QUANTUM MONTE CARLO STUDIES OF MANY-ELECTRON SYSTEMS

by

Håkon Sebastian Bakke Mørk

THESIS

for the degree of

MASTER OF SCIENCE

Faculty of Mathematics and Natural Sciences
University of Oslo

June 2016

Abstract

In this thesis a simple variational Monte Carlo (VMC) method is used to calculate ground state energies and one-body densities of several quantum systems: the atoms helium, beryllium, and neon, molecules dihydrogen and diberyllium, and quantum dots in two and three dimensions. The method has been implemented with the Metropolis algorithm and importance sampling to improve accuracy. For atoms, in addition to using slater type orbitals, Gaussian type orbitals has also been implemented. Furthermore effects in the quantum dots when the frequency in the harmonic oscillator orbitals is lowered has been studied in both two and three dimensions.

The source code for the developed VMC solver can be found at [Mørk, 2016].

Acknowledgements

I would like to thank everyone who have helped me complete this thesis. A big thanks to Gullik and Emilio for the collaboration in creating our first quantum Monte Carlo solver. Another thanks to Stian for interesting discussions and much needed breaks. And above all, a special thanks to my supervisor, Morten Hjorth-Jensen, for providing an interesting topic, constant support and always great mood.

Contents

Introduction

Quantum mechanical systems are complex. As such creating a specific program for each specific system is not a viable method of studying a range of systems. We need to generalize. A generalized solver for quantum mechanical systems must be written without any constraints to specific properties any system may have. To achieve such a feat the program is best implemented by the use of object orientation, creating an easily expandable solver to which simple or complex systems may be added. In this thesis the aim is to write a generalized variational Monte Carlo solver which, by using object orientation, may handle a wide range of quantum mechanical systems, such as confined electrons in so-called quantum dots, atoms, and molecules.

The variational Monte Carlo method poses an attractive way to solve quantum mechanical systems, compared to other more complex methods, while also taking correlation factors into account, as opposed to the Hartree-Fock method. The attractiveness of the variational Monte Carlo method lies in the way it solves the multi-dimensional integrals arising in the many-body quantum mechanical problem, which, as the name implies, is by using the Monte Carlo method. In this thesis a single so-called Slater determinant is used as an ansatz for the trial wave function. This simplicity makes it easy to implement an efficient and flexible program. It is however a compromise, yielding less accurate results, but nevertheless good enough to study a variety of systems.

The solver presented in this thesis was initially made to solve simple atomic systems, as a reference, and thereby expanding to molecules. To further demonstrate the flexibility of the program, quantum dots are studied in two and three dimensions. The reason for choosing quantum dots to be studied is their simple structure, yet multitude of practical uses.

The aim in this thesis is to demonstrate the flexibility of the program by studying the system like atomic helium, beryllium and neon, the helium and beryllium molecules, benchmarking ground state energies against existing references, and studying their one-body densities. Furthermore quantum dots consisting of up

to 56 electrons will be studied in a similar manner, and their frequency will be varied. With a lower frequency the quantum dots will implicitly have a higher correlation, and studying correlations are of great importance when using more elaborate methods than for example the simple Hartree-Fock method.

Ground states of atoms and molecules are compared to experimental results, which should be close to the exact results, offering a good test of the accuracy of the variational Monte Carlo method and the solver created. Because of the popularity of quantum dots several master students have studied them, each with different methods. This provides a wide range of references to which ground state energies may be compared, which should give further insight to the accuracy of the solver.

The thesis is structured in two parts: a theory part, and a results part. A brief description of the chapters is given below.

- In the first chapter a brief introduction to scientific computing is given. In it different types of programming languages will be described, and we will give an introduction to object-oriented programming. This is important because object-oriented programming is used to create a generalized solver. A summary of message passing interface, used to parallelize the computations, is also given.

- The second chapter aims to give an overview of a more basic solution method, the Hartree-Fock method. It also describes other methods derived from the Hartree-Fock method, so-called post Hartree-Fock methods. The Hartree-Fock method can be used as a convenient test for a more simplified variational Monte Carlo program, and some of the post Hartree-Fock methods are used in computations of references when ground state energies computed by the solver is benchmarked.

- Next the variational Monte Carlo method is explained with details around ways of optimization, like the Metropolis algorithm and importance sampling. Details about how to calculate the Slater determinant efficiently and other measures to further optimize the solver are also given. Then the process of blocking to get an accurate estimate of the error is explained.

- In the fourth chapter the modelled systems are described. How quantum dots are modelled in two and three dimensions is described, then a brief description of how each of the atomic systems are modelled is given. A way to replace the Slater type orbitals with gaussian type orbitals is also described.

- In the final chapter of the first part we look at how the solver is structured, and how the variational Monte Carlo method is implemented using object-orientation.
- With the theory in place we look at the results from calculations with the variational Monte Carlo solver created. The program is benchmarked to test the optimizations and also to see how well it scales with an increasing number of processors used. The systems are benchmarked against reference ground state energies, and one-body densities are calculated, which can give insight into the structure of the systems.
- Finally a conclusion is given with final remarks.

The source code for the developed VMC solver can be found at [Mørk, 2016].

Part I

Theory

Chapter 1

Scientific Computing

A great deal of problems cannot be solved using analytical methods. And except for very simple problems, it is impossible to find an analytical solution. For this reason scientific computing is a rapidly growing field, and is now applied in a wide range of fields. These include not only natural sciences and mathematics, but also medicine, sociology, and economics to name a few. In physics, the dawn of computational solution-methods gave rise to the field of computational physics, and is today used in every major field from astrophysics to quantum physics.

In essence scientific computing is taking a real world phenomena, which often are chaotic and noisy, and making a model that is abstract and perhaps simplified. The model is then used in an automated simulation to solve the problem at hand. This means that the model has to be implemented in a computer program and executed on a computer.

1.1 Types of programming languages

A computer program contains a set of instructions to be executed by the computer. These are usually in the form of an algorithm, that is a series of instructions designed to solve a given problem, or perform a specific task.

As with human languages there exists many different computer languages, such as Python, C++ or Java. However the machine is almost never able to perform

the instructions written as they are[1]. To carry out the written instructions the program has to be compiled or interpreted, depending on the language. Using a compiler, the program is compiled in advance of execution and creates an executable program which can be run on the central processing unit (CPU). If the program is written in a scripted language, however, the code is fed to an interpreter at execution time, which processes the written instructions and carries them out on the CPU.

It is well known that the native language of the computer is binary. In other words it is made up of ones and zeros called bits, corresponding to high and low voltage readings in the CPU. Everything from numbers to words to colors and so on is represented in the computer as a sequence of bits. Thus the purpose of the programming language is to convey instructions from the programmer to be carried out by the machine.

There are two kinds of programming languages: High-level and low-level languages. The terms are inherently relative, as there are more than two grades of abstraction of programming languages. A language like C, today regarded as a low-level language, would some decades ago be referred to as a high-level language, when compared to assembly language which was referred to as a low-level language. However assembly language can be regarded as a higher-level language than machine code, which again is slightly higher than the microcode used internally in the processor.

High-level languages

High-level languages have a high level abstraction from the machine language, may use more natural language, and often automate details such as memory management. They focus on usability rather than efficiency, and are therefore preferable when the task is less taxing on the processor, making efficiency less important. High-level languages are often used with a single task in mind, such as handling input and output from other tools, analyzing data, or simple computations. In short, specific codes like these are often referred to as scripts, and consequently high-level languages used for tasks like these are called scripting languages [Langtangen, 2006].

Programs written in high-level languages tend to execute slower than their low-level counterpart. Because of this, for computationally intensive tasks it is favor-

[1] One exception is if the program is written in machine code, which can be executed (almost) directly by the processor. But writing a program in machine code is tiresome and error-prone, and is thus rarely done except in extreme cases.

able to use a low-level language for it's improved efficiency. High level languages also give the programmer reduced control over memory-handling due to the languages' more simplistic style. Some examples of popular high-level languages are Python, Perl, Ruby, PHP, and Visual Basic.

As an example, let us consider the sum

$$\sum_{i=1}^{100} i = 5050. \tag{1.1.1}$$

Using Python, here is the code for calculating the sum, and showing the result on the command line

```
1 print sum(range(101))
```

We see that the command is simple and straightforward, and done in a single line. The `range`-function creates a list of numbers in arithmetic progression. Called with a single argument, like here, it starts in 0 and counts up to $n-1$, where n is the argument.

For an introduction to Python in scientific programming, see for example Ref. [Langtangen, 2011].

Low-level languages

Low-level languages have a low level of abstraction from the technicalities of a computer, and can be described as being closer to the hardware. Languages regarded as low-level include C, C++, Fortran, Basic, machine code or assembly language. The low abstraction gives the programmer more control over for example how to handle memory.

There are several caveats with writing a program using a low-level language, however. The programmer has to remember numerous technical details to make a working program, such as declarations, pointers, compiling and linking, making the task of writing the code more complex. Nonetheless this increased complexity does give the programmer more control over the technical details of how the program works, which makes it possible to write flexible and optimized programs. The reward for writing a program in such a complex language, compared to high-level languages, is ending up with a highly efficient program.

Following the same example as for high-level languages, performing the calculation of the sum in Eq. (1.1.1) by using C++ we have

```cpp
#include <iostream>
int main() {
  int sum = 0;
  for (int i = 1; i <= 100; i++) {
    sum += i;
  }
  std::cout << sum << std::endl;
  return 0;
}
```

Comparing this to the Python case we see that using C++ is more complicated; we need to define the type of all variables, and there is no built-in function to carry out the sum. We also see that everything is done from the `main`-function, the designated starting-point in the C++ language. With these examples it is now clear that low-level languages like C++, although more complicated, give more control in addition to carry out the calculations faster.

The calculations carried out when solving a quantum mechanical system by using variational Monte Carlo, which is the main focus in this thesis, are quite heavy. It is therefore unfeasible to do the calculations in a high-level language. Furthermore a low-level language provides more control and makes it possible to make the program flexible. Therefore the language C++ is chosen as the language used in the variational Monte Carlo calculations, as it is a low-level language and has excellent support for object orientation. The high-level language Python is also used, although only for analyzing the resulting data. Therefore the focus will be on these languages in the following, with emphasis on the C++ language as it is used for the variational Monte Carlo machinery.

1.2 Object-oriented programming

Although some of the terminology and ideas of object-oriented programming appeared earlier, the programming concept of objects was introduced in the 1960s with Simula 67, developed by Ole-Johan Dahl and Kristen Nygård of the Norwegian Computing Center [Holmevik, 1994]. Object-oriented programming has since become the dominant technique of modern programming and is today supported by most if not all of the most popular programming languages.

Objects in programming are similar to everyday objects. They both have two characteristics: a state and a behaviour. A lamp has commonly two states, on and off, and two behaviours, turn on and turn off. Meanwhile a bicycle has more

states, like current speed, current gear, current pedal cadence, and so on, and more behaviour, like change gear, change pedal cadence, and apply breaks. One of the big strengths of object-oriented programming is that it is intuitive in that it translates well to real-world objects.

A class consists of fields, defining data of the current state of the class, and methods, operating on the internal state of the class, that is working with fields of the class. Calling a class creates an instance, or an executable copy of the class, also called an object. This way there can be multiple copies of objects of a given class in memory at any time. There can also be classes within classes, giving us a hierarchy of classes.

By writing the variational Monte Carlo solver by using object-orientation, the program can be made very flexible. Specifically, the solver expects to be connected to a system, such as an atom, or quantum dot, in form of an object. It is therefore easy to expand the variational Monte Carlo machinery to solve additional systems simply by creating a separate class for a new system. Because of the importance of object-orientation in the implementation of the solver, a brief introduction to concepts regarding object-orientation will be given in the following.

Members

It is now clear that a class consists of variables and functions, called members of the class. Every instance of the class has its own set of members, but also a member which points to itself. In Python this is called the `self` member, and in C++ it is a pointer called `this`. This way all the members of the class has access to the class they are part of, and the class can make changes to itself at any time.

Constructors

By creating an instance of a class, the constructor function of the class is called. The constructors job is to initializes the class with its data members, and it may take some arguments. In C++ the constructur matches the name of the class itself, while in Python the constructor is called `__init__()`.

As an example, here is how to create an instance of a class `rectangleClass` in C++, simply by calling the constructor

```
rectangleClass* rectangleObject = new rectangleClass(height, width);
```

Destructors

When the program is done with an object, the object has to be deallocated from the memory. The function of the class handling this is called the destructor. As we have talked about, handling the memory is automatic in Python, thus the memory is automatically deallocated by the so-called garbage collector. In C++ however it is important to make sure that the memory reserved by the object is deallocated. Deallocation of memory allocated by the object is handled by a special function in the class declared as the name of the class preceded by a tilde, thus the destructor for the class in the previous example would be `~rectangleClass()`.

Levels of Accessibility

A class may disallow calling code from outside the class itself. This is called encapsulation. Encapsulation is useful because it prevents external code from being involved in the internal functions of the class. In some languages, like Python, there is limited access to methods of encapsulation. In C++ there are three levels of accessibility. With the `Private` keyword access to the data is restricted to only inside the class, while the `Public` keyword allows access to the data from outside the class. There is also an intermediate level, `Protected`, which restricts access to inside the class and also subclasses of the class.

Inheritance

Another useful feature of classes is inheritance. This allows classes to have a hierarchy, inheriting properties from its parent classes. For example, a class `Student` may inherit from a class `Person`. The `Student` class may have variables such as `Institute`, `Subjects` and `Advisor`, while it may inherit variables `First_name`, `Last_name` and `Gender` from the class `Person`. Methods and variables defined by a superclass can also be overwritten by the child class.

Typecasting and pointers

A variable can be stored in the computer's memory as different types, such as integers, floating point numbers and strings. In Python this classification is handled automatically, and there is no need to cast a variable as a specific type. In low-level languages such as C++ however we have more control over the memory, and the type of the variable has to be specified.

In the memory objects and variables are stored at memory addresses. In C++ and other low-level languages we can interface with this memory address directly, through pointers. So instead of modifying the value itself, we can modify the value which is stored in memory.

Modifying values in memory directly is very useful because memory addresses are shared through the program. Changes done through a pointer is therefore effective everywhere where the object pointed to is used. Passing a variable as an argument to a function would normally make the function create its own copy of the variable, which is destroyed when exiting the function. However passing a pointer to the function would let the function modify the original variable.

Virtual members and polymorphism

In a hierarchy of classes the subclass can redefine a member function of the superclass, if the function is defined as a virtual function in the superclass. As an example, consider this superclass for polygons, and its subclass for rectangles:

```
1  class Polygon{
2    protected:
3      int width, height;
4    public:
5      void set_values(int a, int b)
6        {width=a; height=b;}
7      virtual int area()
8        {return 0;}
9  };

11 class Rectangle: public Polygon {
12   public:
13     int area()
14       {return width * height;}
15 };
```

Because a subclass is type-compatible with a pointer to its superclass, we may typecast the subclass `Rectangle` as

```
1 Polygon* rectangleObject = new Rectangle();
```

Now we may call the function `rectangleObject::set_values()`[2], which is defined in the superclass, to set the width and height of the rectangle. We can also call the function `rectangleObject::area()` to get the area of the rectangle. Note that this function is redefined by the `Rectangle` subclass, and thus the function in the `Polygon` superclass is overwritten.

When classes are organized in a hierarchy and are related by inheritance, in the fashion exemplified above, they are said to be polymorphic. As the name suggests, a polymorphic function may have many different shapes, first defined as a function in the superclass, but redefined through functions in the subclasses.

Polymorphic classes are used in the variational Monte Carlo solver in this thesis to implement the quantum mechanical systems. A class `Trialfunction` defines all the functions needed from the system by the solver, but they have to be overwritten by a subclass of the `Trialfunction`-class. It is in this subclass the system, be it an atom or a quantum dot, is defined. For a more thorough overview of the solver classes, see section 5.1.

1.3 Implementation of MPI

As calculations become increasingly complex and heavy, running computations on a single processor becomes impractical. Because of physical limitations and the difficulty of continuously making processors faster, a different strategy has been developed, that is using multiple processing cores in a single processor. Therefore personal computers today, and even some phones, contain multiple processing cores.

Taking advantage of all available processors on the machine a calculation is carried out is very useful and an easy way to cut computation times to a fraction of how long it would take with a single processor. Because of the heavy calculations in scientific programming it is usual to run computing jobs on a shared cluster computer, often containing thousands or even millions of processing cores. To take advantage of this it is necessary to implement a message passing interface,

[2]The double colon, `::`, is the scope resolution operator, and here it simply means "member function `set_values()` of class `rectangleObject`".

or MPI, to make use of multiple processors.

One of the advantages of using MPI is its good performance and control over message passing between processors. There exists other ways to take advantage of multiple processors, but MPI is one of the most common methods, and is thus available at most computing clusters.

A program written, compiled and run using MPI runs simultaneously on all processes. At the core of an MPI-implementation is the MPI communicator, a communication universe for a group of processes. The default MPI communicator is called `MPI_COMM_WORLD` and is the collection of all processes. Within the communicator each process is identified by a unique rank, an integer identifier used to distinguish one process from another.

By using the MPI communicator each process can send and receive data between each other, using for example the functions `MPI_Send` and `MPI_Recv`. It is also possible to distribute data from one process to all others, or to gather data from all processes to a single process, by using for example `MPI_BCAST` and `MPI_GATHER`, respectively.

Not all computational methods can be easily split up and be distributed on multiple processors. However one of the easier methods is the Monte Carlo method. As we deal with statistical values we can easily split up the problem. Each process will run its own set of samples. The number of samples used by each process is simply n/p, where n is the total number of samples we want to do, and p is the number of processes. In the end a master process gathers the results from all processes and sums up the values, taking the average over all processes.

Chapter 2

Hartree-Fock

Perhaps one of the most common and fundamental methods of approximating wave functions and electronic structure, and solving the many-body Hamiltonian, is the Hartree-Fock method.

The Hartree-Fock method is based on a procedure introduced by D. R. Hartree in 1928 [Hartree, 1928], which he called the self-consistent field method, later known as the Hartree method. With this method Hartree wanted to solve the many-body time-independent Schrödinger equation from first principles. It was incomplete in many respects, such as neglecting antisymmetry of the wave function. This is taken care of by incorporating a Slater determinant in the calculations, giving us the Hartree-Fock method, proposed in 1930 by Fock [Fock, 1930b], and independently Slater the same year [Slater, 1930].

2.1 The Hartree-Fock method

The Hartree-Fock method simplifies the problem by making a series of approximations. This makes the problem reasonably easy to solve iteratively, but can give great deviations from experimental results. It is however a great foundation for the so-called post Hartree-Fock methods, which more accurately describe the many-body problem by including electron correlation in normally approximative ways.

The idea behind the Hartree-Fock method is to think of each electron as a single particle wave function, without correlation from other electrons. This gives us a separable Hamiltonian, and the total wave function of system, called the Hartree

function, becomes the product of the wave functions of each electron,

$$\Phi_H\left(x_1,\, x_2,\, \ldots,\, x_N,\, \alpha,\, \beta,\, \ldots,\, \sigma\right) = \psi_\alpha\left(x_1\right)\psi_\beta\left(x_2\right)\ldots\psi_\sigma\left(x_N\right).$$

However, the Hartree product does not, unfortunately, satisfy the antisymmetry principle. A better approximation to the wave function is by using a Slater determinant,

$$\Phi\left(x_1,\, x_2,\, \ldots,\, x_N,\, \alpha,\, \beta,\, \ldots,\, \sigma\right) = \frac{1}{\sqrt{N}}\begin{vmatrix} \psi_\alpha\left(x_1\right) & \psi_\alpha\left(x_2\right) & \ldots & \psi_\alpha\left(x_N\right) \\ \psi_\beta\left(x_1\right) & \psi_\beta\left(x_2\right) & & \vdots \\ \vdots & \vdots & \ddots & \\ \psi_\sigma\left(x_1\right) & \psi_\sigma\left(x_2\right) & & \psi_\sigma\left(x_N\right) \end{vmatrix}, \tag{2.1.1}$$

named after Slater, who introduced the determinant to handle antisymmetry [Slater, 1929].

We assume that that we can approximate the interacting part of the Hamiltonian by a two-body interaction. This gives us a total Hamiltonian as a sum of a one-body part and a two-body part,

$$\hat{H} = \hat{H}_0 + \hat{H}_I = \sum_{i=1}^{N}\hat{h}_0\left(x_i\right) + \sum_{i<j}^{N}\hat{v}\left(r_{ij}\right),$$

with

$$\hat{H}_0 = \sum_{i=1}^{N}\hat{h}_0\left(x_i\right).$$

The single-particle functions $\psi_\alpha(x_i)$ are eigenfunctions of the one-body Hamiltonian, h_i. The one-body Hamiltonian is

$$\hat{h}_0\left(x_i\right) = \hat{t}\left(x_i\right) + \hat{u}_{ext}\left(x_i\right),$$

with eigenvalues

$$\hat{h}_0\left(x_i\right)\psi_\alpha\left(x_i\right) = \left(\hat{t}\left(x_i\right) + \hat{u}_{ext}\left(x_i\right)\right)\psi_\alpha\left(x_i\right) = \epsilon_\alpha\psi_\alpha\left(x_i\right).$$

The resulting energies ϵ_α are non-interacting single-particle energies. In this case, if there are no many-body interactions, the total energy is the sum over all single-particle energies.

We can rewrite the Slater determinant as

$$\Phi\left(x_1,\, x_2,\, \ldots,\, x_N,\, \alpha,\, \beta,\, \ldots,\, \sigma\right) = \frac{1}{\sqrt{N!}} \sum_P (-)^P \hat{P}\psi_\alpha\left(x_1\right)\psi_\beta\left(x_2\right) \ldots \psi_\sigma\left(x_N\right) = \sqrt{N!}\hat{A}\Phi_H,$$

where $\hat{A}$ is an introduced anti-symmetrization operator, defined as

$$\hat{A} = \frac{1}{N!} \sum_p (-)^p \hat{P},$$

where p denotes number of permutations.

The one-particle and two-particle Hamiltonians are both invariant under all possible permutations of any two particles, thus they commute with $\hat{A}$

$$\left[H_0,\, \hat{A}\right] = \left[H_I,\, \hat{A}\right] = 0. \tag{2.1.2}$$

Furthermore, because every permutation of the Slater determinant reproduces it, $\hat{A}$ satisfies

$$\hat{A}^2 = \hat{A}. \tag{2.1.3}$$

We have the expectation value of $\hat{H}_0$,

$$\int \Phi^* \hat{H}_0 \Phi d\tau = N! \int \Phi_H^* \hat{A}\hat{H}_0\hat{A}\Phi_H d\tau,$$

which, by using Eqs. (2.1.2) and (2.1.3), we can write as

$$\int \Phi^* \hat{H}_0 \Phi d\tau = N! \int \Phi_H^* \hat{H}_0\hat{A}\Phi_H d\tau.$$

Now we replace $\hat{H}_0$ with the sum of one-body operators, and the anti-symmetrization operator with its definition

$$\int \Phi^* \hat{H}_0 \Phi d\tau = \sum_{i=1}^{N} \sum_p (-)^p \int \Phi_H^* \hat{h}_0 \hat{P}\Phi_H d\tau.$$

Because of the orthogonality of the single-particle wave functions, the sum over p vanishes as two or more particles are permuted in one of the Hartree-functions Φ_H, and we get

$$\int \Phi^* \hat{H}_0 \Phi d\tau = \sum_{i=1}^{N} \int \Phi_H^* \hat{h}_0 \Phi_H d\tau.$$

Furthermore, the orthogonality lets us simplify further and get

$$\int \Phi^* \hat{H}_0 \Phi d\tau = \sum_{\mu=1}^{N} \int \psi_\mu^*(x) \hat{h}_0 \psi_\mu(x) dx d\mathbf{r}.$$

This expression can be rewritten with braket notation, giving us

$$\int \Phi^* \hat{H}_0 \Phi d\tau = \sum_{\mu=1}^{N} \langle \mu | \hat{h}_0 | \mu \rangle.$$

Similarly we can obtain the expectation value of the two-body part of the Hamiltonian. Here we have

$$\int \Phi^* \hat{H}_I \Phi d\tau = N! \int \Phi_H^* \hat{A} \hat{H}_I \hat{A} \Phi_H d\tau,$$

which in the same way as the one-body Hamiltonian reduces to

$$\int \Phi^* \hat{H}_I \Phi d\tau = \sum_{i \leq j=1}^{N} \sum_{p} (-)^p \int \Phi_H^* \hat{v}(r_{ij}) \hat{P} \Phi_H d\tau.$$

The expectation value depends on the inter-particle distance, r_{ij}, and therefore permutations of any two particles will not vanish. While still assuming that the single-particle wave functions are orthogonal, we get

$$\int \Phi^* \hat{H}_I \Phi d\tau = \sum_{i<j=1}^{N} \int \Phi_H^* \hat{v}(r_{ij})(1 - P_{ij}) \Phi_H d\tau,$$

where P_{ij} is a permutation operator that interchanges particle i and particle j.

We then get

$$\int \Phi^* \hat{H}_I \Phi d\tau = \frac{1}{2} \sum_{\mu=1}^{N} \sum_{\nu=1}^{N} \left[\int \psi_\mu^*(x_i) \, \psi_\nu^*(x_j) \, \hat{v}(r_{ij}) \, \psi_\mu(x_i) \, \psi_\nu(x_j) \, dx_i dx_j \right.$$
$$\left. - \int \psi_\mu^*(x_i) \, \psi_\nu^*(x_j) \, \hat{v}(r_{ij}) \, \psi_\nu(x_i) \, \psi_\mu(x_j) \, dx_i dx_j \right].$$

The two integrals on the right hand side are called direct or Hartree term, and exchange or Fock term. As we now run over all pairs twice, we have introduced a factor 1/2. We have also used the shorthand $d\tau = dx_1 dx_2 \ldots dx_N$.

The full functional becomes

$$E[\Phi] = \sum_{\mu=1}^{N} \psi_\mu^*(x_i)\,\hat{h}_0\psi_\mu(x_i)\,dx_i + \frac{1}{2}\sum_{\mu=1}^{N}\sum_{\nu=1}^{N}\left[\int \psi_\mu^*(x_i)\,\psi_\nu^*(x_j)\,\hat{v}(r_{ij})\,\psi_\mu(x_i)\,\psi_\nu(x_j)\,dx_i dx_j - \int \psi_\mu^*(x_i)\,\psi_\nu^*(x_j)\,\hat{v}(r_{ij})\,\psi_\nu(x_i)\,\psi_\mu(x_j)\,dx_i dx_j\right].$$

Written with the more manageable braket notation, the expression becomes

$$E[\Phi] = \sum_{\mu}^{N}\langle\mu|\,\hat{h}_0\,|\mu\rangle + \frac{1}{2}\sum_{\mu\nu}^{N}\left[\langle\mu\nu|\,\hat{v}\,|\mu\nu\rangle - \langle\nu\mu|\,\hat{v}\,|\mu\nu\rangle\right]. \tag{2.1.4}$$

The interaction is invariant under the interchange of two particles, and we have

$$\langle\mu\nu|\,\hat{v}\,|\mu\nu\rangle = \langle\nu\mu|\,\hat{v}\,|\nu\mu\rangle.$$

We can write the direct and exchange matrix elements in a more compact way, by defining the anti-symmetrized matrix element,

$$\langle\mu\nu|\,\hat{v}\,|\mu\nu\rangle_{AS} = \langle\mu\nu|\,\hat{v}\,|\mu\nu\rangle - \langle\mu\nu|\,\hat{v}\,|\nu\mu\rangle,$$

which has the symmetry property

$$\langle\mu\nu|\,\hat{v}\,|\mu\nu\rangle_{AS} = -\langle\nu\mu|\,\hat{v}\,|\mu\nu\rangle_{AS} = -\langle\mu\nu|\,\hat{v}\,|\nu\mu\rangle_{AS}.$$

It is also hermitian, that is

$$\langle\mu\nu|\,\hat{v}\,|\sigma\tau\rangle_{AS} = \langle\sigma\tau|\,\hat{v}\,|\mu\nu\rangle_{AS}.$$

We can now rewrite the Hartree-Fock functional as

$$\int \Phi^*\hat{H}_I\Phi d\tau = \frac{1}{2}\sum_{\mu=1}^{N}\sum_{\nu=1}^{N}\langle\mu\nu|\,\hat{v}\,|\mu\nu\rangle_{AS},$$

and adding the contribution from the one-body operator, we get the energy functional

$$E[\Phi] = \sum_{\mu=1}^{N}\langle\mu|\,\hat{h}_0\,|\mu\rangle + \frac{1}{2}\sum_{\mu=1}^{N}\sum_{\nu=1}^{N}\langle\mu\nu|\,\hat{v}\,|\mu\nu\rangle_{AS}. \tag{2.1.5}$$

The Hartree-Fock energy in Eq. (2.1.5) is a decent approximation to the true ground state energy, although leaves a lot to be desired in terms of accuracy. However it may be used to test the variational Monte Carlo program and how the Slater determinant is handled in it. For a small number of particles the energy given by Eq. (2.1.5) can be calculated by pen and paper. Furthermore, by neglecting the so-called Jastrow-factor in the calculation of the wave function in the variational Monte Carlo simulation, the VMC solver should calculate energies which are close to the energies calculated by Eq. (2.1.5). This can be used to benchmark the variational Monte Carlo solver.

2.2 Hartree-Fock for Quantum Dots

In this thesis a main focus is on quantum dots in harmonic oscillators. For Hartree-Fock for such a system we can make some simplifications to the one-body and two-body integrals.

The one-body operator is diagonal for states i and j, each with quantum numbers n, m_l and m_s. The expectation value of the one-body operator is thus

$$\langle i|\hat{h}_0|j\rangle = e_i\delta_{i,j} = e_{n_i m_{l_i}} = \hbar\omega(2n_i + |m_{l_i}| + 1).$$

The quantum numbers n_i and m_{l_i} represent the number of nodes of the wave function, and the projection of the orbital momentum, respectively, while m_{s_i} represents spin. The one-body expectation value is however independent of spin.

Using Laguerre polynomials, with $\alpha = \sqrt{m\omega/\hbar}$ we get the single-particle wave functions

$$\psi_{nm_l}(r,\theta) = \alpha\exp(\imath m\theta)\sqrt{\frac{n!}{\pi(n+|m|)!}}(\alpha r)^{|m|}L_n^{|m|}(\alpha^2 r^2))\exp\big(-\alpha^2 r^2/2\big).$$

The $L_n^{|m|}$ here are the Laguerre polynomials. For the system we are studying we will only need $L_n^{|m|}(x) = 1$ and $L_1^0(x) = -x + 1$. Using these functions we can compute the integral that defines the two-body matrix elements that comes from repulsive Coulomb interaction

$$\langle\alpha\beta|V|\gamma\delta\rangle = \int r_1 dr_1 d\theta_1 \int r_2 dr_2 d\theta_2 \psi^*_{n_\alpha m_{l_\alpha}}(r_1,\theta_1)\psi^*_{n_\beta m_{l_\beta}}(r_2,\theta_2) \frac{1}{|\mathbf{r}_1 - \mathbf{r}_2|}\psi_{n_\gamma m_{l_\gamma}}(r_1,\theta_1)\psi_{n_\delta m_{l_\delta}}(r_2,\theta_2).$$

This is also independent of spin, and doing our Hartree-Fock calculations we have to consider spin degrees of freedom as well.

To calculate the Coulomb matrix elements, we use the expression from article in Ref. [Anisimovas and Matulis, 1998]. For simplicity we write the angular momentum projection quantum number m_l as m. The expression for the Coulomb integral can thus be written as

$$\begin{aligned} V_{1234} &= \delta_{m_1+m_2,m_3+m_4}\sqrt{\left[\prod_{i=1}^{4}\frac{n_i!}{(n_i+|m_i|!)}\right]} \\ &\times \sum_{j_1=0,\ldots,j_4=0}^{n_1,\ldots,n_4}\left[\frac{(-1)^{j_1+j_2+j_3+j_4}}{j_1!j_2!j_3!j_4!}\left[\prod_{k=1}^{4}\binom{n_k+|m_k|}{n_k-j_k}\right]\frac{1}{2^{\frac{G+1}{2}}}\right. \\ &\times \sum_{l_1=0,\ldots,l_4=0}^{\gamma_1=0,\ldots,\gamma_4=0}\left(\delta_{l_1,l_2}\delta_{l_3,l_4}(-1)^{\gamma_2+\gamma_3-l_2-l_3}\left[\prod_{t=1}^{4}\binom{\gamma_t}{l_t}\right]\right. \\ &\left.\left.\times\Gamma\left(1+\frac{\Lambda}{2}\right)\ \Gamma\left(\frac{G-\Lambda+1}{2}\right)\right)\right], \end{aligned}$$

where we have defined

$$\begin{aligned} \gamma_1 &= j_1+j_4+\frac{|m_1|+m_1}{2}+\frac{|m_4|-m_4}{2} \\ \gamma_2 &= j_2+j_3+\frac{|m_2|+m_2}{2}+\frac{|m_3|-m_3}{2} \\ \gamma_3 &= j_3+j_2+\frac{|m_3|+m_3}{2}+\frac{|m_2|-m_2}{2} \\ \gamma_4 &= j_4+j_1+\frac{|m_4|+m_4}{2}+\frac{|m_1|-m_1}{2} \\ G &= \gamma_1+\gamma_2+\gamma_3+\gamma_4 \\ \Lambda &= l_1+l_2+l_3+l_4. \end{aligned}$$

We now expand the single-particle functions and vary the coefficients. This means that the new single-particle wave function is written as a linear expansion in terms of a fixed basis ϕ

$$\psi_p = \sum_{\lambda} C_{p\lambda}\phi_{\lambda}.$$

We minimize with respect to $C^*_{p\alpha}$ and define

$$h^{HF}_{\alpha} = \langle\alpha|h|\gamma\rangle + \sum_{p}\sum_{\beta\delta} C^*_{p\beta}C_{p\delta}\langle\alpha\beta|V|\gamma\delta\rangle_{AS}.$$

The Hartree-Fock equations can now be written as

$$\sum_{\gamma} h^{HF}_{\alpha\gamma} C_{p\gamma} = \epsilon^{\mathrm{HF}}_{p} C_{p\alpha}.$$

This way the energy can be minimized in an iterative way using the coefficients. However the accuracy in the Hartree-Fock method is far from perfect as it neglects correlation between electrons. The Hartree-Fock method includes however the Pauli principle via the exchange term. In the next section we will look at some methods based on the Hartree-Fock method which try to deal with this.

