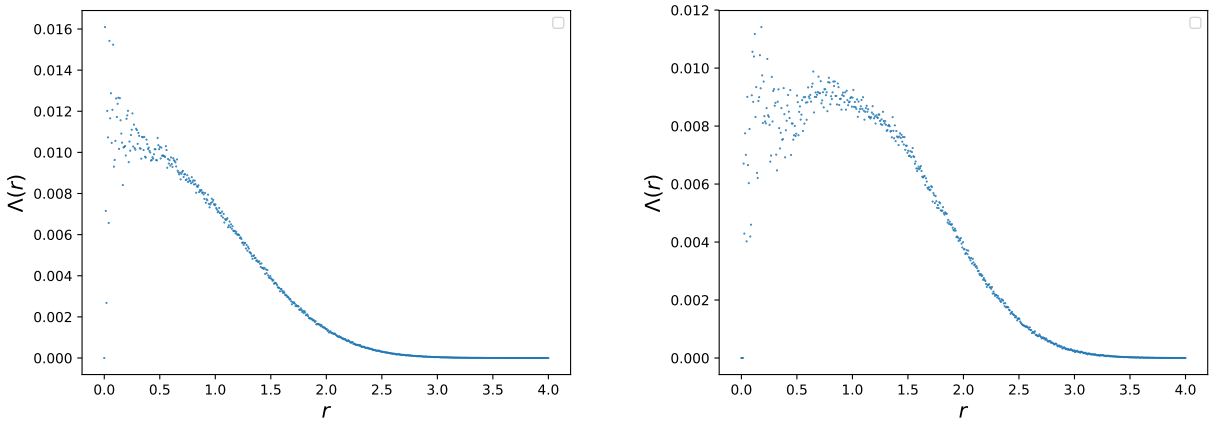
(a) $N = 6$, $\omega = 1.0$.



(b) $N = 8$, $\omega = 1.0$.

Figure 6.6: One-Body density for the two dimensional double-well potential using a Hartree-Fock basis with Padé function and α parameter.

.0



(a) $N = 2$, $\omega = 1.0$.

(b) $N = 4$, $\omega = 1.0$.

Figure 6.7: One-Body density for the three dimensional double-well potential using a Hartree-Fock basis with Padé function and α parameter.

APPENDIX A

THEORY

A.1 Atomic Unit

The units used in the calculations are known as atomic units. The premise is to write the natural length unit in terms of the *Bohr radius*[19]

$$a_0 = \frac{4\pi\epsilon_0\hbar^2}{m_e e^2} = 0.529\text{Å}, \tag{A.1}$$

giving the energy unit known as a *Hartree* and has the value

$$E_{\text{Hartree}} = m_e\left(\frac{4\pi\epsilon_0}{\hbar e}\right)^2 = 27.211\text{eV}, \tag{A.2}$$

and the electronmass m_e, elementary charge e, reduced Plancks constant $\hbar$ and free-space permittivity ϵ_0 are all equal to unity,

$$m_e = e = \hbar = 4\pi\epsilon_0 = 1. \tag{A.3}$$

The kinetic operator can then be rewritten to

$$K = -\frac{1}{2}\sum_i \nabla_i^2, \tag{A.4}$$

and the Coulomb interaction is simply

$$V_{\text{I}}\sum_{i<j}\frac{1}{r_{ij}}, \tag{A.5}$$

with r_{ij} also rescaled in terms of the Bohr radius.

A.2 Interaction-Term in Fock-Operator

Introducing the so-called permutation operator P which interchanges the labels of particles meaning we can define

$$A \equiv \frac{1}{N!}\sum_{p}(-1)^{p}P, \tag{A.6}$$

the so-called *antisymmetrization* operator. This operator has the following traits:

- The Hamiltonian H and A commute since the Hamiltonian is invariant under permutation.
- A applied on itself (that is A^2) is equal to itself since permuting a permuted state reproduces the state.

We can now express our Slater Ψ_T in terms of A as

$$\Psi_T = \sqrt{N!}A\prod_{i,j}\psi_{ij}, \tag{A.7}$$

where $\psi_{ij} = \psi_j(\boldsymbol{r}_i)$ is element i,j of the Slater matrix (the matrix associated with the Slater determinant Ψ_T).

The interaction part of H is then

$$\langle\Psi_T|\,H_I\,|\Psi_T\rangle = N!\prod_{i,j}\left\langle\psi_{ij}\right|AH_IA\left|\psi_{ij}\right\rangle. \tag{A.8}$$

The interaction H_I and A commute since A commutes with H giving

$$AH_IA\left|\psi_{ij}\right\rangle = \frac{1}{N!^2}\sum_{i<j}\sum_{p}(-1)^{2p}f_{ij}P\left|\psi_{ij}\right\rangle \tag{A.9}$$

$$= \frac{1}{N!^2}\sum_{i<j}f_{ij}(1-P_{ij})\left|\psi_{ij}\right\rangle. \tag{A.10}$$

The factor $1-P_{ij}$ comes from the fact that contributions with $i \neq j$ vanishes due to orthogonality when P is applied. The final expression for the interaction term is thus

$$\langle\Psi_T|\,H_I\,|\Psi_T\rangle = \sum_{i<j}\prod_{k,l}\left[\left\langle\psi_{kl}\right|f_{ij}\left|\psi_{kl}\right\rangle - \left\langle\psi_{kl}\right|f_{ij}\left|\psi_{lk}\right\rangle\right]. \tag{A.11}$$

Writing out the product and realizing the double summation over pairs of states we end up with

$$\langle\Psi_T|\,H_I\,|\Psi_T\rangle = \frac{1}{2}\sum_{i,j}\left[\left\langle\psi_{ij}\psi_{ji}\right|f_{ij}\left|\psi_{ij}\psi_{ji}\right\rangle - \left\langle\psi_{ij}\psi_{ji}\right|f_{ij}\left|\psi_{ji}\psi_{ij}\right\rangle\right]. \tag{A.12}$$

More comprehensive details and derivations are given in [26, 56].

