The Hartree-Fock method with gaussian basis sets

Gøran Brekke Svaland[1]*, Audun Skau Hansen[2]*

Abstract
In this project we present an implementation of the *Hartree-Fock method* in *C++* and armadillo using gaussian basis sets. We review some of the theoretical foundation for the method, and discuss some computational aspects of the code. To test the accuracy of the code, we perform some unit testing and benchmarking. Finally we investigate some properties of real physical systems, especially monatomic molecules and H_2O, and we use the results from these investigations to further evaluate the accuracy and performance of the code. The project was written as a final hand-in in the course FYS4411 - Computational Physics for Quantum Mechanical Systems. The full program is publicly available on www.github.com/audunsh/FYS4411.

Keywords
The Hartree-Fock method – Gaussian basis sets – Recursive Gaussian Integral Evaluation – Atomic calculations – Molecular calculations – Computational Quantum Mechanics – Quantum Mechanical Many-Body problems

[1] *Computational Physics, Master student, University of Oslo*
[2] *Computational Physics, Master student, University of Oslo*
***e-mail**: g.b.svaland@fys.uio.no
***e-mail**: audun@loverecords.no

Contents

Introduction

The *Hartree-Fock method* is a variational method that may be used to solve quantum physical many-body problems. Starting out from first principles, the method allows for both approximations of the systems groundstate wavefunction and energy, but the method may also serve as a starting point for more exact approximations such as many-body perturbation theory (MBPT), configuration interaction (CI) and coupled cluster theory (CC)[8].

In this project we use the restricted *Hartree-Fock method* to investigate certain properties of atoms and molecules. In such a system, multiple electrons are interacting with a potential field generated by all the particles present in the system. The method is a minimization procedure that lets us find the lowest energy and eigenstate of the system. The method can be applied to both atoms, molecules and other quantum mechanical systems. We may perform calculations for a system of two such atoms or molecules and vary the distance between them, thus finding the energy of the system as a function of the distance. This way we are able to determine the potential field of the atom or the molecule. The resulting potential can be used to model the interactions of many such atoms or molecules in a molecular dynamics simulation.

A number of approximations and assumptions are made before the method is applied. The nuclei(s) are assumed to be fixed point particles, which is justified by the nuclei being much heavier than the electons orbiting them. This is known as the *Born-Oppenheimer* approximation. It is also assumed that the systems wavefunction may be reasonable represented by a slater determinant (SD), a determinant consisting of permutations of single particle states.

Originally it was commonly distinguished between Hartree-Fock and Hartree-Fock-Roothaan (or self-consistent field), but this distinction has been lost. [8] In this project we shall employ the latter method, meaning that we will expand each spin orbital in a basis set and seek the coefficients that minimizes the energy of the system. In principle this means we will need to solve a eigenvalue problem using linear algebra. To achieve this in a effective way we have implemented the method in a *C++* code that relies heavily on Armadillo - a linear algebra library.

In the following we will give a review of the theory related to quantum many-body problems, the Hartree-Fock method itself and the different basis sets and related integrals. We then go on by discussing the implementation in *C++* and armadillo, and give an overview of the full code. Towards the end we investigate some quantum mechanical problems with the code and present our results. Finally, we adress some issues with possible optimizations of the code, further developements and perspectives on future applications of the code.

1. Theoretical background of the Hartree-Fock method

1.1 The Born-Oppenheimer approximation

The full Hamiltonian for a system of N electrons and K nuclei with charges Z_n reads as eq. (1), where the index i refers to the electrons and n refers to the nucleis. M_n is the mass of nuclei n and m is the electron mass. The first two terms are the kinetic energies of the particles, electrons and cores, respectively. The final three terms represents the Coulumb interaction between the particles, *electron-electron*, *electron-nuclei* and *nuclei-nuclei* repulsion respectively. We will assume that the cores are fixed point particles so that the last term vanish. This is a good approximation because the nuclei are much heavier than the electrons. As a result from this approximation, the second term also vanishes because the cores will have no kinetic energy. [9]. This is known as the *Born-Oppenheimer approximation*.

$$\begin{aligned}\hat{H} = \sum_i^N \frac{p_i^2}{2m} + \sum_n^K \frac{P_n^2}{2M} + \frac{1}{4\pi\varepsilon_0}\left(\frac{1}{2}\sum_{i,j\neq i}^N \frac{e^2}{|\mathbf{r_i}-\mathbf{r_j}|}\right. \\ \left. -\sum_n^K\sum_i^N \frac{Z_n e^2}{|\mathbf{r_i}-\mathbf{R_n}|} + \frac{1}{2}\sum_{n,n'\neq n}^K \frac{Z_n Z_n e^2}{|\mathbf{R_n}-\mathbf{R_{n'}}|}\right)\end{aligned} \tag{1}$$

There is a number of different conventions used to refer to the different operators in many-body quantum physics. We will initially use $\hat{f} = \hat{f}(\mathbf{x_i}) = \hat{f}_i$ to refer to the one body part of the hamiltonian, while we use $\hat{v} = \hat{v}(\mathbf{x_i},\mathbf{x_j}) = \hat{v}_{i,j}$ when referring to the particle-particle interaction.

There might be some other notations involved when we look at the algorithms.

1.2 The many-body wavefunction

We consider a system of interacting fermions in an common spherically symmetric potential, such as electrons in an atom. The time independent wavefunction Ψ of such a system will be a function of the positions and spins $\mathbf{x_i}$ of all particles in the system $\Psi = \Psi(\mathbf{x_1},\mathbf{x_2},...,\mathbf{x_N})$. We know from measurements that this wavefunction is *antisymmetric* [9], so that interchanging particles i and j results in a change of sign in the wavefunction:

$$\Psi(\mathbf{x_1},\mathbf{x_2},..,\mathbf{x_i},..,\mathbf{x_j},..,\mathbf{x_N}) = -\Psi(\mathbf{x_1},\mathbf{x_2},..,\mathbf{x_j},..,\mathbf{x_i},..,\mathbf{x_N})$$

One function that obeys this symmetrization is the Slater determinant (SD). To define it, we fist consider the constituent particles wavefunctions $\Psi_1(\mathbf{x_1}), \Psi_2(\mathbf{x_2}), ..., \Psi_N(\mathbf{x_N})$. The Slater determinant is then defined as eq. (2).

$$\Psi(\mathbf{x_1},\mathbf{x_2},...,\mathbf{x_N}) = \frac{1}{\sqrt{N!}}\begin{vmatrix} \Psi_1(\mathbf{x_1}) & \Psi_2(\mathbf{x_1}) & \cdots & \Psi_N(\mathbf{x_1}) \\ \Psi_1(\mathbf{x_2}) & \Psi_2(\mathbf{x_2}) & \cdots & \Psi_N(\mathbf{x_2}) \\ \vdots & \vdots & \ddots & \vdots \\ \Psi_1(\mathbf{x_N}) & \Psi_2(\mathbf{x_N}) & \cdots & \Psi_N(\mathbf{x_N}) \end{vmatrix} \tag{2}$$

As the SD will contain $N!$ terms of presumably already normalized functions, the factor in front of eq. (2) preserves the normalization. Another equivalent notation for the SD is given in eq. (3).

$$\Psi(\mathbf{x_1},\mathbf{x_2},...,\mathbf{x_N}) = \frac{1}{\sqrt{N!}}\sum_p(-1)^p\hat{P}\prod_{i=1}^N\psi_i(\mathbf{x_i}) = \sqrt{N!}\mathscr{A}\psi_h \tag{3}$$

Where in eq. (3), the antimymmetrization operator $\mathscr{A} = \frac{1}{N!}\sum_p(-1)^p\hat{P}$ performs all possible permutations, keeping the signconvention from the Slater determinant. An important property of the antisymmetrizer is that [9] $\mathscr{A}^2 = \mathscr{A}$ and $\mathscr{A}^\dagger = \mathscr{A}$. Interchanging two particles in the system will obey the symmetry of the system, so that this operation will change the sign of the SD. Another consequence of the antisymmetrization is the *Pauli exclusion principle*, which states that two fermions may not share the same quantum numbers.