2.3 Post Hartree-Fock methods

The main issue with the Hartree-Fock method is that it uses an average potential for interaction between electrons. Therefore the energies calculated using this method are not particularly accurate. Because of interaction between electrons, their motion is correlated. The exact wave function therefore has to depend on the coordinates of all electrons simultaneously. The Hartree-Fock method neglects these additional correlations and assumes that the electrons move independently. However, to improve the accuracy, correlations have to be taken into account. Several methods based on the Hartree-Fock method have been devised, with better management of the electron-electron interaction by attempting to add correlation. In this section an overview of these methods will be given.

2.3.1 Configuration interaction

The configuration interaction method [Shavitt and Bartlett, 2009]
[Maurice and Head-Gordon, 1999] expands upon the Hartree-Fock method by using a linear combination of excited states. An N-electron wave function ground state is then

$$|\Psi_0\rangle = C_0 |\Phi_0\rangle + \sum_{i,a} C^a_i |\Phi^a_i\rangle + \sum_{i<j,a<b} C^{ab}_{ij} |\Phi^{ab}_{ij}\rangle + \ldots \quad \text{(up to } N \text{ excitations)}, \tag{2.3.1}$$

where Φ^a_i is a configuration that has a single excitation, Φ^{ab}_{ij} is a configuration with a double excitation, and so on. This way all possible excitations can be represented. We can rewrite it to a more compact form

$$|\Psi_0\rangle = \sum_{PH} C^P_H \Phi^P_H, \tag{2.3.2}$$

where H is 0, 1, ..., N hole states, and P is 0, 1, ..., N particle states. We have the normalization requirement

$$\langle\Psi_0|\Phi_0\rangle = \sum_{PH} \left|C_H^P\right|^2 = 1, \tag{2.3.3}$$

and the energy can then be written as

$$E = \langle\Psi_0|\,\hat{H}\,|\Phi_0\rangle = \sum_{PP'HH'} C^{*P}_{\ H}\left\langle\Phi_H^P\right|\hat{H}\left|\Phi_{H'}^{P'}\right\rangle C_{H'}^{P'}. \tag{2.3.4}$$

The energy in Eq. (2.3.4) is solved by setting up a Hamiltonian matrix and diagonalizing.

By including all possible configurations this method is called Full Configuration Interaction (FCI). However the size of the matrix to be diagonalized grows exponentially, and quickly becomes impossible for us to solve. The CI expansion therefore needs to be truncated, and is often truncated after the double-excitation level [Shavitt and Bartlett, 2009]. The truncated CI expansion gives a less accurate solution than the Full CI expansion, but has the benefit of being solvable for larger systems.

2.3.2 Many-body perturbation theory

As the name implies, many-body perturbation theory treats parts of the Hamilotian (normally the two-body interaction) as a perturbation in order to account for the correlation in electron-electron interaction. The wave function can be rewritten to

$$|\Psi_0\rangle = \left[|\Phi_0\rangle\langle\Phi_0| + \sum_{m=1}^{\infty}|\Phi_m\rangle\langle Phi_m|\right]|\Psi_0\rangle\,. \tag{2.3.5}$$

We can rewrite this with with the projections operators

$$\hat{P} = |\Phi_0\rangle\langle\Phi_0| \qquad \text{and} \qquad \hat{Q} = \sum_{m=1}^{\infty}|\Phi_m\rangle\langle Phi_m|\,,$$

giving us

$$|\Psi_0\rangle = \left[\hat{P}+\hat{Q}\right]|\Psi_0\rangle\,. \tag{2.3.6}$$

Using commutator relations, adding a factor ω, and inserting into the Schrödinger equation, it can be shown that one ends up with [Shavitt and Bartlett, 2009]

$$|\Psi_0\rangle = |\Phi_0\rangle + \frac{\hat{Q}}{\omega-\hat{H}_0}\left(\omega - E + \hat{H}_I\right)|\Psi_0\rangle \tag{2.3.7}$$

$$= \sum_{n=0}^{\infty}\left[\frac{\hat{Q}}{\omega-\hat{H}_0}\left(\omega - E + \hat{H}_I\right)\right]^n|\Psi_0\rangle\,, \tag{2.3.8}$$

for the wave function, and

$$\Delta E = \langle \Phi_0 | \hat{H}_I | \Psi_0 \rangle \tag{2.3.9}$$

$$= \sum_{n=0}^{\infty} \langle \Psi_0 | \hat{H}_I \left[\frac{\hat{Q}}{\omega - \hat{H}_0} \left(\omega - E + \hat{H}_I \right) \right]^n | \Phi_0 \rangle , \tag{2.3.10}$$

which is a perturbative expansion of the exact energy in terms of the interaction $\hat{H}_I$ and the unperturbed wave function $|\Psi_0\rangle$.

We can choose different values of ω in order to obtain different types of perturbation theory expansions. If we choose $\omega = E$ we get the Brillouin-Wigner perturbation theory [Brillouin, 1932] [Wigner, 1935]. Using the this we need to solve the energy iteratively, which is doable, but impractical.

A better choice is then to set

$$\omega = E_0 = \langle \Phi_0 | \hat{H}_0 | \Phi_0 \rangle ,$$

which yields

$$\Delta E = \sum_{n=0}^{\infty} \langle \Psi_0 | \hat{H}_I \left[\frac{\hat{Q}}{E_0 - \hat{H}_0} \left(\hat{H}_I - \Delta E \right) \right]^n | \Phi_0 \rangle . \tag{2.3.11}$$

This is called the Rayleigh-Schrödinger perturbation theory, which was presented by Schrödinger in a paper in 1926 [Schrödinger, 1926].

The Rayleigh-Schrödinger perturbation theory is a good method for simpler problems, but for higher orders (where n is large) it becomes increasingly impractical.

2.3.3 Coupled cluster

The final post Hartree-Fock method we will look at is the Coupled cluster (CC) method. Initially developed by Fritz Coester and Hermann Kümmel for nuclear physics [Coester and Kümmel, 1960], and modified by Jiři Čížek [Čížek, 1966] [Čížek, 1969], later with Paldus [Čížek and Paldus, 1971], to be used for electronic structure theory, it has become one of the most popular methods for calculating correlations in quantum chemistry.

The coupled cluster method is similar to the configuration interaction method, but CC uses an exponential ansatz for the wave function

$$|\Psi\rangle = e^{\hat{T}} |\Phi_0\rangle = \left(\sum_{n=1}^{N} \frac{1}{n!} \hat{T}^n \right) |\Phi_0\rangle , \tag{2.3.12}$$

where n is the number of particle-hole-excitations (np-nh), and $\hat{T}$ is the cluster operator,

$$\hat{T} = \hat{T}_1 + \hat{T}_2 + \cdots + \hat{T}_N. \tag{2.3.13}$$

Here $\hat{T}_1$ is the one-body cluster operator, $\hat{T}_2$ is the two-body cluster operator, and so on. These n-body cluster operators are given by

$$\hat{T}_n = \frac{1}{(n!)^2} \sum_{a_1,\ldots,a_n,i_1,\ldots,i_n} t_{i_1\ldots i_n}^{a_1\ldots a_n} a_{a_1}^\dagger \ldots a_{a_n}^\dagger a_{i_n} \ldots a_{i_1}, \tag{2.3.14}$$

where $a^\dagger$ and a is the creation operator and the annihilation operator from second quantization, respectively, and $t_{i_1\ldots i_n}^{a_1\ldots a_n}$ is the coefficient which is to be determined.

If we insert Ψ_{CC} into the Schrödinger equation we get

$$\hat{H}\Psi_{CC} = E_{CC}\Psi_{CC}, \tag{2.3.15}$$

from which, by using the reference function $\langle\Phi_0|$, and the intermediate normalization $\langle\Phi_0|\Psi_{CC}\rangle = 1$, we can obtain the energy,

$$E_{CC} = \langle\Phi_0|\,\hat{H}\,|\Psi_{CC}\rangle\,. \tag{2.3.16}$$

By using Eqs. (2.3.12) and (2.3.13) the energy becomes

$$E_{CC} = \langle\Phi_0|\,\hat{H}(1 + \hat{T}_1 + \cdots + \hat{T}_N)\,|\Phi_0\rangle\,. \tag{2.3.17}$$

It is convenient to find the contribution from the CC method, by subtracting E_0 (the Hartree-Fock energy) from both sides of the Schrödinger equation, giving

$$\hat{H_N}\Psi_{CC} = \Delta E_{CC}\Psi_{CC}, \tag{2.3.18}$$

where $\hat{H_N} = \hat{H} - E_0$. The energy is now given as

$$\Delta E_{CC} = \langle\Phi_0|\,\hat{H}_N e^{\hat{T}}\,|\Phi_0\rangle\,. \tag{2.3.19}$$

We have now looked at several different methods for solving the many-body quantum mechanical problem: the simple Hartree-Fock method, and post Hartree-Fock methods which have different approaches to including electron-electron correlation. In the next chapter we will look at another approach, which is used by the solver developed for this thesis, the variational Monte Carlo method.

Chapter 3

Quantum Monte Carlo

Solving the Schrödinger equation for large systems can be a daunting task, and solution methods such as configuration interaction theory would converge very slowly and would render it practically impossible to find a solution for large systems in a reasonable time. Simpler approximations exist, as we have seen with the Hartree-Fock method, but this uses a limited effect of many-body correlations, and would thus give an unsatisfactory result.

To go beyond these approximations we use Quantum Monte Carlo (QMC), which uses statistical Markov Chain (random walk) simulations. It is an intuitive method, as it is statistical. Therefore it is easy to extract quantities such as densities or probability distributions. Several types of the QMC method exist, but the focus in this thesis is on the variational Monte Carlo method.

3.1 Variational Monte Carlo method

The variational Monte Carlo (VMC) method uses Monte Carlo numerical integration to calculate the multidimensional integrals emerging in quantum mechanics [McMillan, 1965] [Ceperley et al., 1977]. In a quantum mechanical system the energy is given by the expectation value of the Hamiltonian. Assuming we have a Hamiltonian H and a proposal for a wave function that can describe the system, called the trial wave function Ψ_T , the expectation value, from the variational principle, is defined by

$$E[\hat{H}] = \langle \Psi_T | \hat{H} | \Psi_T \rangle = \frac{\int d\mathbf{R} \Psi_T^*(\mathbf{R}) \hat{H} \Psi_T(\mathbf{R})}{\int d\mathbf{R} \Psi_T^*(\mathbf{R}) \Psi_T(\mathbf{R})}. \tag{3.1.1}$$

We have here an upper bound to the ground state energy, E_0, of the Hamiltonian that is

$$E_0 \leq \langle H \rangle .$$

We can expand the trial wave function in the eigenstates of the Hamiltonian, as they form a complete set. This gives us

$$\Psi_T(\mathbf{R}) = \sum_i a_i \Psi_i(\mathbf{R}).$$

If we assume this set of eigenfunctions are normalized, we get

$$\frac{\sum_{nm} a_m^* a_n \int d\mathbf{R} \Psi_m^*(\mathbf{R}) H(\mathbf{R}) \Psi_n(\mathbf{R})}{\sum_{nm} a_m^* a_n \int d\mathbf{R} \Psi_m^*(\mathbf{R}) \Psi_n(\mathbf{R})} = \frac{\sum_n a_n^2 E_n}{\sum_n a_n^2} \geq E_0,$$

where we have used that $H(\mathbf{R})\Psi_n(\mathbf{R}) = E_n \Psi_n(\mathbf{R})$. A problem arises with the integrals in the calculation of the expectation values, namely that the integrals generally are multi-dimensional ones.

A wave function will in most cases have a small part in configuration space where its values are significant. A large part of configuration space will therefore contain small values. Initially the Monte Carlo method uses a simple procedure with random points homogeneously distributed in configuration space. This unfocused distribution will lead to poor results as a large portion of the points will hit a portion of the wave function where the values are small.

To make our variational Monte Carlo algorithm more efficient we need a better way to distribute random points. A method to make the random points hit the wave function more effectively is to use importance sampling combined with the Metropolis algorithm. Hopefully this will make the solver find the ground state energy more efficiently by sampling values in the region of configuration space where the wave function has more significant values.

First we need to choose a trial wave function $\psi_T(\mathbf{R})$. From this we create a probability distribution

$$P(\mathbf{R}) = \frac{|\psi_T(\mathbf{R})|^2}{\int |\psi_T(\mathbf{R})|^2 \, d\mathbf{R}}. \tag{3.1.2}$$

Next we introduce a local energy,

$$E_L(\mathbf{R},\alpha) = \frac{1}{\psi_T(\mathbf{R},\alpha)} H\psi_T(\mathbf{R},\alpha)). \tag{3.1.3}$$

Combining Eqs. (3.1.1) and (3.1.3) we can rewrite the energy as

$$E[H(\alpha)] = \int P(\mathbf{R}) E_L(\mathbf{R}) d\mathbf{R} \approx \frac{1}{N}\sum_{i=1}^{N} P(\mathbf{R_i},\alpha) E_L(\mathbf{R_i},\alpha),$$

where N is the number of samples in our Monte Carlo solver.

The algorithm for a simple quantum Monte Carlo solver is given below.

1. To initialize the system give all particles a random position, $\mathbf{R}$, choose variational parameters α, and calculate $|\psi_T^{\alpha}(\mathbf{R})|^2$. We fix also at the beginning the number of Monte Carlo cycles.

2. Start Monte Carlo Calculations

   (a) Propose a move of the particles according to an algorithm, for example $\mathbf{R_{new}} = \mathbf{R_{old}} + \delta * r$, where r is a random number in $[0,1]$

   (b) Accept or reject move according to $P(\mathbf{R_{new}})/P(\mathbf{R_{old}}) \geq r$, where r is a new number. Update position values if accepted.

   (c) Calculate energy and other averages for this cycle.

3. After iterating through all the Monte Carlo cycles, finish computations by computing the final averages.

The basic Monte Carlo algorithm uses the step variable δ to govern a new proposed move for the particles. This is called brute-force sampling. To get more relevant sampling we need importance sampling, which will be discussed in section 3.3.

The trial wave function, Ψ_T, consists of the Slater determinant of all particles in the system, as defined in section 2.1, combined with a correlation function, $f(\mathbf{r}_{ij})$, and we have thus

$$\begin{aligned}\Psi_T(\mathbf{R}) &= \Phi(\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_N, \alpha, \beta, \ldots, \sigma) f(\mathbf{r}_{12}) f(\mathbf{r}_{21}) \ldots f(\mathbf{r}_{kl}) \ldots f\left(\mathbf{r}_{N(N-1)}\right) \\ &= \Phi(\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_N, \alpha, \beta, \ldots, \sigma) \Psi_C,\end{aligned}$$

where Ψ_C is referred to as simply the Jastrow factor. There exists several forms of the correlation function, but a form suited for large systems, and therefore a suitable form for us, is the Padé-Jastrow form, defined as

$$f\left(\mathbf{r}_{ij}\right) = \exp\left(U\right). \tag{3.1.4}$$

Here U is a potential series expansion on both $\mathbf{r}_i$ and $\mathbf{r}_{ij}$. We use here a Padé-Jastrow function function typically used for quantum mechanical Monte Carlo calculations

$$\exp\left(\frac{a\mathbf{r}_{ij}}{(1+\beta\mathbf{r}_{ij})}\right), \tag{3.1.5}$$

where β is a variational parameter and a is a constant depending on the spins of the interacting particles. For atoms and quantum dots in three dimensions the spin constant a is $1/4$ if the interacting spins are parallel, and $1/2$ if they are anti-parallel. For quantum dots in two dimensions a is $1/3$ and 1 for parallel and anti-parallel spins respectively.

3.1.1 The trial wave function

It is now clear that the choice of the trial wave function, Ψ_T, is crucial, and all observables from the Monte Carlo calculations are evaluated with respect to the probability distribution function in Eq. (3.1.2), which is calculated with the trial wave function.

The trial wave function can be chosen to be any normalizable wave function which in some degree overlaps the exact ground state wave function, Ψ_0, and for which the value, the gradient and the laplacian of the wave function can be easily calculated. One of the advantages of using Quantum Monte Carlo methods is their flexibility of the form of the trial wave function.

To get accurate results the trial wave function should behave in a way as similar to the exact trial wave function as possible. It is especially important that the trial wave function and its derivative is well defined at the origin, that is $\Psi_0(|\mathbf{R}| = 0) \neq 0$.

If we for example perform calculations of the energy with two particles that are infinitely close, the Coulomb singularity would make the energy blow up. But by using the exact wave function the diverging terms would cancel each other out. It is therefore necessary to put constraints on the behaviour of the trial wave function when the distance between one electron and the nucleus or two electrons approaches zero. Constraints like these are the so-called "cusp conditions" [Hjorth-Jensen, 2010].

3.2 The Metropolis algorithm

The Metropolis-Hastings algorithm, often simply called the Metropolis algorithm, is named after Nicholas Metropolis, who invented it along with A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller in 1953 [Metropolis et al., 1953], and W. K. Hastings [Hastings, 1970], who extended the method the a more general case. It is a method of sampling a normalized probability distribution by a stochastic process.

The algorithm goes as follows:

1. At step n, choose a random state, with the probability $\mathcal{P}_i^{(n)}$.
2. Sample a possible new state j with some probability $T_{i\to j}$.
3. With a probability $A_{i\to j}$ the move to j is accepted, and j is used as the new state. However, with a probability $1 - A_{i\to j}$ the move is rejected, and the original state i is used as a sample again.

We want to make $\mathcal{P}_i^{(n\to\infty)} \to p_i$, meaning that starting from some distribution, the Metropolis method will converge to the correct distribution. Consequently we have to derive the properties we need from T and A. To take the probabilities p_i over to their corresponding continuum expressions, they are replaced with expressions such as $p(x_i)dx_i$.

The equation of the probability $\mathcal{P}_i^{(n)}$, written in a dynamical way by including the transition probability, is thus written as

$$\mathcal{P}_i^{(n)} = \sum_j \left[\mathcal{P}_j^{(n-1)} T_{j\to i} A_{j\to i} + \mathcal{P}_i^{(n-1)} T_{i\to j} \left(1 - A_{i\to j}\right)\right]. \tag{3.2.1}$$

Here, the probability of being in the state i at a step n is given by the probability of being in the state j at the previous step, and accepting the transition to state i, plus the probability of being in the state i and rejecting a transition to a state j. Furthermore, the probability of making any transition must be 1, in other words $\sum_j T_{i\to j} = 1$. This means that we can rewrite Eq. (3.2.1) to

$$\mathcal{P}_i^{(n)} = \mathcal{P}_i^{(n-1)} + \sum_j \left[\mathcal{P}_j^{(n-1)} T_{j\to i} A_{j\to i} - \mathcal{P}_i^{(n-1)} T_{i\to j} A_{i\to j}\right]. \tag{3.2.2}$$

A requirement to Eq. (3.2.2) is that for large n we have $\mathcal{P}_i^{(n\to\infty)} = p_i$, that is, that it gives the desired probability distribution. Taking the limit $n \to \infty$ on Eq. (3.2.2) gives the balance requirement

$$\sum_j \left[p_j T_{j\to i} A_{j\to i} - p_i T_{i\to j} A_{i\to j}\right] = 0.$$

This is a very weak balance requirement, however. Typically, each term is independently set to zero and used to determine the acceptance probabilities, rather than simply the sum being set to zero. We can rearrange the equation, grouping the probabilities to move from one state to another on the left hand side, and the transition probabilities on the right hand side, resulting in

$$\frac{A_{j\to i}}{A_{i\to j}} = \frac{p_i T_{i\to j}}{p_j T_{j\to i}}.$$

The choice made in the Metropolis algorithm is maximizing the probabilities to move from one state to another, A, yielding

$$A_{j\to i} = \min\left(1, \frac{p_i T_{i\to j}}{p_j T_{j\to i}}\right).$$

This means that we multiply $A_{j\to i}$ and $A_{i\to j}$ by the same constant, which is smaller than unity. By choosing acceptance probabilities in this manner we have assured that if the probability $\mathcal{P}_i^{(n)}$ is equal to p_i, and is thus in equilibrium, it will remain equilibrated.

Subsequently we need to find the conditions for when the probability $\mathcal{P}_i^{(n)}$ converges to equilibrium. We rewrite Eq. (3.2.2) as

$$\mathcal{P}_i^{(n)} = \sum_j M_{ij} \mathcal{P}_j^{(n-1)},$$

where we have introduced the matrix M, defined as

$$M_{ij} = \delta_{ij}\left[1 - \sum_k T_{i\to k} A_{i\to k}\right] + T_{j\to i} A_{j\to i}.$$

If we sum over i we find that $\sum_i M_{ij} = 1$. We also know that $\sum_k T_{i\to k}$ and $A_{i\to k} \leq 1$, which means that all elements of the matrix will satisfy $M_{ij} \geq 0$, thus showing that the matrix M is a stochastic matrix.

The Metropolis algorithm, simply put, gives us the method for computing the right eigenvector of matrix M with the largest magnitude eigenvalue. It is created in such a way that the correct probability distribution is a right eigenvector with eigenvalue 1. This means that in order for the Metropolis algorithm to converge to the desired result, the matrix M must have one and only one eigenvalue with this magnitude, with all other eigenvalues smaller.

3.3 Importance sampling

In many cases using the Monte Carlo method, we encounter the scenario that we want to compute the expected value of a function $f(\mathbf{X})$, but $f(x)$ yields very small values outside a region A, for which the probability to have $\mathbf{X} \in A$ is small. In such a scenario the set A may have a small volume, or may be poorly covered by the distribution $\mathbf{X}$. Therefore a sample from a basic Monte Carlo method from the distribution $\mathbf{X}$ could fail to have even one point inside the region A.

Clearly we need a better way to get Monte Carlo samples, that is by sampling the important region. To achieve this we sample from a weighted distribution, which prioritizes the important region. This is called importance sampling.

To implement importance sampling we replace the brute force Metropolis algorithm with a walk in coordinate space biased by the trial wave function, an approach based on the Fokker-Planck equation [Fokker, 1914] [Planck, 1917] and the Langevin equation [Langevin, 1908] for generating a trajectory in coordinate space. The Fokker-Planck equation, named after Adriaan Fokker and Max Planck, is often used as a model for more general Markov processes.

3.3.1 Importance sampling

By using the Fokker-Planck equation and the Langevin equation we can derive and implement importance sampling in our Metropolis algorithm.

For one particle or walker, a diffusion process characterized by a time-dependent

probability density $P(x,t)$ in one dimension, we have the Fokker-Planck equation

$$\frac{\partial P}{\partial t} = D\frac{\partial}{\partial x}\left(\frac{\partial}{\partial x} - F\right)P(x,t), \tag{3.3.1}$$

where F is a drift term and D is the diffusion coefficient.

The new positions in coordinate space are found using the Langevin equation with Euler's method. We go from the Langevin equation

$$\frac{\partial x(t)}{\partial t} = DF(x(t)) + \eta, \tag{3.3.2}$$

where η is a random variable. This gives us a new position

$$y = x + DF(x)\Delta t + \xi\sqrt{\Delta t}. \tag{3.3.3}$$

Here ξ is a Gaussian random variable and Δt is a chosen time step. D comes from the factor $1/2$ in the kinetic energy operator, and is therefore equal to $1/2$ in atomic units.

The process of isotropic diffusion characterized by a time-dependent probability density $P(\mathbf{x},t)$ will, as an approximation, obey the Fokker-Planck equation

$$\frac{\partial P}{\partial t} = \sum_i D\frac{\partial}{\partial \mathbf{x_i}}\left(\frac{\partial}{\partial \mathbf{x_i}} - \mathbf{F_i}\right)P(\mathbf{x},t), \tag{3.3.4}$$

where $\mathbf{F}_i$ is component number i of the drift term caused by an external potential, and D is the diffusion coefficient. We set the left hand side equal to zero and obtain the convergence to a stationary probability density

$$\frac{\partial^2 P}{\partial \mathbf{x_i}^2} = P\frac{\partial}{\partial \mathbf{x_i}}\mathbf{F_i} + \mathbf{F_i}\frac{\partial}{\partial \mathbf{x_i}}P. \tag{3.3.5}$$

Inserting the drift vector, $\mathbf{F} = g(\mathbf{x})\frac{\partial P}{\partial \mathbf{x}}$, we get

$$\frac{\partial^2 P}{\partial \mathbf{x_i}^2} = P\frac{\partial g}{\partial P}\left(\frac{\partial P}{\partial \mathbf{x}_i}\right)^2 + Pg\frac{\partial^2 P}{\partial \mathbf{x}_i^2} + g\left(\frac{\partial P}{\partial \mathbf{x}_i}\right)^2. \tag{3.3.6}$$

To meet the condition of stationary density the left hand side has to be zero. This means that the terms containing first and second order derivatives has to cancel each other, which is only possible if $g = 1/P$. This yields

$$\mathbf{F} = 2\frac{1}{\Psi_T}\nabla\Psi_T, \tag{3.3.7}$$

known as the quantum force. This so-called force pushes the walker towards regions of configuration space where the trial wave function is large, thus increasing the efficiency of the simulation. This is a great improvement on the Metropolis algorithm where the walker otherwise would have the same probability to move in every direction.

From the Fokker-Planck equation we get a transition probability given by Green's function

$$G(y,x,\Delta t) = \frac{1}{(4\pi D\Delta t)^{3N/2}} \exp\left(-\frac{(y-x-D\Delta t F(x))^2}{4D\Delta t}\right). \tag{3.3.8}$$

This means that we now have the Metropolis algorithm

$$A(y,x) = \min(1, q(y,x))), \tag{3.3.9}$$

where

$$q(y,x) = \frac{|\Psi_T(y)|^2}{|\Psi_T(x)|^2}, \tag{3.3.10}$$

is replaced by the Metropolis-Hastings algorithm,

$$q(y,x) = \frac{G(x,y,\Delta t)|\Psi_T(y)|^2}{G(y,x,\Delta t)|\Psi_T(x)|^2}. \tag{3.3.11}$$

3.4 Calculating the Slater determinant

To describe the wave function of multiple fermions in the VMC method we use a Slater determinant, as described in Eq. (2.1.1). Writing out the Slater determinant for a four-fermionic system while including spin, we have

$$\Phi(\mathbf{r}_1,\mathbf{r}_2,\mathbf{r}_3,\mathbf{r}_4,\alpha,\beta,\gamma,\delta) = \frac{1}{\sqrt{4!}} \begin{vmatrix} \psi_{100\uparrow}(\mathbf{r}_1) & \psi_{100\uparrow}(\mathbf{r}_2) & \psi_{100\uparrow}(\mathbf{r}_3) & \psi_{100\uparrow}(\mathbf{r}_4) \\ \psi_{100\downarrow}(\mathbf{r}_1) & \psi_{100\downarrow}(\mathbf{r}_2) & \psi_{100\downarrow}(\mathbf{r}_3) & \psi_{100\downarrow}(\mathbf{r}_4) \\ \psi_{200\uparrow}(\mathbf{r}_1) & \psi_{200\uparrow}(\mathbf{r}_2) & \psi_{200\uparrow}(\mathbf{r}_3) & \psi_{200\uparrow}(\mathbf{r}_4) \\ \psi_{200\downarrow}(\mathbf{r}_1) & \psi_{200\downarrow}(\mathbf{r}_2) & \psi_{200\downarrow}(\mathbf{r}_3) & \psi_{200\downarrow}(\mathbf{r}_4) \end{vmatrix}. \tag{3.4.1}$$

Note, however, that the electron-electron interaction is spin-independent. This means that we don't need to consider degrees of freedom arising from different spins when calculating the interaction, effectively halving the number of dimensions in our calculations compared to if the interaction had been dependent on spin.

We can rewrite the Slater determinant as a product of two Slater determinants, one for spin up and one for spin down. This gives us

$$\begin{aligned}\Phi(\mathbf{r}_1,\mathbf{r}_2,,\mathbf{r}_3,\mathbf{r}_4,\alpha,\beta,\gamma,\delta) &= \det\uparrow(1,2)\det\downarrow(3,4)-\det\uparrow(1,3)\det\downarrow(2,4)\\ &\quad-\det\uparrow(1,4)\det\downarrow(3,2)+\det\uparrow(2,3)\det\downarrow(1,4)\\ &\quad-\det\uparrow(2,4)\det\downarrow(1,3)+\det\uparrow(3,4)\det\downarrow(1,2).\end{aligned}$$

Here we have defined the Slater determinant for spin up as

$$\det\uparrow(1,2)=\frac{1}{\sqrt{2}}\begin{vmatrix}\psi_{100\uparrow}(\mathbf{r}_1) & \psi_{100\uparrow}(\mathbf{r}_2)\\ \psi_{200\uparrow}(\mathbf{r}_1) & \psi_{200\uparrow}(\mathbf{r}_2)\end{vmatrix}, \tag{3.4.2}$$

and the Slater determinant for spin down as

$$\det\downarrow(3,4)=\frac{1}{\sqrt{2}}\begin{vmatrix}\psi_{100\downarrow}(\mathbf{r}_3) & \psi_{100\downarrow}(\mathbf{r}_4)\\ \psi_{200\downarrow}(\mathbf{r}_3) & \psi_{200\downarrow}(\mathbf{r}_4)\end{vmatrix}. \tag{3.4.3}$$

Further, it can be shown that for the variational energy we can approximate the Slater determinant as [Moskowitz and Kalos, 1981]

$$\Phi(\mathbf{r}_1,\mathbf{r}_2,\dots\mathbf{r}_N)\propto\det\uparrow\det\downarrow. \tag{3.4.4}$$

We now have the Slater determinant as a product of two determinants, one containing the electrons with only spin up, and one containing the electrons of spin down. This approach has certain limits as the ansatz isn't antisymmetric under the exchange of electrons with opposite spins, but it gives the same expectation value for the energy as the full Slater determinant as long as the Hamiltonian is spin independent. We thus avoid summing over spin variables.

Now we have the Slater determinant written as a product of a determinant for spin up and a determinant for spin down. The next step is to invert the matrices using LU decomposition. We can thus rewrite a matrix $\hat{A}$ as a product of two matrices, $\hat{B}$ and $\hat{C}$

$$\begin{pmatrix}a_{11}&a_{12}&a_{13}&a_{14}\\ a_{21}&a_{22}&a_{23}&a_{24}\\ a_{31}&a_{32}&a_{33}&a_{34}\\ a_{41}&a_{42}&a_{43}&a_{44}\end{pmatrix}=\begin{pmatrix}1&0&0&0\\ b_{21}&1&0&0\\ b_{31}&b_{32}&1&0\\ b_{41}&b_{42}&b_{43}&1\end{pmatrix}\begin{pmatrix}c_{11}&c_{12}&c_{13}&c_{14}\\ 0&c_{22}&c_{23}&c_{24}\\ 0&0&c_{33}&c_{34}\\ 0&0&0&c_{44}\end{pmatrix}.$$

LU factorization exists for $\hat{A}$ if the determinant is nonzero. If $\hat{A}$ also is non-singular, then the LU factorization is unique and the determinant is given by

$$|\hat{A}|=c_{11}c_{22}\dots c_{nn}. \tag{3.4.5}$$

Using this we can calculate the spin up determinant, the spin down determinant, and by putting them together, the Slater determinant.

3.5 Optimization

Because the computational cost of the Variational Monte Carlo method scales up quickly with increasing number of particles it is important to make the computations as optimized as possible, to reduce the time needed for the computations and therefore be able to produce more accurate results.

There are three ways we improve the efficiency of the VMC method. The first two are importance sampling and multithreaded computations, as has already been discussed. The third way is improving efficiency when computing ratios and local energy. This will reduce the computing cost of the Slater determinants and remove the need for slow numerical derivations.

In this section ways to improve efficiency when calculating the ratios and local energy will be presented.

3.5.1 Metropolis-Hastings-Ratio

In the Metropolis-Hasting algorithm a ratio of the new probability and the old probability is calculated to evaluate if a suggested move is accepted or rejected, given by

$$\frac{P(\mathbf{r_{new}})}{P(\mathbf{r_{old}})} \geq r. \tag{3.5.1}$$

For our probability function it can be written as

$$\frac{|D_{\uparrow}^{new}|}{|D_{\uparrow}^{old}|}\frac{|D_{\downarrow}^{new}|}{|D_{\downarrow}^{old}|}\frac{\Psi_C^{new}}{\Psi_C^{old}} \geq r. \tag{3.5.2}$$

3.5.2 Slater-Determinant-Ratio

The Slater-Determinants ratios are slow to calculate by brute force, for example by LU-decomposition (see section 3.4), but recalculating it after each move can be done in a computationally easier way.

To tackle the determinant ratios we need to introduce some notation. Let an element in the determinant matrix, $|D|$, be described by

$$D_{ij} = \phi_j(\mathbf{r}_i), \tag{3.5.3}$$

where ϕ_j is the j'th single particle wave function and $\mathbf{r}_i$ is the position of the i'th particle.