A.3 Multi-Index Notation

This section will give a brief overlook of a notation which compresses indices running in similar fashion, the so-called *multi-index notation*[61]. We will make use of this to reduce indices in each dimension down to one.
The rules are stated as, given a n-tuple $(x_1,\dots x_n)$ over any field $\mathbb{F}$ (real, complex, etc.), a multi index is defined to be

$$i=(i_1,\dots i_n)\in\mathbb{Z}_+^n. \tag{A.13}$$

with expansions;

$$\begin{aligned} |i| &= |i_1|+\cdots+|i_n| \\ i! &= i_1!\dots i_n! \\ x^i &= x_1^{i_1}\dots x_n^{i_n}\in\mathbb{F}[x] \\ i\pm j &= (i_1\pm j_1,\dots,i_n+j_n)\in\mathbb{Z} \end{aligned} \tag{A.14}$$

In essence the notation just wraps the notion of element-wise operations into one single index variable.

APPENDIX

B.1 Derivatives of Hermite Functions

The gradient of Hermite functions on the form

$$\psi_n(r) = \prod_d \psi_{n_d}(x_d) = \prod_d H_{n_d}(\sqrt{\omega}x_d)\mathrm{e}^{-\frac{\omega}{2}x_d^2}, \tag{B.1}$$

is

$$\begin{aligned}\nabla\psi_n(r) &= \sum_d \hat{e}\prod_{d'\neq d}\psi_{n_{d'}}\frac{\partial\psi_{n_d}}{\partial x_d}\\ &= \sum_d \hat{e}\prod_{d'\neq d}\psi_{n_{d'}}\psi_{n_d}\sqrt{\omega}\left(\frac{\partial H_{n_d}}{\partial u}\frac{1}{H_{n_d}} - x_d\right)\\ &= \psi_n\sqrt{\omega}\sum_d \hat{e}\left(2n_d\frac{H_{n_d-1}(\sqrt{\omega}x_d)}{H_{n_d}(\sqrt{\omega}x_d)} - x_d\right),\end{aligned} \tag{B.2}$$

and the Laplacian follows

$$\begin{aligned}\nabla^2\psi_n(r) &= \sum_d\prod_{d'\neq d}\psi_{n_{d'}}\frac{\partial^2\psi_{n_d}}{\partial {x_d}^2}\\ &= \psi_n\omega\sum_d\left(\frac{\left(4n_d(n_d-1)H_{n_d-2}(\sqrt{\omega}x_d) - \sqrt{\omega}x_d H_{n_d-1}(\sqrt{\omega}x_d)\right)}{H_{n_d}(\sqrt{\omega}x_d)} + \omega x_d^2 - 1\right).\end{aligned} \tag{B.3}$$

The derivative with respect to the variational parameter α is

$$\begin{aligned}\frac{\partial \psi_n}{\partial \alpha} &= \frac{\partial}{\partial \alpha}\prod_d H_{n_d}(\sqrt{\omega}x_d)\mathrm{e}^{-\frac{\omega}{2}x_d^2} \\ &= \sum_d \prod_{d'\neq d} \psi_{n_{d'}}(x_{d'})\frac{\partial \psi_{n_d}(x_d)}{\partial \alpha} \\ &= \psi_n(r)\sum_d \omega x_d\left(\frac{n_d}{\sqrt{\alpha\omega}}\frac{H_{n_d-1}\left(\sqrt{\alpha\omega}x_d\right)}{H_{n_d}\left(\sqrt{\alpha\omega}x_d\right)} - \frac{\omega x_d^2}{2}\right) \\ &= \psi_n(r)\left(\sqrt{\frac{\omega}{\alpha}}\sum_d x_d n_d \frac{H_{n_d-1}\left(\sqrt{\alpha\omega}x_d\right)}{H_{n_d}\left(\sqrt{\alpha\omega}x_d\right)} - \frac{\omega}{2}r^2\right).\end{aligned} \tag{B.4}$$

B.2 Derivatives of Padé-Jastrow Function

Given the Padé-Jastrow function

$$J = \exp\left(\sum_{i<j} f_{ij}\right), \qquad f_{ij} = \frac{a_{ij}r_{ij}}{1+\beta r_{ij}}. \tag{B.5}$$

The general expression for the gradient and Laplacian with respect to particle position k is

$$\begin{aligned}\nabla_k J &= J\sum_{j\neq k}\frac{\boldsymbol{r}_{kj}}{r_{kj}}\frac{\partial f_{kj}}{\partial r_{kj}} \\ \nabla_k^2 J &= \frac{(\nabla_k J)^2}{J} + J\sum_{j\neq k}\left(\frac{\partial f_{kj}}{\partial r_{kj}}\frac{D-1}{r_{kj}} + \frac{\partial^2 f_{kj}}{\partial {r_{kj}}^2}\right).\end{aligned} \tag{B.6}$$

Notice that the sum with $j \neq k$ is only a sum over j with k fixed. The derivatives of f are

$$\begin{aligned}\frac{\partial f_{kj}}{\partial r_{kj}} &= \frac{a_{kj}}{\left(1+\beta r_{kj}\right)^2} \\ \frac{\partial^2 f_{kj}}{\partial {r_{kj}}^2} &= -\frac{2a_{kj}\beta}{\left(1+\beta r_{kj}\right)^3}\end{aligned}. \tag{B.7}$$

And the derivative with respect to the variational parameter β

$$\frac{\partial f_{kj}}{\partial \beta} = -\frac{a_{kj}r_{kj}^2}{\left(1+\beta r_{kj}\right)^2} \tag{B.8}$$

B.3 Derivatives of NQS-Wavefunction

Given the NQS-Wavefunction

$$J = \exp\left(-\sum_{i=1}^{N}\frac{(\boldsymbol{r}_i - \boldsymbol{a}_i)^2}{2\sigma^2}\right)\prod_j^M\left(1+E_j\right). \tag{B.9}$$