We also introduced the Hartree function in eq. (4). This function (shown in eq. (3)) places all particles in the unpermuted state.

$$\psi_h = \prod_{i=1}^N\psi_i(\mathbf{x_i}) \tag{4}$$

In summary, we conclude that the wavefunction of a many-body system may be exactly represented by an Slater determinant constructed from the single-particle wavefunctions (which, of course, we do not know much about yet).

1.3 The particle interactions

The Hamiltonian (after introducing the Born-Oppenheimer approximation) for a system of fermions interacting with each other and a central symmetric common potential is given in dimensionless form by eq. (5). This Hamiltonian is conveniently separated in two terms, with all particle-particle interactions isolated in the second term. A system where no inter-particle interactions occured would therefore be represented by the first part only. Note the factor $1/2$, indicating that we sum over each pair of particles twice. Also we exclude the terms where $i = j$, as there are no interaction between a particle and itself.

$$\hat{H} = \sum_i^N\Big(-\frac{\hat{p}_i}{2m}+V(\mathbf{x_i})\Big)+\frac{1}{2}\sum_{i,j\neq i}^N\left(\frac{1}{|\mathbf{r_i}-\mathbf{r_j}|}\right) \tag{5}$$

As the first part only acts on one particle at the time while the second acts on pairs, it is even more convinient to use the notation in eq. (6) as for eq. (5).

$$\hat{H} = \sum_i^N\hat{f}(\mathbf{x_i})+\frac{1}{2}\sum_{i,j\neq i}^N\hat{v}(\mathbf{x_i},\mathbf{x_j}) \tag{6}$$

The Hamiltonian commutes with the antisymmetrizer, $\big[\mathscr{A},\hat{H}\big] = \mathscr{A}\hat{H}-\hat{H}\mathscr{A} = 0$.

1.4 Many-body time-independent Schrödinger equation

We may now use the Hamiltonian and Slater determinant to inspect the solution of the many body Schrödinger equation. Solving the Schrödinger equation means to find a set of wavefunctions Ψ_i with corresponding eigenvalues ε_i, so that $\hat{H}|\Psi_i\rangle = \varepsilon_i|\Psi_i\rangle$. The standard procedure is to multipy by the bra $\langle\Psi_j|$ from the left, so that, due to the orthonormality in the wavefunctions, we get eq. (7).

$$\langle\Psi_j|\hat{H}|\Psi_i\rangle = \varepsilon_i\langle\Psi_j|\Psi_i\rangle = \varepsilon_i\delta_{ij} \tag{7}$$

From now on we seek the ground state energy of the system, meaning that we want to solve eq. (8).

$$\langle\Psi_0|\hat{H}|\Psi_0\rangle = \varepsilon_0 \tag{8}$$

Considering what we have discussed so far, we may rewrite the above inner product, using eq. (3) and the fact that the Hamiltonian and the antisymmetrization operator commutes, into eq. (9). An integral form of this equation if found in the Appendix section 10.1.

$$\begin{aligned}&\varepsilon_0 = \langle\sqrt{N!}\mathscr{A}\psi_h|\hat{H}|\sqrt{N!}\mathscr{A}\psi_h\rangle = \\ &N!\langle\psi_h|\mathscr{A}^\dagger\hat{H}\mathscr{A}\psi_h\rangle = N!\langle\psi_h|\hat{H}\mathscr{A}\psi_h\rangle = \\ &\langle\psi_h|\hat{H}|\sum_p(-1)^p\hat{P}\psi_h\rangle = \langle\psi_h|\hat{H}|\sum_p(-1)^p\hat{P}\psi_h\rangle\end{aligned} \tag{9}$$

We would like to write this equation (eq. (9)) in terms of the notation used in eq. (6). The equation will then look like eq. (10).

$$\varepsilon_0 = \sqrt{N!}\Big(\langle\psi_h|\sum_i^N\hat{f}(\mathbf{x_i})|\Psi_0\rangle+\frac{1}{2}\langle\psi_h|\sum_{i,j\neq i}^N\hat{v}(\mathbf{x_i},\mathbf{x_j})|\Psi_0\rangle\Big) \tag{10}$$

The problem is now broken down into two parts, the one body part f, and the two body part v, where $\varepsilon_0 = \varepsilon_f+\varepsilon_v$. In the following subsections we will derive the form of the solution to these two problems.

1.5 The one body problem

The one body operator $\hat{f}(\mathbf{x_i})$ acts exclusively on one particle at the time. This means that when acting on any term in the SD, we find that:

$$\hat{f}(\mathbf{x_i})\prod_{j=1}^{N}\psi_j(\mathbf{x_j}) = \left(\prod_{j=1}^{N-1}\psi_j(\mathbf{x_j})\right)\hat{f}(\mathbf{x_i})\psi_i(\mathbf{x_i})$$

For one of the terms in the integrals previously discussed, it follows that:

$$\langle\psi_h|\hat{f}|\Psi_0\rangle = \int d\tau\left(\prod_{j=1}^{N}\psi_i^*(\mathbf{x_i})\right)\hat{f}(\mathbf{x_j})\left(\prod_{k=1}^{N}\psi_k(\mathbf{x_k})\right) = \prod_{i\neq j}^{N-1}\left(\int d\mathbf{x_i}|\psi_i(\mathbf{x_i})|^2\right)\int d\mathbf{x_j}\left(\psi_j^*(\mathbf{x_j})\hat{f}(\mathbf{x_j})\psi_j(\mathbf{x_j})\right)$$

Here, we used that $d\tau = d\mathbf{x_0}d\mathbf{x_1}...d\mathbf{x_N}$.