The inverse of the Slater matrix is given by its adjugate by

$$\mathbf{D}^{-1} = \frac{1}{|\mathbf{D}|}\text{adj}\mathbf{D}.$$

The adjugate of a matrix is the cofactor matrix C, $\text{adj}\mathbf{D} = \mathbf{C}^T$, and

$$\begin{aligned} \mathbf{D}^{-1} &= \frac{\mathbf{C}^T}{|\mathbf{D}|} \\ \mathbf{D}_{ji}^{-1} &= \frac{\mathbf{C}_{ij}}{|\mathbf{D}|}. \end{aligned}$$

We can express the determinant as a cofactor expansion around row i, using Cramer's rule

$$|\mathbf{D}| = \sum_j \mathbf{D}_{ij}\mathbf{C}_{ij}. \tag{3.5.4}$$

This gives the ratio of the new and old Slater determinants the following

$$R_{SD} = \frac{|\mathbf{D}^{new}|}{|\mathbf{D}^{old}|} = \frac{\sum_{j=0}^{N} D_{ij}^{new} C_{ij}^{new}}{\sum_{j=0}^{N} D_{ij}^{old} C_{ij}^{old}}. \tag{3.5.5}$$

Since we are only moving one particle at a time and the cofactor term relies on the other rows it doesn't change, $C_{ij}^{new} = C_{ij}^{old}$ in one movement. Combining this with Eq. (3.5.4) we get

$$R_{SD} = \frac{\sum_{j=0}^{N} D_{ij}^{new} (D_{ji}^{old})^{-1} |D^{old}|}{\sum_{j=0}^{N} D_{ij}^{old} (D_{ji}^{old})^{-1} |D^{old}|}. \tag{3.5.6}$$

Since $\mathbf{D}$ is invertible, $\mathbf{D}\mathbf{D}^{-1} = \mathbf{1}$, the ratio becomes

$$\begin{aligned} R_{SD} &= \sum_{j=0}^{N} D_{ij}^{new} (D_{ji}^{old})^{-1} \\ &= \sum_{j=0}^{N} \phi_j(\mathbf{x}_i^{new}) D_{ji}^{-1}(\mathbf{x}^{old}). \end{aligned}$$

3.5.3 Correlation-to-correlation ratio

We have $N(N-1)/2$ relative distances r_{ij}. We can write these in a matrix storage format, where they form a strictly upper triangular matrix

$$\mathbf{r} \equiv \begin{pmatrix} 0 & r_{1,2} & r_{1,3} & \dots & r_{1,N} \\ \vdots & 0 & r_{2,3} & \dots & r_{2,N} \\ \vdots & \vdots & 0 & \ddots & \vdots \\ \vdots & \vdots & \vdots & \ddots & r_{N-1,N} \\ 0 & 0 & 0 & \dots & 0 \end{pmatrix}.$$

This upper triangular matrix form also applies to $g = g(r_{ij})$.

The correlation-to-correlation ratio, or ratio between Jastrow factors, is given by

$$R_C = \frac{\Psi_C^{new}}{\Psi_C^{cur}} = \prod_{i=1}^{k-1} \frac{g_{ik}^{new}}{g_{ik}^{cur}} \prod_{i=k+1}^{N} \frac{g_{ki}^{new}}{g_{ki}^{cur}}, \tag{3.5.7}$$

or in the Padé-Jastrow form

$$R_C = \frac{\Psi_C^{\text{new}}}{\Psi_C^{\text{cur}}} = \frac{\exp(U_{new})}{\exp(U_{cur})} = \exp(\Delta U), \tag{3.5.8}$$

where

$$\Delta U = \sum_{i=1}^{k-1} \left(f_{ik}^{\text{new}} - f_{ik}^{\text{cur}}\right) + \sum_{i=k+1}^{N} \left(f_{ki}^{\text{new}} - f_{ki}^{\text{cur}}\right). \tag{3.5.9}$$

This efficient calculation of the correlation-to-correlation ratio is part of the calculation of the energy of the system. In the VMC solver program developed it is situated in the `Derivatives` class, see section 5.1 and Fig. 5.2.

3.5.4 Efficient calculation of derivatives

Calculating the derivatives involved in the VMC calculation numerically is slow in that they entail several calls to the wave functions in addition to introducing an extra numerical error. Here we will show how to divide up the derivatives and how to find analytical expressions for all the parts using the derivatives.

The trial-function can be factorized as

$$\Psi_T(\mathbf{x}) = \Psi_D \Psi_C = |D_\uparrow||D_\downarrow|\Psi_C, \tag{3.5.10}$$

where $D_\uparrow$, $D_\downarrow$ and Ψ_C is the spin up and down part of the Slater determinant and the Jastrow factor respectively.

Gradient-Ratio

For the quantum force, and in the final expression for the local energy, we need to calculate the gradient ratio of the trial-function which is given by

$$\frac{\nabla\Psi_T}{\Psi_T} = \frac{\nabla(\Psi_D\Psi_C)}{\Psi_D\Psi_C} = \frac{\nabla\Psi_D}{\Psi_D} + \frac{\nabla\Psi_C}{\Psi_C} \tag{3.5.11}$$

$$= \frac{\nabla|D_\uparrow|}{|D_\uparrow|} + \frac{\nabla|D_\downarrow|}{|D_\downarrow|} + \frac{\nabla\Psi_C}{\Psi_C}. \tag{3.5.12}$$

Laplacian-Ratio

From the Hamiltonians and the expression for the local energy the local kinetic energy of electron i is given by the following

$$K_i = -\frac{1}{2}\frac{\nabla_i^2\Psi}{\Psi}. \tag{3.5.13}$$

Using the factorization of the trial-function from Eq. (3.5.10) we can calculate the ratio needed for the kinetic energy,

$$\frac{1}{\Psi_T}\frac{\partial^2\Psi_T}{\partial x_k^2} = \frac{1}{\Psi_D\Psi_C}\frac{\partial^2(\Psi_D\Psi_C)}{\partial x_k^2} = \frac{1}{\Psi_D\Psi_C}\frac{\partial}{\partial x_k}\left(\frac{\partial\Psi_D}{\partial x_k}\Psi_C + \Psi_D\frac{\partial\Psi_C}{\partial x_k}\right) \tag{3.5.14}$$

$$= \frac{1}{\Psi_D\Psi_C}\left(\frac{\partial^2\Psi_D}{\partial x_k^2}\Psi_C + 2\frac{\partial\Psi_D}{\partial x_k}\frac{\partial\Psi_C}{\partial x_k} + \Psi_D\frac{\partial^2\Psi_C}{\partial x_k^2}\right) \tag{3.5.15}$$

$$= \frac{1}{\Psi_D}\frac{\partial^2\Psi_D}{\partial x_k^2} + 2\frac{1}{\Psi_D}\frac{\partial\Psi_D}{\partial x_k}\cdot\frac{1}{\Psi_C}\frac{\partial\Psi_C}{\partial x_k} + \frac{1}{\Psi_C}\frac{\partial^2\Psi_C}{\partial x_k^2}. \tag{3.5.16}$$

Since the Slater determinant part of the trial-function is separable into a spin up and spin down part, we can simplify it further

$$\frac{1}{\Psi_D}\frac{\partial^2\Psi_D}{\partial x_k^2} = \frac{1}{|D_\uparrow||D_\downarrow|}\frac{\partial^2|D_\uparrow||D_\downarrow|}{\partial x_k^2} = \frac{1}{|D_\uparrow|}\frac{\partial^2|D_\uparrow|}{\partial x_k^2} + \frac{1}{|D_\downarrow|}\frac{\partial^2|D_\downarrow|}{\partial x_k^2} \tag{3.5.17}$$

$$\frac{1}{\Psi_D}\frac{\partial\Psi_D}{\partial x_k} = \frac{1}{|D_\uparrow||D_\downarrow|}\frac{\partial|D_\uparrow||D_\downarrow|}{\partial x_k} = \frac{1}{|D_\uparrow|}\frac{\partial|D_\uparrow|}{\partial x_k} + \frac{1}{|D_\downarrow|}\frac{\partial|D_\downarrow|}{\partial x_k}. \tag{3.5.18}$$

Inserting Eqs. (3.5.18) and (3.5.17) into Eq. (3.5.16) we get

$$\frac{\nabla^2\Psi_T}{\Psi_T} = \frac{\nabla^2|D_\uparrow|}{|D_\uparrow|} + \frac{\nabla^2|D_\downarrow|}{|D_\downarrow|} + 2\left(\frac{\nabla|D_\uparrow|}{|D_\uparrow|} + \frac{\nabla|D_\downarrow|}{|D_\downarrow|}\right)\cdot\frac{\nabla\Psi_C}{\Psi_C} + \frac{\nabla^2\Psi_C}{\Psi_C}. \tag{3.5.19}$$

So to calculate the Laplacian-ratio and the gradient-ratio we need to find expressions for the gradient Slater ratio, $\nabla|D|/|D|$, the Laplacian Slater ratio, $\nabla^2|D|/|D|$, the gradient Jastrow ratio, $\nabla\Psi_C/\Psi_C$, and the Laplacian Jastrow ratio $\nabla^2\Psi_C/\Psi_C$.

The gradient Slater ratio

By the same argument as in section 3.5.2 the gradient-Slater-ratio can be written as

$$\frac{\nabla|D|}{|D|} = \sum_{j=0}^{N}\nabla(D_{ij})D_{ji}^{-1} = \sum_{j=0}^{N}\nabla\phi_j(\mathbf{x}_i)D_{ji}^{-1}(\mathbf{x}). \tag{3.5.20}$$

The Laplacian Slater ratio

As in the previous section the Laplacian can be calculated by a similar method to the one used in section 3.5.2, and we end up with

$$\frac{\nabla^2|D|}{|D|} = \sum_{j=0}^{N} \nabla^2(D_{ij})D_{ji}^{-1} = \sum_{j=0}^{N} \nabla^2\phi_j(\mathbf{x}_i)D_{ji}^{-1}(\mathbf{x}). \tag{3.5.21}$$

The gradient Jastrow ratio

We continue by finding a useful expression for the quantum force and kinetic energy, the ratio $\nabla\Psi_C/\Psi_C$. It has, for all dimensions, the form

$$\frac{\nabla_i\Psi_C}{\Psi_C} = \frac{1}{\Psi_C}\frac{\partial\Psi_C}{\partial x_i}, \tag{3.5.22}$$

where i runs over all particles. Since the terms of the trial-function that are not differentiated cancel with their corresponding terms in the denominator, so only $N-1$ terms survive the first derivative. We thus get

$$\frac{1}{\Psi_C}\frac{\partial\Psi_C}{\partial x_k} = \sum_{i=1}^{k-1}\frac{1}{g_{ik}}\frac{\partial g_{ik}}{\partial x_k} + \sum_{i=k+1}^{N}\frac{1}{g_{ki}}\frac{\partial g_{ki}}{\partial x_k}. \tag{3.5.23}$$

For the exponential form we get almost the same, by just replacing g_{ij} with $\exp(f_{ij})$, and we get

$$\frac{1}{\Psi_C}\frac{\partial\Psi_C}{\partial x_k} = \sum_{i=1}^{k-1}\frac{\partial g_{ik}}{\partial x_k} + \sum_{i=k+1}^{N}\frac{\partial g_{ki}}{\partial x_k}. \tag{3.5.24}$$

We now use the identity

$$\frac{\partial}{\partial x_i}g_{ij} = -\frac{\partial}{\partial x_j}g_{ij}, \tag{3.5.25}$$

and get expressions where the derivatives that act on the particle are represented by the second index of g

$$\frac{1}{\Psi_C}\frac{\partial \Psi_C}{\partial x_k} = \sum_{i=1}^{k-1}\frac{1}{g_{ik}}\frac{\partial g_{ik}}{\partial x_k} - \sum_{i=k+1}^{N}\frac{1}{g_{ki}}\frac{\partial g_{ki}}{\partial x_i}, \tag{3.5.26}$$

and for the exponential case

$$\frac{1}{\Psi_C}\frac{\partial \Psi_C}{\partial x_k} = \sum_{i=1}^{k-1}\frac{\partial g_{ik}}{\partial x_k} - \sum_{i=k+1}^{N}\frac{\partial g_{ki}}{\partial x_i}. \tag{3.5.27}$$

Since we have that the correlation function is depending on the relative distance we use the chain rule

$$\frac{\partial g_{ij}}{\partial x_j} = \frac{\partial g_{ij}}{\partial r_{ij}}\frac{\partial r_{ij}}{\partial x_j} = \frac{x_j - x_i}{r_{ij}}\frac{\partial g_{ij}}{\partial r_{ij}}. \tag{3.5.28}$$

After substitution we get

$$\frac{1}{\Psi_C}\frac{\partial \Psi_C}{\partial x_k} = \sum_{i=1}^{k-1}\frac{1}{g_{ik}}\frac{\mathbf{r_{ik}}}{r_{ik}}\frac{\partial g_{ik}}{\partial r_{ik}} - \sum_{i=k+1}^{N}\frac{1}{g_{ki}}\frac{\mathbf{r_{ki}}}{r_{ki}}\frac{\partial g_{ki}}{\partial r_{ki}}. \tag{3.5.29}$$

For the Padé-Jastrow form we set $g_{ij} \equiv g(r_{ij}) = e^{f(r_{ij})} = e^{f_{ij}}$ and

$$\frac{\partial g_{ij}}{\partial r_{ij}} = g_{ij}\frac{\partial f_{ij}}{\partial r_{ij}}, \tag{3.5.30}$$

and arrive at

$$\frac{1}{\Psi_C}\frac{\partial \Psi_C}{\partial x_k} = \sum_{i=1}^{k-1}\frac{\mathbf{r_{ik}}}{r_{ik}}\frac{\partial f_{ik}}{\partial r_{ik}} - \sum_{i=k+1}^{N}\frac{\mathbf{r_{ki}}}{r_{ki}}\frac{\partial f_{ki}}{\partial r_{ki}}, \tag{3.5.31}$$

where we have the relative vectorial distance

$$\mathbf{r}_{ij} = |\mathbf{r}_j - \mathbf{r}_i| = (x_j - x_i)\mathbf{e}_1 + (y_j - y_i)\mathbf{e}_2 + (z_j - z_i)\mathbf{e}_3. \tag{3.5.32}$$

With a linear Padé-Jastrow we set

$$f_{ij} = \frac{ar_{ij}}{(1+\beta r_{ij})}, \tag{3.5.33}$$

with the corresponding closed form expression

$$\frac{\partial f_{ij}}{\partial r_{ij}} = \frac{a}{(1+\beta r_{ij})^2}. \tag{3.5.34}$$

The Laplacian Jastrow ratio

For the kinetic energy we also need the second derivative of the Jastrow factor divided by the Jastrow factor. We start with this

$$\left[\frac{\nabla^2\Psi_C}{\Psi_C}\right]_x = 2\sum_{k=1}^{N}\sum_{i=1}^{k-1}\frac{\partial^2 g_{ik}}{\partial x_k^2} + \sum_{k=1}^{N}\left(\sum_{i=1}^{k-1}\frac{\partial g_{ik}}{\partial x_k} - \sum_{i=k+1}^{N}\frac{\partial g_{ki}}{\partial x_i}\right)^2. \tag{3.5.35}$$

But we have another, simpler form for the function

$$\Psi_C = \prod_{i<j}\exp f(r_{ij}) = \exp\left\{\sum_{i<j}\frac{ar_{ij}}{1+\beta r_{ij}}\right\}, \tag{3.5.36}$$

and for particle k we have

$$\frac{\nabla_k^2\Psi_C}{\Psi_C} = \sum_{ij\neq k}\frac{(\mathbf{r}_k - \mathbf{r}_i)(\mathbf{r}_k - \mathbf{r}_j)}{r_{ki}r_{kj}}f'(r_{ki})f'(r_{kj}) + \sum_{j\neq k}\left(f''(r_{kj}) + \frac{2}{r_{kj}}f'(r_{kj})\right). \tag{3.5.37}$$

We use

$$f(r_{ij}) = \frac{ar_{ij}}{1+\beta r_{ij}}, \tag{3.5.38}$$

and with

$$g'(r_{kj}) = dg(r_{kj})/dr_{kj} \qquad \text{and} \quad g''(r_{kj}) = d^2g(r_{kj})/dr_{kj}^2 \ , \tag{3.5.39}$$

we find that for particle k we have

$$\frac{\nabla_k^2\Psi_C}{\Psi_C} = \sum_{ij\neq k} \frac{(\mathbf{r}_k-\mathbf{r}_i)(\mathbf{r}_k-\mathbf{r}_j)}{r_{ki}r_{kj}} \frac{a}{(1+\beta r_{ki})^2}\frac{a}{(1+\beta r_{kj})^2} + \sum_{j\neq k}\left(\frac{2a}{r_{kj}(1+\beta r_{kj})^2} - \frac{2a\beta}{(1+\beta r_{kj})^3}\right).$$

And for the linear Padé-Jastrow we get the closed form result

$$\frac{\partial^2 f_{ij}}{\partial r_{ij}^2} = -\frac{2a_{ij}\beta_{ij}}{(1+\beta_{ij}r_{ij})^3}. \tag{3.5.40}$$

3.6 Blocking

The data produced by Monte Carlo methods are typically a finite time series of correlated data. Many results, like the energy in the variational Monte Carlo solver, are averages over a finite time, and thus vary from simulation to simulation. Estimating the error on a time average of correlated data is therefore of great importance to be able to have information about how accurate the results produced are. There are several estimators of this error, like those described in Refs. [Binder, 1984] and [Daniell et al., 1984]. A more effortless method, however, is the blocking method.

The exact origin of the blocking method is uncertain, but it is plausible that it was invented by K. Wilson [Wilson, 1980]. It is also briefly described by

Whitmer [Whitmer, 1984] and Gottlieb et al. [Gottlieb et al., 1986]. For a more thorough description of the method, see the article by Flyvbjerg and Petersen [Flyvbjerg and Petersen, 1989].

The blocking method is independent from the VMC computation, and can be used afterwards to get a more robust estimate of the variance. The basic idea lies on the correlations between all the measurements. If these are important enough, they will produce an increase in the error that needs to be taken into account. The reason behind this is related to the effective amount of measurements, if there are correlations there will be measurements that will contain less information, so these won't be as valuable as the rest and it will be as if there are *less* measurements than we actually have. Obviously this is a problem, the usual identification of the error with $\sqrt{\sigma/n}$ will be overly optimistic and a correction is needed.

We introduce

$$f_d = \frac{1}{n-d}\sum_{k=1}^{n-d}\left(x_k - \bar{x}_n\right)\left(x_{k+d} - \bar{x}_n\right), \tag{3.6.1}$$

where f_d is the correlation between measurements separated by a distance of d. This can be used to give an actual form to the correction factor,

$$\tau = 1 + 2\sum_{d=1}^{n-1}\frac{f_d}{var\left(x\right)}. \tag{3.6.2}$$

This is the autocorrelation time and it relates the error with the variance,

$$err^2 = \frac{\tau}{n} var\left(x\right). \tag{3.6.3}$$

The inverse of the first factor is the number of effective measurements (that are useful since they contain information),

$$n_{eff} = \frac{n}{\tau}. \tag{3.6.4}$$

The expression that relates the standard deviation with this correlation time is thus

$$\sigma = \sqrt{\left(\frac{1 + 2\tau/\Delta t}{n}\left(\bar{x^2} - \bar{x}^2\right)\right)}, \tag{3.6.5}$$

where Δt is the time between samples, commonly smaller than τ. The main problem is that to compute τ a lot of time is needed, and this is not feasible in most cases.

The solution is to use blocking, and the algorithm for it is fairly straightforward

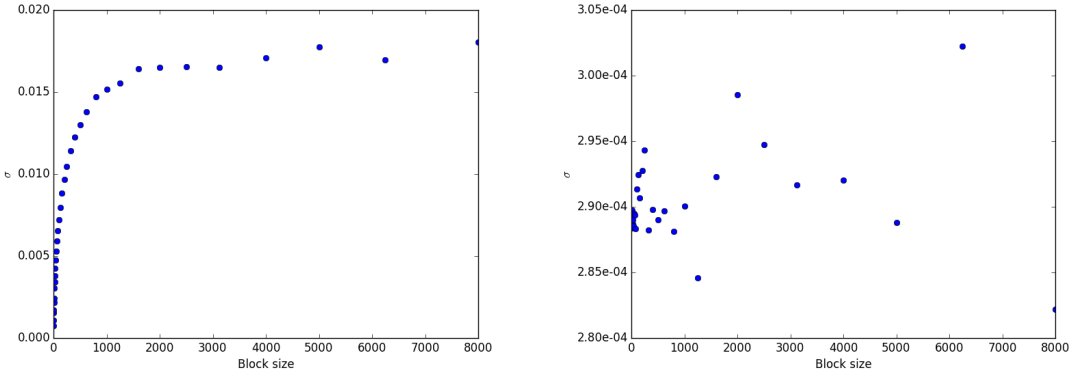
- Take a sequence of measured data and divide the sequence into blocks.

- Calculate the total mean and variance from the mean, $\bar{x}_i$, of block $i = 1 \ldots n_{blocks}$, making sure the block-size is large enough so that sample j of block i is not correlated with sample j of block $i + 1$.

- A good choice for the size would be the correlation time, τ. However, it is unknown. Therefore make a plot of the standard deviation σ, as a function of the block-size.

- When the standard deviation reaches a plateau the blocks are uncorrelated.

With the standard deviation reaching the plateau and the blocks thus are uncorrelated, we have an approximation to the true error, in form of the covariance. In some cases the data is fairly uncorrelated, however. In cases like that there is no plateau, only a seemingly random spread of the standard deviations with respect to block size. For an example of the two cases, see figure Fig. 3.1.



(a) **(b)**

Figure 3.1: **(a)**: Result from blocking of a VMC simulation of a 2-particle quantum dot in two dimensions, with frequency $\omega = 1.0$. There is a clear plateau, which means that the data is correlated, and the true error is approximately 1.6×10^{-2}. **(b)**: Result from blocking of data from a Monte Carlo integration. The data is uncorrelated, resulting in no plateau and a distribution of standard deviations spread out around 2.9×10^{-4}. The definite integral calculated is $\int_0^\pi \sin(x)dx = 2$, where the Monte-Carlo integrator gives the result 2.00003, in agreement with the magnitude of error.

3.7 Energy minimization

As it can be expected, the values of the energy can depend heavily with respect to the variational parameters, so the optimal value must be found. This usually means finding the minimum value, and the first option would obviously be just a brute force search, but this is not efficient and there are better alternatives. Among these, we can find the steepest descent method, the conjugate gradient method and the Newton-Raphson method (also known as simply Newton's method[1]) [Newton, 1671] [Raphson, 1702]. The first two offer the possibility of finding a minimum in a multivariate space, while the Newton-Raphson method only allows searching in one dimension, but is simpler and faster.

Since these methods don't find minima but the zeros of a function, the derivative

[1]It should be remarked that what we today commonly refer to as "Newton's method" in fact is Raphson's simplification of the method. However, Newton isn't mentioned in Raphson's presentation of the method; Raphson evidently considered his method to be fairly different than Newton's. That Raphson worked independently of Newton is however doubtful. But the "Newton-Raphson method" is therefore a more fitting denomination for the method than "Newton's method" [Cajori, 1911].

of said function is needed. But there is no analytical expression for the local energy, so a workaround must be used. It is possible to find an analytical expression of this derivative as a function of the values of the local energy and the logarithmic derivative of the wave function. Following [Hjorth-Jensen, 2013] the expression for the derivative with respect to parameter c is

$$\frac{\partial E}{\partial c} = 2\left[\langle E_L \frac{\partial \ln \Psi}{\partial c}\rangle - E\langle \frac{\partial \ln \Psi}{\partial c}\rangle\right], \tag{3.7.1}$$

or more explicitly

$$\frac{\partial E}{\partial c} = \frac{2}{N}\left[\sum_{i=1}^{N}\left(\left[E_L\left(c\right)\right]_i\left[\frac{\partial \ln \Psi}{\partial c}\right]_i\right) - \frac{1}{N}\sum_{i=1}^{N}\left(\left[E_L\left(c\right)\right]_i\sum_{j=1}^{N}\left[\frac{\partial \ln \Psi}{\partial c}\right]_j\right)\right], \tag{3.7.2}$$

where the indices's i and j run through all the time steps independently of each other. In our case, we are going to derive with respect to β only, because we are only interested in minimization in that direction. Since the derivative is composed of three parts, two for the Slater determinants and one for the Padé-Jastrow interaction part, we get

$$\begin{aligned}\frac{\partial \ln \Psi}{\partial \beta} &= \frac{\partial \ln \Psi_{SD\uparrow}}{\partial \beta} + \frac{\partial \ln \Psi_{SD\downarrow}}{\partial \beta} + \frac{\partial \ln \Psi_J}{\partial \beta} = \frac{\partial \ln \Psi_J}{\partial \beta} \\ &= \frac{\partial\left(\sum_{i<j}\frac{ar_{ij}}{1+\beta r_{ij}}\right)}{\partial \beta} = \sum_{i<j}\frac{-ar_{ij}^2}{\left(1+\beta r_{ij}\right)^2}.\end{aligned}$$

So the resulting expression is

$$\begin{aligned}\frac{\partial E}{\partial \beta} =& \frac{2}{N}\left[\sum_{i=1}^{N}\left(\left[E_L\left(\beta\right)\right]_i\left[\sum_{k<l}\frac{-ar_{kl}^2}{\left(1+\beta r_{kl}\right)^2}\right]_i\right)\right. \\ &\left. -\frac{1}{N}\sum_{i=1}^{N}\left(\left[E_L\left(\beta\right)\right]_i\sum_{j=1}^{N}\left[\sum_{k<l}\frac{-ar_{kl}^2}{\left(1+\beta r_{kl}\right)^2}\right]_j\right)\right],\end{aligned} \tag{3.7.3}$$

with k and l running through the electrons. We assume that we need to minimize with respect to one variable, β, and choose the Newton-Raphson method for

its simplicity. This method additionally requires the derivative of the function ($\partial E/\partial\beta$ in this case), but it's not possible to simply derive with respect to β again because there is no analytical expression for $E_L(\beta)$. Therefore we must resort to numerical derivation. A simple, first oder finite difference method can be used for this

$$\frac{\partial^2 E}{\partial \beta^2} = \lim_{h\to 0} \frac{\frac{\partial^2 E(\beta+h)}{\partial \beta^2} - \frac{\partial^2 E(\beta)}{\partial \beta^2}}{h}. \tag{3.7.4}$$

With this the Newton-Raphson method can be implemented in a straight forward manner with one simple equation

$$\beta_{i+1} = \beta_i - \frac{\frac{\partial E}{\partial \beta}}{\frac{\partial^2 E}{\partial \beta^2}}, \tag{3.7.5}$$

where the index i represents the iterations of the Newton-Raphson method, not the iterations of the Monte Carlo loop; for each iteration of this index a whole Monte Carlo loop is performed. In the first iteration a seed must be provided for the random generator. Depending on the guess for this seed, the method's performance will be better or worse.

There is one catch, the values of the first derivative are computed as a part of the Monte Carlo loop, and are thus susceptible to a certain degree of randomness. This variability introduces some uncertainty in each iteration. Basically, we are not applying the method to a function, but to a scattered point cloud from which we sample points one at a time. This, coupled with the fact that the second derivative is obtained numerically from two points of that cloud, made the method show poor results, namely slow convergence to values that apparently depended heavily on the seed choice. To circumvent this issue, a different, yet similar method was used, the bisection method

The bisection method is similar to the Newton-Raphson method, but it only needs the function that has the roots we want to find. And, instead of a point seed, it needs an interval seed. The method exploits Bolzano's theorem in said interval: if the values of the function (the derivative of the local energy in this case) in the extremes have different signs, the existence of at least one root is guaranteed. By evaluating the function in the midpoint of the interval, it is possible to know in which half of it the root lies, and thus the interval can be reduced to one of its halves. This process is repeated until a certain tolerance is reached and the root is obtained. In our case we don't have to worry about

multiple roots because we know that there is only one minimum, the only problem is choosing the appropriate intervals so that the method can find the root.

The main advantage here compared to the Newton-Raphson method is that the sensibility to the values of the function is much smaller because only the sign of the function values is important (and this only becomes a problem when we are already very close to the root). The other advantage is the robustness and simplicity, the method will converge no matter what if the appropriate interval is chosen, which is not something very complicated to guess and can be checked quite fast in case it's not so obvious. The downside is the linear, comparatively slow convergence. But in this case, due to the the imprecision in obtaining the values of the derivative of the local energy, the Newton-Raphson method becomes relatively slow as well, so it is not actually a problem for our purposes.

Chapter 4

Modelled systems

We will in this thesis look at two kinds of systems: quantum dots and atomic systems. The atom- and quantum dot-systems we look at all have closed form solutions in the non-interacting case. Furthermore the quantum dots we look at all consist of a so-called magic number of electrons, meaning that all the nuclear shells are filled.

4.1 Quantum dots

Quantum dots, a term coined by M. Reed in 1988 [Reed et al., 1988], are semiconductors made up of electrons that are tightly confined in all three spatial dimensions by a potential well. This well can be tuned by making the well broader or narrower, and thus changing the material properties of the quantum dot, which acts like a semiconductor. This gives quantum dots a great range of applications, for example in solar cells [Kim et al., 2015], LEDs [Neidhardt et al., 2015], lasers [Strauf et al., 2006], and medical imaging [Tokumasu et al., 2005]. It is also possible that quantum dots can be used as so-called qbits in quantum computing [Fedichkin et al., 2000]. The applications of quantum dots are however not the focus of this thesis.

By studying electrons confined in a potential well in this manner, we may gain insight into important physics regarding many-body theory. Therefore, the motivation to study the quantum dot system purely academically is to explore how the system behaves resulting from how tightly the electrons are confined, and the number of electrons.

As a model for a quantum dot we use electrons trapped in a harmonic potential. The potential has a frequency ω. Without any electron-electron interaction this potential can be solved analytically, which will become convenient as a verification of the VMC solver.

4.1.1 Harmonic Oscillators

The harmonic oscillator potential is a natural choice to describe the quantum dots, because they are essentially electrons without a nucleus. This oscillator potential has the form

$$\hat{\mathbf{v}}_{ext}(\mathbf{r}) = \frac{1}{2}\omega^2 r^2,$$

where ω is the frequency of the oscillator. The single particle Hamiltonian for the harmonic oscillator is

$$\hat{\mathbf{h}}_0(\mathbf{r}) = -\frac{1}{2}\nabla^2 + \frac{1}{2}\omega^2 r^2,$$

with corresponding eigenfunction in two and three dimensions

$$\begin{aligned}
\phi_{n_x,n_y}(\mathbf{r}) &= H_{n_x}\left(\sqrt{\omega}x\right) H_{n_y}\left(\sqrt{\omega}y\right) e^{-\frac{1}{2}\omega r^2} \\
\phi_{n_x,n_y,n_z}(\mathbf{r}) &= H_{n_x}\left(\sqrt{\omega}x\right) H_{n_y}\left(\sqrt{\omega}y\right) H_{n_z}\left(\sqrt{\omega}z\right) e^{-\frac{1}{2}\omega r^2}.
\end{aligned}$$

Here $H_n(x)$ is the n-level Hermite polynomial. Different combinations of n_x and n_y for two dimensions, and n_x, n_y and n_z for three dimensions, with the same total n gives the shell structure of a quantum dot.

We introduce a variational parameter α with $\omega \to \alpha\omega$, and define $k \equiv \sqrt{\alpha\omega}$. Now the eigenfunctions used as single-particle orbitals are

$$\begin{aligned}
\phi_{n_x,n_y}(\mathbf{r}) &= H_{n_x}(kx) H_{n_y}(ky) e^{-\frac{1}{2}k^2 r^2} \\
\phi_{n_x,n_y,n_z}(\mathbf{r}) &= H_{n_x}(kx) H_{n_y}(ky) H_{n_z}(kz) e^{-\frac{1}{2}k^2 r^2}.
\end{aligned}$$

A set of quantum numbers are mapped to an integer i, representing different single-particle wave functions. For a complete list of orbital functions used, see appendices C and D

4.1.2 Quantum dots in two and three dimensions

The model we use for the quantum dot is N electrons, interacting with each other, confined in a harmonic oscillator potential with frequency ω. The Hamiltonian

of the quantum dot is then

$$\begin{aligned}\hat{\mathbf{H}}_{QD}(\mathbf{r}) &= \sum_{i=1}^{N} \hat{\mathbf{h}}_0(\mathbf{r}_i) + \sum_{i<j} \frac{1}{r_{ij}} \\ &= \sum_{i=1}^{N} \left[-\frac{1}{2}\nabla_i^2 + \frac{1}{2}\omega^2 r_i^2 \right] + \sum_{i<j} \frac{1}{r_{ij}}.\end{aligned}$$

Here r_{ij} is the distance between electron i and electron j, that is $r_{ij} = |\mathbf{r}_i - \mathbf{r}_j|$.

If we disregard electron-electron interaction and thus exclude the Coulomb term, we can decouple the Hamiltonian into single-particle terms. For the quantum dot the total ground state energy becomes [Griffiths, 2005]

$$E_0 = \omega \sum_{i=1}^{N} \left(n_i + \frac{d}{2} \right), \tag{4.1.1}$$

where d represents the number of dimensions. For two dimensions we have $n_i = n_x + n_y \geq 0$, and for three dimensions we have $n_i = n_x + n_y + n_z \geq 0$.