The gradient is

$$\begin{aligned}\frac{\nabla_k J}{J} &= \nabla_k \ln(J) \\ &= -\frac{\boldsymbol{r}_k - \boldsymbol{a}_k}{\sigma^2} + \sum_{j=0}^{M}\sum_{d=1}^{D} \hat{\boldsymbol{e}}_d \frac{w_{k+d,j}}{\sigma^2\left(1+\frac{1}{E_j}\right)}\end{aligned} \tag{B.10}$$

and the Laplacian

$$\begin{aligned}\frac{\nabla_k^2 J}{J} &= \nabla_k^2 \ln(J) \\ &= \left(\frac{\nabla_k J}{J}\right)^2 - \frac{D}{\sigma^2} + \sum_{j=1,d=1}^{M,D} \frac{w_{k+d,j}^2 E_j}{\sigma^4\left(1+E_j\right)^2},\end{aligned} \tag{B.11}$$

with

$$\begin{aligned}W_j &\equiv \sum_{i=1}^{N}\sum_{d=1}^{D} \frac{x_i^{(d)} w_{i+d,j}}{\sigma^2} \\ E_j &\equiv \exp\left(b_j + W_j\right)\end{aligned}. \tag{B.12}$$

The derivative with respect to the visible bias is

$$\frac{1}{J}\frac{\partial J}{\partial a_l} = \frac{x_l - a_l}{\sigma^2}, \tag{B.13}$$

and the hidden bias

$$\begin{aligned}\frac{1}{J}\frac{\partial J}{\partial b_l} &= \frac{\partial}{\partial b_l}\ln(J) \\ &= \frac{1}{1+\frac{1}{E_j}},\end{aligned} \tag{B.14}$$

and the weights

$$\begin{aligned}\frac{1}{J}\frac{\partial}{\partial w_{kl}} &= \frac{\partial}{\partial w_{kl}}\ln(J) \\ &= \frac{x_k}{\sigma^2\left(1+\frac{1}{E_j}\right)}.\end{aligned} \tag{B.15}$$

Notice that the indices k and l are here indiscriminate towards the dimensional component in relation to x. This means that x_l or x_k represents just a particle position in a dimension.

APPENDIX C

DOUBLE-WELL

Table 1: Eigenvalues for double-well functions in 2D with $R = 2.0$ and $\omega = 1.0$

L	1	3	6	10	15	21	28	36	45	55	66	78	91	105	120
1	0.93581	0.93581	0.81130	0.81130	0.80989	0.80989	0.80959	0.80959	0.80954	0.80954	0.80951	0.80951	0.80950	0.80950	0.80949
2	-	1.37162	1.37162	1.23653	1.23653	1.23441	1.23441	1.23426	1.23426	1.23425	1.23425	1.23424	1.23424	1.23424	1.23424
3	-	1.93581	1.93581	1.81130	1.81130	1.80989	1.80989	1.80959	1.80959	1.80954	1.80954	1.80951	1.80951	1.80950	1.80950
4	-	-	2.21404	2.21404	2.00420	2.00420	2.00055	2.00055	2.00013	2.00013	2.00008	2.00008	2.00005	2.00005	2.00004
5	-	-	2.37162	2.37162	2.23653	2.23653	2.23441	2.23441	2.23426	2.23426	2.23425	2.23425	2.23424	2.23424	2.23424
6	-	-	2.93581	2.93581	2.81130	2.70373	2.70373	2.69787	2.69787	2.69751	2.69751	2.69749	2.69749	2.69748	2.69748
7	-	-	-	2.94252	2.94252	2.81130	2.80989	2.80989	2.80959	2.80959	2.80954	2.80954	2.80951	2.80951	2.80950
8	-	-	-	3.21404	3.21404	3.00420	3.00420	3.00055	3.00055	3.00013	3.00013	3.00008	3.00008	3.00005	3.00005
9	-	-	-	3.37162	3.37162	3.23653	3.23653	3.23441	3.23441	3.23426	3.23426	3.23425	3.23425	3.23424	3.23424
10	-	-	-	3.93581	3.80710	3.80710	3.50765	3.50765	3.49933	3.49933	3.49871	3.49871	3.49867	3.49867	3.49865
11	-	-	-	-	3.93581	3.81130	3.70373	3.70373	3.69787	3.69787	3.69751	3.69751	3.69749	3.69749	3.69748
12	-	-	-	-	3.94252	3.94252	3.81130	3.80989	3.80989	3.80959	3.80959	3.80954	3.80954	3.80951	3.80951
13	-	-	-	-	4.21404	4.21404	4.00420	4.00420	4.00055	4.00055	4.00013	4.00013	4.00008	4.00008	4.00005
14	-	-	-	-	4.37162	4.37162	4.23653	4.23653	4.23441	4.23441	4.23426	4.23426	4.23425	4.23425	4.23424
15	-	-	-	-	4.93581	4.62520	4.62520	4.29240	4.29240	4.28089	4.28089	4.28024	4.28024	4.28023	4.28023
16	-	-	-	-	-	4.80710	4.80710	4.50765	4.50765	4.49933	4.49933	4.49871	4.49871	4.49867	4.49867
17	-	-	-	-	-	4.93581	4.81130	4.70373	4.70373	4.69787	4.69787	4.69751	4.69751	4.69749	4.69749
18	-	-	-	-	-	4.94252	4.94252	4.81130	4.80989	4.80989	4.80959	4.80959	4.80954	4.80954	4.80951
19	-	-	-	-	-	5.21404	5.21404	5.00420	5.00420	5.00055	5.00055	5.00013	5.00013	5.00008	5.00008
20	-	-	-	-	-	5.37162	5.37162	5.23653	5.12518	5.12518	5.11027	5.11027	5.10933	5.10933	5.10930
21	-	-	-	-	-	5.93581	5.51138	5.51138	5.23653	5.23441	5.23441	5.23426	5.23426	5.23425	5.23425
22	-	-	-	-	-	-	5.62520	5.62520	5.29240	5.29240	5.28089	5.28089	5.28024	5.28024	5.28023
23	-	-	-	-	-	-	5.80710	5.80710	5.50765	5.50765	5.49933	5.49933	5.49871	5.49871	5.49867
24	-	-	-	-	-	-	5.93581	5.81130	5.70373	5.70373	5.69787	5.69787	5.69751	5.69751	5.69749
25	-	-	-	-	-	-	5.94252	5.94252	5.81130	5.80989	5.80989	5.80959	5.80959	5.80954	5.80954
26	-	-	-	-	-	-	6.21404	6.21404	6.00420	5.94989	5.94989	5.93092	5.93092	5.92987	5.92987
27	-	-	-	-	-	-	6.37162	6.37048	6.23653	6.00420	6.00055	6.00055	6.00013	6.00013	6.00008
28	-	-	-	-	-	-	6.93581	6.37162	6.37048	6.12518	6.12518	6.11027	6.11027	6.10933	6.10933
29	-	-	-	-	-	-	-	6.51138	6.51138	6.23653	6.23441	6.23441	6.23426	6.23426	6.23425