Assuming that the constituent parts is properly normalized, it is straight forward to see that:

$$\prod_{i\neq j}^{N-1}\left(\int d\mathbf{x_i}|\psi_i(\mathbf{x_i})|^2\right) = 1$$

If we perform one or more permutations in the right-most Hartree-functions, i.e considering terms in the full SD beyond the Hartree-function, we either end up with a factor of:

$$\int d\mathbf{x_j}\left(\psi_j^*(\mathbf{x_j})\hat{f}(\mathbf{x_j})\psi_i(\mathbf{x_j})\right) = 0 \quad or:$$
$$\int d\mathbf{x_j}\left(\psi_j^*(\mathbf{x_j})\psi_i(\mathbf{x_j})\right) = 0$$

Either way, the contribution is zero. Only the first term survives the integral, we therefore finally end up with eq. (11).

$$\varepsilon_f = \sqrt{N!}\sum_i^N\langle\psi_h|\hat{f}(\mathbf{x_i})|\Psi_0\rangle = \sum_i^N\langle\psi_h|\hat{f}(\mathbf{x_i})|\psi_h\rangle = \sum_i^N\varepsilon_i \quad (11)$$

since all permuted terms will vanish. The inner product may be written in a more convenient notation. eq. (12).

$$\varepsilon_f = \sum_i\langle i|\hat{f}_i|i\rangle \quad (12)$$

1.6 The two body problem

For the two body part of the problem, we want to find eq. (13).

$$\varepsilon_v = \frac{1}{2}\sum_{i,j\neq i}^{N}\langle\psi_h|v_{ij}|\mathscr{A}\psi_h\rangle \quad (13)$$

The two body operator acts on two particles at a time. In the case of no permutations, we find eq. (14).

$$\langle\psi_h|v_{ij}|\psi_h\rangle = \prod_{k\neq(i,j)}^{N}\left(\int d\mathbf{x_k}|\psi_k(\mathbf{x_k})|^2\right) \int d\mathbf{x_i}d\mathbf{x_j}\left(\psi_i^*(\mathbf{x_i})\psi_j^*(\mathbf{x_j})\hat{v}(\mathbf{x_i},\mathbf{x_j})\psi_i(\mathbf{x_i})\psi_j(\mathbf{x_j})\right) \quad (14)$$

Again, the factor in front of the integral vanishes due to the normalization conditions. We now need to consider the permutations of particles i and j in the last Hartree function in the integral in the above equation (eq. (14)). The equation will now look like eq. (15).

$$\langle\psi_h|v_{ij}|\hat{P}_{ij}\psi_h\rangle = \int d\mathbf{x_i}d\mathbf{x_j}\left(\psi_i^*(\mathbf{x_i})\psi_j^*(\mathbf{x_j})\hat{v}(\mathbf{x_i},\mathbf{x_j})\psi_i(\mathbf{x_j})\psi_j(\mathbf{x_i})\right) \quad (15)$$

Where in eq. (15), $\hat{P}_{ij}$ is the permutation operator interchanging particle i and j. The integral in this equation is not zero, so we need to include it in the final evaluation.

In summary the two body problem will contribute with the term given in equation eq. (16).

$$\varepsilon_v = \frac{1}{2}\sum_{i,j\neq i}^{N}\langle\psi_h|\hat{v}_{ij}|(1-\hat{P}_{ij})\psi_h\rangle \quad (16)$$

Or, in a more convenient notation, eq. (16) can be expressed as in eq. (17).

$$\varepsilon_v = \frac{1}{2}\sum_{i,j\neq i}^{N}\left(\langle ij|\hat{v}|ij\rangle - \langle ij|\hat{v}|ji\rangle\right) \quad (17)$$

1.7 The full many body problem

In conclusion, we have now shown that the general form of the solution to the Schrödinger equation for the ground state SD is given by the sums eq. (18). Where i and j refers to particle i and j respectively.

$$\varepsilon_0 = \varepsilon_f + \varepsilon_v = \sum_i\langle i|\hat{f}_i|i\rangle + \frac{1}{2}\sum_{i,j\neq i}^{N}\left(\langle ij|\hat{v}|ij\rangle - \langle ij|\hat{v}|ji\rangle\right) \quad (18)$$

The last two innerproduct terms (in the two body problem) is commonly refered to as the *direct*- and the *exchange* term. The two terms are conventionally written in shorthand as $\langle ij|\hat{v}|ij\rangle - \langle ij|\hat{v}|ji\rangle = \langle ij|\hat{v}|ij\rangle_{AS}$, where AS refers to the antisymmetry.
It is a noteworthy relief that so many of the integrals contained in the previous discussions either normalized to one, or just

vanished, but we still haven't adressed the problem of how we may find, or know, the constituent wavefunctions in the Slater determinant.
At this point, we should also remind ourselves that no approximate techniques has been involved so everything outlined above is analytically exact. That is, given that the constituent wavefunctions ψ_i are the fermions true wavefunctions.

As most problems we set out to solve will contain a known one body part, with a spherically symmetric potential such as the hydrogen-like potential or the harmonic oscillator, we normally start out by knowing all these terms for a single particle. The challenge is the particle-particle interaction term. In the following we set out to tackle these problems.

1.8 The variational principle

The variational principle states that any trial wavefunction will represent an upper bound to the energy, thus allowing for an approximate scheme:

$$E_0 \leq \langle \Phi_{trial} | \hat{H} | \Phi_{trial} \rangle$$

We therefore need to adress the problem as how to find a reasonable trial wavefunction.

We have previously derived the form of the many body Hamiltonian (see eq. (10)), and by reviewing the one body and the two body problems, we have found that:

$$\varepsilon_0 = \langle \Psi_0 | \sum_i^N \hat{f}(\mathbf{x_i}) | \Psi_0 \rangle + \frac{N!}{2} \langle \Psi_0 | \sum_{i,j \neq i}^N \hat{v}(\mathbf{x_i}, \mathbf{x_j}) | \Psi_0 \rangle$$

Now, observing that the first term is the solution to the one body problem, we may consider the second term as a small pertubation due to the particle-particle interaction. If we are dealing with electrons in an atom, this is justified by the fact that the nucleus-electron interaction exceeds the electron-electron interaction (both being coulumb interactions, only the electrons being much more lighter compared to the protons).

This fact suggests that the constituent single particle wavefunctions Ψ_i could have a considerable overlap with a linear combination of the solutions to the one body problem, thus being such a reasonable trial wavefunction as we seek.

We may therefore proceed by setting up a trial SD where each constituent wavefunction in the SD is expanded in a basis set made up of the eigenstates to the one body problem. In effect, this means that we set $|\Psi_i\rangle = \sum_a^A C_{a,i} |\phi_{a,i}\rangle$. The expansion is truncated at the Ath term, as to keep the approximation computationally possible. Greek letter indices now refer to the expanded states, and we find by insertion eq. (19).

$$\varepsilon_0 = \sum_{\alpha,\beta}^A C^*_{\alpha,i} C_{\beta,i} \sum_i^N \langle \phi_i | \hat{f}(\mathbf{x_i}) | \phi_i \rangle + \\ \sum_{\alpha,\beta,\gamma,\delta}^A C^*_{\alpha,i} C^*_{\beta,j} C_{\gamma,i} C_{\delta,j} \frac{1}{2} \sum_{i,j \neq i}^N \langle \phi_{\alpha,i} \phi_{\beta,j} | \hat{v}(\mathbf{x_i}, \mathbf{x_j}) | \phi_{\gamma,i} \phi_{\delta,j} \rangle \tag{19}$$

We could now proceed by setting all the C constants to one, and invoking the variational principle to declare the resulting eigenenergy to be an upper bound to the states real ground state energy. We may, however, do even better: by finding the linear combination that gives the lowest possible eigenengergy, we can approximate the ground state and its energy to high precision.

1.9 Lagrange multiplers

In the upcoming derivation of the Hartree Fock equations, we will need the application of Lagrange multiplers. This is a brief outline of the method in general.

Lagrange multiplers is a method to find extremal values of functions subject to some constraint. The following example from a wikipedia article , see [1], outlines the process as follows.

Suppose we want to find the extremal of a function $f(x,y)$, subject to a constraint $g(x,y) = \alpha$. The functional we need to examine is then:

$$F\big(f(x,y), g(x,y)\big) = f(x,y) - \lambda \big(g(x,y) - \alpha\big)$$

Where the function λ is a variable called a *Lagrangian multipler*. A extremal of $f(x,y)$ will be a point on the constraint where $\nabla F(x,y,\lambda) = 0$. Solving this resulting set of equations will give the values of x and y on g where $f(x,y)$ is at an extremal. The method is readly generalizable to more variables and/or constraints.