4.2 Atoms and Molecules

Atomic systems consist of a positively charged nucleus and a number of negatively charged electrons surrounding it. We know that the mass of the nucleus is many orders of magnitude larger than the mass of the electron. Therefore we can do the approximation that the position of the nucleus is independent on the electrons, and we keep the position of the nucleus fixed. This is called the Born-Oppenheimer Approximation [Born and Oppenheimer, 1927].

The dimensionless Hamiltonian for the case of electrons around a nucleus is given by

$$\hat{H} = \sum_{i=1}^{N} -\frac{\nabla_i^2}{2} - \frac{Z}{r_i} + \sum_{i<j} \frac{1}{r_{ij}}, \tag{4.2.1}$$

where r_i is the distance from electron i to the nucleus, Z is the nucleus charge, and $r_{ij} = |\mathbf{r}_i - \mathbf{r}_j|$. The kinetic energy for electron i is represented by $-\nabla_i^2/22$, $-Z/r_i$, the potential energy with respect to the nucleus, and $1/r_{ij}$, the repulsive energy between the electrons i and j.

In the VMC calculation the local energy, E_L is a useful quantity and needs to be finite at all points to be normalizable. So by looking at the limits where the the local energy diverges we can guess the form the wave functions should follow,

$$E_L(r_i, r_{ij}) = \frac{1}{\Psi_T}\hat{H}\Psi_T. \tag{4.2.2}$$

In the cases where $r_i \to 0$ or $r_{ij} \to 0$ we need to make sure that the local energy does not diverge

$$\lim_{r_i \to 0} E_L(r_i, r_{ij}) = \frac{1}{R_i(r_i)}\left(-\frac{1}{2}\frac{\partial^2}{\partial x_k^2} - \frac{Z}{r_i}\right)R_i(r_i) + G(r_i, r_{ij}) \tag{4.2.3}$$

$$\lim_{r_i \to 0} E_L(r_i, r_{ij}) = \frac{1}{R_i(r_i)}\left(-\frac{1}{2}\frac{\partial^2}{\partial r_k^2} - \frac{1}{r_i}\frac{\partial}{\partial r_i} - \frac{Z}{r_i}\right)R_i(r_i) + G(r_i, r_{ij}) \tag{4.2.4}$$

Given a well behaved wave function $\partial^2/(\partial r_k^2)$ is finite, and we have

$$\lim_{r_i \to 0} E_L(r_i, r_{ij}) = \frac{1}{R_i(r_i)}\left(-\frac{1}{r_i}\frac{\partial}{\partial r_i} - \frac{Z}{r_i}\right)R_i(r_i). \tag{4.2.5}$$

This is finite given when the following differential equation is fulfilled

$$\frac{1}{R_i(r_i)}\frac{\partial}{\partial r_i}R_i(r_i) = -Z \qquad \text{with solution} \qquad R_i(r_i) = Ae^{-Z}. \tag{4.2.6}$$

A similar calculation applies for $r_{12} \to 0$ and a trial function of the form

$$\Psi_T(r_i, r_j, r_{ij}) = e^{-\alpha\sum_N r_i}\prod_{i<j}^{N} e^{\beta r_{ij}} = e^{-\alpha\sum_N r_i}\prod_{i<j}^{N} e^{\frac{ar_{ij}}{1+\beta r_{ij}}}$$

should fulfill the condition that the local energy is finite.

4.2.1 Hydrogenic wave functions

A hydrogen atom is analytically solvable and we have exact wave functions corresponding to the electron being in the different shells. When we are building atoms containing more electrons, thus turning it into a many-body problem, we

	ψ_i	$\nabla\psi_i$	$\nabla^2\psi_i$
ψ_{1S}	$e^{-\alpha r i}$	$-\frac{\alpha}{r_i}\mathbf{r_i}e^{-\alpha r_i}$	$\frac{\alpha}{r_i}\left(\alpha r_i - 2\right)e^{-\alpha r_i}$
ψ_{2S}	$\left(-\frac{\alpha r_i}{2}+1\right)e^{-\frac{\alpha r_i}{2}}$	$\frac{\alpha}{4r_i}\left(\alpha r_i - 4\right)\mathbf{r_i}e^{-\frac{\alpha r_i}{2}}$	$-\frac{\alpha}{8r_i}\left(\alpha^2 r_i^2 - 10\alpha r_i + 16\right)e^{-\frac{\alpha r_i}{2}}$
ψ_{2Px}	$\alpha x_i e^{-\frac{\alpha r_i}{2}}$	$-\frac{\alpha x_i}{2r_i}\left(\alpha\mathbf{r_i} - 2r_i\hat{\mathbf{i}}\right)e^{-\frac{\alpha r_i}{2}}$	$\frac{\alpha^2 x_i}{4r_i}\left(\alpha r_i - 8\right)e^{-\frac{\alpha r_i}{2}}$
ψ_{2Py}	$\alpha x_i e^{-\frac{\alpha r_i}{2}}$	$-\frac{\alpha y_i}{2r_i}\left(\alpha\mathbf{r_i} - 2r_i\hat{\mathbf{i}}\right)e^{-\frac{\alpha r_i}{2}}$	$\frac{\alpha^2 x_i}{4r_i}\left(\alpha r_i - 8\right)e^{-\frac{\alpha r_i}{2}}$
ψ_{2Pz}	$\alpha x_i e^{-\frac{\alpha r_i}{2}}$	$-\frac{\alpha z_i}{2r_i}\left(\alpha\mathbf{r_i} - 2r_i\hat{\mathbf{i}}\right)e^{-\frac{\alpha r_i}{2}}$	$\frac{\alpha^2 x_i}{4r_i}\left(\alpha r_i - 8\right)e^{-\frac{\alpha r_i}{2}}$

Table 4.1: The different hydrogenic wave functions along with the gradients and laplacians. The derivatives is computed by a python script, using SciPy.

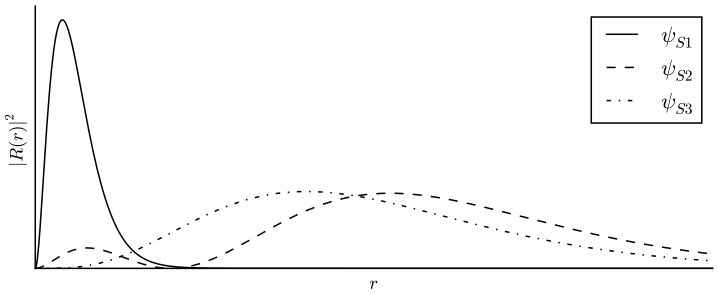


Figure 4.1: The radial probability distribution of the first three orbitals of a hydrogen atom. The radial distribution functions is taken from [Pauling and Bright, 1935].

base our guess at the trial function on the solutions to the hydrogen atom. So we need the wave functions for the five lowest states to calculate up to the neon atom, which consists of 10 electrons. The hydrogenic wave functions along with their gradients and laplacians is contained in Tab. 4.1. The radial distribution of the first three hydrogen orbitals, which our trial functions is based on, is depicted in Fig. 4.1.

4.2.2 The helium atom

Let us start with the simplest atom we can use in many-body calculations, the helium atom. The Hamiltonian for the helium atom is given by Eq. (4.2.1) and a trial function that fulfills the cusp conditions discussed earlier is

$$\Psi_{T2}(\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_{12}) = e^{-\alpha(r_1+r_2)}e^{\frac{r_{12}}{2(1+\beta r_{12})}}. \tag{4.2.7}$$

We have also done calculations for a simplified version of the trial function by removing the electron-electron interaction from the trial function, so we end up with the trial function

$$\Psi_{T1}(\mathbf{r}_1, \mathbf{r}_2) = e^{-\alpha(r_1+r_2)}. \tag{4.2.8}$$

The local energies for these two trial functions have been calculated, and they are for Ψ_{T1} and Ψ_{T2},

$$\begin{aligned} E_{L1} &= (\alpha - Z)\left(\frac{1}{r_2} + \frac{1}{r_2}\right) + \frac{1}{r_{12}} - \alpha^2 \\ E_{L2} &= E_{L1} + \frac{1}{2\,(1+\beta r_{12})}\left[\frac{\alpha(r_1+r_2)}{r_{12}}\left(1 - \frac{\mathbf{r_1}\mathbf{r_2}}{r_1 r_2}\right)\right. \\ &\qquad \left. - \frac{1}{2\,(1+\beta r_{12})} - \frac{2}{r_{12}} + \frac{2\beta}{(1+\beta r_{12})}\right]. \end{aligned}$$

See appendix A for a derivation of the local energies.

4.2.3 The beryllium atom

It is fairly simple to extend the machinery of variational Monte Carlo to other systems than the helium atom. To show this flexibility the beryllium atom will be studied. As beryllium has four electrons compared to the two in helium, we need to construct a Slater determinant from the hydrogenic wave functions. Sticking to hydrogen-like wave functions, we can write the trial wave function for beryllium as

$$\psi_T(\mathbf{r_1}, \mathbf{r_2}, \mathbf{r_3}, \mathbf{r_4}) = Det\left(\phi_1(\mathbf{r_1}), \phi_2(\mathbf{r_2}), \phi_3(\mathbf{r_3}), \phi_4(\mathbf{r_4})\right) \prod_{i<j}^{4} \exp\left\{\left(\frac{a r_{ij}}{(1+\beta r_{ij})}\right)\right\}, \tag{4.2.9}$$

where *Det* is a Slater determinant and the single-particle wave functions are the hydrogen wave functions for the 1s and 2s orbitals. With the variational ansatz these are for 1s

$$\phi_{1s}(\mathbf{r_i}) = e^{-\alpha r_i}, \tag{4.2.10}$$

and for 2s

$$\phi_{2s}(\mathbf{r_i}) = (1 - \alpha r_i/2)\, e^{-\alpha r_i/2}. \tag{4.2.11}$$

The Slater determinant is calculated using these ansatzes, and can for beryllium be written out as

$$|D| \propto \left[\psi_{1s}(\mathbf{r_1})\psi_{2s}(\mathbf{r_2}) - \psi_{1s}(\mathbf{r_2})\psi_{2s}(\mathbf{r_1})\right]\left[\psi_{1s}(\mathbf{r_3})\psi_{2s}(\mathbf{r_4}) - \psi_{1s}(\mathbf{r_4})\psi_{2s}(\mathbf{r_3})\right]. \tag{4.2.12}$$

Furthermore, for the Jastrow factor,

$$\Psi_C = \prod_{i<j} g(r_{ij}) = \exp\left\{\sum_{i<j} \frac{a r_{ij}}{1+\beta r_{ij}}\right\}, \tag{4.2.13}$$

we need to take into account the spins of the electrons. We fix electrons 1 and 2 to have spin up, and electron 3 and 4 to have spin down. This means that when the electrons have equal spins we get a factor

$$a = \frac{1}{4}, \tag{4.2.14}$$

and if they have opposite spins we get a factor

$$a = \frac{1}{2}. \tag{4.2.15}$$

4.2.4 The neon atom

Wishing to extend the variational Monte Carlo machinery further neon is implemented. Neon has ten electrons, so it is a big jump from helium and beryllium. Now we need better methods to handle the Slater determinant, which are described in section 3.4. The trial wave function for neon can be written as

$$\psi_T(\mathbf{r_1}, \mathbf{r_2}, \ldots, \mathbf{r_{10}}) = Det\left(\phi_1(\mathbf{r_1}), \phi_2(\mathbf{r_2}), \ldots, \phi_{10}(\mathbf{r_{10}})\right) \prod_{i<j}^{10} \exp\left\{\left(\frac{r_{ij}}{2(1+\beta r_{ij})}\right)\right\}, \tag{4.2.16}$$

Now we need to include the $2p$ wave function as well. It is given as

$$\phi_{2p}(\mathbf{r_i}) = \alpha \mathbf{r_i} e^{-\alpha r_i/2}, \tag{4.2.17}$$

where $r_i = \sqrt{r_{ix}^2 + r_{iy}^2 + r_{iz}^2}$.

4.2.5 The hydrogen molecule

Because of its flexibility the VMC machinery can also handle molecules, with some modifications to the trial function considering that we now have two nuclei. So we need a slightly different Hamiltonian, where we we need include a few more terms in the potential energy of the system. If we let $\mathbf{R}$ be the vector between the nuclei we can write the positions, $\mathbf{r_{ipk}}$, of electron i in relation to nucleus k

$$\mathbf{r_{ip1}} = \mathbf{r_i} + \frac{\mathbf{R}}{2} \tag{4.2.18}$$

$$\mathbf{r_{ip2}} = \mathbf{r_i} - \frac{\mathbf{R}}{2}. \tag{4.2.19}$$

Then we add all the terms for the potential energy with the kinetic energy and get the Hamiltonian

$$\hat{H} = \sum_{i=1}^{2} -\frac{\nabla_i^2}{2} - \frac{Z_1}{r_{ip1}} - \frac{Z_2}{r_{ip2}} + \sum_{i<j} \frac{1}{r_{ij}} + \frac{Z_1 Z_2}{|\mathbf{R}|}. \tag{4.2.20}$$

Here Z_1 and Z_2 are given by the charge of their respective nuclei. Assuming that the trial-function for the hydrogen molecule is similar to two hydrogen atoms we can base our guess of the trial-function on a linear combination of the two hydrogenic 1s wave functions around each nuclei. Disregarding factors, since they disappear in the VMC computation and assuming symmetry about the nuclei, we end up with

$$\Psi_T(\mathbf{r_1}, \mathbf{r_2}, \mathbf{R}) = \psi(\mathbf{r_1}, \mathbf{R})\psi(\mathbf{r_2}, \mathbf{R}) \exp\left\{ \frac{r_{12}}{2(1+\beta r_{12})} \right\}, \tag{4.2.21}$$

where the hydrogenic wave functions is given by

$$\psi(\mathbf{r_i}, \mathbf{R}) = \left[\exp\{-\alpha r_{ip1}\} \pm \exp\{-\alpha r_{ip2}\} \right]. \tag{4.2.22}$$

Here we should add together the 1s wave functions, as subtracting corresponds to the electrons having the same spin. In a Hydrogen molecule the electrons will have different spin if possible.

4.2.6 The Beryllium Molecule

The beryllium molecule consists of four electrons shared between two nuclei with a charge of $Z = 4$, we will use the same method to calculate it as in Hydrogen molecule and it also shares the same Hamiltonian. As in the description of the beryllium atom we will need to construct the trial-function out of an Slater Determinant consisting of linear combinations of the hydrogenic wave functions ψ_{1s} and ψ_{2s}, and a correlation term, yielding

$$\Psi_T(\mathbf{r_i}, \mathbf{R}) = |D| \prod_{i<j}^{4} \exp\left\{ \left(\frac{a r_{ij}}{(1 + \beta r_{ij})} \right) \right\}, \tag{4.2.23}$$

$$\tag{4.2.24}$$

where the Slater determinant is constructed by the following wave functions

$$\psi_{1S1}(\mathbf{r_i}) = [\exp\{-\alpha r_{ip1}\} + \exp\{-\alpha r_{ip2}\}] \tag{4.2.25}$$
$$\psi_{1S2}(\mathbf{r_i}) = [\exp\{-\alpha r_{ip1}\} - \exp\{-\alpha r_{ip2}\}] \tag{4.2.26}$$
$$\psi_{2S1}(\mathbf{r_i}) = \left[(1 - \alpha r_{i1p}/2)\, e^{-\alpha r_{i1p}/2} + (1 - \alpha r_{i2p}/2)\, e^{-\alpha r_{i2p}/2} \right] \tag{4.2.27}$$
$$\psi_{2S2}(\mathbf{r_i}) = \left[(1 - \alpha r_{i1p}/2)\, e^{-\alpha r_{i1p}/2} - (1 - \alpha r_{i2p}/2)\, e^{-\alpha r_{i2p}/2} \right]. \tag{4.2.28}$$

Here we need to use both subtracting and addition in the combinations in the construction of the wave functions as each beryllium will have two electrons in each shell so there will need to be two of the same spin in each of the shells.

4.3 Gaussian type orbitals

As proposed in a paper by Boys in 1950 [Boys, 1950], Gaussian Type Orbitals, or GTOs, can be used in electronic structure theory, and it is today common to use them, especially in computational chemistry. Most importantly for us they make us able to limit the variational calculations of complex atoms to one variational variable, thus reducing the number of variational possibilities significantly. This is obviously a great advantage, because as the number of particles in our calculations grow, the energy takes considerably longer time to compute for each combination of variables.

4.3.1 Using GTOs to replace the Slater type orbitals

To replace our current orbitals we need to calculate primitive GTOs and contract them to contracted GTOs. A contracted GTO, ϕ, is defined as

$$\phi(x, y, z) = \sum_i N_i \chi_i(x, y, z),$$

where χ_i is a primitive GTO and N_i is a normalization constant. The primitive GTO is defined as

$$\chi_i(x, y, z) = c_i x^m y^n z^o e^{-\alpha_i R^2}.$$

Here x, y, and z are Cartesian coordinates representing the distance to a nucleus, and $R^2{=}x^2 + y^2 + z^2$. The quantum numbers m, n and o depend on the angular momentum of the orbital. The numbers c_i and α_i are variational parameters, which can be fetched from an online library, namely from the Environmental Molecular Sciences Laboratory [J. S. Binkley, 1980][EMS, 2015]. From this library we use the 3-21G basis to represent the orbitals.

When combining contracted primitives into orbitals that are to mimic and replace the Slater type orbitals additional constants are needed, one for each contracted GTO. These are calculated using Hartree-Fock calculations. For a complete list of the parameters used, see Tabs. B.1, B.2, and B.3 in appendix B.

The contracted GTO primitives are not a perfect representation for the Slater Type Orbitals, however, as we can see in Fig. 4.2 and Fig. 4.3. The shape of the contracted GTO resembles the STO, but the top of the GTO is still not as sharp as the STO. A way to get a closer resemblance to the STO is to use another basis with a bigger set of primitives.

As an example, for helium, we find the parameters to be

	α_i	c_i
χ_1	13.62670	0.175230
χ_2	1.999350	0.893483
χ_3	0.382993	1.000000

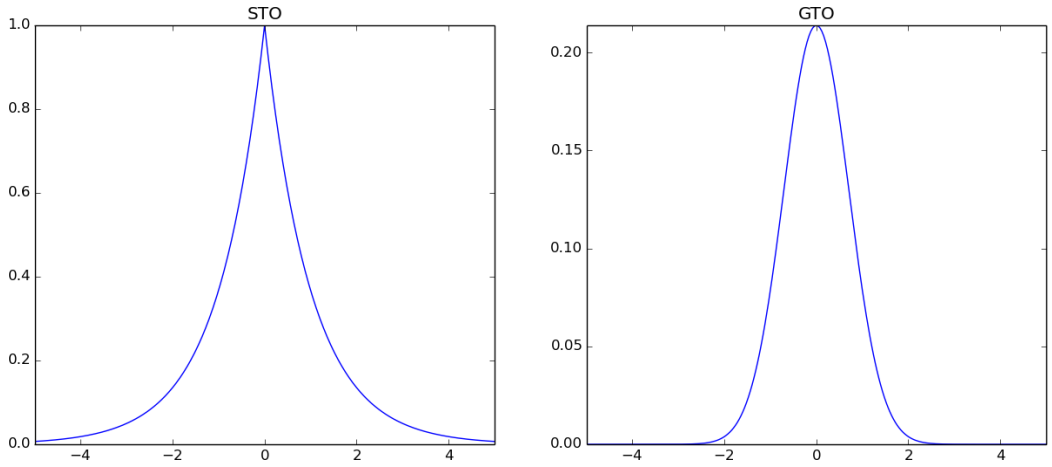


Figure 4.2: The shape of a Slater Type Orbital (left) compared to the shape of a GTO primitive (right).

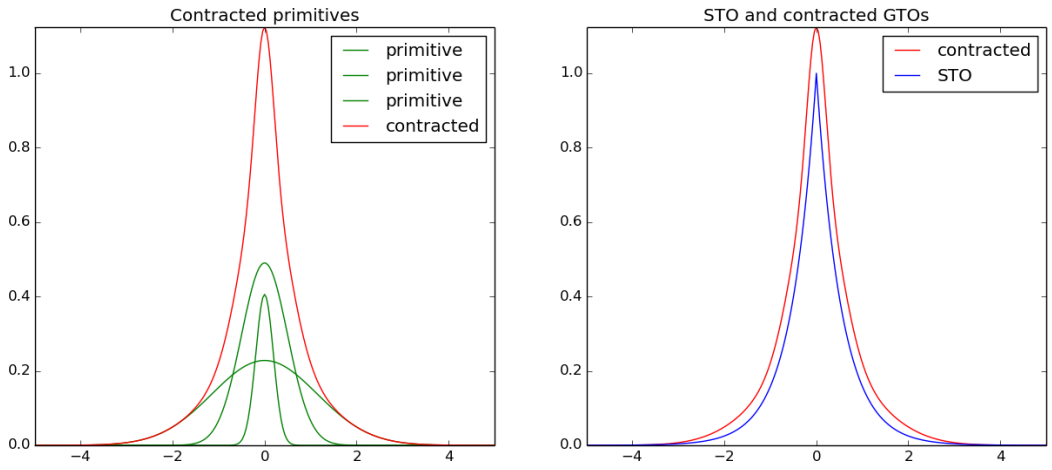


Figure 4.3: Construction of a contracted GTO from primitives (left) and comparison of the contracted GTO to the Slater Type Orbital (right).

Thus the Gaussian orbital becomes

$$\begin{aligned}\phi(x,y,z) &= 0.4579 \times \left(\left(\frac{2 \times 13.62670}{\pi} \right)^{3/4} 0.17523 e^{-13.62670 \times R^2} \right. \\ &\qquad \left. + \left(\frac{2 \times 1.99935}{\pi} \right)^{3/4} 0.893483 \times e^{-1.99935 \times R^2} \right) \\ &\quad +0.6573 \times \left(\left(\frac{2 \times 0.382993}{\pi} \right)^{3/4} 1.0 e^{-0.382993 \times R^2} \right).\end{aligned}$$

Chapter 5

Implementation

In this chapter the Variational Monte Carlo program will be presented. We will look at the structure and the implementation of the program, and on the class structure. Details about the code will be given and finally a description of the classes making up the program will be presented.

5.1 Structure

For short, straightforward programs the structure of the code is more or less irrelevant. But in scientific programming a program easily becomes large and complex, and it is therefore important to structure the code in an orderly manner. This makes it not only easier to debug, but also makes it easier for others to read and understand what all parts of the program does.

For large programs, like the VMC solver program in this thesis, it is generally recommended to use object orientation. This makes the program tidy and flexible. From the main function the solver program in this thesis initializes the VMC solver class, `VMCSolver`. For an overview of the `VMCSolver`-class, see Fig. 5.1. This class will in turn initialize a class for the trial function, a class for the Slater determinant, and a class for the derivatives associated with the trial function. See Fig. 5.2 for an overview of the subclasses.

The job of the `main`-function in the variational Monte Carlo solver is to gather input parameters like number of cycles in the Monte Carlo simulation, and what kind of test should be run. The tests include finding optimal α- and β-values and testing for different oscillator frequencies, ω, and calculating the ground state

energy with the found variational parameters while saving detailed data like positions of each particle. Each of the tests will call the `VMCSolver`-class as needed, and thus start the simulation process. When initialized by the `main`-function, the `VMCSolver`-class will initialize the other classes `SlaterDeterminant`, `Derivatives`, `Orbitals`, and `Trialfunction`.

The `Trialfunction`-class acts as a wrapper for the `QuantumDots`-class. This is not strictly necessary for our needs, as the only trialfunction we use is the quantum dots one. However, we wish to make the program flexible and to make it simple to add additional parts. Therefore, to be able to choose between different trialfunctions easily, we need a class like the `Trialfunction`-class.

Consisting of a number of Hermite polynomials, the orbital functions are generated by a Python script. By using SymPy [SymPy Development Team, 2016] we can easily find the analytical expressions of the derivatives and gradients of the orbital functions. The `Orbitals`-class and its subclasses are therefore maintained by the Python script. The `Orbitals`-class itself acts as an interface, where values for the orbitals and their derivative functions can be extracted.

5.1.1 Class structure

Classes in a large program can be many hundred or even thousands of lines long. This makes it messy and cumbersome to keep all classes in a single file. A standard way to deal with this is to have a file for each class and subclass. This is not only much more tidy than having everything in a single file, but has the practical benefit of not needing to recompile the entire program on every compilation; it suffices to compile only the changed classes. This can cut down compilation times considerably.

For this reason every class created as part of the VMC solver has its own source file and header file, and are neatly stored in a hierarchy of folders. This way it is also easy to implement new systems, in form of a new class, simply by adding its class and corresponding files to the main program.

5.2 Code

The VMC solver in this thesis is written in C++. This language is chosen mainly because it is more efficient than high-level languages like Python. It is also especially suitable because of its flexibility as an object-oriented language. To

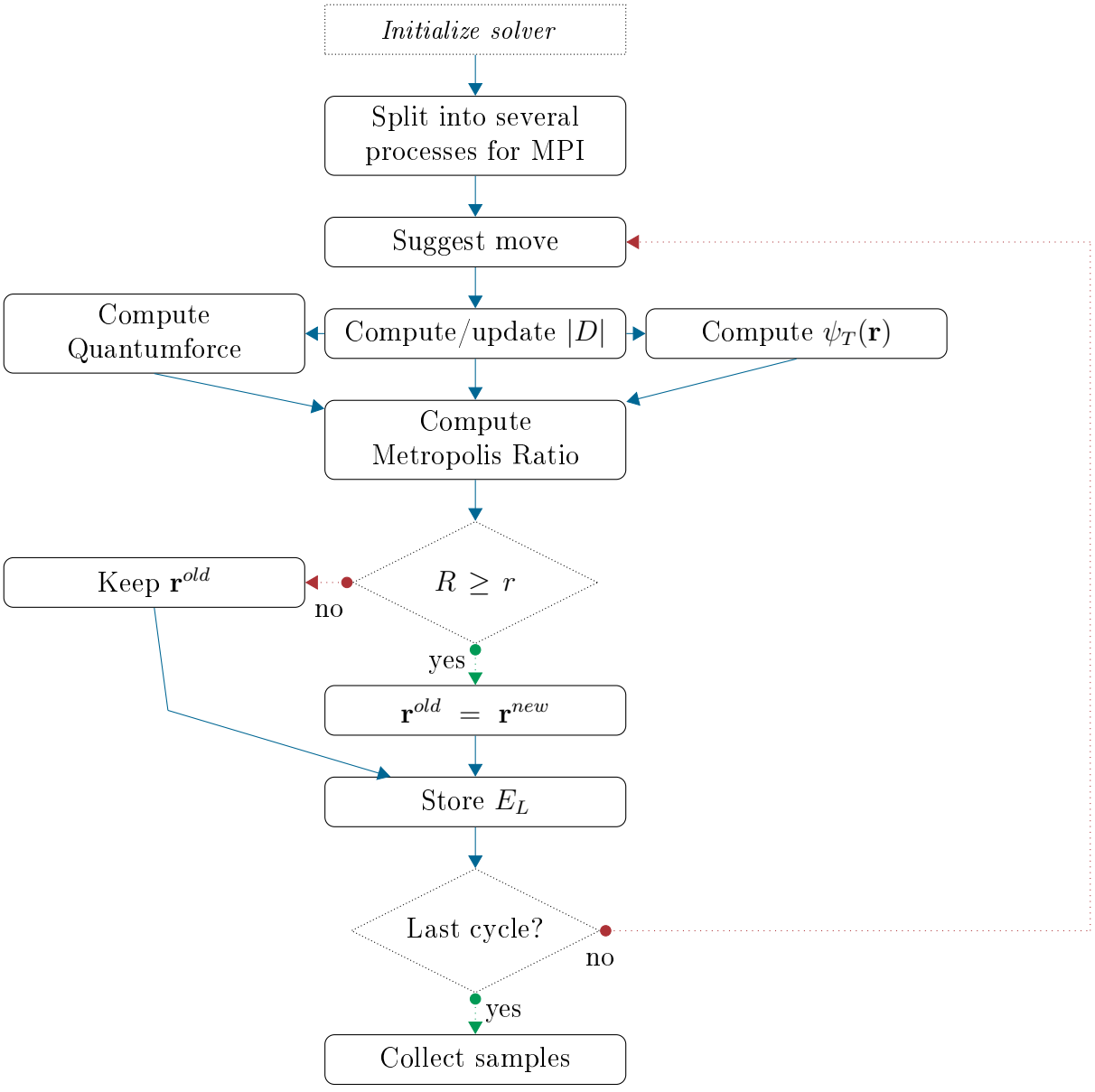


Figure 5.1: Schematic overview over the workflow of the VMC solver. The solver is initialized and called by the `main`-function.

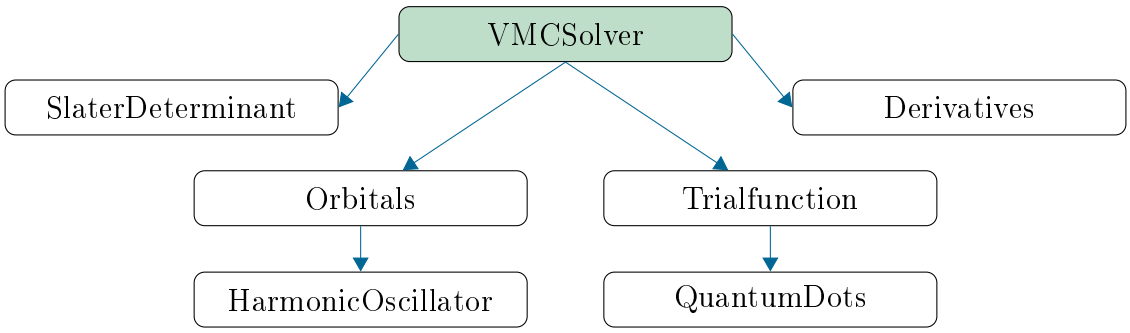


Figure 5.2: Class and subclass structure used by the program. The `VMCSolver`-class initializes the other classes `SlaterDeterminant`, `Derivatives`, `Orbitals`, and `Trialfunction`. Classes `Orbitals` and `Trialfunction` each has their own subclasses.

handle matrices efficiently the solver uses the Armadillo library [Sanderson, 2010], which provides easy methods for creating and filling matrices and vectors, and doing linear algebra.

The main program of the simulator handles running of different tests on the chosen system. To find the optimal values for α and β it performs several simulations over a range of α- and β-values, and iteratively zooms in on the part of the range with the best fit. This way it gains accuracy with each iteration. This can continue until a given accuracy is met, or for a given number of iterations.

To run tests for different oscillator frequencies the program simply iterates over given numbers of electrons and runs simulations with different ω- and α-values.

The `VMCSolver`-class

At the heart of the simulator is the `VMCSolver` class. During initialization it also initializes the `SlaterDeterminant` class, the `Derivatives` class and the classes containing the harmonic orbitals for two and three dimensions. The main function of the `VMCSolver` class is of course to run the Monte Carlo cycles, where it follows the workflow outlined in Fig. 5.1.

The solver starts by setting initial trial positions, and calculating the Slater matrices.

```
134   void VMCSolver::runMonteCarloIntegrationIS() {
      ...
162   //initial trial positions
163   for(int i = 0; i < nParticles; i++) {
```

```
164       for(int j = 0; j < nDimensions; j++) {
165           rOld(i,j) = GaussianDeviate(&idum)*sqrt(stepLength);
166       }
167     }
168     //Set up the Slater Matrices after the move
169     determinant()->updateSlaterMatrices(rOld,this);
170     rNew = rOld;
        ...
```

Then it goes into the Monte Carlo cycles loop.

```
175     for(int cycle = 0; cycle < nCycles; cycle++) {
176       //Store the current value of the wave function
177         waveFunctionOld = trialFunction()->waveFunction(rOld, this);
178         derivatives()->numericalGradient(QForceOld,rOld, this);
          ...
```

For each iteration we find a new position for the particles, using function for the quantum force.

```
200   for(int i = 0; i < nParticles; i++) {
201     for(int j = 0; j < nDimensions; j++) {
202         rNew(i,j) = rOld(i,j) +
              GaussianDeviate(&idum)*sqrt(stepLength)+QForceOld(i,j)*stepLength*D;
203     }
      ...
```

To run the Metropolis-Hastings algorithm we first have to compute the ratio of Greens functions

```
224   GreensFunction = 0.0;
225   for (int j=0; j < nDimensions; j++) {
226       GreensFunction += 0.5*(QForceOld(i,j)+QForceNew(i,j))*
227         (D*stepLength*0.5*(QForceOld(i,j)-QForceNew(i,j))-rNew(i,j)+rOld(i,j));
228   }
229   GreensFunction = exp(GreensFunction);

231   // The Metropolis test is performed by moving one particle at the time
232   MHRatio = GreensFunction*(waveFunctionNew*waveFunctionNew) /
          (waveFunctionOld*waveFunctionOld);
233   if(ran2(&idum) <= MHRatio) {
234       acc_moves += 1;
235       for(int j = 0; j < nDimensions; j++) {
236           rOld(i,j) = rNew(i,j);
237           QForceOld(i,j) = QForceNew(i,j);
```

```
238           waveFunctionOld = waveFunctionNew;
239           }
240         }

242         else {
243         for(int j = 0; j < nDimensions; j++) {
244           rNew(i,j) = rOld(i,j);
245           QForceNew(i,j) = QForceOld(i,j);
246           determinant()->updateSlaterMatrices(rOld,this);
247         }
248       }
          ...
```

After performing the Metropolis test on all the particles, the Monte Carlo cycle is complete. After completing all cycles the energy, variance, and average distances are gathered and written to an output file.