30	-	-	-	-	-	-	-	6.62520	6.62520	6.29240	6.29240	6.28089	6.28089	6.28024	6.28024
31	-	-	-	-	-	-	-	6.80710	6.80710	6.50765	6.50765	6.49933	6.49933	6.49871	6.49871
32	-	-	-	-	-	-	-	6.93581	6.81130	6.70373	6.70373	6.69787	6.69787	6.69751	6.69751
33	-	-	-	-	-	-	-	6.94252	6.94252	6.81130	6.80315	6.80315	6.77991	6.77991	6.77851
34	-	-	-	-	-	-	-	7.21404	7.21404	7.00420	6.80989	6.80989	6.80959	6.80959	6.80954
35	-	-	-	-	-	-	-	7.37162	7.27239	7.23653	6.94989	6.94989	6.93092	6.93092	6.92987
36	-	-	-	-	-	-	-	7.93581	7.37048	7.27239	7.00420	7.00055	7.00055	7.00013	7.00013
37	-	-	-	-	-	-	-	-	7.37162	7.37048	7.12518	7.12518	7.11027	7.11027	7.10933
38	-	-	-	-	-	-	-	-	7.51138	7.51138	7.23653	7.23441	7.23441	7.23426	7.23426
39	-	-	-	-	-	-	-	-	7.62520	7.62520	7.29240	7.29240	7.28089	7.28089	7.28024
40	-	-	-	-	-	-	-	-	7.80710	7.80710	7.50765	7.50765	7.49933	7.49933	7.49871
41	-	-	-	-	-	-	-	-	7.93581	7.81130	7.70373	7.65216	7.65216	7.62410	7.62410
42	-	-	-	-	-	-	-	-	7.94252	7.94252	7.81130	7.70373	7.69787	7.69787	7.69751
43	-	-	-	-	-	-	-	-	8.21404	8.15560	8.00420	7.80315	7.80315	7.77991	7.77991
44	-	-	-	-	-	-	-	-	8.37162	8.21404	8.15560	7.80989	7.80989	7.80959	7.80959
45	-	-	-	-	-	-	-	-	8.93581	8.27239	8.23653	7.94989	7.94989	7.93092	7.93092
46	-	-	-	-	-	-	-	-	-	8.37048	8.27239	8.00420	8.00055	8.00055	8.00013
47	-	-	-	-	-	-	-	-	-	8.37162	8.37048	8.12518	8.12518	8.11027	8.11027
48	-	-	-	-	-	-	-	-	-	8.51138	8.51138	8.23653	8.23441	8.23441	8.23426
49	-	-	-	-	-	-	-	-	-	8.62520	8.62520	8.29240	8.29240	8.28089	8.28089
50	-	-	-	-	-	-	-	-	-	8.80710	8.80710	8.50765	8.50765	8.49933	8.48750
51	-	-	-	-	-	-	-	-	-	8.93581	8.81130	8.70373	8.52060	8.52060	8.49933
52	-	-	-	-	-	-	-	-	-	8.94252	8.94252	8.81130	8.65216	8.65216	8.62410
53	-	-	-	-	-	-	-	-	-	9.21404	9.06914	9.00420	8.70373	8.69787	8.69787
54	-	-	-	-	-	-	-	-	-	9.37162	9.15560	9.06914	8.80315	8.80315	8.77991
55	-	-	-	-	-	-	-	-	-	9.93581	9.21404	9.15560	8.80989	8.80989	8.80959
56	-	-	-	-	-	-	-	-	-	-	9.27239	9.23653	8.94989	8.94989	8.93092
57	-	-	-	-	-	-	-	-	-	-	9.37048	9.27239	9.00420	9.00055	9.00055
58	-	-	-	-	-	-	-	-	-	-	9.37162	9.37048	9.12518	9.12518	9.11027
59	-	-	-	-	-	-	-	-	-	-	9.51138	9.51138	9.23653	9.23441	9.23441
60	-	-	-	-	-	-	-	-	-	-	9.62520	9.62520	9.29240	9.29240	9.28089
61	-	-	-	-	-	-	-	-	-	-	9.80710	9.80710	9.50765	9.38649	9.38649
62	-	-	-	-	-	-	-	-	-	-	9.93581	9.81130	9.70373	9.50765	9.49933
63	-	-	-	-	-	-	-	-	-	-	9.94252	9.94252	9.81130	9.52060	9.52060
64	-	-	-	-	-	-	-	-	-	-	10.21404	9.96852	9.96852	9.65216	9.65216
65	-	-	-	-	-	-	-	-	-	-	10.37162	10.06914	10.00420	9.70373	9.69787
66	-	-	-	-	-	-	-	-	-	-	10.93581	10.15560	10.06914	9.80315	9.80315
67	-	-	-	-	-	-	-	-	-	-	-	10.21404	10.15560	9.80989	9.80989
68	-	-	-	-	-	-	-	-	-	-	-	10.27239	10.23653	9.94989	9.94989
69	-	-	-	-	-	-	-	-	-	-	-	10.37048	10.27239	10.00420	10.00055
70	-	-	-	-	-	-	-	-	-	-	-	10.37162	10.37048	10.12518	10.12518
71	-	-	-	-	-	-	-	-	-	-	-	10.51138	10.51138	10.23653	10.23441
72	-	-	-	-	-	-	-	-	-	-	-	10.62520	10.62520	10.29240	10.26669
73	-	-	-	-	-	-	-	-	-	-	-	10.80710	10.80710	10.50765	10.29240
74	-	-	-	-	-	-	-	-	-	-	-	10.93581	10.81130	10.70373	10.38649
75	-	-	-	-	-	-	-	-	-	-	-	10.94252	10.89097	10.81130	10.50765
76	-	-	-	-	-	-	-	-	-	-	-	11.21404	10.94252	10.89097	10.52060
77	-	-	-	-	-	-	-	-	-	-	-	11.37162	10.96852	10.96852	10.65216
78	-	-	-	-	-	-	-	-	-	-	-	11.93581	11.06914	11.00420	10.70373
79	-	-	-	-	-	-	-	-	-	-	-	-	11.15560	11.06914	10.80315
80	-	-	-	-	-	-	-	-	-	-	-	-	11.21404	11.15560	10.80989
81	-	-	-	-	-	-	-	-	-	-	-	-	11.27239	11.23653	10.94989
82	-	-	-	-	-	-	-	-	-	-	-	-	11.37048	11.27239	11.00420
83	-	-	-	-	-	-	-	-	-	-	-	-	11.37162	11.37048	11.12518
84	-	-	-	-	-	-	-	-	-	-	-	-	11.51138	11.51138	11.23653