1.10 Variational methods

Considering the coefficients C_α as variables. We now seek the combination of C's yielding the lowest possible eigenvalue for the Hamiltonian. The condition for this being true is that the set of eigenstates in the expansion remains orthonormal through the variation. That is, we must demand:

$$\langle i | j \rangle = \delta_{ij} = \sum_\alpha^A C^*_{\alpha,i} C_{\alpha,i}$$

We consider this to be our constraint. The so called Hartree-Fock energy is as we have already derived in eq. (19), but now assuming it's a function of the C's. Thus, the functional we want to minimize is given by eq. (20)·

$$F\big(E(C), \lambda\big) = E\big(C\big) - \sum_i^N \lambda_i \sum_\alpha^A C^*_{\alpha,i} C_{\alpha,i} \tag{20}$$

To find a minimum means to solve the set of equations from eq. (21).

$$\frac{\partial}{\partial C^*_{\alpha,i}} F\big(E(C),\lambda\big) = 0 \tag{21}$$

1.11 The Hartree-Fock equations

Solving the equations in the preceeding discussion, we end up with the Hartree-Fock equations. These are given by eqs. (22) to (26).

$$\begin{aligned} &\lambda_k C_{\alpha,i} = \\ &\sum_{\beta}^{A}\sum_{i}^{N} C_{\beta,i}\langle\alpha|\hat{f}(\mathbf{x_i})|\beta\rangle + \\ &\sum_{\beta,\gamma,\delta}^{A}\sum_{i}^{N} C^*_{\beta,j}C_{\gamma,i}C_{\delta,j}\langle\alpha\beta|\hat{v}(\mathbf{x_i},\mathbf{x_j})|\gamma\delta\rangle AS \end{aligned} \tag{22}$$

For a full derivation the reader is referred to [10]. This expression may be rewritten as

$$\begin{aligned} &\varepsilon_i^{HF} C_{\alpha,i} = \\ &\sum_{\beta}\langle\alpha|\hat{f}(\mathbf{x_i})|\beta\rangle + \sum_{j}^{N}\sum_{\gamma\delta} C_{j\gamma}C_{j\delta}\langle\alpha\gamma|\hat{v}(\mathbf{x_i})|\beta\delta\rangle \end{aligned} \tag{23}$$

Where the lagrangian multiplier now is identified as the *Hartree-Fock* energy. [2]. If we define the *Hartree-Fock matrix*

$$h^{HF}_{\alpha\beta} = \langle\alpha|\hat{f}(\mathbf{x_i})|\beta\rangle + \sum_{j}^{N}\sum_{\gamma\delta} C^*_{j\gamma}C_{j\delta}\langle\alpha\gamma|\hat{v}(\mathbf{x_i})|\beta\delta\rangle_{AS} \tag{24}$$

The expression can be further compacted down to

$$\sum_{\beta} h^{HF}_{\alpha\beta} C_{i\beta} = \varepsilon_i^{HF} C_{\alpha,i} \tag{25}$$

This finally forms our eigenvalue problem, namely:

$$\hat{h}^{HF}\hat{C} = \varepsilon^{HF}\hat{C} \tag{26}$$

The *field* referred to in the *self consistent field method*, is the potential set up by the linear expansion with the coefficients from the matrix $\hat{C}$ and the chosen basis. The *self consistency* is referring to the requirement of $\hat{C}$ being the same on both sides of the equation above. By making an initial guess for the coefficients, possibly using $C = \mathbb{I}$, and solving the eigenvalue problem, we obtain a new set of coefficients. The essence of the *Hartree-Fock* method is to do this iteratively, until the eigenvalues returned by the diagonalization differ from previously calculations below a given threshold.

2. Basis sets

2.1 Choosing a basis

When representing any function as a linear combination of a truncated (or incomplete) basis, it is a desireable feature of the set of basis functions that they have a considerable overlap with the function we want to represent. In theory, a perfect overlap yields the exact result.

In problems with few electrons, such as the helium atom or the harmonic oscillator, it is sometimes beneficial to populate the slater orbital with the wavefunctions from the one-electron problem. The particle-particle integrals may then be tackled by a precalculated table [2], and each orbitals interaction with the potential field is known partly from the one-electron problem. When the problem gets scaled up and more electrons interact, the overlap with the onebody solutions gets poorer, so that the wavefunctions no longer suffice.

We are then faced with the challenge of finding a basis set that is both suited to represent the orbitals and at the same time lets us calculate the integrals in an exact and efficient way. We could arguably choose *any* truncated basis set and implement some numerical approximative integrations, such as Monte Carlo integration, but this would at the same time introduce even more approximations in the problem.

Luckily for us, quantum chemists and physicists have already developed a multitude of basis sets with related frameworks for dealing with the integrations. [6] In our implementation of the algorithms in *C++* we have employed a tabulated one particle basis set [2], and a framework for dealing with so-called Gaussian Type Orbitals.

2.2 Gaussian Type Orbitals

Gaussian Type Orbitals *GTOs* are linear combinations of Gaussian functions, intended to approximate the *Slater Type Orbitals* (STO's). The STO's are functions used as atomic orbitals to approximate the probability density distribution of electrons "orbiting"some central symmetric potential, and works as a basis for the system of particles. The advantage of using GTO's is that it reduces the total computational calculation time used since the integral of two GTO's results in another GTO and thereby one can iterate for new Gaussian Type Orbitals. The normally intensive integral calculations of the basis functions becomes computationally more efficient.
A single Gaussian function that is used in a linear combination to build a contracted basis funciton, is called a *primitive* basis function. A general primitive basis function is shown in eqs. (27) and (28).

$$G_{ijk}(a,\mathbf{r_A}) = x_A^i y_A^j z_A^k exp(-ar_A^2) \tag{27}$$

$$G_{ijk}(a,\mathbf{r_A}) = G_i(a,x_A)G_j(a,y_A)G_k(a,z_A) \tag{28}$$

The factors in front of the exponential in eq. (27) is defined by $x_A G_i = G_{i+1}$, and the total angular momentum is

$$l = i + j + k$$

The primitives is used to form a *contracted* basis function, a *Contracted Gaussian Type Orbital*, and is on the form of eq. (29). These are also refered to as *STO-LG* basis funtions, because they imitate an STO, and is built up by L Gaussian types.

$$\chi^{CGTO}(\mathbf{r_A}, i, j, k) = \sum_{p=1}^{n} G_{ijk}(a_p, \mathbf{r_A}) \tag{29}$$

2.3 Properties of GTOs

There is some notable differences of the *GTOs* and wavefunctions arising from the solution of the Schrödinger equation , most importantly that the gaussian primitives is not orthonormal, so that

$$S_{ab} = \langle G_a | G_b \rangle \neq 0 \tag{30}$$

We will adress the issues arising from this when solving the *Hartree-Fock* equation later on, but for now we note that these overlapping quantities will need to be calculated to properly solve the *Hartree-Fock* equation with GTOs.

Another difference is that the *GTOs* are real-valued, as opposed to the complex-valued wavefunctions in Hilbert Space. A consequence is that the complex conjugation no longer plays in when evaluating the integrals. This introduces some nice symmetries, which we discuss some more in the section on optimizing the code.