The `SlaterDeterminant`-class

The class for the Slater determinant has several functions. One of its main functions is to update the Slater Matrices based on the positions of the electrons. This is done by simply calling a function of the `Orbitals` class for each element of the matrix. The other main function is of course to calculate the Slater determinant itself, which it does using LU decomposition. The class also contains functions to calculate the gradient and the laplacian of the Slater determinant. To achieve this we call the functions `get_gradient` and `get_laplacian` from the `Orbitals` class.

Calculating the determinant is simplified by splitting the Slater determinant by using LU decomposition, as found in Ref. [Press, 2007]. This gives us two matrices, in the code below named `detUp` and `detDown`, from which we can easily find the determinant of the Slater matrix.

```
167 double SlaterDeterminantHO::calculateDeterminant(const mat &r,double alpha,
        VMCSolver *solver){
    ...
195   ludcmp(detUp, Nhalf, indx, &d1);
196   ludcmp(detDown, Nhalf, indx, &d2);

198   // compute SD as c00*c11*..*cnn
199   SD = 1;
200   for (i = 0; i < Nhalf; ++i)
201   {
202     SD *= detUp(i, i)*detDown(i, i);
```

```
203     }
204     return d1*d2*SD;
205   }
```

The `Derivatives`-class

In VMC, sampling and updating our matrices efficiently involves calculating a number of derivatives. This is evident in the Quantum Monte Carlo section. To comply with the tidy structure we wish the program to have, the derivatives are grouped in the class `Derivatives`. However, for efficiency, some of the derivatives of the orbitals are already calculated, and resides in the `Orbitals`-class. For these, the `Derivatives`-class merely calculates the derivatives based on the precalculated derived functions.

The `Orbitals`-classes

There are two classes for the harmonic orbitals, one for two dimensions, and one for three dimensions. The functions of the two classes are similar, the only difference is the extra dimension accommodated in the three dimensional version of the class. The functions of the class is exclusively called by the Slater determinant class. The main contents of the class, that is the harmonic oscillator-functions and their derivatives, are generated by a Python script using SymPy.

Part II

Results

Chapter 6

Results

In this chapter results from the calculations will be presented. Results are produced using atomic units, meaning $\hbar = e = m_e = 4\pi\epsilon_0 = 1$, where m_e is the electron mass, and e_0 is the vacuum permittivity.

6.1 Atoms and Molecules

The attention concerning atoms has been on three single atoms, helium, beryllium and neon, and two molecules, dihelium (He_2) and diberyllium (Be_2). In this section different optimizations of the VMC solver will be tested, and ground state energies and one-body densities will be calculated. There will also be a comparison to calculations using Gaussian Type Orbitals (GTOs), and a statistical discussion, using blocking.

6.1.1 Optimization results

By using an analytical expression for the local energy instead of using a numerical derivation in the calculation the program becomes more efficient. For helium we achieve a speedup of nearly 48 percent, while for beryllium a speedup of approximately 80 percent is achieved, see Table 6.1.

Trial function	Numerical (s)	Analytical (s)	Ratio
Helium ψ_T	29.7288	20.1189	0.6767
Beryllium ψ_{T2}	58.9623	32.4622	0.5505

Table 6.1: The time to run a Monte Carlo run with 10^7 cycles for helium, and 10^6 cycles for beryllium. Using an analytical expression for the local energy decreases the computation time by a significant degree for both trial functions tested.

Num. of processes	1	2	3	4
Speedup	1.0	1.97	2.90	3.35

Table 6.2: Speedup achieved by using MPI to run computations on multiple processors.

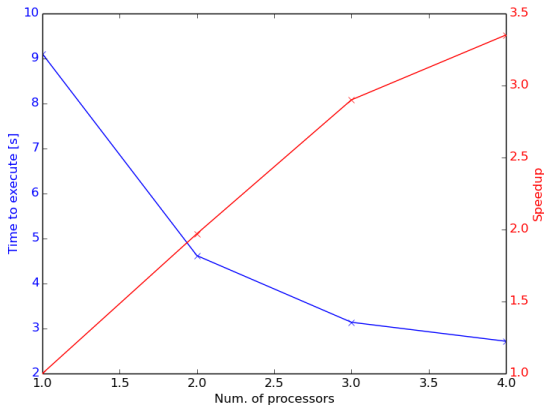


Figure 6.1: Using MPI, the time used as a function of number of processors used, and the corresponding speedup achieved.

6.1.2 Parallel scaling performance

While it is easy to think that parallelization always gives an increase in performance equal to the number of processors used, this is not always the case. As stated in section **??** not all algorithms are parallelized equally well, and heavy use of communication will slow down the calculations. However the variational Monte Carlo method is easily parallelized, and uses minimal communication. It should therefore both be possible and desirable to have a speedup as close as possible to the number of processors used.

The speedup measured by our VMC program running on one to four processors is shown in Table 6.2 and visualized in Fig. 6.1. We see that the speedup is nearly perfect for two and three processes, in that it is almost equal to the number of processes used. However when running four processes the speedup gain suffers somewhat. This is because the computer only has four processors, so running on all processors the VMC program has to share resources with the operating system and other programs. However, this simple implementation of our parallel version of the code on standard quad-core PC which can be bought in any supermarket, demonstrates the ease by which Monte Carlo methods can be run in parallel.

6.1.3 Using brute force Monte Carlo sampling

As a first attempt to solve the ground state energy for the helium atom we perform Variational Monte Carlo calculation with a brute force Metropolis sampling. We do this with two trial wave functions

$$\psi_T(\mathbf{r_1}, \mathbf{r_2}, \mathbf{r_{12}}) = \exp\{(-\alpha(r_1 + r_2))\} \exp\left\{\left(\frac{r_{12}}{2(1+\beta r_{12})}\right)\right\},$$

using α and β as variational parameters. We run the Variational Monte Carlo calculation over different values for the two variables α and β, with 2×10^7 cycles in the Monte Carlo simulation, and the resulting energies are presented in Fig. 6.2.

We find the optimal values to be $\alpha = 1.843$ and $\beta = 0.34$, as we can see in the figures. Using these values for α and β we run the brute force variational Monte Carlo calculation. The program finds an optimal value for the steplength, δ, which results in roughly 50 percent accepted moves. Using 10^8 cycles the algorithm finds the steplength to be $\delta = 1.5$, giving 48.8 percent accepted moves. The energy found with this method is -2.89024, with a variance of 3.77402×10^{-5}, as presented in Tab. 6.3. The parameter α can be interpreted as a parameter for the force pulling the electron to the nucleus.

Atom	E_{VMC}	Variance	E_{ref}
Helium	-2.89024	3.77402×10^{-5}	-2.9037

Table 6.3: Ground state energy for helium calculated with variational Monte Carlo without using Importance Sampling. The reference energy is from Ref. [Binkley and Pople, 1975].

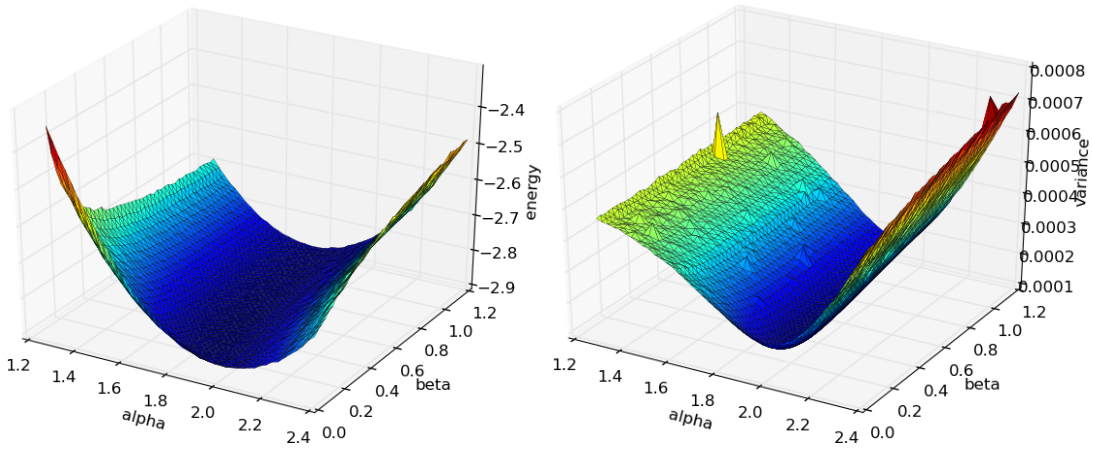


Figure 6.2: For helium using ψ_T, the energy as a function of α and β (left), and the variance as a function of α and β (right).

6.1.4 Ground state energies

We now introduce importance sampling to our calculations of the Helium atom, and expand the calculations to the Beryllium and the Neon atoms, and the Helium and the Beryllium molecules.

Expanding the system to larger atoms, we use the trial functions given in Eq. (4.2.9) for Beryllium, and Eq. (4.2.16) for Neon. Optimal values for α and β are found by variation until the ground energy is minimized. The resulting ground state energies of the systems are presented in Table 6.4. There is a relatively large uncertainty of the ground state energies, especially for Neon and the molecules, because the variance of the energy compared to the difference in energy caused by varying the parameters is quite high. However, as a function of the variational parameters the variance is more smooth, see fig 6.2, and it is therefore easier to determine the best values for the variables from variance as a function of α and β. The energy and variance as a function of the timestep, δt is shown in Fig. 6.3.

The ground state energies found with the variational Monte Carlo method are in good agreement with the reference energies. However the method generally underestimates the ground state energies. There is also a big statistical error,

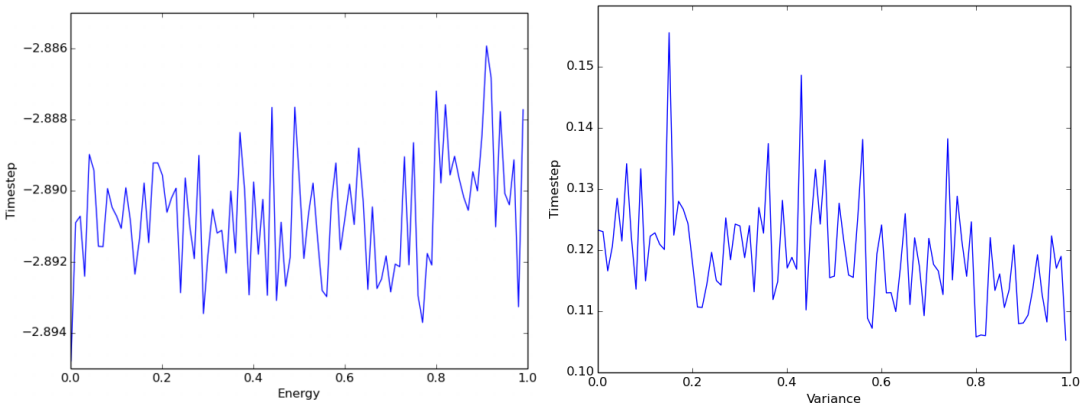


Figure 6.3: For helium ψ_T, timestep shown as a function of energy (left) and variance (right), using $\alpha = 1.84$ and $\beta = 0.34$.

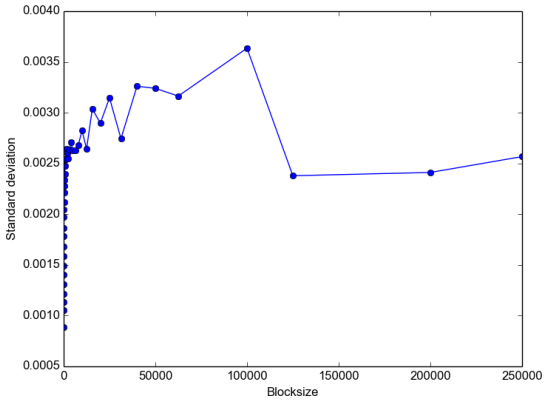


Figure 6.4: Statistical analysis for the helium case, using blocking. The blocking behaviour can be seen very clearly as it plateaus rapidly with increasing blocksize.

increasing with the size of the systems, which stems from a relatively low number of Monte Carlo samples.

To visualize the search for optimal variational parameters for the atoms we run VMC with ranges of different values for α and β. The resulting plots of the variance and the energy for different combinations are given in Fig. 6.5 for Beryllium and Fig. 6.6 for Neon. As VMC runs slowly for Neon, because it has ten electrons, we made runs over the range of Alpha and Beta values with 10^5 cycles. This is reflected in the higher variance, and the spikes in the variance plot.

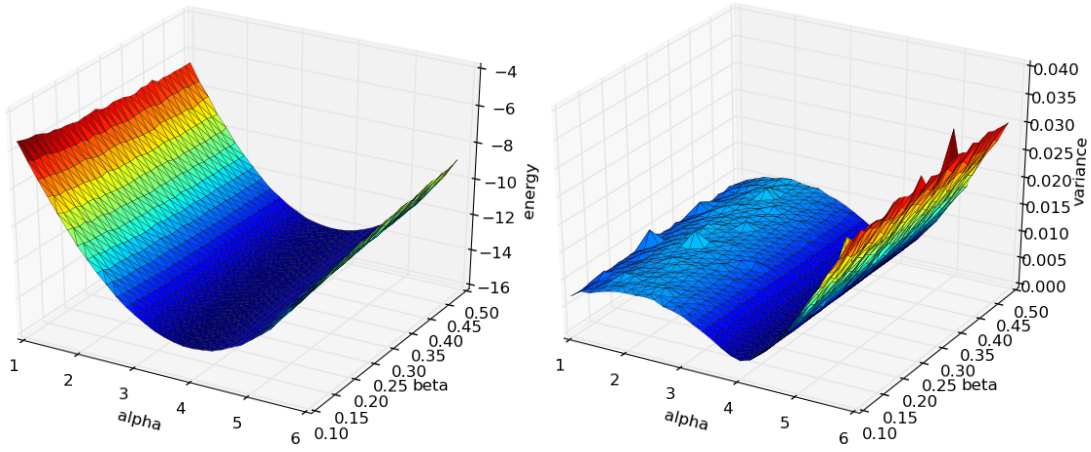


Figure 6.5: Energy (left) and variance (right) as a function of α and β for beryllium, using 10^6 cycles.

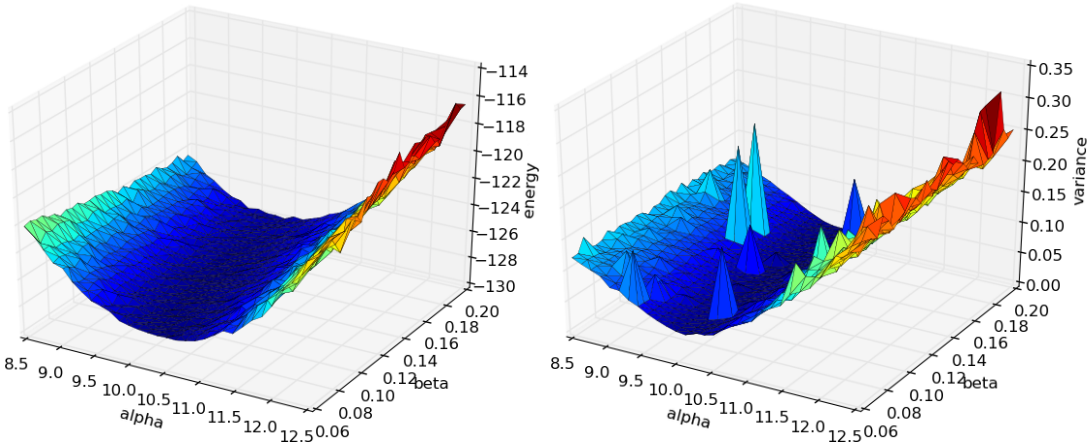


Figure 6.6: Energy (left) and variance (right) as a function of α and β for neon, using 10^5 cycles.

Atom	E_{VMC}	Variance	$E_{ref}^{(a)}$	$E_{ref}^{(b)}$
He	-2.89012	7.77×10^{-5}	-2.9037	$-2.9036(2)$
Be	-14.3902	9.09×10^{-4}	-14.667	$-14.657(2)$
Ne	-127.875	0.0132	-128.928	$-128.765(4)$
H_2	-1.15828	0.00023	-1.17540	$-1.1745(3)$
Be_2	-31.349	0.0076	-29.339	$-29.301(5)$

Table 6.4: Ground state energies of atoms and molecules, computed with the variational Monte Carlo method. For H_2 and Be_2 we used a nuclei distance of 1.40 and 4.63, respectively. Refs. (a): [Koput, 2011] [Binkley and Pople, 1975], (b): [Høgberget, 2013] (using diffusion Monte Carlo).

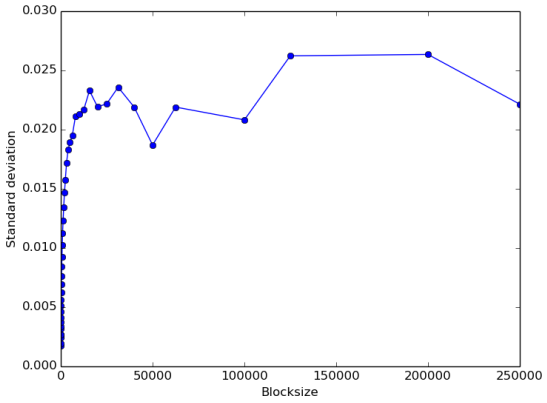


Figure 6.7: Statistical analysis of beryllium using blocking with 10^6 Monte Carlo cycles. There is a clear plateauing behaviour as the blocksize increases after rising dramatically for small blocksizes. This is due to the blocksize catching up to the correlation length between the measurements.

Atom	E_{VMC}	Var.
He	-2.757	0.0030
Be	-13.710	0.0072
Ne	-110.301	0.0477

Table 6.5: Energies for the helium, beryllium and neon atoms using simple helium wave functions with only one variational parameter. Note that the variance is the variance for the local energy between each particle move, not the true variance.

6.1.5 One-body densities

The helium atom

Simulations of the helium atom have been studied using several different trial functions. First the electrons are regarded as having pure hydrogenlike wave functions. Then the hydrogenlike wave functions are expanded by optimizing the α value to get a better ground state energy. For the final calculations the Padé-Jastrow correlation factor is added to include the effects of the electron-electron repulsion. The results of the case without Padé-Jastrow correlation is presented in Table 6.5. Note that the variance listed is the variance for the local energy between each particle moves in the VMC calculations, not the true variance, which is obtained through the blocking routine described in section 3.6.

In Fig. 6.8 histograms of the positions the electrons occupy during the simulations are presented, and we can see that for the pure hydrogenic wave function the electrons are generally closer to the nucleus than in the other two simulations. The ground-state energy improved and got closer to the reference value, as listed in 6.4, as the wave function got more sophisticated, and we have a decreasing variance. In the wave function the α parameter represents the electrical attraction of the nucleus on the electron, and represents an effective charge. In a real Helium atom the attraction felt by the electrons is smaller than the charge of the nucleus because it is also under the influence of the other electrons. For the Helium atom the effect of the correlation factor is not very pronounced, because there are only two electrons, and they can be quite far away from each other. However the correlation factor is still significant in the ground-state energy and the variance.

As expected, since the trial functions used here is based only on the ψ_{1S} orbitals, the radial distribution is similar to the ψ_{1S} hydrogen orbital in 4.1.

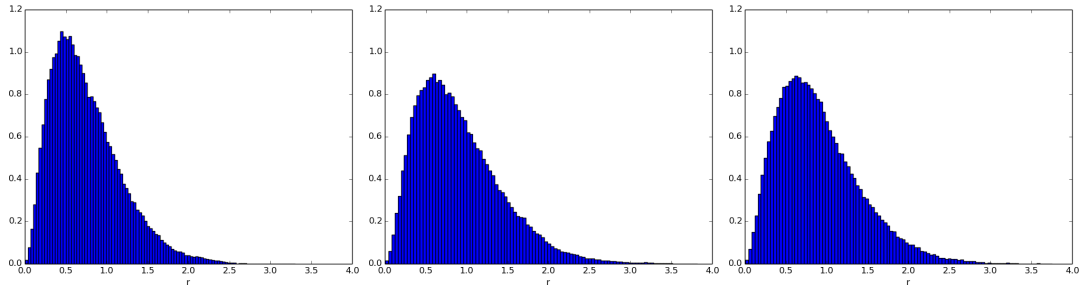


Figure 6.8: Radial densities of an helium atom simulated with different trial functions. From the left the plots represents pure hydrogenic wave functions, hydrogenic wave functions with an optimized alpha, and hydrogenic wave functions with a Jastrow factor and optimized variational parameters.

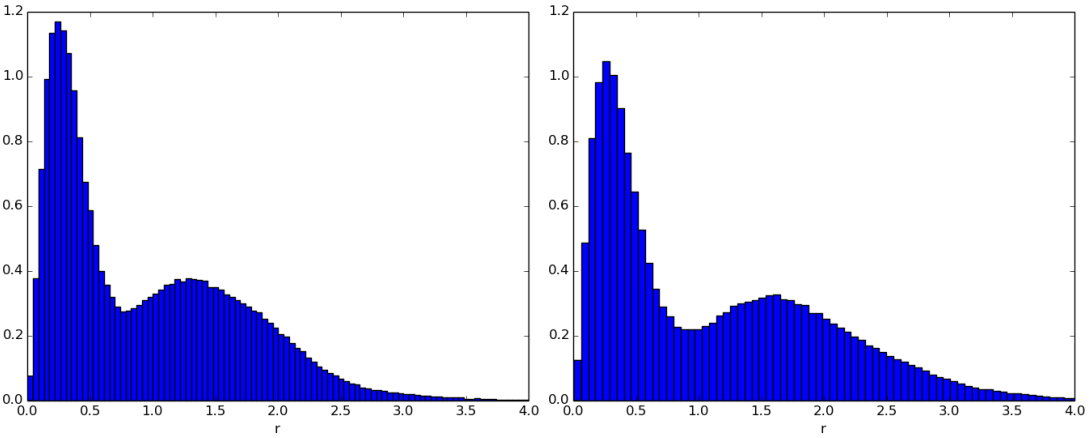


Figure 6.9: The radial densities of a beryllium atom simulated with different trial functions. The left plot is with hydrogenic wave functions and the right plot is with hydrogenic wave functions using a Jastrow factor and optimized parameters.

The beryllium atom

There is a similar drop in the energy going from a purely hydrogenic wave function to a wave function where we have introduced a variational parameter, comparing values in Tabs. 6.5 and 6.4. The hydrogenic wave function for beryllium consists of a Slater determinant constructed of the first two hydrogenic orbitals, plotted in Fig. 4.1. The traces of the 1s orbital is there in the first maximum and is closer to the nucleus than in an hydrogen atom, which is expected since the distance is a function of the nucleus charge. The contribution from the second orbital 2s is visible as the second maximum in the distribution 6.9. When the correlation factor is included the distribution is smeared out and the orbitals are not as sharp.

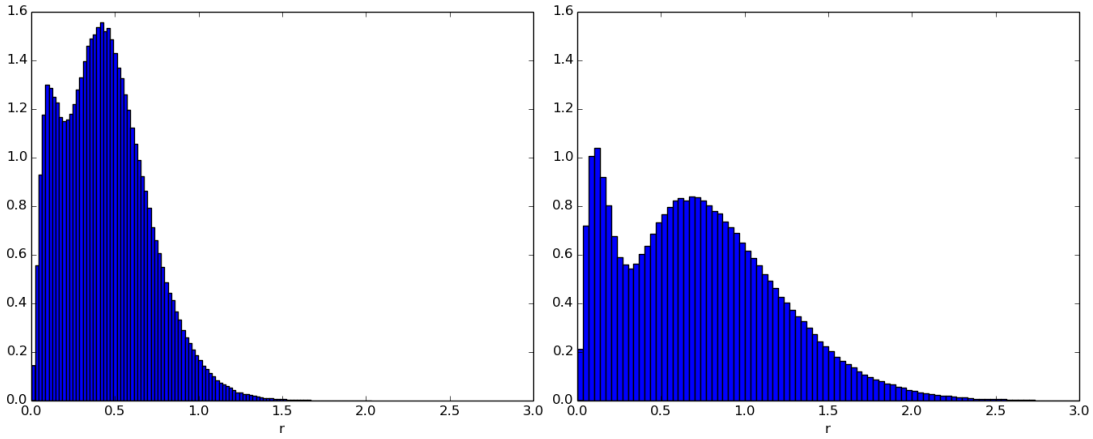


Figure 6.10: Radial densities of an Neon atom simulated with different trial functions. The left plot is with hydrogenic wave functions and the right plot is with hydrogenic wave functions with a Jastrow factor and optimized parameters.

The neon atom

As with helium and beryllium there is a drop in the energy as we introduce a variational parameter. The trial functions for neon include all the three orbitals and the distribution, presented in Fig. 6.10 is contracted closer to the nucleus than in the other atoms. Here we can see the largest effect due to the inclusion of the correlation factor, both in the ground state energy and the radial distribution. This is because the electrons are closer together than in the other atoms due to the higher charge of the nucleus.

The hydrogen molecule

Density plots for the hydrogen molecule H_2 are presented in Fig. 6.11. For this molecule, which is comparable to a Helium atom with the protons separated, the radial distribution from the mass center appears slightly more smeared out than in the Helium atom because the protons are slightly apart. In the one-body density plot, sliced about the xz-plane, the electron density around the nuclei were higher.

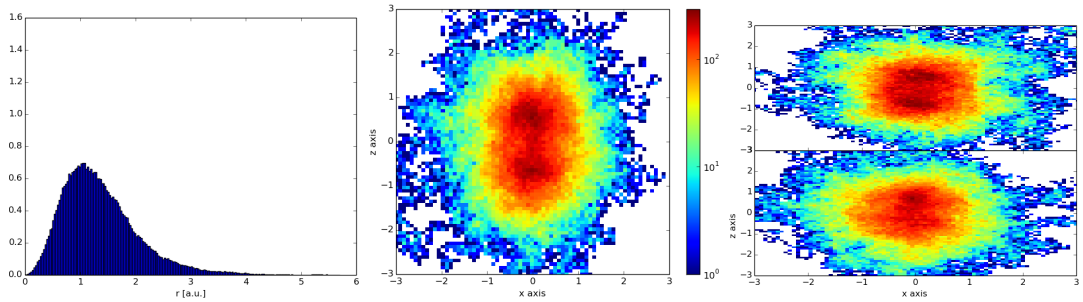


Figure 6.11: On the left the the radial distribution of the hydrogen molecule is presented, where both nuclei are placed $R = 1.40$ atomic units way from each other on the z axis. In the center a slice of the density in the xz plane with a width of 1.0 atomic units is shown. On the right one-body densities of each of the electrons are shown. Both the nuclei are visible on the one-body density plot as a denser concentration.

The beryllium molecule

For the beryllium molecule, Be_2, shown in Fig. 6.12, the concentration of the one-body density around the nuclei is sharper than in the hydrogen molecule, which is likely caused by the higher charge of the nuclei. In the beryllium atom we also see that the electron density is higher closer to the nucleus. The one-body density plots here are not to be completely trusted because there is a bug in the implementation of the molecule which causes a too low binding energy, but much of the expected physics still seem to be captured by the VMC computation. By following only one of the electrons in the one-body densities presented in Fig. 6.12 we can see that, compared to the Hydrogen molecule, the nuclei in the molecule trap the electrons more efficiently.

6.1.6 Using Gaussian Type Orbitals

For larger systems it is cumbersome to use Slater Type Orbitals, as has been done until now. An alternative, which should ease speed up the overall calculations, is using Gaussian Type Orbitals. Using GTOs the energies calculated using Slater Type Orbitals are nearly reproduced, although using GTOs generally does not reach quite as low energies. This is due to the GTOs being an imperfect representation of the STOs.

Using GTOs in stead of STOs is also considerably slower, as seen in 6.6. This can be due to inefficient code, and there is also room for improvements by calculating analytical derivatives of the GTOs. However using GTOs reduces the variables to

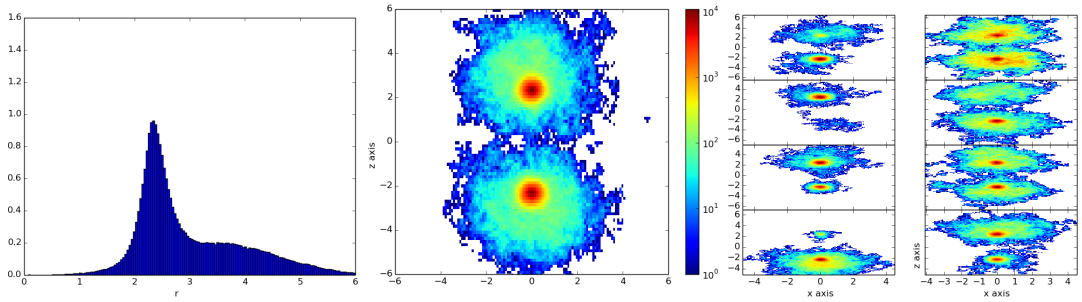


Figure 6.12: On the left the radial distribution of the beryllium molecule is presented, where both nuclei where placed $R = 4.63$ atomic units away from each other on the z axis. In the center the one-body density of a slice of density in the the xz plane with with of 1.0 atomic units is shown. On the right one-body densities of each of the electrons is shown. The concentration of the one-body density is very sharp around the two nuclei.

only β, thus we don't have to look for energy minima by varying α. This means that it is possible to easily use a gradient method to more efficiently find the variable that gives minimum energy. By optimizing the GTO calculations they should be more comparable with the STO case. And by reducing the variational parameters to only β introduces the possibility to examine much larger systems.

Atom	E_{VMC}	Var.	$E_{ref}^{(a)}$	$E_{ref}^{(b)}$	GTO [s]	STO [s]
He	-2.85482	0.004	-2.9037	$-2.9036(2)$	15.0	6.5
Be	-14.0182	0.002	-14.667	$-14.657(2)$	48321	4141
Ne	-113.542	0.498	-128.928	$-128.765(4)$	2821	203

Table 6.6: Comparison of energies found using bisection method with the refenrence energy [Koput, 2011] [Binkley and Pople, 1975] and comparison of the time used running the computation with the given number of cycles using GTOs and STOs.

6.2 Quantum Dots

The attention concerning quantum dots has been on studying one-body densities for quantum dots consisting of different magic numbers of electrons, and with different frequencies. Ground state energies have also been calculated, and are compared to energies calculated with other many-body methods.

6.2.1 Verification and validation

It is important to test the program, to see if all parts work as intended. This is done by running a simulation of a system of whitch the solution is already known. A great way to test if the program indeed gives the results it should is by testing against a case where there is an exact solution.

As a first test we run a simulation where the interaction is neglected. In VMC electron-electron interaction is accounted for by a Coulomb part, and the Jastrow factor. So by ignoring these in our calculations, we can run simulations without interaction. In this case the exact wave function is represented by the trial wave function. The expected energies without interaction are the sum over each occupied orbital, and the energy becomes that of Eq. (4.1.1).

As we see in Tab. 6.7 the VMC solver yields the expected results. This seemingly simple test actually tests most parts of the VMC machinery. Getting positive results means that vital parts like the `VMCSolver`-class, which runs the Monte Carlo cycles with Metropolis sampling, the class handling the Slater determinant, and the class handling the derivatives all work properly. It also means that the orbitals, generated by a Python script, are implemented correctly. However, as we have neglected the interaction part, some parts are not tested. We therefore need a more thorough test.

The next step is to test more parts of the solver by including electron-electron interaction. If we calculate the interaction while ignoring the Jastrow-factor, we should get an energy similar to the energy in the Hartree-Fock case, given by Eq. (2.1.5). By neglecting the Jastrow-factor the variational Monte Carlo solver works as a numerical integrator, using the Slater determinant, and the resulting energy is thus an approximation to the energy which can be calculated manually with Eq. (2.1.5). The results are presented in Tab. 6.8. With the exception of the two-particle case, all the energies calculated with VMC without the Jastrow-factor are slightly higher than then ones calculated using Hartree-Fock. However they are all very close to the HF-energies, which is what was required.

The remaining part of the solver is the inclusion of electron-electron correlation in the calculations. Correlations are implemented with the so-called Jastrow-factor in the trial function, which relies on parts of the derivatives-, orbitals-, and Slater determinant-classes. As we now include correlations in our calculations, there is no way to analytically calculate the resulting energy. Therefore we have to check our results against other, independent calculations of the same system as refenrence.

2D					3D				
ω	N	α	E_{VMC}	E_0	ω	N	α	E_{VMC}	E_0
0.5	2	1.0	1.0	1	0.5	2	1.0	1.5	1.5
1.0		1.0	2.0	2	1.0		1.0	3.0	3
0.5	6	1.0	5.0	5	0.5	8	1.0	9.0	9
1.0		1.0	10.0	10	1.0		1.0	18.0	18
0.5	12	1.0	14.0	14	0.5	20	1.0	30.0	30
1.0		1.0	28.0	28	1.0		1.0	60.0	60
0.5	20	1.0	30.0	30	0.5	40	1.0	75.0	75
1.0		1.0	60.0	60	1.0		1.0	150.0	150
0.5	30	1.0	55.0	55					
1.0		1.0	110.0	110					
0.5	42	1.0	91.0	91					
1.0		1.0	182.0	182					
0.5	56	1.0	140.0	140					
1.0		1.0	280.0	280					

Table 6.7: Results from VMC calculations without electron-electron interaction. N is number of particles used in the quantum dot, and ω is the frequency of the harmonic oscillator. Results for both two dimensional cases and three dimensional cases are tested. For reference the exact energies are given in column E_0.

N	E_{VMC}	E_{HF}
2	3.17151	3.25331
6	20.8133	20.71922
12	67.1167	66.91132
20	158.446	158.0043

Table 6.8: Energies computed with VMC without the Jastrow-factor compared to the Hartree-Fock energies calculated with Eq. (2.1.5).