85	-	-	-	-	-	-	-	-	-	-	-	-	11.62520	11.62520	11.29240
86	-	-	-	-	-	-	-	-	-	-	-	-	11.80710	11.80208	11.50765
87	-	-	-	-	-	-	-	-	-	-	-	-	11.93581	11.80710	11.70373
88	-	-	-	-	-	-	-	-	-	-	-	-	11.94252	11.81130	11.80208
89	-	-	-	-	-	-	-	-	-	-	-	-	12.21404	11.89097	11.81130
90	-	-	-	-	-	-	-	-	-	-	-	-	12.37162	11.94252	11.89097
91	-	-	-	-	-	-	-	-	-	-	-	-	12.93581	11.96852	11.96852
92	-	-	-	-	-	-	-	-	-	-	-	-	-	12.06914	12.00420
93	-	-	-	-	-	-	-	-	-	-	-	-	-	12.15560	12.06914
94	-	-	-	-	-	-	-	-	-	-	-	-	-	12.21404	12.15560
95	-	-	-	-	-	-	-	-	-	-	-	-	-	12.27239	12.23653
96	-	-	-	-	-	-	-	-	-	-	-	-	-	12.37048	12.27239
97	-	-	-	-	-	-	-	-	-	-	-	-	-	12.37162	12.37048
98	-	-	-	-	-	-	-	-	-	-	-	-	-	12.51138	12.51138
99	-	-	-	-	-	-	-	-	-	-	-	-	-	12.62520	12.62520
100	-	-	-	-	-	-	-	-	-	-	-	-	-	12.80710	12.73158
101	-	-	-	-	-	-	-	-	-	-	-	-	-	12.93581	12.80208
102	-	-	-	-	-	-	-	-	-	-	-	-	-	12.94252	12.80710
103	-	-	-	-	-	-	-	-	-	-	-	-	-	13.21404	12.81130
104	-	-	-	-	-	-	-	-	-	-	-	-	-	13.37162	12.89097
105	-	-	-	-	-	-	-	-	-	-	-	-	-	13.93581	12.94252
106	-	-	-	-	-	-	-	-	-	-	-	-	-	-	12.96852
107	-	-	-	-	-	-	-	-	-	-	-	-	-	-	13.06914
108	-	-	-	-	-	-	-	-	-	-	-	-	-	-	13.15560
109	-	-	-	-	-	-	-	-	-	-	-	-	-	-	13.21404
110	-	-	-	-	-	-	-	-	-	-	-	-	-	-	13.27239
111	-	-	-	-	-	-	-	-	-	-	-	-	-	-	13.37048
112	-	-	-	-	-	-	-	-	-	-	-	-	-	-	13.37162
113	-	-	-	-	-	-	-	-	-	-	-	-	-	-	13.51138
114	-	-	-	-	-	-	-	-	-	-	-	-	-	-	13.62520
115	-	-	-	-	-	-	-	-	-	-	-	-	-	-	13.80710
116	-	-	-	-	-	-	-	-	-	-	-	-	-	-	13.93581
117	-	-	-	-	-	-	-	-	-	-	-	-	-	-	13.94252
118	-	-	-	-	-	-	-	-	-	-	-	-	-	-	14.21404
119	-	-	-	-	-	-	-	-	-	-	-	-	-	-	14.37162
120	-	-	-	-	-	-	-	-	-	-	-	-	-	-	14.93581

Table 2: Eigenvalues for double-well functions in 3D with $R = 2.0$ and $\omega = 1.0$

L	1	4	10	20	35	56	84	120
1	1.43581	1.43581	1.31130	1.31130	1.30989	1.30989	1.30959	1.30959
2	-	1.87162	1.87162	1.73653	1.73653	1.73441	1.73441	1.73426
3	-	2.43581	2.43581	2.31130	2.31130	2.30989	2.30989	2.30959
4	-	2.43581	2.43581	2.31130	2.31130	2.30989	2.30989	2.30959
5	-	-	2.71404	2.71404	2.50420	2.50420	2.50055	2.50055
6	-	-	2.87162	2.87162	2.73653	2.73653	2.73441	2.73441
7	-	-	2.87162	2.87162	2.73653	2.73653	2.73441	2.73441
8	-	-	3.43581	3.43581	3.31130	3.20373	3.20373	3.19787
9	-	-	3.43581	3.43581	3.31130	3.31130	3.30989	3.30989
10	-	-	3.43581	3.43581	3.31130	3.31130	3.30989	3.30989
11	-	-	-	3.44252	3.44252	3.31130	3.30989	3.30989