Another nice feature of the GTOs is that the product of gaussian primitives equals a new gaussian primitive centered between the constituents, as presented in the section on the Gaussian product theorem (Equation 46).

This is one of the things that facilitates effective integral evaluations of products of gaussian primitives.

2.4 The Overlap Integral

The integral between two primitives such as eqs. (27) and (28) are given by eq. (31). The overlap integral between the primitive functions at hand results in a overlap matrix which we are going to come back to later. First we will have a look at how we may compute such an integral.

$$S_{ab} = \langle G_a | G_b \rangle = \langle G_i | G_j \rangle \langle G_k | G_l \rangle \langle G_m | G_n \rangle = S_{ij} S_{kl} S_{mn} \tag{31}$$

So let's have a look at a overlap integral such as $S_{ij} = \langle G_i | G_j \rangle$, which is for the x-component. In eq. (32) we rewrite the expression of the integral at hand in terms of a Hermite Gaussian using the theory of Gauss-Hermite quadrature, since as we shall see, this gives us a simple integral. From eq. (32) to eq. (33), we have used the Leibniz integration rule (see Appendix section 10.3). Since the integral is independent of **P** we are left with the Hermite functions.

$$S_{ij} = \int dx \Omega_{ij}(x) = \sum_{t=0}^{i+j} E_t^{ij} \Lambda_t(p, x_P) \tag{32}$$

$$S_{ij} = \sum_{t=0}^{i+j} E_t^{ij} \sqrt{\frac{\pi}{p}} \delta_{t0} = E_0^{ij} \sqrt{\frac{\pi}{p}} \tag{33}$$

From eq. (33), we easily see that the final overlap integral will look like eq. (34).

$$S_{ab} = E_0^{ij} E_0^{kl} E_0^{mn} \left(\frac{\pi}{p}\right)^{3/2} \tag{34}$$

The integral is simple enough, but with some simplifications somewhere, difficulties elsewhere emerge. We need to compute the coefficients in eq. (34). This is done by the recurrence relation in eq. (35), and knowing that $E_0^{0,0} = K_{AB}^x$

$$\begin{aligned} E_t^{i+1,j} &= \frac{1}{2p} E_{t-1}^{i,j} + X_{PA} E_t^{i,j} + (t+1) E_{t+1}^{i,j} \\ E_t^{i,j+1} &= \frac{1}{2p} E_{t-1}^{i,j} + X_{PB} E_t^{i,j} + (t+1) E_{t+1}^{i,j} \end{aligned} \tag{35}$$

2.5 Implementation of the Hermite coefficients

We are going to iterate for a set of coefficients, $E_t^{i,j}$. These coefficients are called Hermite coefficients, and has to be iterated for in a "cubic"way, suggesting that we first set $j = 0$, and find every $E_t^{i,0}$ first, since these are given by the previous and next t-value (see eq. (35)). It is important to know that if $t > i + j \Rightarrow E_t^{i,j} = 0$, and that if $t < 0 \Rightarrow E_t^{i,j} = 0$. This method is illustrated in the matrix in eq. (36).

$$\begin{pmatrix} t \mid i & 0 & 1 & 2 & 3 & \cdots \\ 0 & E_0^{0,0} & E_0^{1,0} & E_0^{2,0} & E_0^{3,0} & \cdots \\ 1 & 0 & E_1^{1,0} & E_1^{2,0} & E_1^{3,0} & \cdots \\ 2 & 0 & 0 & E_2^{2,0} & E_2^{3,0} & \cdots \\ 3 & 0 & 0 & 0 & E_3^{3,0} & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix} \tag{36}$$

The cube slices of (t, i) as in matrix eq. (36), where $j = 0$, can be set up for all j's. For $j = 0$ we will ofcourse have the same matrix, so it is not nesessary to run for this over again.

2.6 Integral evaluation of the kinetic energy

$$\begin{aligned} T_{ab} &= -\frac{1}{2} G_a \nabla^2 G_b \\ T_{ab} &= -\frac{1}{2} G_{ikm}(a, \mathbf{r}_A) \nabla^2 G_{jln}(b, \mathbf{r}_B) \\ T_{ab} &= -\frac{1}{2} \Big(T_{ij} S_{kl} S_{mn} + S_{ij} T_{kl} S_{mn} + S_{ij} S_{kl} T_{mn} \Big) \end{aligned} \tag{37}$$

Where in eq. (37), T_{ij} is given by eq. (38).

$$T_{ij} = 4b^2 S_{i,j+2} - 2b(2j+1)S_{i,j} + j(j-1)S_{i,j-2} \quad (38)$$

2.6.1 Integral evaluation of the Coulumb interaction

This part is rather cumbersome, and we would therefore like to refere to the lecture notes and lecture slides of [2] and [14] for further information. The main results are presented in this section.

When we are going to solve the one-body coulumb interaction, we want to find eq. (39). This integral is found through the Hermite coefficients found in eq. (36), and the components of a four-dimentional matrix R^n_{tuv}. This matrix is found through the iterations shown in eq. (41).

$$V_{ab} = \langle G_a | \frac{1}{r_C} | G_b \rangle = \sum_{tuv} E_t^{ij} E_u^{kl} E_v^{mn} R^0_{tuv} \quad (39)$$

Where r_C is the radial distance from the nuclei.
For the two-body integral, we have eq. (40).

$$\begin{aligned} &\langle G_a G_b | \frac{1}{r_{1,2}} | G_c G_d \rangle = \\ &\mathcal{K} \sum_{tuv} E^{ab}_{tuv} \sum_{\tau\mu\phi} E^{cd}_{\tau\mu\phi} (-1)^{(\tau\mu\phi)} R_{(t+\tau,u+\mu,v+\phi)}(\alpha, R_{PQ}) \quad (40) \\ &where,\ \mathcal{K} = \frac{2\pi^{5/2}}{pq\sqrt{p+q}} \end{aligned}$$

We have to iterate to find R^0_{tuv}. Where we find the first values from the Boys-function.

$$\begin{aligned} R^n_{(t+1,u,v)} &= tR^{n+1}_{(t-1,u,v)} + X_{PC} R^{n+1}_{(t,u,v)} \\ R^n_{(t,u+1,v)} &= uR^{n+1}_{(t,u-1,v)} + Y_{PC} R^{n+1}_{(t,u,v)} \\ R^n_{(t,u,v+1)} &= vR^{n+1}_{(t,u,v-1)} + Z_{PC} R^{n+1}_{(t,u,v)} \end{aligned} \quad (41)$$

As a summary we can list the progress of the one-body Coulumb interaction calculation:

1. Calculate the Hermite expansion coefficients $E^t_{ij}, E^t_{kl}, E^t_{mn}$
2. Calculate or find tabulated values for the Boys-function.
3. Iterate for the Hermite integrals $R^n_{(t,u,v)}$
4. Calculate the Cartesian integral $\langle G_a | \frac{1}{r_C} | G_b \rangle$

3. Implementation of equations

3.1 Overview

As we have presented in the previous sections, the Hartree-Fock (Roothaan) equations is a set of linear equations, suited for a linear algebra environment. The iterative schemes involving gaussian primitives, also demand flexible management of multidimensional arrays.