6.2.2 Ground state energies

Ground state energies of quantum dots are presented in table 6.9. They are compared to results from similar studies using diffusion Monte Carlo, Similarity Renormalization Group Theory, Coupled Cluster singles and doubles, and full configuration interaction.

The energies calculated are generally higher than the energies presented from similar studies. This can be due to the fact that the variational Monte Carlo method is unable to reach as low energies and the same accuracy as for example diffusion Monte Carlo or the other methods presented. Another source for the error may be insufficiency in the search for optimal variational parameters because of limited time and resources, especially for quantum dots consisting of a higher number of electrons. Nonetheless, the energies found using the variational Monte Carlo method are in fairly good agreement with the reference energies.

A case where the variational Monte Carlo method does better than the reference is against the full configuration interaction for 12 particles. This is due to the number of shells above the highest filled shell used in the FCI calculations, which is low in this case, leading to inaccurate results.

For low number of electrons in the quantum dot we see that the energies are in good agreement with the references, with results accurate to up to two decimals for the two-electron case. However the error grows with increasing number of electrons in the system.

Next we look at the ratio between the potential and kinetic energy as a function of $\hbar\omega$. The ratios are presented in Fig. 6.13. Unfortunately the results here are somewhat jumpy, despite using plenty of Monte Carlo cycles. However it still shows that when we lower the frequency, ω, the ratio breaks down, and the potential energy becomes dominating. With lower frequency the electrons no longer possess as much kinetic energy, and the repulsive forces makes the potential energy the predominant energy.

The relationship between the kinetic and potential energy is related to what in classical mechanics is known as the virial theorem. For quantum mechanical systems the virial theorem gives the relation between the kinetic and potential energy operators, $\hat{\mathbf{T}}$ and $\hat{\mathbf{V}}$, with the relation

$$\hat{\mathbf{V}}(\mathbf{r}) \propto r^{\gamma} \rightarrow \langle \hat{\mathbf{T}} \rangle = \frac{\gamma}{2} \langle \hat{\mathbf{V}} \rangle, \tag{6.2.1}$$

first shown by Fock [Fock, 1930a]. Thus two systems which have the same ratio of kinetic to potential energy will follow the same effective potential and similar eigenstates.

N	ω	E_{VMC}	$E_{ref}^{(a)}$	$E_{ref}^{(b)}$	$E_{ref}^{(c)}$	$E_{ref}^{(d)}$
2	0.01	0.0797255(1)	-	0.0738{23}	0.07373505 {19}	0.073839(2)
	0.1	0.45156772(1)	-	0.4408{23}	0.44079191 {19}	0.44079(1)
	0.28	1.0264107(2)	0.99263{19}	1.0217{23}	1.0216441 {19}	1.02164(1)
	0.5	1.6633891(2)	1.643871{19}	1.6599{23}	1.6597723 {19}	1.65977(1)
	1.0	3.0010648(3)	2.9902683{19}	3.0002{23}	3.0000001 {19}	3.00000(1)
6	0.1	3.6712647(1)	3.49991 {18}	3.5805 {22}	3.551776 {9}	3.55385(5)
	0.28	7.7496318(1)	7.56972 {18}	7.6254{22}	7.599579{6}	7.60019(6)
	0.5	11.956646(2)	11.76228 {18}	11.8055 {22}	11.785915 {6}	11.78484(6)
	1.0	20.376948(2)	20.14393 {18}	20.1734 {22}	20.160472 {8}	20.15932(8)
12	0.1	12.568683(1)	12.2253 {17}	12.3497 {21}	12.850344 {3}	12.26984(8)
	0.28	26.056874(1)	25.61084 {17}	25.7095{21}	26.482570 {2}	25.63577(9)
	0.5	39.684993(2)	39.13899 {17}	39.2194 {21}	39.922693 {2}	39.1596(1)
	1.0	66.660247(2)	65.68304 {17}	65.7399 {21}	66.076116 {3}	65.7001(1)
20	0.1	33.959997(1)	29.95345 {16}	30.2700 {8}	-	29.9779(1)
	0.28	62.958144(1)	61.91368 {16}	62.0676{20}	-	61.9268(1)
	0.5	95.405501(1)	93.86145 {16}	93.9889 {20}	-	93.8752(1)
	1.0	158.4896(2)	155.8665 {16}	155.9569 {20}	-	155.8822(1)
30	0.1	-	60.43000 {15}	61.3827{9}	-	60.4205(2)
	0.28	-	123.9733 {15}	124.2111{9}	-	123.9683(2)
	0.5	194.11612(1)	187.0408 {15}	187.2231 {19}	-	187.0426(2)
	1.0	313.26857(2)	308.5536 {15}	308.6810 {19}	-	308.5627(2)

Table 6.9: Ground state results for N-electron quantum dots in two dimensions, with frequency ω. Refs. (a): [Reimann, 2013] (using Similarity Renormalization Group Theory), (b): [Hirth, 2012] (using Coupled Cluster singles and doubles), (c): [Olsen, 2012] (using full configuration interaction), (d): [Høgberget, 2013] (using diffusion MC). Numbers in curly brackets represent the number of shells above the highest filled shell, referred to as the Fermi-level [Shavitt and Bartlett, 2009], which has been used to construct the basis.

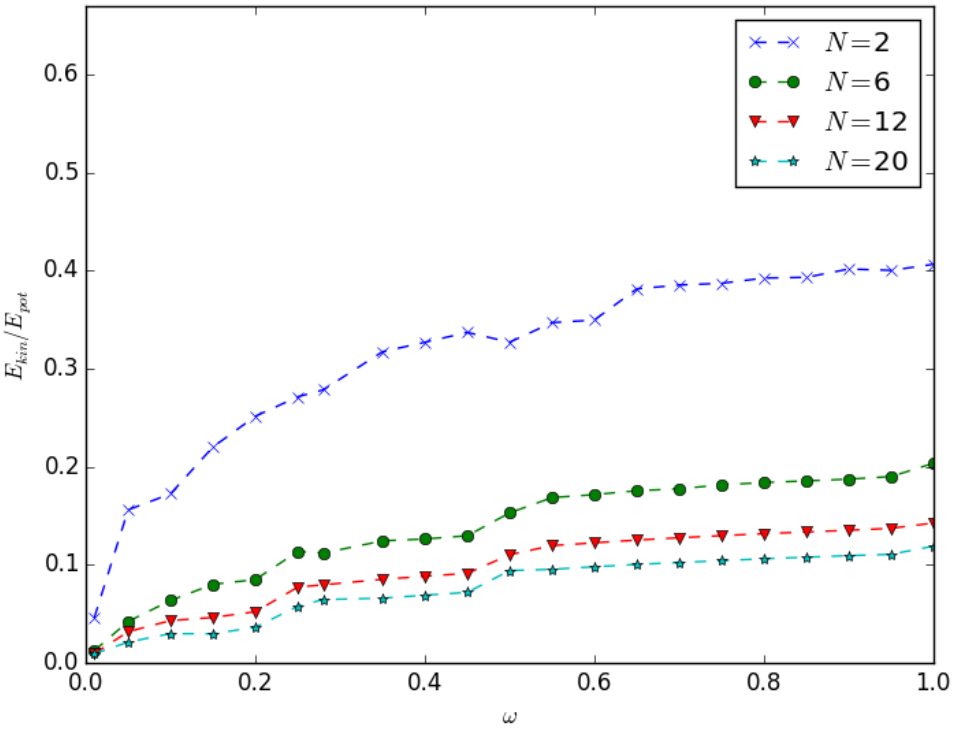


Figure 6.13: Ratio of $\langle T \rangle / \langle V \rangle$, that is the ratio of the expectation value for the kinetic energy and the expectation value for the potential energy, for quantum dots consisting of 2, 6, 12, and 20 electrons.

N	ω	E_{VMC}	$E_{ref}^{(a)}$	$E_{ref}^{(b)}$
2	0.1	0.5033689(2)	0.5	0.499997(3)
	0.28	1.2037916(2)	-	1.201725(2)
	0.5	2.0018226(2)	2.0	2.000000(2)
	1.0	3.7355844(3)	-	3.730123(3)
8	0.1	5.7612847(1)	-	5.7028(1)
	0.28	12.320225(2)	-	12.1927(1)
	0.5	19.043405(2)	-	18.9611(1)
	1.0	33.317234(2)	-	32.6680(1)
20	0.1	27.603613(1)	-	27.2717(2)
	0.28	56.829274(2)	-	56.3868(2)
	0.5	86.667247(2)	-	85.6555(2)
	1.0	145.13357(2)	-	142.8875(2)

Table 6.10: Ground state results for N-electron quantum dots in three dimensions, with frequency ω. Refs. (a): [Taut, 1993] (using analytical solutions) (b): [Høgberget, 2013] (using diffusion Monte Carlo).

Ground state energies for quantum dots in three dimensions are presented in table 6.10. Quantum dot systems in three dimensions are less studied than their two-dimension counterpart, and as such there are less reference energies to test against. We see however, against the reference energies listed, that as with the two-dimensional cases the ground state energies computed with the variational Monte Carlo method are in fairly good agreement with the reference energies, but are generally higher. With a two-particle system the energies are accurate to two decimal places compared to the references, which is in agreement with the indicated accuracy. However the gap between the computed energy and the reference energies grows with the number of electrons in the system.

6.2.3 One-body densities

In Fig. 6.14 the radial one-body densities for some of the quantum dots are presented. For the two-particle case there is a single peak, slightly smeared with increasing radius. For six particles the distribution is more smeared and the peak is shifted to a higher radius, because of the inclusion of a second filled shell. We see in the 12, 20, and 30 particle cases that the peaks are shifted further with the inclusion of more filled shells. Thus, as more shells are filled, the density distribution gains extrema points.

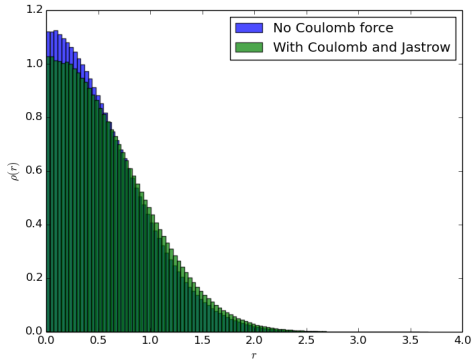
Additionally the densities without Coulomb interaction are shown. For two particles the difference is small as effects from interaction are minimal with so few particles. However it shows a slightly sharper peak for the non-interacting case. As we add more filled shells we see that the same effect, and with more particles the effect becomes more pronounced. Removing the Coulomb force from the calculations thus makes the distribution become more localized and shifted towards the center, due to the reduced potential energy between the energies, meaning that the electrons can be situated closer together.

To study the importance of the Coulomb forces in the interaction, a factor $e^{-\mu r}$ is introduced, where μ is a constant. This is called the Yukawa interaction, named after its creator [Yukawa, 1935]. With this modified interaction the reach of the Coulomb force, which usually is virtually infinite, is killed off by the exponential factor. Looking at the difference in Fig. 6.15 it is evident that the density is shifted towards the center when using the Yukawa factor. Because the Coulomb interaction is killed off for larger distances, each particle no longer feels as strong repulsive force from the other electrons. They can thus be closer together, as the density shows.

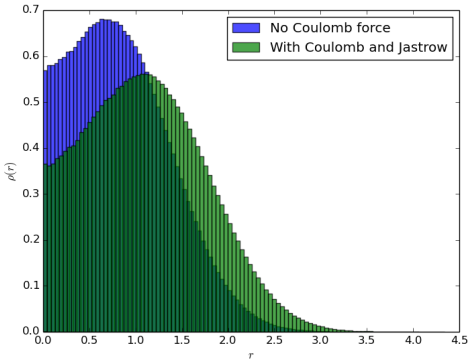
One-body densities for quantum dots in two dimensions with a frequency $\omega = 1$ are presented in Fig. 6.16. It is easy to see how the densities correspond to the radial densities. There is a single peak in the center in the two particle case, corresponding to the single filled shell. Meanwhile in the six particle case there is no peak in the center, but a ring of high density surrounding the center, corresponding to the single peak in the radial density. Another such ring can be seen in the 12 particle case, together with a high density in the center. There is a clear pattern emerging, with a wavelike behaviour with increasing numbers of filled shells. The high density center in the two electron case can thus be seen in the 12 electron case, which again can be seen in the center of the 30 electron case (although the center in the 30 electron case is not as defined). The same can also be seen for the six and 20 particle cases. This similar shape in the middle is a result of the systems being in energetically favourable configurations.

One-body densities for lower frequencies, presented in Fig. 6.17, shows similar shapes, but with larger radial size. As we lower the frequency, the radial size grows. For the $\omega = 0.1$ cases the one-body densities for higher number of particles does not exhibit as clear wave-like shapes anymore due to a limited number of samplings in the calculations.

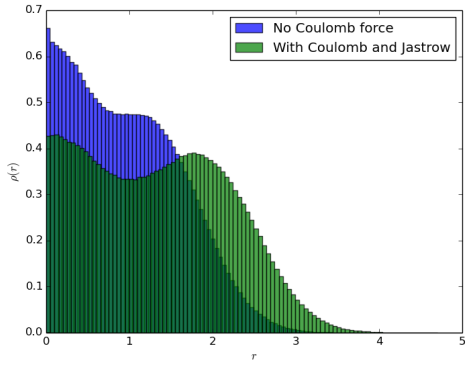
An interesting feature arising in the one-body densities is that for very low frequencies the densities change structure. In the two-particle case, with $\omega = 0.01$, there is no longer a peak in the middle, the one-body density resembles more the six-particle case with a decreased density in the center. Similarly in the



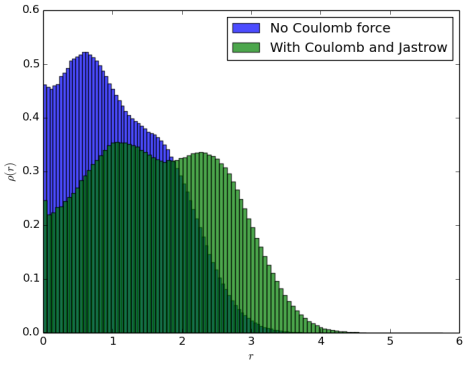
(a) $N = 2$



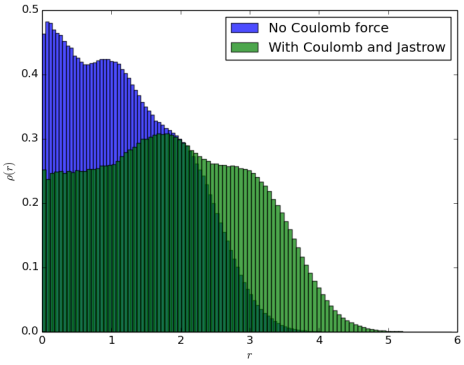
(b) $N = 6$



(c) $N = 12$



(d) $N = 20$



(e) $N = 30$

Figure 6.14: Radial one-body densities of quantum dots in two dimensions with frequency $\omega = 1$, consisting of 2, 6, 12, 20, and 30 particles, with and without electron-electron interaction.

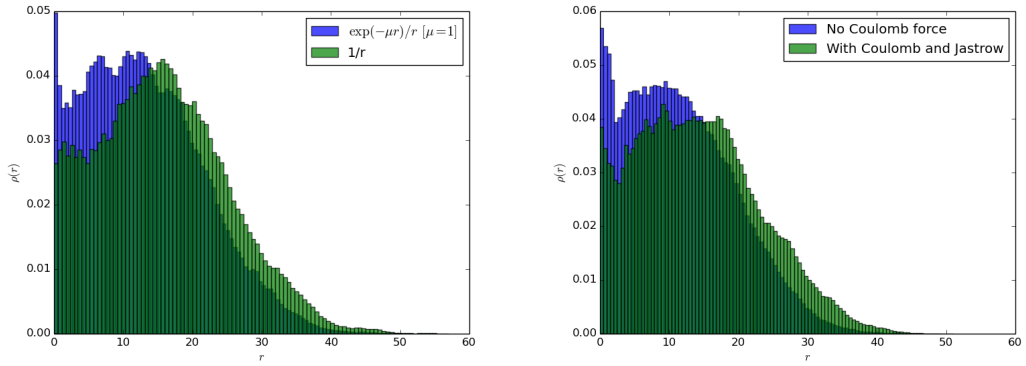


Figure 6.15: Radial one-body density of a two-dimensional quantum dot consisting of six particles and frequency $\omega = 0.01$. To the left is a comparison with and without a Yukawa factor in the Coulomb interaction. To the right is a comparison with and without Coulomb interaction.

six-particle case with $\omega = 0.01$ there is a peak in the center, unlike for higher frequencies, making it resemble the 12-particle case.

It is evident from the one-body densities that lowering the frequencies makes the density more smeared out, from the increasing size as frequency is lowered. The density thus becomes more even and more localized as the frequency is lowered, which suggests that the electrons are, on average, more evenly spread across the shell structure. This interpretation is further supported by Fig. 6.13, where we see the ratio between the kinetic and potential energy break down as the frequency is lowered, suggesting that the potential energy becomes dominating.

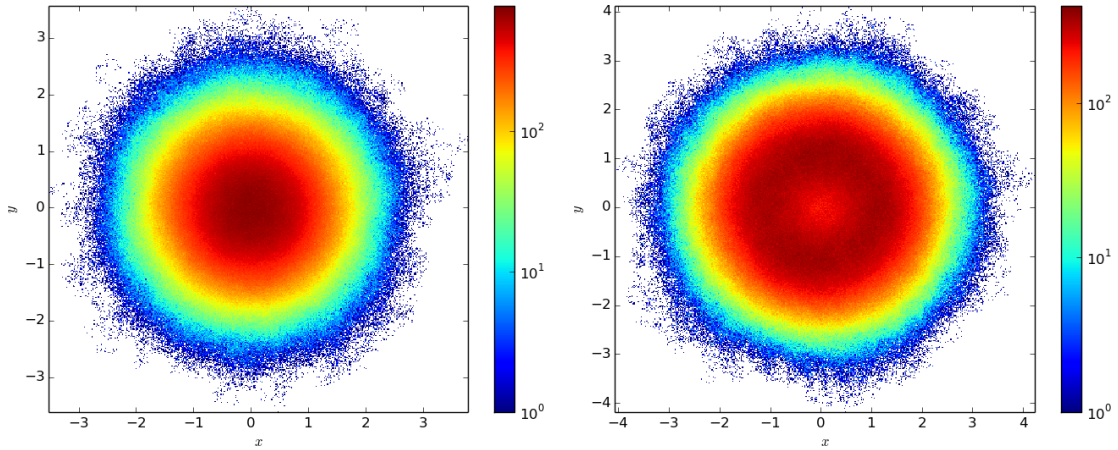
In three dimensions the one-body densities, presented in Fig. 6.18, are evidently quite similar to their two-dimensional counterparts, exhibiting the same wave-like distribution of the density. However the similarity only holds for the same number of closed shells, meaning that for example the 8 particle case in three dimensions corresponds to the 6 particle case in two dimensions, as presented in Fig. 6.16, because they both have two filled shells.

Comparing two and three dimensions it is therefore clear that for two particles we have a single peak in both dimensions, for 6 particles in two dimensions and 8 in three dimensions we have a single band of high density surrounding a low point, and for 12 particles in two dimensions and 20 in three dimensions we have a peak in the center and a band of high density surrounding it.

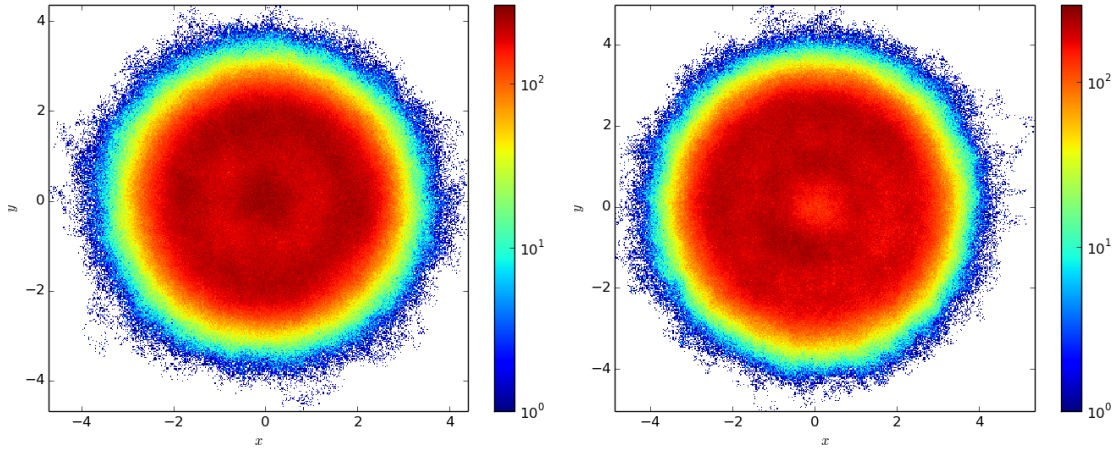
With a very low frequency, $\omega = 0.01$, it is from Fig. 6.17 evident that the density becomes localized and that the particles gets evenly distributed over the shells. That is, for example for a very low frequency for the six particle case, the shape

containing a single band of high density shows an elevated density in the center, from the lowest filled shell. Looking at an example for the three dimensional case, Fig. 6.19, also with two filled shells and a frequency of $\omega = 0.01$, the same effect does not apply in the three dimensional systems studied; there are no signs of an elevated density in the center. We thus get a breaking of symmetry. However, from this we can only conclude that the three-dimensional system does not exhibit localization for frequencies down to $\omega = 0.01$ when studied by this method.

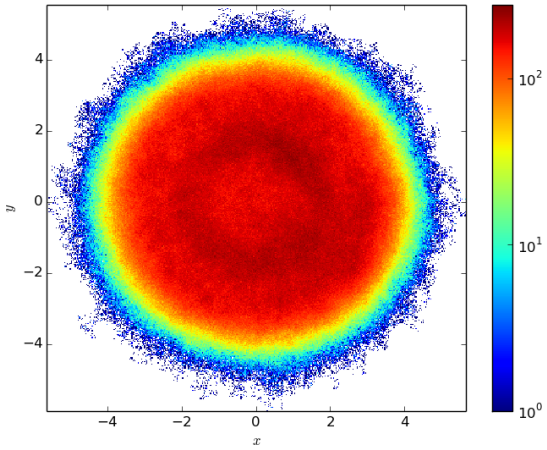
For low frequencies in two dimensions, where the density becomes localized and evenly distributed in shells, it is apparent that the quantum dots become crystallized in what is called a Wigner crystal, named after Wigner who predicted them [Wigner, 1934]. This effect is expected for the quantum dots for low electron densities, where the potential energy dominate over the kinetic energy [Yannouleas and Landman, 2007].



(a) $N = 2$ **(b)** $N = 6$

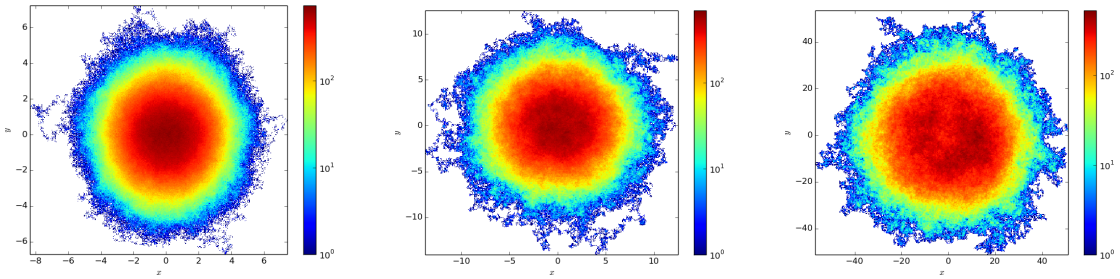


(c) $N = 12$ **(d)** $N = 20$



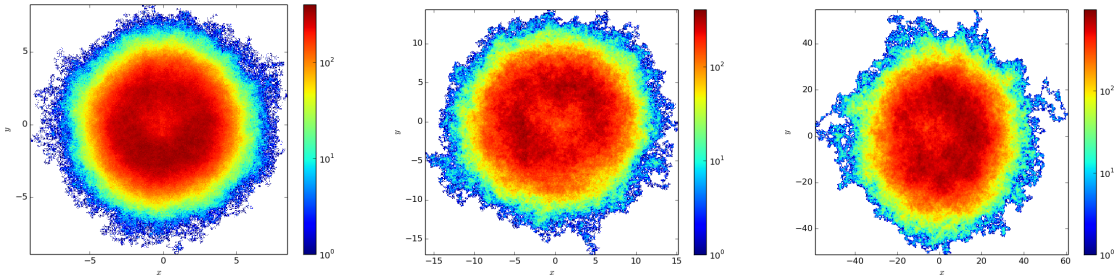
(e) $N = 30$

Figure 6.16: One-body density with $\omega = 1.0$



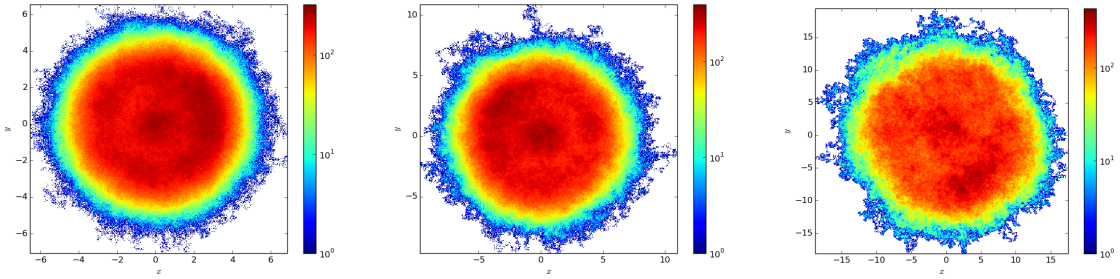
$\omega = 0.28$ $\omega = 0.1$ $\omega = 0.01$

(a) One-body densities for $N = 2$



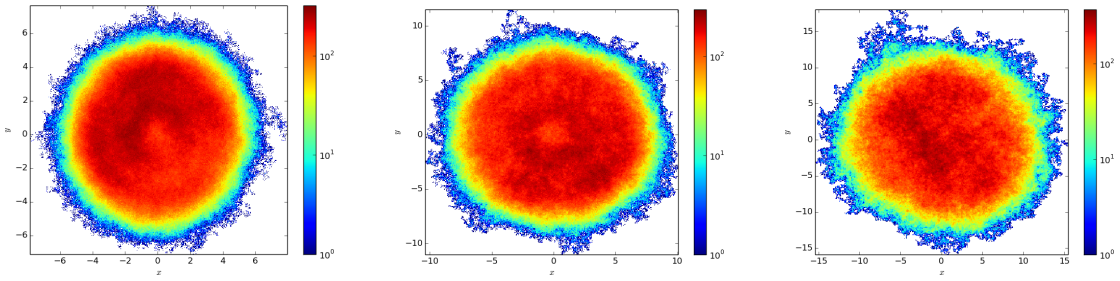
$\omega = 0.28$ $\omega = 0.1$ $\omega = 0.01$

(b) One-body densities for $N = 6$



$\omega = 0.5$ $\omega = 0.28$ $\omega = 0.1$

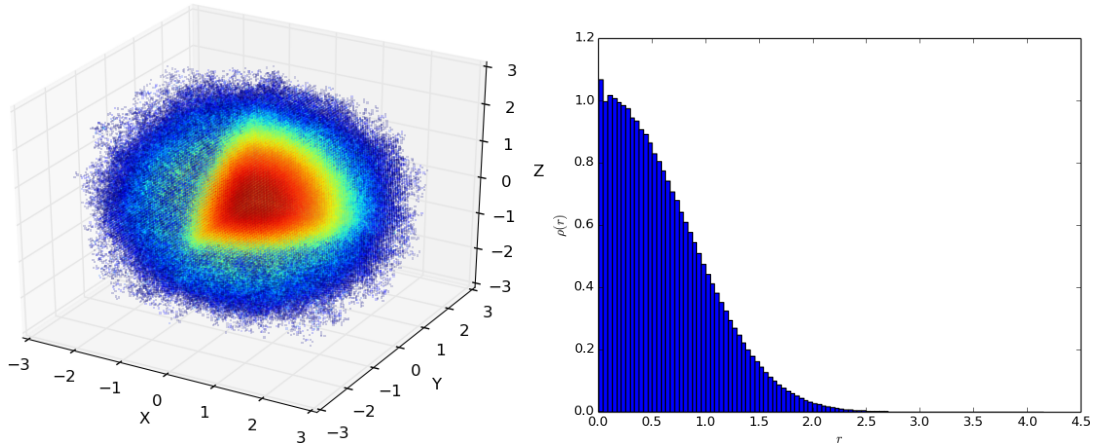
(c) One-body densities for $N = 12$



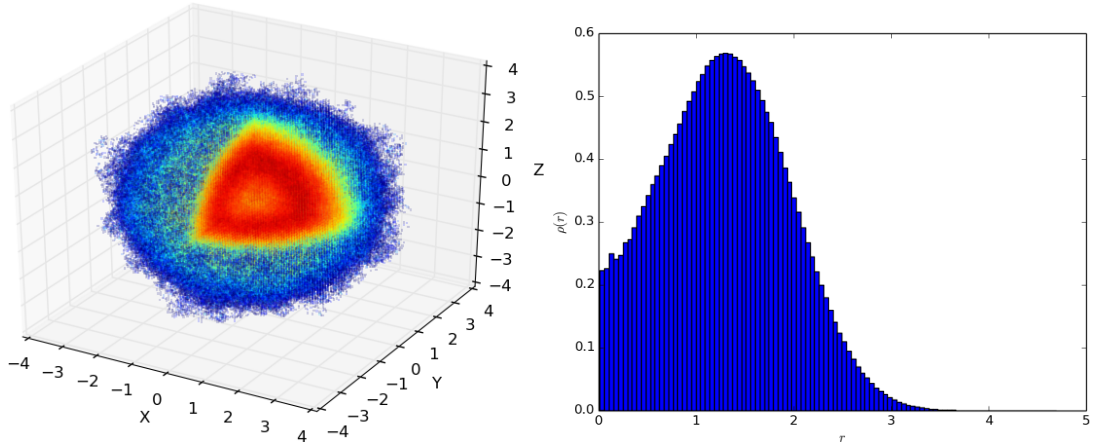
$\omega = 0.5$ $\omega = 0.28$ $\omega = 0.1$

(d) One-body densities for $N = 20$

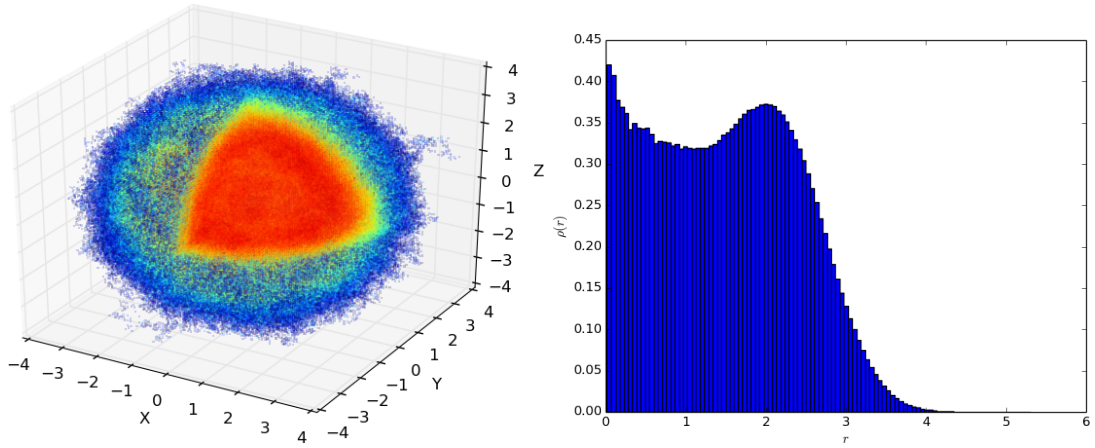
Figure 6.17: One-body densities for different ω-values



(a) $N = 2$



(b) $N = 8$



(c) $N = 20$

Figure 6.18: One-body densities of quantum dots in three dimensions, consisting of 2, 8, and 20 electrons, with frequency $\omega = 1$. One eight of the density-sphere has been removed to reveal the distribution of density to the center. The radial density distribution for each case is also presented on the right.

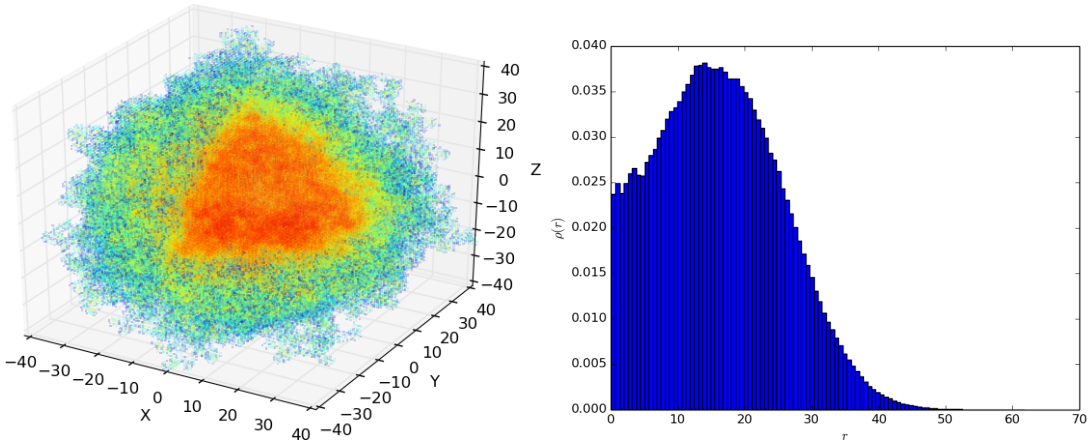


Figure 6.19: One-body density of a three-dimensional quantum dot consisting of 8 electrons, with frequency $\omega = 0.01$. One eight of the density-sphere has been removed to reveal the distribution of density to the center. On the right the corresponding radial density is shown.