12	-	-	-	3.71404	3.71404	3.50420	3.50420	3.50055
13	-	-	-	3.71404	3.71404	3.50420	3.50420	3.50055
14	-	-	-	3.87162	3.87162	3.73653	3.73653	3.73441
15	-	-	-	3.87162	3.87162	3.73653	3.73653	3.73441
16	-	-	-	3.87162	3.87162	3.73653	3.73653	3.73441
17	-	-	-	4.43581	4.30710	4.30710	4.00765	4.00765
18	-	-	-	4.43581	4.43581	4.31130	4.20373	4.20373
19	-	-	-	4.43581	4.43581	4.31130	4.20373	4.20373
20	-	-	-	4.43581	4.43581	4.31130	4.31130	4.30989
21	-	-	-	-	4.43581	4.31130	4.31130	4.30989
22	-	-	-	-	4.44252	4.44252	4.31130	4.30989
23	-	-	-	-	4.44252	4.44252	4.31130	4.30989
24	-	-	-	-	4.71404	4.71404	4.50420	4.50420
25	-	-	-	-	4.71404	4.71404	4.50420	4.50420
26	-	-	-	-	4.71404	4.71404	4.50420	4.50420
27	-	-	-	-	4.87162	4.87162	4.73653	4.73653
28	-	-	-	-	4.87162	4.87162	4.73653	4.73653
29	-	-	-	-	4.87162	4.87162	4.73653	4.73653
30	-	-	-	-	4.87162	4.87162	4.73653	4.73653
31	-	-	-	-	5.43581	5.12520	5.12520	4.79240
32	-	-	-	-	5.43581	5.30710	5.30710	5.00765
33	-	-	-	-	5.43581	5.30710	5.30710	5.00765
34	-	-	-	-	5.43581	5.43581	5.31130	5.20373
35	-	-	-	-	5.43581	5.43581	5.31130	5.20373
36	-	-	-	-	-	5.43581	5.31130	5.20373
37	-	-	-	-	-	5.43581	5.31130	5.31130
38	-	-	-	-	-	5.43581	5.31130	5.31130
39	-	-	-	-	-	5.44252	5.44252	5.31130
40	-	-	-	-	-	5.44252	5.44252	5.31130
41	-	-	-	-	-	5.44252	5.44252	5.31130
42	-	-	-	-	-	5.71404	5.71404	5.50420
43	-	-	-	-	-	5.71404	5.71404	5.50420
44	-	-	-	-	-	5.71404	5.71404	5.50420
45	-	-	-	-	-	5.71404	5.71404	5.50420
46	-	-	-	-	-	5.87162	5.87162	5.73653
47	-	-	-	-	-	5.87162	5.87162	5.73653
48	-	-	-	-	-	5.87162	5.87162	5.73653
49	-	-	-	-	-	5.87162	5.87162	5.73653
50	-	-	-	-	-	5.87162	5.87162	5.73653
51	-	-	-	-	-	6.43581	6.01138	6.01138
52	-	-	-	-	-	6.43581	6.12520	6.12520
53	-	-	-	-	-	6.43581	6.12520	6.12520
54	-	-	-	-	-	6.43581	6.30710	6.30710
55	-	-	-	-	-	6.43581	6.30710	6.30710
56	-	-	-	-	-	6.43581	6.30710	6.30710
57	-	-	-	-	-	-	6.43581	6.31130
58	-	-	-	-	-	-	6.43581	6.31130
59	-	-	-	-	-	-	6.43581	6.31130
60	-	-	-	-	-	-	6.43581	6.31130
61	-	-	-	-	-	-	6.43581	6.31130
62	-	-	-	-	-	-	6.43581	6.31130
63	-	-	-	-	-	-	6.44252	6.44252
64	-	-	-	-	-	-	6.44252	6.44252
65	-	-	-	-	-	-	6.44252	6.44252
66	-	-	-	-	-	-	6.44252	6.44252

67	-	-	-	-	-	-	6.71404	6.71404
68	-	-	-	-	-	-	6.71404	6.71404
69	-	-	-	-	-	-	6.71404	6.71404
70	-	-	-	-	-	-	6.71404	6.71404
71	-	-	-	-	-	-	6.71404	6.71404
72	-	-	-	-	-	-	6.87162	6.87048
73	-	-	-	-	-	-	6.87162	6.87162
74	-	-	-	-	-	-	6.87162	6.87162
75	-	-	-	-	-	-	6.87162	6.87162
76	-	-	-	-	-	-	6.87162	6.87162
77	-	-	-	-	-	-	6.87162	6.87162
78	-	-	-	-	-	-	7.43581	6.87162
79	-	-	-	-	-	-	7.43581	7.01138
80	-	-	-	-	-	-	7.43581	7.01138
81	-	-	-	-	-	-	7.43581	7.12520
82	-	-	-	-	-	-	7.43581	7.12520
83	-	-	-	-	-	-	7.43581	7.12520
84	-	-	-	-	-	-	7.43581	7.30710
85	-	-	-	-	-	-	-	7.30710
86	-	-	-	-	-	-	-	7.30710
87	-	-	-	-	-	-	-	7.30710
88	-	-	-	-	-	-	-	7.43581
89	-	-	-	-	-	-	-	7.43581
90	-	-	-	-	-	-	-	7.43581
91	-	-	-	-	-	-	-	7.43581
92	-	-	-	-	-	-	-	7.43581
93	-	-	-	-	-	-	-	7.43581
94	-	-	-	-	-	-	-	7.43581
95	-	-	-	-	-	-	-	7.44252
96	-	-	-	-	-	-	-	7.44252
97	-	-	-	-	-	-	-	7.44252
98	-	-	-	-	-	-	-	7.44252
99	-	-	-	-	-	-	-	7.44252
100	-	-	-	-	-	-	-	7.71404
101	-	-	-	-	-	-	-	7.71404
102	-	-	-	-	-	-	-	7.71404
103	-	-	-	-	-	-	-	7.71404
104	-	-	-	-	-	-	-	7.71404
105	-	-	-	-	-	-	-	7.71404
106	-	-	-	-	-	-	-	7.87162
107	-	-	-	-	-	-	-	7.87162
108	-	-	-	-	-	-	-	7.87162
109	-	-	-	-	-	-	-	7.87162
110	-	-	-	-	-	-	-	7.87162
111	-	-	-	-	-	-	-	7.87162
112	-	-	-	-	-	-	-	7.87162
113	-	-	-	-	-	-	-	8.43581
114	-	-	-	-	-	-	-	8.43581
115	-	-	-	-	-	-	-	8.43581
116	-	-	-	-	-	-	-	8.43581
117	-	-	-	-	-	-	-	8.43581
118	-	-	-	-	-	-	-	8.43581
119	-	-	-	-	-	-	-	8.43581
120	-	-	-	-	-	-	-	8.43581

BIBLIOGRAPHY

[1] E. Anisimovas and A. Matulis. “Energy spectra of few-electron quantum dots”. In: *Journal of Physics: Condensed Matter* (1998), p. 601.