To efficiently employ the computational power available on laptop and desktop computers, we have therefore implemented all of the equations in the *C++* programming language. For matrix handling and diagonalization we utilize the Armadillo Library [13] - a flexible and effective linear algebra library. Some of its advantages is the handling of dynamic memory allocation for multidimensional objects, the efficient eig_sym[13] algorithm, and the comfortable ways of transferring data to other environments, such as Python.

There is a number of equations that needs to be implemented, most notable:

- The diagonalization of the Fock matrix
- The overlap integral
- The kinetic integral
- The coulomb integral for the electron-nucleus interaction
- The coulomb integral for the electron-electron interaction

There is also a number of support equations needed to fully implement the algorithm, such as:

- The tranformation of the Fock matrix
- The calculation of the energy
- The normalization of the coefficient matrix
- The nucleus-nucleus interaction (for molecular calculations)

3.2 A note on object handling

As a lot of the integrals involved requires multidimensional iterative methods, an important aspect of the implementation is the handling of such dynamic objects. To simplify this process, Armadillo is extensively used throughout the code. For example, the E^t_{ij}'s is initialized as shown in fig. 1.

```
    field <cube> Eij;
    Eij.set_size(3);
    for(int i=0; i<3; i++){
        Eij(i).zeros(N,N,N);
    }
```

Figure 1. Initializing a field of cubes.

The first line initializes a field of cubes, meaning that the object will contain a number of cubic arrays with three indices. The second lines sets the number of cubes in the field to 3, and inside the for-loop each of the cubes have its size set to $N*N*N$. Other Armadillo types includes matrices and vectors.

3.3 The iterative solver

The Hartree-Fock solver performs a number of operations alongside the diagonalization itself. An comprehensive overview is given in chapter 4 of Computational Physic [9], and our iterative code is structured in a similar manner. Codewise this means that we perform the following operations (in pseudocode):

```
Set up unit matrices
Set up initial density matrix
while Not converged do
    Previous energy = current energy
    Set up the Fock matrix
    Transform the Fock matrix to $F'$
    Solve eigenvalue problem for $F'$
    Normalize the new coefficient matrix
    Update the density matrix
end while
Return Energy
```

Figure 2. *Hartree-Fock* Algorithm. F' is the transposed *Hartree-Fock* matrix F.

3.4 Setting up the unit matrices

The GTOs are not orthonormal, as we required from the basis set we defined when deriving the *Hartree-Fock* equations. To take this fact into account, the *Hartree-Fock* equations must be rewritten to include the *overlap matrix* $\hat{S}$, consisting of all possible overlaps between the basis functions [9]:

$$\hat{h}^{HF} C_k = \varepsilon_k \hat{S} C_k \tag{42}$$

Our task is then to bring the overlap matrix to unit form, so that the problem takes the same form as the original equation. To do this we find the set of eigenvectors $\hat{V}$ that diagonalizes the overlap matrix. We may then transform the problem in the following manner:

$$\hat{V}\hat{h}^{HF}\hat{V}^T C_k = \varepsilon_k \hat{V}\hat{S}\hat{V}^T C_k \tag{43}$$

$$\hat{F}' C_k = \varepsilon_k C_k \tag{44}$$

When the Fock matrix is properly set up, the diagonalization is easily performed with armadillo as in fig. 3.

```
eig_sym(epsilon, Cprime, Fprime)
```

Figure 3. Calling the eigenvalue problem solver sig_sym.

This operation diagonalizes the matrix *F'* so that its eigenvectors and eigenvalues now are contained in *C'* and ε, respectively.

The inverse transformation of the coefficient matrix is ofcourse performed at a later stage to let us calculate the eigenenergies.

3.5 The initial guess for the expansion coefficients

A common choice is to set the initial expansion coefficients to unity, meaning that each expanded state is equally weighted. If a weighting closer to the final state is chosen, the algorithm will converge faster. A possible choice for the coefficients is therefore to weight states below the fermilevel more than those above. [2]

3.6 The integrals

All of the integrals are implemented in an iterative way as previously discussed. As many of the multidimensional tensors will be common to more than one set of gaussian primitives, we have choosen to create a class *integrator* that may be initialized for two primitives. The class can return the overlap- and kinetic integral, and by suppliying the position of the core it can return the coulomb interaction with the nucleus. By passing two more primtives, it can return the particle-particle interaction due to coulomb repulsion amongst the electrons.

Some computational aspects of the integrals is how to tackle cases where the indices are referring to non-existent elements in the array. As all of the elements except the initial ones rely on previous values, there will be cases where an array is called with negative indices. We chose to shift all indices by 1, so that no if-testing was required inside the iterative loops. A backside of this is that one must be very careful with indices acting as parameters in the expressions, as they will have to be subtracted by one to compensate for the shifting.

4. Structure of the Hartree-Fock code

4.1 Overview

An graphic overview of the full code is given in Figure 7. To ensure flexibility and reusability of different parts of the code, it is written in an object oriented way. The code consists of 3 classes for performing the calculations:

- The Hartree Fock Sover
- The Integrals
- The Boys Function

There is also a number of support classes to maintain the basis and datastructure:

- The Basis
- Gaussian Primitive
- Gaussian Contracted

```
//create a basis object ↩
    HydrogenSystem:
basis HydrogenSystem;
HydrogenSystem.init_H2({0,0,0}, ↩
    {1.4,0,0});
HydrogenSystem.init_integrals();
```

Figure 4. Creating the object HydrogenSystem.

```
Basis.init_H(...)
Basis.init_He(...)
Basis.init_Be(...)
Basis.init_O(...)
Basis.init_Ne(...)
Basis.init_Si(...)

Basis.init_H2(...)
Basis.init_He2(...)
Basis.init_Be2(...)
Basis.init_O2(...)
Basis.init_H2O(...)
Basis.init_H2Be(...)
```

Figure 5. Available basis initalizers.

4.2 The Basis

When running the code, the main task to perform is to (1) set up a basis, and (2) minimize this basis for a number of electrons. When initializing the basis class, there is a number of available choices of basisfunctions and systems. For example, one may initialize and set up an STO-3G basis for hydrogen molecule as in the codesnippet in fig. 4.

The second line in the codesnippet above creates two gaussian contracted centered on the vectors in the call to init_H2(vec3 A, vec3 B). It is this line of code that actually initializes the basis that we want to minimize. At the current moment, the systems shown in fig. 5 are callable.

In principle there is however no limit as to how many atoms and configurations are available, and more complex systems are easily set up by directly accessing the functions inside the basis class. For example, a system of multiple atoms interacting may be set up as in fig. 6.

Note the final function call above. When all basis functions are added to the full contracted basis, all the integrals may be calculated by calling `._init_integral()`, in effect setting up the coupled and uncoupled interactions. This is what we need to pass to the solver class.

4.3 The Solver

The solver performs the iterative Hartree-Fock algorithm. As it has no need to know how the integrals are calculated, it is natural to pass the basis class to the solver, and let the solver read out the needed objects for the calculations. When the

```
basis AnySystem;
AnySystem.add_atom("Be", {0,0,0});
AnySystem.add_atom("Ne", {0,0,1});
AnySystem.add_atom("H", {0,2,0});
AnySystem.add_atom("H", {0,−2,0});
AnySystem.init_integrals();
```

Figure 6. Adding atoms to system.

```
basis AnySystem;
(...)
hartreefocksolver System (AnySystem↩
    , nElectrons, nProtons);
double energy = System.solve();
```

basis is properly set up, the solver is utilized as in section 4.3.