Chapter 7

Conclusion

In this thesis the main aim has been to develop a variational Monte Carlo solver. Furthermore, emphasis has been put on making the solver as general as possible, thus making it possible to study a wide variety of systems. To achieve this the solver was written in C++ using object-oriented programming. To ease the handling of vectors and matrices being handled by the solver it utilizes the linear algebra library Armadillo [Sanderson, 2010], which provides efficient methods for this.

As presented in section 4.2.1, modelling atoms is fairly simple and only a handful of orbitals are used to create the trial functions. With quantum dots however, harmonic oscillator orbitals are used, and with the aim of using a great number of particles in the calculations, in both two and three dimensions, a way to automate the creation of the orbitals was needed. This was done using the python library SymPy, with which the orbitals could easily be created, their gradient and laplacian functions could be calculated automatically, and source files used by the variational Monte Carlo solver could seamlessly be generated.

A way to handle the immense amounts of data produced by the solver program was also needed. For this purpose a script was created, which converted the data files with the size of numerous gigabytes to something visualizable and with a more manageable size.

Over the years many Master projects have had focus on solving quantum mechanical many-body systems, using different methods, like the Multi-Configuration time-dependent Hartree-Fock method [Skattum, 2013], post Hartree-Fock methods like coupled cluster theory [Hirth, 2012], full configuration interaction theory [Olsen, 2012], and quantum Monte Carlo methods [Høgberget, 2013]. The focus in this thesis has thus not been something entirely new, but nonetheless gave

interesting insight to quantum mechanical many-body systems. Creating a flexible solver allowed the study of many systems: atoms with a size up to neon, consisting of 10 particles, molecules with a size up to diberyllium, consisting of 12 particles, and quantum dots with even greater size, in both two and three dimensions.

By successfully implementing and studying a wide variety of quantum mechanical systems the generality and flexibility of the VMC solver is demonstrated. This shows that the solver that has been developed has reached the goal which was set, by performing calculations on atoms, molecules, and quantum dots.

The variational Monte Carlo method gives fairly accurate results, both for atoms and molecules, and also for quantum dots. With atoms it is demonstrated how the so-called Jastrow factor efficiently deals with the correlation factors, thus yielding energies comparable with those computed using more exact methods. In the case of helium the relative error is less than 0.5 percent compared to the optimized theoretical reference energies. For neon the relative error is just under 1 percent, and for the largest molecule used, the relative error is about 6.5 percent, which is fairly good. Due to limited resources the larger atoms and molecules had to be run with fewer Monte Carlo cycles, so it is probable that the results could have been better. There is however another limitation, which is the simple trial function used. But considering this with the reduced number of Monte Carlo cycles, the results are very good.

For all of the atoms the radial distribution of the wave functions showed traces of the hydrogenic wave functions they were constructed from. As the binding energy of the atoms during the VMC computations are close to the experimental values, it is likely that the atomic structure of the real atoms resemble the atomic structures of the VMC simulations.

Results for quantum dots are for a comparable number of particles even more accurate than for atoms and molecules. For the two particle case the relative error is less than 0.04 percent and less than 0.2 percent for two and three dimensions, respectively. For the quantum dots consisting of the largest number of electrons, 30, the relative error is also low, at about 1.5 percent. Unfortunately the number of quantum dot systems had to be limited due to insufficient time. Despite this the characteristics of the quantum dot system could still be demonstrated.

In comparing the radial distribution of quantum dots in two and three dimensions which have the same number of closed shells the similarity is striking, as long as the frequency is high. For very low frequencies there is a break in symmetry, where the two dimensional case shows electrons that become strongly localized , while the three dimensional case does not exhibit the same degree of localization.

From the results presented in this thesis it is clear that the VMC method computes fairly accurate results by utilizing the Jastrow factor in computing correlations. The low error in the ground state energies shows that the results are very good considering the amount of resources that went into calculating them. Many of the calculations for a smaller number of particles could be run on a single computer in a handful of hours. For larger systems however a cluster computer is used.

Considering that a relatively simple and very well tested method was used to obtain the results presented it is easy to see why VMC is so popular for solving quite complicated problems, like calculating the ground state energies and one-body densities of non-trivial quantum mechanical systems. There is however a trade off in that the trial function used is a limiting factor on the accuracy. Choosing better trial functions would thus result in more accurate results, especially for larger systems like molecules.

Because the variational Monte Carlo method uses a statistical approach to the quantum mechanical many-body problem, it is maybe not as efficient as some other methods, like for example the Hartree-Fock method or the Density functional theory method [Kohn and Sham, 1965]. It does however take into account correlation, unlike the Hartree-Fock method. However other, more efficient methods exist, like the diffusion Monte Carlo method.

Perspectives for future work

Looking ahead, to improve the Variational Monte Carlo program for atoms, more analytical functions for the Slater Determinant and for the derivatives of the GTO functions could be calculated. These parts are used a lot during the program and switching from numerical to analytical solution would speed things up. Further, as the Gaussian Type Orbitals gave slightly disappointing energies, another basis set, which is larger than the 3-21G basis, could be implemented, such as the 6-311G basis. This would give the GTO functions a shape that more closely resembles Slater Type Orbitals and therefore give more accurate energies.

The solver could also be made more efficient, and given more time, systems of greater size could be studied. A possibility is to study a higher scale, that is studying molecules. The break in symmetry of quantum dots in two and three dimensions could also be studied in greater detail. Our code is also flexible enough to accommodate double-well quantum dots, of great interest for computing computing. The implementation of a diffusion Monte Carlo solver is also something which can be added to the code, providing thereby almost exact ground state energies.

Appendices

Appendix A

Closed expression for noncorrelation helium trial function

A.1 Derivation of local energies, using radial coordinates

The local energy of is dependant on the Hamiltonian and the wave function describing the system, the Hamiltonian incorporates both a kinetic energy part given by $\nabla_i^2/2$ for each particle and a potential energy part given by Z/r_i and $1/r_{ij}$, where Z is the charge of the center, r_i is the distance for electron i to the atom center and r_{ij} is the distance between electron l and m. Then the local energy is thus given by

$$E_L = \sum_{i,i<j} \frac{1}{\Psi_T(\mathbf{r_i}, \mathbf{r_{ij}})} \hat{H} \Psi_T(\mathbf{r_i}, \mathbf{r_{ij}}) \tag{A.1.1}$$

$$= \sum_{i,i<j} \frac{1}{\Psi_T(\mathbf{r_i}, \mathbf{r_{ij}})} \left(-\frac{\nabla_i^2}{2} - \frac{Z}{r_i} - \frac{Z}{r_j} + \frac{1}{r_{ij}} \right) \Psi_T(\mathbf{r_i}, \mathbf{r_{ij}}) \tag{A.1.2}$$

$$= \sum_{i,i<j} -\frac{1}{2\Psi_T} \left(\nabla_i^2 \Psi_T \right) - \frac{Z}{r_i} - \frac{Z}{r_j} + \frac{1}{r_{ij}}, \tag{A.1.3}$$

where Ψ_T is the trial wave function. Let us change derivation variables

$$-\frac{1}{2\Psi_T}\left(\nabla_i^2\Psi_T\right) = \sum_{m=1}^{3} -\frac{1}{2\Psi_T}\left(\frac{\partial^2\Psi_T}{\partial x_m^2}\right)_i \tag{A.1.4}$$

$$= \sum_{m=1}^{3} -\frac{1}{2\Psi_T}\left(\frac{\partial}{\partial x_m}\left(\frac{\partial\Psi_T}{\partial r_i}\frac{\partial r_i}{\partial x_m}\right)\right)_i. \tag{A.1.5}$$

Since $r_i = \left(x_1^2 + x_2^2 + x_3^2\right)^{1/2}$ then

$$\frac{\partial r_i}{\partial x_m} = \frac{\partial\left(x_1^2 + x_2^2 + x_3^2\right)^{1/2}}{\partial x_m} = \frac{x_m}{r_i}, \tag{A.1.6}$$

and we have

$$-\frac{1}{2\Psi_T}\left(\nabla_i^2\Psi_T\right) = \sum_{m=1}^{3} -\frac{1}{2\Psi_T}\left(\frac{\partial}{\partial x_m}\left(\frac{\partial\Psi_T}{\partial r_i}\frac{x_m}{r_i}\right)\right)_i \tag{A.1.7}$$

$$= \sum_{m=1}^{3} -\frac{1}{2\Psi_T}\left(\frac{\partial^2\Psi_T}{\partial x_m\partial r_i}\frac{x_m}{r_i} + \frac{\partial\Psi_T}{\partial r_i}\frac{\partial}{\partial x_m}\left(\frac{x_m}{r_i}\right)\right)_i. \tag{A.1.8}$$

For different values for m the last partial derivative in the last term becomes

$$\frac{\partial}{\partial x_1}\left(\frac{x_1}{\left(x_1^2 + x_2^2 + x_3^2\right)^{1/2}}\right) = \frac{x_2^2 + x_3^2}{r_i^3}, \tag{A.1.9}$$

so summing over all values for m it should sum up to

$$\frac{2(x_1^2 + x_2^2 + x_3^2)}{r_i^3}.$$

We thus get

$$-\frac{1}{2\Psi_T}\left(\nabla_i^2\Psi_T\right) = -\frac{1}{2\Psi_T}\left(\frac{\partial^2\Psi_T}{\partial r_i^2}\frac{x_1^2 + x_2^2 + x_3^2}{r_i^2} + \frac{\partial\Psi_T}{\partial r_i}\frac{2(x_1^2 + x_2^2 + x_3^2)}{r_i^3}\right)_i \tag{A.1.10}$$

$$= -\frac{1}{2\Psi_T}\left(\frac{\partial^2\Psi_T}{\partial r_i^2} + \frac{\partial\Psi_T}{\partial r_i}\frac{2}{r_i}\right). \tag{A.1.11}$$

Now the local energy becomes

$$E_L = \sum_{i,i<j} -\frac{1}{2\Psi_T}\left(\frac{\partial^2\Psi_T}{\partial r_i^2} + \frac{\partial\Psi_T}{\partial r_i}\frac{2}{r_i}\right) - \frac{Z}{r_i} - \frac{Z}{r_j} + \frac{1}{r_{ij}}. \tag{A.1.12}$$

We can apply this to the simple helium trial function with no electronic interaction to obtain the local energy.

Helium: Simple trial function

The simple version of the trial function is only dependant on one parameter α and does not take into account interaction between the two electrons, it is of the form

$$\Psi_T(\mathbf{r_1}, \mathbf{r_2}) = \exp\{-\alpha(r_1 + r_2)\}.$$

Let us set this trial function into the equation for the local energy (A.1.12).

$$E_L = \sum_{i,i<j} -\frac{1}{2\Psi_T}\left(\frac{\partial^2 e^{-\alpha(r_i+r_j)}}{\partial r_i^2} + \frac{\partial e^{-\alpha(r_i+r_j)}}{\partial r_i}\frac{2}{r_i}\right) - \frac{Z}{r_i} - \frac{Z}{r_j} + \frac{1}{r_{ij}} \tag{A.1.13}$$

$$E_L = -\frac{1}{2\Psi_T}\sum_{i=1}^{2}\left(\alpha^2 - \alpha\frac{2}{r_i}\right)\Psi_T - \frac{Z}{r_i} + \frac{1}{r_{ij}} \tag{A.1.14}$$

$$E_L = -\alpha^2 + (\alpha - Z)\left(\frac{1}{r_1} + \frac{1}{r_2}\right) + \frac{1}{r_{12}}. \tag{A.1.15}$$

Appendix B

GTO constants

In the following tables of constants used to combine the contracted Gaussian type orbitals for helium, beryllium, and neon are presented.

1s
0.4579
0.6573

Table B.1: Constants for combining contracted GTOs for helium.

1s	2s
-9.9281e-01	-2.1571e-01
-7.6425e-02	2.2934e-01
2.8727e-02	8.2235e-01
1.2898e-16	5.1721e-16
-2.3257e-19	4.5670e-18
5.6097e-19	-1.1040e-17
1.2016e-16	8.5306e-16
-4.6874e-19	7.0721e-18
1.1319e-18	-1.7060e-17

Table B.2: Constants for combining contracted GTOs for beryllium.

1s	2s	$2p_x$	$2p_y$	$2p_z$
-9.8077e-01	-2.6062e-01	1.1596e-16	-8.3716e-18	-1.9554e-17
-9.3714e-02	2.5858e-01	-2.0106e-16	-9.7173e-17	-7.3738e-17
2.2863e-02	8.1619e-01	-3.2361e-16	1.3237e-16	1.5789e-16
-9.9519e-19	-5.6186e-18	2.7155e-02	-4.0320e-01	3.9171e-01
-1.2125e-18	-2.8615e-16	-5.6207e-01	-2.5833e-02	1.2375e-02
-4.1800e-19	4.6199e-17	9.1139e-03	-3.9180e-01	-4.0392e-01
-1.6696e-19	-4.2405e-18	2.8890e-02	-4.2895e-01	4.1673e-01
1.2125e-18	-2.9426e-16	-5.9797e-01	-2.7482e-02	1.3166e-02
3.8779e-19	5.0519e-17	9.6959e-03	-4.1683e-01	-4.2972e-01

Table B.3: Constants for combining contracted GTOs for neon.

Appendix C

Harmonic oscillator orbitals in two dimensions

Harmonic oscillator orbitals in two dimensions are a combination of hermite polynomials of the x-coordinate, hermite polynomials of the y-coordinate, and an exponential factor. The orbitals are thus created as

$$\phi(\mathbf{r})_{n_x,n_y} = H_{n_x}(kx)H_{n_y}(ky)e^{-\frac{1}{2}k^2r^2}.$$

Here $k = \sqrt{\omega\alpha}$, where ω is the oscillator frequency and α is the variational parameter. In the following the hermite polynomials and resulting orbitals will be listed.

$H_0(kx)$	1
$H_1(kx)$	$2kx$
$H_2(kx)$	$4k^2x^2 - 2$
$H_3(kx)$	$8k^3x^3 - 12x$
$H_4(kx)$	$16k^4x^4 - 48k^2x^2 + 12$
$H_5(kx)$	$32k^5x^5 - 160k^3x^3 + 120$
$H_6(kx)$	$64k^6x^6 - 480k^4x^4 + 720k^2x^2 - 120$
$H_0(ky)$	1
$H_1(ky)$	$2ky$
$H_2(ky)$	$4k^2y^2 - 2$
$H_3(ky)$	$8k^3y^3 - 12y$
$H_4(ky)$	$16k^4y^4 - 48k^2y^2 + 12$
$H_5(ky)$	$32k^5y^5 - 160k^3y^3 + 120$
$H_6(ky)$	$64k^6y^6 - 480k^4y^4 + 720k^2y^2 - 120$

Table C.1: Hermite polynomials used to create the harmonic oscillator orbitals in two dimensions.

$\phi_0 \to \phi_{0,0}$	
$\phi(\mathbf{r})$	1
$\mathbf{i} \cdot \nabla\phi(\mathbf{r})$	$-k^2x$
$\mathbf{j} \cdot \nabla\phi(\mathbf{r})$	$-k^2y$
$\nabla^2\phi(\mathbf{r})$	$k^2(k^2r^2 - 2)$

Table C.2: Harmonic orbital expressions made with $H_0(kx)$ and $H_0(ky)$. Omitted factor $e^{-\frac{k^2}{2}(x^2+y^2)}$.

$\phi_1 \to \phi_{0,1}$	
$\phi(\mathbf{r})$	$2ky$
$\mathbf{i} \cdot \nabla\phi(\mathbf{r})$	$-2k^3xy$
$\mathbf{j} \cdot \nabla\phi(\mathbf{r})$	$-2k(ky - 1)(ky + 1)$
$\nabla^2\phi(\mathbf{r})$	$2k^3y(k^2r^2 - 4)$

Table C.3: Harmonic orbital expressions made with $H_0(kx)$ and $H_1(ky)$. Omitted factor $e^{-\frac{k^2}{2}(x^2+y^2)}$.

$\phi_2 \to \phi_{1,0}$	
$\phi(\mathbf{r})$	$2kx$
$\mathbf{i} \cdot \nabla\phi(\mathbf{r})$	$-2k(kx-1)(kx+1)$
$\mathbf{j} \cdot \nabla\phi(\mathbf{r})$	$-2k^3xy$
$\nabla^2\phi(\mathbf{r})$	$2k^3x(k^2r^2-4)$

Table C.4: Harmonic orbital expressions made with $H_1(kx)$ and $H_0(ky)$. Omitted factor $e^{-\frac{k^2}{2}(x^2+y^2)}$.

$\phi_3 \to \phi_{0,2}$	
$\phi(\mathbf{r})$	$4k^2y^2-2$
$\mathbf{i} \cdot \nabla\phi(\mathbf{r})$	$-2k^2x(2k^2y^2-1)$
$\mathbf{j} \cdot \nabla\phi(\mathbf{r})$	$-2k^2y(2k^2y^2-5)$
$\nabla^2\phi(\mathbf{r})$	$2k^2(k^2r^2-6)(2k^2y^2-1)$

Table C.5: Harmonic orbital expressions made with $H_0(kx)$ and $H_2(ky)$. Omitted factor $e^{-\frac{k^2}{2}(x^2+y^2)}$.

$\phi_4 \to \phi_{2,0}$	
$\phi(\mathbf{r})$	$4k^2x^2-2$
$\mathbf{i} \cdot \nabla\phi(\mathbf{r})$	$-2k^2x(2k^2x^2-5)$
$\mathbf{j} \cdot \nabla\phi(\mathbf{r})$	$-2k^2y(2k^2x^2-1)$
$\nabla^2\phi(\mathbf{r})$	$2k^2(k^2r^2-6)(2k^2x^2-1)$

Table C.6: Harmonic orbital expressions made with $H_2(kx)$ and $H_0(ky)$. Omitted factor $e^{-\frac{k^2}{2}(x^2+y^2)}$.

$\phi_5 \to \phi_{1,1}$	
$\phi(\mathbf{r})$	$4k^2xy$
$\mathbf{i} \cdot \nabla\phi(\mathbf{r})$	$-4k^2y(kx-1)(kx+1)$
$\mathbf{j} \cdot \nabla\phi(\mathbf{r})$	$-4k^2x(ky-1)(ky+1)$
$\nabla^2\phi(\mathbf{r})$	$4k^4xy(k^2r^2-6)$

Table C.7: Harmonic orbital expressions made with $H_1(kx)$ and $H_1(ky)$. Omitted factor $e^{-\frac{k^2}{2}(x^2+y^2)}$.

$\phi_6 \to \phi_{0,3}$	
$\phi(\mathbf{r})$	$8k^3y^3 - 12ky$
$\mathbf{i} \cdot \nabla\phi(\mathbf{r})$	$-4k^3xy(2k^2y^2 - 3)$
$\mathbf{j} \cdot \nabla\phi(\mathbf{r})$	$-4k(2k^4y^4 - 9k^2y^2 + 3)$
$\nabla^2\phi(\mathbf{r})$	$4k^3y(k^2r^2 - 8)(2k^2y^2 - 3)$

Table C.8: Harmonic orbital expressions made with $H_0(kx)$ and $H_3(ky)$. Omitted factor $e^{-\frac{k^2}{2}(x^2+y^2)}$.

$\phi_7 \to \phi_{3,0}$	
$\phi(\mathbf{r})$	$8k^3x^3 - 12kx$
$\mathbf{i} \cdot \nabla\phi(\mathbf{r})$	$-4k(2k^4x^4 - 9k^2x^2 + 3)$
$\mathbf{j} \cdot \nabla\phi(\mathbf{r})$	$-4k^3xy(2k^2x^2 - 3)$
$\nabla^2\phi(\mathbf{r})$	$4k^3x(k^2r^2 - 8)(2k^2x^2 - 3)$

Table C.9: Harmonic orbital expressions made with $H_3(kx)$ and $H_0(ky)$. Omitted factor $e^{-\frac{k^2}{2}(x^2+y^2)}$.

$\phi_8 \to \phi_{1,2}$	
$\phi(\mathbf{r})$	$2kx(4k^2y^2 - 2)$
$\mathbf{i} \cdot \nabla\phi(\mathbf{r})$	$-4k(kx - 1)(kx + 1)(2k^2y^2 - 1)$
$\mathbf{j} \cdot \nabla\phi(\mathbf{r})$	$-4k^3xy(2k^2y^2 - 5)$
$\nabla^2\phi(\mathbf{r})$	$4k^3x(k^2r^2 - 8)(2k^2y^2 - 1)$

Table C.10: Harmonic orbital expressions made with $H_1(kx)$ and $H_2(ky)$. Omitted factor $e^{-\frac{k^2}{2}(x^2+y^2)}$.

$\phi_9 \to \phi_{2,1}$	
$\phi(\mathbf{r})$	$2ky(4k^2x^2 - 2)$
$\mathbf{i} \cdot \nabla\phi(\mathbf{r})$	$-4k^3xy(2k^2x^2 - 5)$
$\mathbf{j} \cdot \nabla\phi(\mathbf{r})$	$-4k(ky - 1)(ky + 1)(2k^2x^2 - 1)$
$\nabla^2\phi(\mathbf{r})$	$4k^3y(k^2r^2 - 8)(2k^2x^2 - 1)$

Table C.11: Harmonic orbital expressions made with $H_2(kx)$ and $H_1(ky)$. Omitted factor $e^{-\frac{k^2}{2}(x^2+y^2)}$.

$\phi_{10} \to \phi_{0,4}$	
$\phi(\mathbf{r})$	$16k^4y^4 - 48k^2y^2 + 12$
$\mathbf{i} \cdot \nabla\phi(\mathbf{r})$	$-4k^2x\left(4k^4y^4 - 12k^2y^2 + 3\right)$
$\mathbf{j} \cdot \nabla\phi(\mathbf{r})$	$-4k^2y\left(4k^4y^4 - 28k^2y^2 + 27\right)$
$\nabla^2\phi(\mathbf{r})$	$4k^2\left(k^2r^2 - 10\right)\left(4k^4y^4 - 12k^2y^2 + 3\right)$

Table C.12: Harmonic orbital expressions made with $H_0(kx)$ and $H_4(ky)$. Omitted factor $e^{-\frac{k^2}{2}\left(x^2+y^2\right)}$.

$\phi_{11} \to \phi_{4,0}$	
$\phi(\mathbf{r})$	$16k^4x^4 - 48k^2x^2 + 12$
$\mathbf{i} \cdot \nabla\phi(\mathbf{r})$	$-4k^2x\left(4k^4x^4 - 28k^2x^2 + 27\right)$
$\mathbf{j} \cdot \nabla\phi(\mathbf{r})$	$-4k^2y\left(4k^4x^4 - 12k^2x^2 + 3\right)$
$\nabla^2\phi(\mathbf{r})$	$4k^2\left(k^2r^2 - 10\right)\left(4k^4x^4 - 12k^2x^2 + 3\right)$

Table C.13: Harmonic orbital expressions made with $H_4(kx)$ and $H_0(ky)$. Omitted factor $e^{-\frac{k^2}{2}\left(x^2+y^2\right)}$.

$\phi_{12} \to \phi_{1,3}$	
$\phi(\mathbf{r})$	$2kx\left(8k^3y^3 - 12ky\right)$
$\mathbf{i} \cdot \nabla\phi(\mathbf{r})$	$-8k^2y\left(kx - 1\right)\left(kx + 1\right)\left(2k^2y^2 - 3\right)$
$\mathbf{j} \cdot \nabla\phi(\mathbf{r})$	$-8k^2x\left(2k^4y^4 - 9k^2y^2 + 3\right)$
$\nabla^2\phi(\mathbf{r})$	$8k^4xy\left(k^2r^2 - 10\right)\left(2k^2y^2 - 3\right)$

Table C.14: Harmonic orbital expressions made with $H_1(kx)$ and $H_3(ky)$. Omitted factor $e^{-\frac{k^2}{2}\left(x^2+y^2\right)}$.

$\phi_{13} \to \phi_{3,1}$	
$\phi(\mathbf{r})$	$2ky\left(8k^3x^3 - 12kx\right)$
$\mathbf{i} \cdot \nabla\phi(\mathbf{r})$	$-8k^2y\left(2k^4x^4 - 9k^2x^2 + 3\right)$
$\mathbf{j} \cdot \nabla\phi(\mathbf{r})$	$-8k^2x\left(ky - 1\right)\left(ky + 1\right)\left(2k^2x^2 - 3\right)$
$\nabla^2\phi(\mathbf{r})$	$8k^4xy\left(k^2r^2 - 10\right)\left(2k^2x^2 - 3\right)$

Table C.15: Harmonic orbital expressions made with $H_3(kx)$ and $H_1(ky)$. Omitted factor $e^{-\frac{k^2}{2}\left(x^2+y^2\right)}$.

$\phi_{14} \to \phi_{2,2}$	
$\phi(\mathbf{r})$	$(4k^2x^2-2)(4k^2y^2-2)$
$\mathbf{i}\cdot\nabla\phi(\mathbf{r})$	$-4k^2x(2k^2x^2-5)(2k^2y^2-1)$
$\mathbf{j}\cdot\nabla\phi(\mathbf{r})$	$-4k^2y(2k^2x^2-1)(2k^2y^2-5)$
$\nabla^2\phi(\mathbf{r})$	$4k^2(k^2r^2-10)(2k^2x^2-1)(2k^2y^2-1)$

Table C.16: Harmonic orbital expressions made with $H_2(kx)$ and $H_2(ky)$. Omitted factor $e^{-\frac{k^2}{2}(x^2+y^2)}$.

$\phi_{15} \to \phi_{0,5}$	
$\phi(\mathbf{r})$	$32k^5y^5-160k^3y^3+120ky$
$\mathbf{i}\cdot\nabla\phi(\mathbf{r})$	$-8k^3xy(4k^4y^4-20k^2y^2+15)$
$\mathbf{j}\cdot\nabla\phi(\mathbf{r})$	$-8k(4k^6y^6-40k^4y^4+75k^2y^2-15)$
$\nabla^2\phi(\mathbf{r})$	$8k^3y(k^2r^2-12)(4k^4y^4-20k^2y^2+15)$

Table C.17: Harmonic orbital expressions made with $H_0(kx)$ and $H_5(ky)$. Omitted factor $e^{-\frac{k^2}{2}(x^2+y^2)}$.

$\phi_{16} \to \phi_{5,0}$	
$\phi(\mathbf{r})$	$32k^5x^5-160k^3x^3+120kx$
$\mathbf{i}\cdot\nabla\phi(\mathbf{r})$	$-8k(4k^6x^6-40k^4x^4+75k^2x^2-15)$
$\mathbf{j}\cdot\nabla\phi(\mathbf{r})$	$-8k^3xy(4k^4x^4-20k^2x^2+15)$
$\nabla^2\phi(\mathbf{r})$	$8k^3x(k^2r^2-12)(4k^4x^4-20k^2x^2+15)$

Table C.18: Harmonic orbital expressions made with $H_5(kx)$ and $H_0(ky)$. Omitted factor $e^{-\frac{k^2}{2}(x^2+y^2)}$.

$\phi_{17} \to \phi_{1,4}$	
$\phi(\mathbf{r})$	$2kx(16k^4y^4-48k^2y^2+12)$
$\mathbf{i}\cdot\nabla\phi(\mathbf{r})$	$-8k(kx-1)(kx+1)(4k^4y^4-12k^2y^2+3)$
$\mathbf{j}\cdot\nabla\phi(\mathbf{r})$	$-8k^3xy(4k^4y^4-28k^2y^2+27)$
$\nabla^2\phi(\mathbf{r})$	$8k^3x(k^2r^2-12)(4k^4y^4-12k^2y^2+3)$

Table C.19: Harmonic orbital expressions made with $H_1(kx)$ and $H_4(ky)$. Omitted factor $e^{-\frac{k^2}{2}(x^2+y^2)}$.

$\phi_{18} \rightarrow \phi_{4,1}$	
$\phi(\mathbf{r})$	$2ky\left(16k^4x^4 - 48k^2x^2 + 12\right)$
$\mathbf{i}\cdot\nabla\phi(\mathbf{r})$	$-8k^3xy\left(4k^4x^4 - 28k^2x^2 + 27\right)$
$\mathbf{j}\cdot\nabla\phi(\mathbf{r})$	$-8k\left(ky-1\right)\left(ky+1\right)\left(4k^4x^4 - 12k^2x^2 + 3\right)$
$\nabla^2\phi(\mathbf{r})$	$8k^3y\left(k^2r^2 - 12\right)\left(4k^4x^4 - 12k^2x^2 + 3\right)$

Table C.20: Harmonic orbital expressions made with $H_4(kx)$ and $H_1(ky)$. Omitted factor $e^{-\frac{k^2}{2}\left(x^2+y^2\right)}$.

$\phi_{19} \rightarrow \phi_{2,3}$	
$\phi(\mathbf{r})$	$\left(4k^2x^2 - 2\right)\left(8k^3y^3 - 12ky\right)$
$\mathbf{i}\cdot\nabla\phi(\mathbf{r})$	$-8k^3xy\left(2k^2x^2 - 5\right)\left(2k^2y^2 - 3\right)$
$\mathbf{j}\cdot\nabla\phi(\mathbf{r})$	$-8k\left(2k^2x^2 - 1\right)\left(2k^4y^4 - 9k^2y^2 + 3\right)$
$\nabla^2\phi(\mathbf{r})$	$8k^3y\left(k^2r^2 - 12\right)\left(2k^2x^2 - 1\right)\left(2k^2y^2 - 3\right)$

Table C.21: Harmonic orbital expressions made with $H_2(kx)$ and $H_3(ky)$. Omitted factor $e^{-\frac{k^2}{2}\left(x^2+y^2\right)}$.

$\phi_{20} \rightarrow \phi_{3,2}$	
$\phi(\mathbf{r})$	$\left(4k^2y^2 - 2\right)\left(8k^3x^3 - 12kx\right)$
$\mathbf{i}\cdot\nabla\phi(\mathbf{r})$	$-8k\left(2k^2y^2 - 1\right)\left(2k^4x^4 - 9k^2x^2 + 3\right)$
$\mathbf{j}\cdot\nabla\phi(\mathbf{r})$	$-8k^3xy\left(2k^2x^2 - 3\right)\left(2k^2y^2 - 5\right)$
$\nabla^2\phi(\mathbf{r})$	$8k^3x\left(k^2r^2 - 12\right)\left(2k^2x^2 - 3\right)\left(2k^2y^2 - 1\right)$

Table C.22: Harmonic orbital expressions made with $H_3(kx)$ and $H_2(ky)$. Omitted factor $e^{-\frac{k^2}{2}\left(x^2+y^2\right)}$.

$\phi_{21} \rightarrow \phi_{0,6}$	
$\phi(\mathbf{r})$	$64k^6y^6 - 480k^4y^4 + 720k^2y^2 - 120$
$\mathbf{i}\cdot\nabla\phi(\mathbf{r})$	$-8k^2x\left(8k^6y^6 - 60k^4y^4 + 90k^2y^2 - 15\right)$
$\mathbf{j}\cdot\nabla\phi(\mathbf{r})$	$-8k^2y\left(8k^6y^6 - 108k^4y^4 + 330k^2y^2 - 195\right)$
$\nabla^2\phi(\mathbf{r})$	$8k^2\left(k^2r^2 - 14\right)\left(8k^6y^6 - 60k^4y^4 + 90k^2y^2 - 15\right)$

Table C.23: Harmonic orbital expressions made with $H_0(kx)$ and $H_6(ky)$. Omitted factor $e^{-\frac{k^2}{2}\left(x^2+y^2\right)}$.

$\phi_{22} \to \phi_{6,0}$	
$\phi(\mathbf{r})$	$64k^6x^6 - 480k^4x^4 + 720k^2x^2 - 120$
$\mathbf{i} \cdot \nabla\phi(\mathbf{r})$	$-8k^2x\left(8k^6x^6 - 108k^4x^4 + 330k^2x^2 - 195\right)$
$\mathbf{j} \cdot \nabla\phi(\mathbf{r})$	$-8k^2y\left(8k^6x^6 - 60k^4x^4 + 90k^2x^2 - 15\right)$
$\nabla^2\phi(\mathbf{r})$	$8k^2\left(k^2r^2 - 14\right)\left(8k^6x^6 - 60k^4x^4 + 90k^2x^2 - 15\right)$

Table C.24: Harmonic orbital expressions made with $H_6(kx)$ and $H_0(ky)$. Omitted factor $e^{-\frac{k^2}{2}\left(x^2+y^2\right)}$.

$\phi_{23} \to \phi_{1,5}$	
$\phi(\mathbf{r})$	$2kx\left(32k^5y^5 - 160k^3y^3 + 120ky\right)$
$\mathbf{i} \cdot \nabla\phi(\mathbf{r})$	$-16k^2y\left(kx - 1\right)\left(kx + 1\right)\left(4k^4y^4 - 20k^2y^2 + 15\right)$
$\mathbf{j} \cdot \nabla\phi(\mathbf{r})$	$-16k^2x\left(4k^6y^6 - 40k^4y^4 + 75k^2y^2 - 15\right)$
$\nabla^2\phi(\mathbf{r})$	$16k^4xy\left(k^2r^2 - 14\right)\left(4k^4y^4 - 20k^2y^2 + 15\right)$

Table C.25: Harmonic orbital expressions made with $H_1(kx)$ and $H_5(ky)$. Omitted factor $e^{-\frac{k^2}{2}\left(x^2+y^2\right)}$.