[2] “Atomic Units”. In: *Springer Handbook of Atomic, Molecular, and Optical Physics*. New York, NY: Springer NY, 2006, pp. 1–6. ISBN: 978-0-387-26308-3. DOI: \url{https://doi.org/10.1007/978-0-387-26308-3}.

[3] Dacorogna B. *Introduction to the calculus of variations*. Imperial College Press; World Scientific, 2004. ISBN: 9781860945083.

[4] M. L. Boas. *Mathematical Methods in the Physical Sciences, 3rd Edition*. Wiley, 2005. ISBN: 978-0-471-19826-0.

[5] C. G. Broyden. “The Convergence of a Class of Double-rank Minimization Algorithms 1. General Considerations”. In: *IMA Journal of Applied Mathematics* (1970). DOI: 10.1093/imamat/6.1.76.

[6] van Brunt B. *The Calculus of Variations*. Springer, 2004. ISBN: 9780387402475.

[7] G. Carleo and M. Troyer. “Solving the Quantum Many-Body Problem with Artificial Neural Networks”. In: *Science* 355.6325 (2017), pp. 602–606. ISSN: 0036-8075. DOI: 10.1126/science.aag2302.

[8] M. S. Carroll. “Implications Of Simultaneous Requirements For Low-Noise Exchange Gates In Double Quantum Dots”. In: *Phys. Rev. B* (2010). DOI: 10.1103/PhysRevB.82.075319.

[9] UnitTest++ developers. *UnitTest++*. https://github.com/unittest-cpp/unittest-cpp/wiki.

[10] YAML developers. *YAML*. http://yaml.org/.

[11] W. H. Dickhoff and D. Van Neck. *Many-body theory exposed!: Propagator Description of Quantum Mechanics in Many-Body Systems*. World Scientific, 2005. ISBN: 981256294X.

[12] Svenn-Arne Dragly. *Hartree-Fock*. https://github.com/dragly/hartree-fock. 2018.

[13] N. D. Drummond, M. D. Towler, and R. J. Needs. “Jastrow Correlation Factor for Atoms, Molecules, and Solids”. In: *Phys. Rev. B* 70 (2004), p. 235119. DOI: 10.1103/PhysRevB.70.235119.

[14] A. L. Fetter and J. D. Walecka. *Quantum Theory of Many-particle Systems*. Mcgraw-Hill College, 1971. ISBN: 9780070206533.

[15] R. Fletcher. "A New Approach To Variable Metric Algorithms". In: *The Computer Journal* (1970). DOI: 10.1093/comjnl/13.3.317.

[16] H. Flyvbjerg and H. G. Petersen. "Error estimates on averages of correlated data". In: *The Journal of Chemical Physics* 91.1 (1989), pp. 461–466. DOI: 10.1063/1.457480.

[17] D. Goldfarb. "A Family of Variable-Metric Methods Derived by Variational Means". In: *Math. Comp.* 24 (1970). DOI: \url{https://doi.org/10.1090/S0025-5718-1970-0258249-6}.

[18] D. Goldfarb. "Stochastic Adaptive Quasi-Newton Methods for Minimizing Expected Values". In: *Proceedings of the 34th International Conference on Machine Learning*. Proceedings of Machine Learning Research. PMLR, 2017.

[19] D.J. Griffiths. *Introduction to quantum mechanics*. 2nd Edition. Pearson PH, 2005. ISBN: 0131911759.

[20] Gaël Guennebaud, Benoît Jacob, et al. *Eigen v3*. http://eigen.tuxfamily.org. 2010.

[21] W. A. Hammond B. L. Lester Jr. and P.J. Reynolds. *Monte Carlo Methods in Ab Initio Quantum Chemistry*. World Scientific, 1994. ISBN: 9810203225.

[22] "Hartree-Fock Theory". In: *Computational Chemistry and Molecular Modeling: Principles and Applications*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 93–113. ISBN: 978-3-540-77304-7. DOI: 10.1007/978-3-540-77304-7_5.

[23] T. Helgaker. *Molecular Integral Evaluation*. 2006.

[24] T. Helgaker and P. R. Taylor. *Gaussian Basis Sets and Molecular Integrals*. 1995.

[25] G. Hinton. *A Practical Guide to Training Restricted Boltzmann Machines*. https://www.cs.toronto.edu/~hinton/absps/guideTR.pdf.

[26] M. Hjort-Jensen. *Computational Physics: Hartree-Fock methods and introduction to Many-Body Theory*. https://www.github.com/CompPhysics/ComputationalPhysics2/blob/gh-pages/doc/pub/basicMB/pdf. 2017.

[27] M. Hjort-Jensen. *Computational Physics: Variational Monte Carlo methods*. https://www.github.com/CompPhysics/ComputationalPhysics2/blob/gh-pages/doc/pub/vmc/pdf. 2017.

[28] Høgberget J. *Quantum Monte-Carlo Studies of Generalized Many-body Systems*. MA Thesis. University of Oslo, 2013.

[29] Rogers J. “Global Optimization of Statistical Functions With Simulated Annealing”. In: *Journal of Econometrics* (1994), pp. 65–99. ISSN: 0304-4076. DOI: https://doi.org/10.1016/0304-4076(94)90038-8.

[30] J. D. Jacksjon. *Classical Electrodynamics*. Wiley, 1998. ISBN: 978-0-471-30932-1.

[31] R. Jastrow. “Many-Body Problem with Strong Forces”. In: *Phys. Rev.* (1955). DOI: 10.1103/PhysRev.98.1479.