In the example above, the basis object AnySystem is sent to the solver with two integer values *nElectrons, nProtons*, the number of electrons and protons in the system. The nProtons integer is a rudimentary parameter not yet removed from the class constructor as there is some dependencies in other parts of the code. The calculations does not rely on this number, so it may be anything.

The final line in the example above is the one that performs the actual Hartree-Fock calculations. It returns a double that is the minimized eigenenergy of the groundstate of the system. We have also implemented some functionality for analyzing the result, so that one may prompt the solver to create a density plot of the system. This functionality remains to be made more flexible, so that when looking at different parts of the systems in 3D or 2D, one needs to alter the hardcoded paramerers inside the solver itself. The functioncall for this is simply *System.createDensityMapstring filename*.

4.4 Restrictions in the code

The code has some limitations in its current implementation. By following the methods described by Thijssen [9], we explicitly included the spin dependence in the solver. The payoff is smaller matrices and faster calculations. At the same time this makes the code for doing open-shell atoms and molecules different than in the case of closed shells. Our code is a Restricted Hartree-Fock solver only, at least in its current state. This means that our solver only works for atoms and molecules where all electrons are paired, such as oxygen, neon, hydrogen and helium.

The structure of our *Hartree-Fock* program is illustrated in fig. 7. The program is split into the source code and the tests. The aim of the tests is ofcourse to test varius implementations, funcitons and classes in the source code up agains

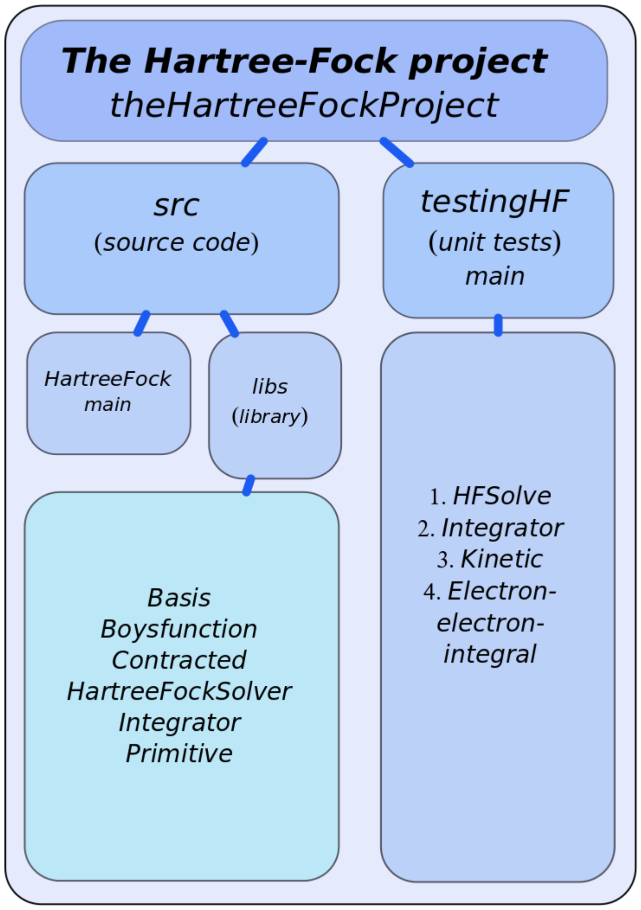


Figure 7. Structure of the Hartree-Fock project code. *theHartreeFockProject* is firstly divided into a source part (*src*) and a unit test part (*testingHF*). In the unit testing part, we have implemented a number of tests to compare specific parts of the code with pre-calculated values. The source code (src) is divided into the main program and the library, the library *libs* is used to choose a basis set, set up the basis, calculate the coulumb contribution for electron-electron repulsion, the nuceus-nucleus repulsion and the nucleus-electron attraction. Use the Hartree-Fock solver to minimize the energy of the system, and thereby find ε_0.

pre-calculated values or comparing calculations. The source code is further divided into the library which contains classes that performs specific tasks.

5. Unit testing

To benchmark the full solver with both the hydrogen like orbitals and the GTOs, a number of unit tests were performed during developement and on the final solver. We employed the UnitTest ++ framework [3]. These tests both ensured the integrity of the code while it was developed, and could serve as a measure in the debugging process.

We also did some manual benchmarking, by comparing our results for different atoms to similar calculations found in the literature . These results are shown in the table below.

Atom	STO-3G	Hydrogenlike	General HF
He	-2.8078	-2.8336	-2.904
Be	-14.3519	-14.5150	-14.67
O	-73.6618	-	-74.8093
Ne	-126.6045	-	-128.5266

The column “General HF” in the table above contains so-called experimental values from the best possible calculations available. For helium and beryllium, we obtained this values from the project description [2], while oxygen and neon was found in an article by Gill et. al. [12], describing a hybrid method using Hartree-Fock combined with Density Functional Theory.

A comparison of our solver to the experimental values are shown in the table below.

Atom	Discrepancy (%)
He	3.31
Be	2.17
O	1.53
Ne	1.50

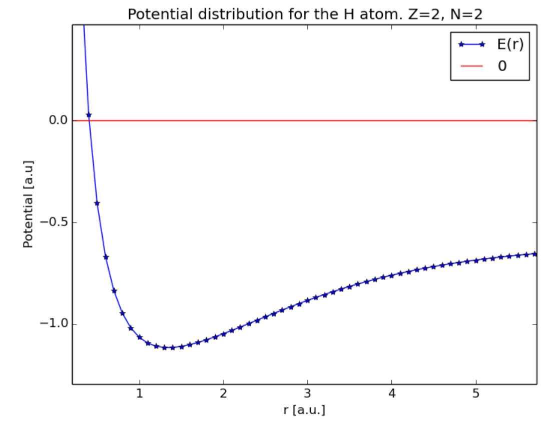
All of the above atoms are so-called closed shell atoms, meaning that no electrons occured in an unpaired state.

6. Experiment: Monatomic molecules

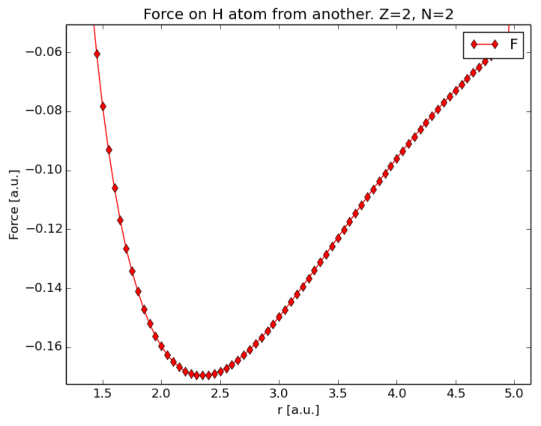
6.1 Calculating the ground state energy for a system of two atoms

We made a python script that calls the *Hartree-Fock code* with five commandline arguments, the number of protons Z, and electrons N, in the system, the type of interaction, an example could be to calculate for the *Hydrogen-Hydrogen* system, and finally the radial distance between the atoms. The program then uses the STO-3G basis set for the Hydrogen system and calls on the integrator to calculate the integrals, and then send the results to the *Hartree-Fock* solver which minimizes the energy of the system. This value is the one returned to our python script. We run for different distances between the atoms. Some results are plotted in fig. 9. An illustration of the experimental setup is shown in fig. 8.