$\phi_{24} \to \phi_{5,1}$	
$\phi(\mathbf{r})$	$2ky\left(32k^5x^5 - 160k^3x^3 + 120kx\right)$
$\mathbf{i} \cdot \nabla\phi(\mathbf{r})$	$-16k^2y\left(4k^6x^6 - 40k^4x^4 + 75k^2x^2 - 15\right)$
$\mathbf{j} \cdot \nabla\phi(\mathbf{r})$	$-16k^2x\left(ky - 1\right)\left(ky + 1\right)\left(4k^4x^4 - 20k^2x^2 + 15\right)$
$\nabla^2\phi(\mathbf{r})$	$16k^4xy\left(k^2r^2 - 14\right)\left(4k^4x^4 - 20k^2x^2 + 15\right)$

Table C.26: Harmonic orbital expressions made with $H_5(kx)$ and $H_1(ky)$. Omitted factor $e^{-\frac{k^2}{2}\left(x^2+y^2\right)}$.

$\phi_{25} \to \phi_{2,4}$	
$\phi(\mathbf{r})$	$\left(4k^2x^2 - 2\right)\left(16k^4y^4 - 48k^2y^2 + 12\right)$
$\mathbf{i} \cdot \nabla\phi(\mathbf{r})$	$-8k^2x\left(2k^2x^2 - 5\right)\left(4k^4y^4 - 12k^2y^2 + 3\right)$
$\mathbf{j} \cdot \nabla\phi(\mathbf{r})$	$-8k^2y\left(2k^2x^2 - 1\right)\left(4k^4y^4 - 28k^2y^2 + 27\right)$
$\nabla^2\phi(\mathbf{r})$	$8k^2\left(k^2r^2 - 14\right)\left(2k^2x^2 - 1\right)\left(4k^4y^4 - 12k^2y^2 + 3\right)$

Table C.27: Harmonic orbital expressions made with $H_2(kx)$ and $H_4(ky)$. Omitted factor $e^{-\frac{k^2}{2}\left(x^2+y^2\right)}$.

$\phi_{26} \to \phi_{4,2}$	
$\phi(\mathbf{r})$	$(4k^2y^2-2)(16k^4x^4-48k^2x^2+12)$
$\mathbf{i}\cdot\nabla\phi(\mathbf{r})$	$-8k^2x(2k^2y^2-1)(4k^4x^4-28k^2x^2+27)$
$\mathbf{j}\cdot\nabla\phi(\mathbf{r})$	$-8k^2y(2k^2y^2-5)(4k^4x^4-12k^2x^2+3)$
$\nabla^2\phi(\mathbf{r})$	$8k^2(k^2r^2-14)(2k^2y^2-1)(4k^4x^4-12k^2x^2+3)$

Table C.28: Harmonic orbital expressions made with $H_4(kx)$ and $H_2(ky)$. Omitted factor $e^{-\frac{k^2}{2}(x^2+y^2)}$.

$\phi_{27} \to \phi_{3,3}$	
$\phi(\mathbf{r})$	$(8k^3x^3-12kx)(8k^3y^3-12ky)$
$\mathbf{i}\cdot\nabla\phi(\mathbf{r})$	$-16k^2y(2k^2y^2-3)(2k^4x^4-9k^2x^2+3)$
$\mathbf{j}\cdot\nabla\phi(\mathbf{r})$	$-16k^2x(2k^2x^2-3)(2k^4y^4-9k^2y^2+3)$
$\nabla^2\phi(\mathbf{r})$	$16k^4xy(k^2r^2-14)(2k^2x^2-3)(2k^2y^2-3)$

Table C.29: Harmonic orbital expressions made with $H_3(kx)$ and $H_3(ky)$. Omitted factor $e^{-\frac{k^2}{2}(x^2+y^2)}$.

Appendix D

Harmonic oscillator orbitals in three dimensions

Harmonic oscillator orbitals in three dimensions are a combination of hermite polynomials of the x-coordinate, hermite polynomials of the y-coordinate, hermite polynomials of the z-coordinate, and an exponential factor. The orbitals are thus created as

$$\phi(\mathbf{r})_{n_x,n_y,n_z} = H_{n_x}(kx)H_{n_y}(ky)H_{n_z}(kz)e^{-\frac{1}{2}k^2r^2}.$$

Here $k = \sqrt{\omega\alpha}$, where ω is the oscillator frequency and α is the variational parameter. In the following the hermite polynomials and resulting orbitals will be listed.

$H_0(kx)$	1
$H_1(kx)$	$2kx$
$H_2(kx)$	$4k^2x^2 - 2$
$H_3(kx)$	$8k^3x^3 - 12x$
$H_0(ky)$	1
$H_1(ky)$	$2ky$
$H_2(ky)$	$4k^2y^2 - 2$
$H_3(ky)$	$8k^3y^3 - 12y$
$H_0(kz)$	1
$H_1(kz)$	$2kz$
$H_2(kz)$	$4k^2z^2 - 2$
$H_3(kz)$	$8k^3z^3 - 12z$

Table D.1: Hermite polynomials used to create the harmonic oscillator orbitals in three dimensions.

$\phi_0 \rightarrow \phi_{0,0,0}$	
$\phi(\mathbf{r})$	1
$\mathbf{i} \cdot \nabla\phi(\mathbf{r})$	$-k^2x$
$\mathbf{j} \cdot \nabla\phi(\mathbf{r})$	$-k^2y$
$\mathbf{k} \cdot \nabla\phi(\mathbf{r})$	$-k^2z$
$\nabla^2\phi(\mathbf{r})$	$k^2(k^2r^2 - 3)$

Table D.2: Harmonic orbital expressions made with $H_0(kx)$, $H_0(ky)$ and $H_0(kz)$. Omitted factor $e^{-\frac{k^2r^2}{2}}$.

$\phi_1 \rightarrow \phi_{0,0,1}$	
$\phi(\mathbf{r})$	$2kz$
$\mathbf{i} \cdot \nabla\phi(\mathbf{r})$	$-2k^3xz$
$\mathbf{j} \cdot \nabla\phi(\mathbf{r})$	$-2k^3yz$
$\mathbf{k} \cdot \nabla\phi(\mathbf{r})$	$-2k(kz - 1)(kz + 1)$
$\nabla^2\phi(\mathbf{r})$	$2k^3z(k^2r^2 - 5)$

Table D.3: Harmonic orbital expressions made with $H_0(kx)$, $H_0(ky)$ and $H_1(kz)$. Omitted factor $e^{-\frac{k^2r^2}{2}}$.

$\phi_2 \rightarrow \phi_{0,1,0}$	
$\phi(\mathbf{r})$	$2ky$
$\mathbf{i} \cdot \nabla\phi(\mathbf{r})$	$-2k^3xy$
$\mathbf{j} \cdot \nabla\phi(\mathbf{r})$	$-2k(ky-1)(ky+1)$
$\mathbf{k} \cdot \nabla\phi(\mathbf{r})$	$-2k^3yz$
$\nabla^2\phi(\mathbf{r})$	$2k^3y(k^2r^2-5)$

Table D.4: Harmonic orbital expressions made with $H_0(kx)$, $H_1(ky)$ and $H_0(kz)$. Omitted factor $e^{-\frac{k^2r^2}{2}}$.

$\phi_3 \rightarrow \phi_{1,0,0}$	
$\phi(\mathbf{r})$	$2kx$
$\mathbf{i} \cdot \nabla\phi(\mathbf{r})$	$-2k(kx-1)(kx+1)$
$\mathbf{j} \cdot \nabla\phi(\mathbf{r})$	$-2k^3xy$
$\mathbf{k} \cdot \nabla\phi(\mathbf{r})$	$-2k^3xz$
$\nabla^2\phi(\mathbf{r})$	$2k^3x(k^2r^2-5)$

Table D.5: Harmonic orbital expressions made with $H_1(kx)$, $H_0(ky)$ and $H_0(kz)$. Omitted factor $e^{-\frac{k^2r^2}{2}}$.

$\phi_4 \rightarrow \phi_{0,0,2}$	
$\phi(\mathbf{r})$	$4k^2z^2-2$
$\mathbf{i} \cdot \nabla\phi(\mathbf{r})$	$-2k^2x(2k^2z^2-1)$
$\mathbf{j} \cdot \nabla\phi(\mathbf{r})$	$-2k^2y(2k^2z^2-1)$
$\mathbf{k} \cdot \nabla\phi(\mathbf{r})$	$-2k^2z(2k^2z^2-5)$
$\nabla^2\phi(\mathbf{r})$	$2k^2(k^2r^2-7)(2k^2z^2-1)$

Table D.6: Harmonic orbital expressions made with $H_0(kx)$, $H_0(ky)$ and $H_2(kz)$. Omitted factor $e^{-\frac{k^2r^2}{2}}$.

$\phi_5 \to \phi_{0,2,0}$	
$\phi(\mathbf{r})$	$4k^2y^2 - 2$
$\mathbf{i} \cdot \nabla\phi(\mathbf{r})$	$-2k^2x(2k^2y^2 - 1)$
$\mathbf{j} \cdot \nabla\phi(\mathbf{r})$	$-2k^2y(2k^2y^2 - 5)$
$\mathbf{k} \cdot \nabla\phi(\mathbf{r})$	$-2k^2z(2k^2y^2 - 1)$
$\nabla^2\phi(\mathbf{r})$	$2k^2(k^2r^2 - 7)(2k^2y^2 - 1)$

Table D.7: Harmonic orbital expressions made with $H_0(kx)$, $H_2(ky)$ and $H_0(kz)$. Omitted factor $e^{-\frac{k^2r^2}{2}}$.

$\phi_6 \to \phi_{2,0,0}$	
$\phi(\mathbf{r})$	$4k^2x^2 - 2$
$\mathbf{i} \cdot \nabla\phi(\mathbf{r})$	$-2k^2x(2k^2x^2 - 5)$
$\mathbf{j} \cdot \nabla\phi(\mathbf{r})$	$-2k^2y(2k^2x^2 - 1)$
$\mathbf{k} \cdot \nabla\phi(\mathbf{r})$	$-2k^2z(2k^2x^2 - 1)$
$\nabla^2\phi(\mathbf{r})$	$2k^2(k^2r^2 - 7)(2k^2x^2 - 1)$

Table D.8: Harmonic orbital expressions made with $H_2(kx)$, $H_0(ky)$ and $H_0(kz)$. Omitted factor $e^{-\frac{k^2r^2}{2}}$.

$\phi_7 \to \phi_{0,1,1}$	
$\phi(\mathbf{r})$	$4k^2yz$
$\mathbf{i} \cdot \nabla\phi(\mathbf{r})$	$-4k^4xyz$
$\mathbf{j} \cdot \nabla\phi(\mathbf{r})$	$-4k^2z(ky - 1)(ky + 1)$
$\mathbf{k} \cdot \nabla\phi(\mathbf{r})$	$-4k^2y(kz - 1)(kz + 1)$
$\nabla^2\phi(\mathbf{r})$	$4k^4yz(k^2r^2 - 7)$

Table D.9: Harmonic orbital expressions made with $H_0(kx)$, $H_1(ky)$ and $H_1(kz)$. Omitted factor $e^{-\frac{k^2r^2}{2}}$.

$\phi_8 \rightarrow \phi_{1,0,1}$	
$\phi(\mathbf{r})$	$4k^2xz$
$\mathbf{i}\cdot\nabla\phi(\mathbf{r})$	$-4k^2z(kx-1)(kx+1)$
$\mathbf{j}\cdot\nabla\phi(\mathbf{r})$	$-4k^4xyz$
$\mathbf{k}\cdot\nabla\phi(\mathbf{r})$	$-4k^2x(kz-1)(kz+1)$
$\nabla^2\phi(\mathbf{r})$	$4k^4xz(k^2r^2-7)$

Table D.10: Harmonic orbital expressions made with $H_1(kx)$, $H_0(ky)$ and $H_1(kz)$. Omitted factor $e^{-\frac{k^2r^2}{2}}$.

$\phi_9 \rightarrow \phi_{1,1,0}$	
$\phi(\mathbf{r})$	$4k^2xy$
$\mathbf{i}\cdot\nabla\phi(\mathbf{r})$	$-4k^2y(kx-1)(kx+1)$
$\mathbf{j}\cdot\nabla\phi(\mathbf{r})$	$-4k^2x(ky-1)(ky+1)$
$\mathbf{k}\cdot\nabla\phi(\mathbf{r})$	$-4k^4xyz$
$\nabla^2\phi(\mathbf{r})$	$4k^4xy(k^2r^2-7)$

Table D.11: Harmonic orbital expressions made with $H_1(kx)$, $H_1(ky)$ and $H_0(kz)$. Omitted factor $e^{-\frac{k^2r^2}{2}}$.

$\phi_{10} \rightarrow \phi_{0,0,3}$	
$\phi(\mathbf{r})$	$8k^3z^3-12kz$
$\mathbf{i}\cdot\nabla\phi(\mathbf{r})$	$-4k^3xz(2k^2z^2-3)$
$\mathbf{j}\cdot\nabla\phi(\mathbf{r})$	$-4k^3yz(2k^2z^2-3)$
$\mathbf{k}\cdot\nabla\phi(\mathbf{r})$	$-4k(2k^4z^4-9k^2z^2+3)$
$\nabla^2\phi(\mathbf{r})$	$4k^3z(kr-3)(kr+3)(2k^2z^2-3)$

Table D.12: Harmonic orbital expressions made with $H_0(kx)$, $H_0(ky)$ and $H_3(kz)$. Omitted factor $e^{-\frac{k^2r^2}{2}}$.

$\phi_{11} \rightarrow \phi_{0,3,0}$	
$\phi(\mathbf{r})$	$8k^3y^3 - 12ky$
$\mathbf{i} \cdot \nabla\phi(\mathbf{r})$	$-4k^3xy(2k^2y^2 - 3)$
$\mathbf{j} \cdot \nabla\phi(\mathbf{r})$	$-4k(2k^4y^4 - 9k^2y^2 + 3)$
$\mathbf{k} \cdot \nabla\phi(\mathbf{r})$	$-4k^3yz(2k^2y^2 - 3)$
$\nabla^2\phi(\mathbf{r})$	$4k^3y(kr - 3)(kr + 3)(2k^2y^2 - 3)$

Table D.13: Harmonic orbital expressions made with $H_0(kx)$, $H_3(ky)$ and $H_0(kz)$. Omitted factor $e^{-\frac{k^2r^2}{2}}$.

$\phi_{12} \rightarrow \phi_{3,0,0}$	
$\phi(\mathbf{r})$	$8k^3x^3 - 12kx$
$\mathbf{i} \cdot \nabla\phi(\mathbf{r})$	$-4k(2k^4x^4 - 9k^2x^2 + 3)$
$\mathbf{j} \cdot \nabla\phi(\mathbf{r})$	$-4k^3xy(2k^2x^2 - 3)$
$\mathbf{k} \cdot \nabla\phi(\mathbf{r})$	$-4k^3xz(2k^2x^2 - 3)$
$\nabla^2\phi(\mathbf{r})$	$4k^3x(kr - 3)(kr + 3)(2k^2x^2 - 3)$

Table D.14: Harmonic orbital expressions made with $H_3(kx)$, $H_0(ky)$ and $H_0(kz)$. Omitted factor $e^{-\frac{k^2r^2}{2}}$.

$\phi_{13} \rightarrow \phi_{0,1,2}$	
$\phi(\mathbf{r})$	$2ky(4k^2z^2 - 2)$
$\mathbf{i} \cdot \nabla\phi(\mathbf{r})$	$-4k^3xy(2k^2z^2 - 1)$
$\mathbf{j} \cdot \nabla\phi(\mathbf{r})$	$-4k(ky - 1)(ky + 1)(2k^2z^2 - 1)$
$\mathbf{k} \cdot \nabla\phi(\mathbf{r})$	$-4k^3yz(2k^2z^2 - 5)$
$\nabla^2\phi(\mathbf{r})$	$4k^3y(k^2r^2 - 9)(2k^2z^2 - 1)$

Table D.15: Harmonic orbital expressions made with $H_0(kx)$, $H_1(ky)$ and $H_2(kz)$. Omitted factor $e^{-\frac{k^2r^2}{2}}$.

$\phi_{14} \rightarrow \phi_{0,2,1}$	
$\phi(\mathbf{r})$	$2kz\left(4k^2y^2-2\right)$
$\mathbf{i}\cdot\nabla\phi(\mathbf{r})$	$-4k^3xz\left(2k^2y^2-1\right)$
$\mathbf{j}\cdot\nabla\phi(\mathbf{r})$	$-4k^3yz\left(2k^2y^2-5\right)$
$\mathbf{k}\cdot\nabla\phi(\mathbf{r})$	$-4k\left(kz-1\right)\left(kz+1\right)\left(2k^2y^2-1\right)$
$\nabla^2\phi(\mathbf{r})$	$4k^3z\left(k^2r^2-9\right)\left(2k^2y^2-1\right)$

Table D.16: Harmonic orbital expressions made with $H_0(kx)$, $H_2(ky)$ and $H_1(kz)$. Omitted factor $e^{-\frac{k^2r^2}{2}}$.

$\phi_{15} \rightarrow \phi_{1,0,2}$	
$\phi(\mathbf{r})$	$2kx\left(4k^2z^2-2\right)$
$\mathbf{i}\cdot\nabla\phi(\mathbf{r})$	$-4k\left(kx-1\right)\left(kx+1\right)\left(2k^2z^2-1\right)$
$\mathbf{j}\cdot\nabla\phi(\mathbf{r})$	$-4k^3xy\left(2k^2z^2-1\right)$
$\mathbf{k}\cdot\nabla\phi(\mathbf{r})$	$-4k^3xz\left(2k^2z^2-5\right)$
$\nabla^2\phi(\mathbf{r})$	$4k^3x\left(k^2r^2-9\right)\left(2k^2z^2-1\right)$

Table D.17: Harmonic orbital expressions made with $H_1(kx)$, $H_0(ky)$ and $H_2(kz)$. Omitted factor $e^{-\frac{k^2r^2}{2}}$.

$\phi_{16} \rightarrow \phi_{1,2,0}$	
$\phi(\mathbf{r})$	$2kx\left(4k^2y^2-2\right)$
$\mathbf{i}\cdot\nabla\phi(\mathbf{r})$	$-4k\left(kx-1\right)\left(kx+1\right)\left(2k^2y^2-1\right)$
$\mathbf{j}\cdot\nabla\phi(\mathbf{r})$	$-4k^3xy\left(2k^2y^2-5\right)$
$\mathbf{k}\cdot\nabla\phi(\mathbf{r})$	$-4k^3xz\left(2k^2y^2-1\right)$
$\nabla^2\phi(\mathbf{r})$	$4k^3x\left(k^2r^2-9\right)\left(2k^2y^2-1\right)$

Table D.18: Harmonic orbital expressions made with $H_1(kx)$, $H_2(ky)$ and $H_0(kz)$. Omitted factor $e^{-\frac{k^2r^2}{2}}$.

$\phi_{17} \to \phi_{2,0,1}$	
$\phi(\mathbf{r})$	$2kz\left(4k^2x^2-2\right)$
$\mathbf{i}\cdot\nabla\phi(\mathbf{r})$	$-4k^3xz\left(2k^2x^2-5\right)$
$\mathbf{j}\cdot\nabla\phi(\mathbf{r})$	$-4k^3yz\left(2k^2x^2-1\right)$
$\mathbf{k}\cdot\nabla\phi(\mathbf{r})$	$-4k\left(kz-1\right)\left(kz+1\right)\left(2k^2x^2-1\right)$
$\nabla^2\phi(\mathbf{r})$	$4k^3z\left(k^2r^2-9\right)\left(2k^2x^2-1\right)$

Table D.19: Harmonic orbital expressions made with $H_2(kx)$, $H_0(ky)$ and $H_1(kz)$. Omitted factor $e^{-\frac{k^2r^2}{2}}$.

$\phi_{18} \to \phi_{2,1,0}$	
$\phi(\mathbf{r})$	$2ky\left(4k^2x^2-2\right)$
$\mathbf{i}\cdot\nabla\phi(\mathbf{r})$	$-4k^3xy\left(2k^2x^2-5\right)$
$\mathbf{j}\cdot\nabla\phi(\mathbf{r})$	$-4k\left(ky-1\right)\left(ky+1\right)\left(2k^2x^2-1\right)$
$\mathbf{k}\cdot\nabla\phi(\mathbf{r})$	$-4k^3yz\left(2k^2x^2-1\right)$
$\nabla^2\phi(\mathbf{r})$	$4k^3y\left(k^2r^2-9\right)\left(2k^2x^2-1\right)$

Table D.20: Harmonic orbital expressions made with $H_2(kx)$, $H_1(ky)$ and $H_0(kz)$. Omitted factor $e^{-\frac{k^2r^2}{2}}$.

$\phi_{19} \to \phi_{1,1,1}$	
$\phi(\mathbf{r})$	$8k^3xyz$
$\mathbf{i}\cdot\nabla\phi(\mathbf{r})$	$-8k^3yz\left(kx-1\right)\left(kx+1\right)$
$\mathbf{j}\cdot\nabla\phi(\mathbf{r})$	$-8k^3xz\left(ky-1\right)\left(ky+1\right)$
$\mathbf{k}\cdot\nabla\phi(\mathbf{r})$	$-8k^3xy\left(kz-1\right)\left(kz+1\right)$
$\nabla^2\phi(\mathbf{r})$	$8k^5xyz\left(k^2r^2-9\right)$

Table D.21: Harmonic orbital expressions made with $H_1(kx)$, $H_1(ky)$ and $H_1(kz)$. Omitted factor $e^{-\frac{k^2r^2}{2}}$.

Bibliography

[EMS, 2015] (2015). EMSL basis set exchange. https://bse.pnl.gov/bse/portal. Accessed: 2015-05-13.

[Anisimovas and Matulis, 1998] Anisimovas, E. and Matulis, A. (1998). Energy spectra of few-electron quantum dots. *Journal of Physics: Condensed Matter*, 10:601.

[Binder, 1984] Binder, K. (1984). Applications of the monte carlo method in statistical physics. *Applications of the Monte Carlo Method in Statistical Physics. Series: Topics in Current Physics, ISBN: 978-3-642-96790-0. Springer Berlin Heidelberg (Berlin, Heidelberg), Edited by Kurt Binder, vol. 36*, 36.

[Binkley and Pople, 1975] Binkley, J. S. and Pople, J. A. (1975). Møller–plesset theory for atomic ground state energies. *International Journal of Quantum Chemistry*, 9(2):229–236.

[Born and Oppenheimer, 1927] Born, M. and Oppenheimer, R. (1927). Zur quantentheorie der molekeln. *Annalen der Physik*, 389(20):457–484.

[Boys, 1950] Boys, S. F. (1950). Electronic wave functions. i. a general method of calculation for the stationary states of any molecular system. *Proceedings of the Royal Society of London. Series A*, 200(1063):542–554.

[Brillouin, 1932] Brillouin, L. (1932). Les problèmes de perturbations et les champs self-consistents. *Journal de Physique et Le Radium*, 3(9):373–389.

[Cajori, 1911] Cajori, F. (1911). Historical note on the newton-raphson method of approximation. *The American Mathematical Monthly*, 18(2):29–32.

[Ceperley et al., 1977] Ceperley, D., Chester, G., and Kalos, M. (1977). Monte carlo simulation of a many-fermion study. *Physical Review B*, 16:3081.

[Čížek, 1966] Čížek, J. (1966). On the correlation problem in atomic and molecular systems. calculation of wavefunction components in ursell-type expansion

using quantum-field theoretical methods. *The Journal of Chemical Physics*, 45:4256–4266.

[Čížek, 1969] Čížek, J. (1969). On the use of the cluster expansion and the technique of diagrams in calculations of correlation effects in atoms and molecules. *Advances in Chemical Physics: Correlation Effects in Atoms and Molecules*, 14:35–89.

[Čížek and Paldus, 1971] Čížek, J. and Paldus, J. (1971). Correlation problems in atomic and molecular systems III. Rederivation of the coupled-pair many-electron theory using the traditional quantum chemical methods. *International Journal of Quantum Chemistry*, 5:359–379.

[Coester and Kümmel, 1960] Coester, F. and Kümmel, H. (1960). Short-range correlations in nuclear wave functions. *Nuclear Physics*, 17:477–485.

[Daniell et al., 1984] Daniell, G., Hey, A. J., and Mandula, J. (1984). Error analysis for correlated monte carlo data. *Physical Review D*, 30:2230.

[Fedichkin et al., 2000] Fedichkin, L., Yanchenko, M., and Valiev, K. (2000). Novel coherent quantum bit using spatial quantization levels in semiconductor quantum dot. *Quantum Computers and Computing*, 1:58–76.

[Flyvbjerg and Petersen, 1989] Flyvbjerg, H. and Petersen, H. G. (1989). Error estimates on averages of correlated data. *The Journal of Chemical Physics*, 91:461–466.

[Fock, 1930a] Fock, V. (1930a). Bemerkung zum virialsatz. *Zeitschrift für Physik*, 63:855–858.

[Fock, 1930b] Fock, V. (1930b). Näherungsmethode zur losung des quantenmechanischen mehrkörperprobleme. *Zeitschrift für Physik*, 61:126–148.

[Fokker, 1914] Fokker, A. D. (1914). Die mittlere energie rotierender elektrischer dipole im strahlungsfeld. *Annalen der Physik*, 348:810–820.

[Gottlieb et al., 1986] Gottlieb, S., MacKenzie, P. B., Thacker, H. B., and Weingarten, D. (1986). Hadronic coupling constants in lattice gauge theory. *Nuclear Physics B*, 263:704–730.

[Griffiths, 2005] Griffiths, D. (2005). *Introduction to Quantum Mechanics*. Pearson international edition. Pearson Prentice Hall.

[Hartree, 1928] Hartree, D. R. (1928). The wave mechanics of an atom with a non-coulomb central field. part i. theory and methods. *Mathematical Proceedings of the Cambridge Philosophical Society*, 24:89–110.

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

[Høgberget, 2013] Høgberget, J. (2013). Quantum monte-carlo studies of generalized many-body systems. Master's thesis, University of Oslo.

[Hirth, 2012] Hirth, C. (2012). Ab initio coupledcluster analysis using opencl and gpu programming. Master's thesis, University of Oslo.

[Hjorth-Jensen, 2010] Hjorth-Jensen, M. (2010). *Computational Physics*.

[Hjorth-Jensen, 2013] Hjorth-Jensen, M. (2013). *Computational Physics: Lecture notes*.

[Holmevik, 1994] Holmevik, J. R. (1994). Compiling simula: a historical study of technological genesis. *IEEE Annals of the History of Computing*, 16:25–37.

[J. S. Binkley, 1980] J. S. Binkley, J. A. Pople, W. J. H. (1980). Self-consistent molecular orbital methods. 21. small split-valence basis sets for first-row elements. *Journal of the American Chemical Society*, 102:939–947.

[Kim et al., 2015] Kim, G.-H., de Arquer, F. P. G., Yoon, Y. J., Lan, X., Liu, M., Voznyy, O., Yang, Z., Fan, F., Ip, A. H., Kanjanaboos, P., Hoogland, S., Kim, J. Y., and Sargent, E. H. (2015). High-efficiency colloidal quantum dot photovoltaics via robust self-assembled monolayers. *Nano Letters*, 15(11):7691–7696. PMID: 26509283.

[Kohn and Sham, 1965] Kohn, W. and Sham, L. J. (1965). Self-consistent equations including exchange and correlation effects. *Physical review*, 140:A1133.

[Koput, 2011] Koput, J. (2011). The ground-state potential energy function of a beryllium dimer determined using the single-reference coupled-cluster approach.

[Langevin, 1908] Langevin, P. (1908). Sur la théorie du mouvement brownien. *Comptes Rendus de l'Académie des Sciences (Paris)*, 146:530–533.

[Langtangen, 2006] Langtangen, H. P. (2006). *Python scripting for computational science*, volume 3. Springer.

[Langtangen, 2011] Langtangen, H. P. (2011). *A primer on scientific programming with Python*, volume 6. Springer.

[Maurice and Head-Gordon, 1999] Maurice, D. and Head-Gordon, M. (1999). Analytical second derivatives for excited electronic states using the single excitation configuration interaction method: theory and application to benzo[a]pyrene and chalcone. *Molecular Physics*, 96:1533–1541.

[McMillan, 1965] McMillan, W. L. (1965). Ground state of liquid he^4. *Physical Review*, 138:A442–A451.

[Metropolis et al., 1953] Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., and Teller, E. (1953). Equation of state calculations by fast computing machines. *Journal of Chemical Physics*, 21:1087–1092.

[Moskowitz and Kalos, 1981] Moskowitz, J. W. and Kalos, M. (1981). A new look at correlations in atomic and molecular systems. I. Application of fermion Monte Carlo variational method. *International Journal of Quantum Chemistry*, 20:1107–1119.

[Mørk, 2016] Mørk, H. (2016). Git repository. http://github.com/hakonsbm/vmc-solver.

[Neidhardt et al., 2015] Neidhardt, H., Wilhelm, L., and Zagrebnov, V. A. (2015). A new model for quantum dot light emitting-absorbing bevices: Proofs and supplements. *Nanosystems: Physics, Chemistry, Mathematics*, 6(1):6–45.

[Newton, 1671] Newton, I. (1671). *Methodus fluxionum et serierum infinitarum.*

[Olsen, 2012] Olsen, V. K. B. (2012). Full configuration interaction simulation of quantum dots. Master's thesis, University of Oslo.

[Pauling and Bright, 1935] Pauling, L. and Bright, W. E. (1935). *Introduction to Quantum Mechanics.* McGraw Hill Book Company Inc.

[Planck, 1917] Planck, M. (1917). Über einen satz der statistischen dynamik und seine erweiterung in der quantentheorie. *Sitzungsberichte der Preussischen Akademie der Wissenschaften*, pages 324–341.

[Press, 2007] Press, W. H. (2007). *Numerical recipes 3rd edition: The art of scientific computing.* Cambridge university press.

[Raphson, 1702] Raphson, J. (1702). *Analysis aequationum universalis.* typis TB prostant venales apud A. & I. Churchill.

[Reed et al., 1988] Reed, M. A., Randall, J. N., Aggarwal, R. J., Matyi, R. J., Moore, T. M., and Wetsel, A. E. (1988). Observation of discrete electronic states in a zero-dimensional semiconductor nanostructure. *Physical Review Letters*, 60:535–537.

[Reimann, 2013] Reimann, S. (2013). Quantum-mechanical systems in traps and similarity renormalization group theory. Master's thesis, University of Oslo.

[Sanderson, 2010] Sanderson, C. (2010). Armadillo: An open source c++ linear algebra library for fast prototyping and computationally intensive experiments.

[Schrödinger, 1926] Schrödinger, E. (1926). Quantisierung als Eigenwertproblem. *Annalen der Physik*, 385:437–490.

[Shavitt and Bartlett, 2009] Shavitt, I. and Bartlett, R. J. (2009). *Many-Body Methods in Chemistry and Physics*. Cambridge University Press.

[Skattum, 2013] Skattum, S. B. (2013). Time evolution of quantum dots. Master's thesis, University of Oslo.

[Slater, 1929] Slater, J. C. (1929). The theory of complex spectra. *Physical Review*, 34:1293–1322.

[Slater, 1930] Slater, J. C. (1930). Note on hartree's method. *Physical Review*, 35:210–211.

[Strauf et al., 2006] Strauf, S., Hennessy, K., Rakher, M. T., Choi, Y.-S., Badolato, A., Andreani, L. C., Hu, E. L., Petroff, P. M., and Bouwmeester, D. (2006). Self-tuned quantum dot gain in photonic crystal lasers. *Physical Review Letters*, 96:127404.

[SymPy Development Team, 2016] SymPy Development Team (2016). *SymPy: Python library for symbolic mathematics*.

[Taut, 1993] Taut, M. (1993). Two electrons in an external oscillator potential: Particular analytic solutions of a coulomb correlation problem. *Physical Review A*, 48:3561.

[Tokumasu et al., 2005] Tokumasu, F., Fairhurst, R. M., Ostera, G. R., Brittain, N. J., Hwang, J., Wellems, T. E., and Dvorak, J. A. (2005). Band 3 modifications in plasmodium falciparum-infected aa and cc erythrocytes assayed by autocorrelation analysis using quantum dots. *Journal of Cell Science*, 118(5):1091–1098.

[Whitmer, 1984] Whitmer, C. (1984). Over-relaxation methods for monte carlo simulations of quadratic and multiquadratic actions. *Physical Review D*, 29:306–311.

[Wigner, 1934] Wigner, E. (1934). On the interaction of electrons in metals. *Physical Review*, 46:1002.

[Wigner, 1935] Wigner, E. (1935). On a modification of the rayleigh–schrödinger perturbation theory. *Mathematischer und Naturwissenschaftlicher Anzeiger der Ungarischen Akademie der Wissenschaften*, 53:477–482.

[Wilson, 1980] Wilson, K. G. (1980). *Recent Developments in Gauge Theories*, chapter Monte-Carlo Calculations for the Lattice Gauge Theory, pages 363–402. Springer US, Boston, MA.

[Yannouleas and Landman, 2007] Yannouleas, C. and Landman, U. (2007). Symmetry breaking and quantum correlations in finite systems: studies of quantum dots and ultracold bose gases and related nuclear and chemical methods. *Reports on Progress in Physics*, 70:2067.

[Yukawa, 1935] Yukawa, H. (1935). On the interaction of elementary particles. i. *Nippon Sugaku-Buturigakkwai Kizi Dai 3 Ki*, 17:48–57.