[32] A. Jeffrey, ed. *Handbook of Mathematical Formulas and Integrals (Third Edition)*. Academic Press, 2004. ISBN: 978-0-12-382256-7. DOI: https://doi.org/10.1016/B978-012382256-7/50034-8.

[33] Marius Jonsson. *block*. https://github.com/computative/block. 2018.

[34] S. Kvaal. “Harmonic Oscillator Eigenfunction Expansions, Quantum dots, and Effective Interactions”. In: *Phys. Rev. B* 80 (4 2009), p. 045321. DOI: 10.1103/PhysRevB.80.045321.

[35] D. C. Lay. *Linear Algebra and Its Applications (4th Edition)*. Addison Wesley / Pearson, 2011. ISBN: 9780321385178.

[36] D. S. Lemons and A. Gythiel. “Paul Langevin’s 1908 paper “On the Theory of Brownian Motion” [“Sur la théorie du mouvement brownien,” C. R. Acad. Sci. (Paris) 146, 530–533 (1908)]”. In: *American Journal of Physics* 65 (1997), pp. 1079–1081. DOI: 10.1119/1.18725.

[37] L. E. Lervåg. *VMC Calculations of Two-Dimensional Quantum Dots*. MA Thesis. University of Oslo, 2010.

[38] Magnus Pedersen Lohne. *Coupled-Cluster Studies of Quantum Dots*. MA Thesis. University of Oslo, 2010.

[39] P Löwdin. “Quantum Theory of Many-Particle Systems. I. Physical Interpretations by Means of Density Matrices, Natural Spin-Orbitals, and Convergence Problems in the Method of Configurational Interaction”. In: *Phys. Rev.* 97 (6 1955), pp. 1474–1489. DOI: 10.1103/PhysRev.97.1474.

[40] Patrick Merlot. *Many-Body Approaches to Quantum Dots*. MA Thesis. University of Oslo, 2009.

[41] J. J. Moré and D. J. Thuente. “Line Search Algorithms with Guaranteed Sufficient Decrease”. In: *ACM Trans. Math. Softw.* (1994). DOI: 10.1145/192115.192132.

[42] J. W. Moskowitz and M. H. Kalos. “A New Look At Correlations In Atomic And Molecular Systems. I. Application Of Fermion Monte Carlo Variational Method”. In: *International Journal of Quantum Chemistry* (). DOI: 10.1002/qua.560200508.

[43] J. W. Negele and Orland H. *Quantum Many-Particle Systems*. Perseus BOOKs, 1998. ISBN: 9780738200521.

[44] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer-Verlag New York, 2006. ISBN: 978-1-4939-3711-0. DOI: 10.1007/978-0-387-40065-5.

[45] F. Pederiva. "Ab Initio Computation of the Energies of Circular Quantum Dots". In: *Phys. Rev. B* 84 (11 2011), p. 115302. DOI: 10.1103/PhysRevB.84.115302.

[46] F. Pederiva. "Variational Monte Carlo for Spin-orbit Interacting Systems". In: *Phys. Rev. B* (2012). DOI: 10.1103/PhysRevB.85.045115.

[47] E. Platen and N. Bruti-Liberati. *Numerical Solution of Stochastic Differential Equations with Jumps in Finance*. Springer-Verlag Berlin Heidelberg, 2010. ISBN: 9783642120572.

[48] P. Pulay. "Convergence Acceleration of Iterative Sequences. The Case of SCF Iteration". In: *Chemical Physics Letters* 73.2 (1980), pp. 393–398. ISSN: 0009-2614. DOI: \url{https://doi.org/10.1016/0009-2614(80)80396-4}.

[49] P. Pulay. "Improved SCF convergence acceleration". In: *Journal of Computational Chemistry* 3.4 (), pp. 556–560. DOI: 10.1002/jcc.540030413.

[50] Jack S. and M. Winifred. "Adjustment of an Inverse Matrix Corresponding to a Change in One Element of a Given Matrix". In: *The Annals of Mathematical Statistics* 21 (1950). DOI: 10.1214/aoms/1177729893.

[51] M. J. A. Schuetz et al. "Nuclear Spin Dynamics in Double Quantum Dots: Multistability, Dynamical Polarization, Criticality, and Entanglement". In: *Phys. Rev. B* 89 (19 2014), p. 195310. DOI: 10.1103/PhysRevB.89.195310.

[52] J. P. Sethna. *Statistical Mechanics: Entropy, Order parameters and complexity*. Oxford University Press, USA, 2006. ISBN: 9780198566779.

[53] D. F. Shanno. "Conditioning of quasi-Newton methods for function minimization". In: *Math. Comp.* 0274029 (1970). DOI: 10.2307/2004840.

[54] S. Surjanovic and D. Bingham. *Virtual Library of Simulation Experiments: Test Functions and Datasets*. http://www.sfu.ca/~ssurjano. 2018.

[55] J. Olsen T. Helgaker P. Jørgensen. *Molecular Electronic-Structure Theory*. Wiley, 2014. ISBN: 978-0-47-196755-2. DOI: 10.1002/9781119019572.

[56] J. M. Thijssen. *Computational Physics*. 2nd Edition. Cambridge University Press, 2013. ISBN: 9781107677135.

[57] M. A. Viergever. "Adaptive Stochastic Gradient Descent Optimisation for Image Registration". In: *Int J Comput. Vision* (2009). DOI: 10.1007/s11263-008-0168-y.

[58] Yang Min Wang. *Coupled-Cluster Studies of Double Quantum Dots*. MA Thesis. University of Oslo, 2011.

[59] Wolfram|Alpha. *Chebyshev-Gauss Quadrature*. http://mathworld.wolfram.com/Chebyshev-GaussQuadrature.html. 2018.

[60] Wolfram|Alpha. *Modified Bessel Function of the First Kind*. http://mathworld.wolfram.com/ModifiedBesselFunctionoftheFirstKind.html. 2018.

[61] S. R. Xavier. *Elementary Introduction to Theory of Pseudodifferential Operators*. 1st ed. CRC Press, 1991. ISBN: 9780849371585.