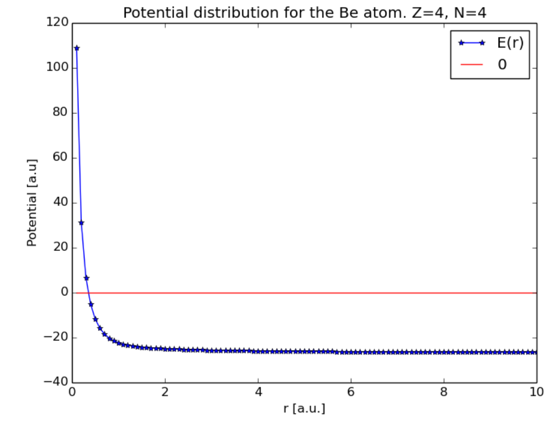
We see that for the H_2 system, a potential well emerge, meaning that the prefered distance for a Hydrogen atom from another is about $1.5a.u.$ Calculating the force exerted on eachother ($F = -\nabla U$), we see that the force is attractive from this point, and becomes most attractive at about $2.3a.u.$ This fits well with our intuition, since in nature, Hydrogen gas exists as H_2. Our experimental groundstate value lands at $-1.12hartrees$, while another *Hartree-Fock* simulation from reference [16] lands at $-1.13hartrees$.
When we look at the other experimental results we rapidly conclude that we must have a bug in our program for Beryllium and Oxygene. We observe a discontinuity for both the



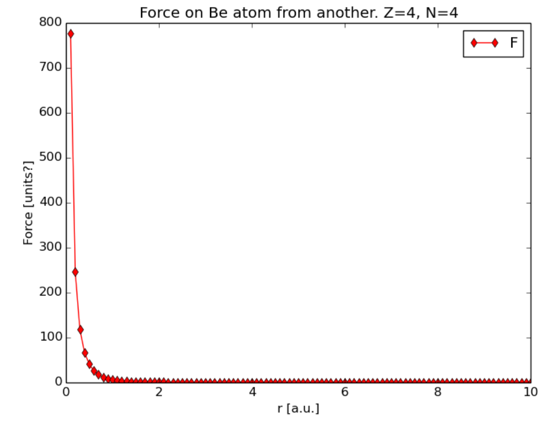
(a) Potential of Hydrogene



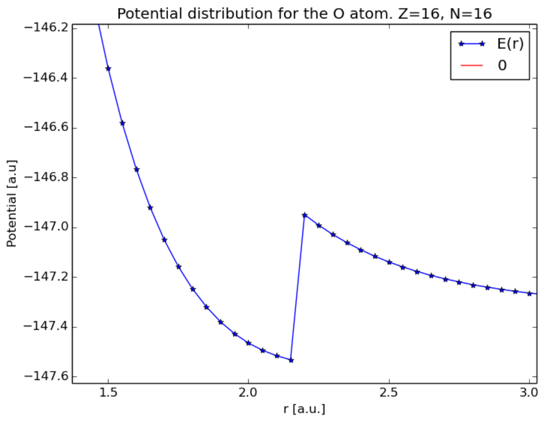
(b) Force from Helium on Hydrogene



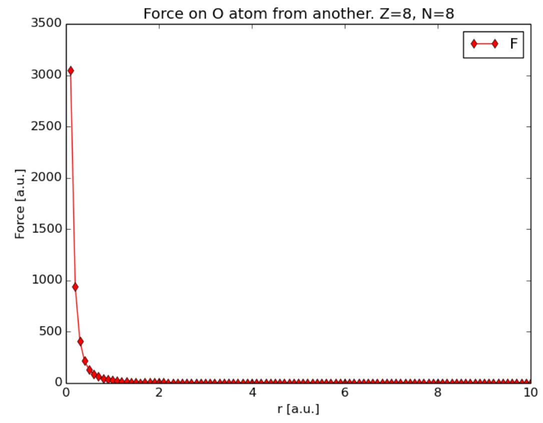
(c) Potential of Beryllium



(d) Force from Beryllium on Beryllium



(e) Potential of Oxygene



(f) Force from Oxygene on Oxygene

Figure 9. Figure 9a show the potential energy for *Hydrogene* as a function of the distance from the core. Figure 9b shows the force exerted from a *Hydrogene* atom on another *Hydrogene* atom as a function of the distance between them. The same yields for Figure 9c and 9d for *Beryllium* and Figure 9e and 9f is for *Oxygene*. The units of force can be obtained from the unitless value by multiplying with the constant $Eh/a_0 = 8.2387225(14) \times 10^{-8}N$, where Eh/a_0 is hartrees per bohr radius. Since we get the answere in atomic units $(a.u.)$, multiplying the answere with hartrees per bohr radius should give us units of force, measured in Newton $[N]$. According to [15], $E_h = \frac{\hbar^2}{m_e a_0} = m_e \left(\frac{e^2}{4\pi\varepsilon_0\hbar}\right)^2 = m_e c^2 \alpha^2 = \frac{\hbar c \alpha}{a_0}$.

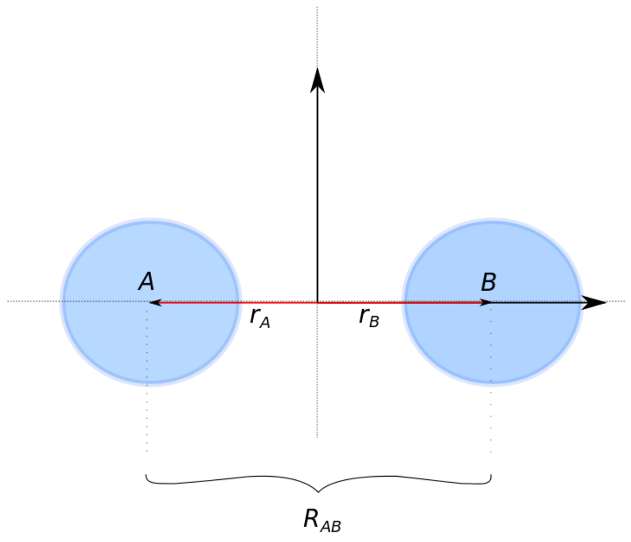


Figure 8. Illustration of the distance between two atoms. The origin is placed in the middle of the atoms A and B, where the position of the atoms $\mathbf{r_A} = -\mathbf{r_B}$. The distance between the atoms is $\mathbf{R_{AB}}$.

Beryllium system and in the Oxygene system. At first we did not expect that there would be any error in the Beryllium system because these should be repulsive. Our suspicion arised when we saw the plot for the Oxygen system, because also Oxygen exists as a molecule in nature, nameley O_2.
Hopefully this is just due to some typo's in the basis for the Oxygen and Beryllium. We did not find such an error for Helium.

7. Experiment: the H_2O molecule

7.1 Experimental background

The H_2O molecule is a well known system that often occurs as an example and benchmark in the literature [10] [9]. The system consist of one oxygen atom and two hydrogen atoms. From experimental data, we know it to have an angluar geometry of 104.5 degrees [9], and a bond length of 1.809 borh (a.u) between the oxygen and hydrogen atoms. [11] The Hartree-Fock limit for H_2O is at -76.065 a.u., while STO-3G is known to converge at -74.963 a.u. [10] The STO-3G basis is known to estimate the angle of H_2O to be 100.0 degrees.[11]

For our purpose, it is a interesting system because of its heterogenous atomic structure and the occurence of a more complex angular geometry. Any major discrepancies from experimental data in such complex calculations may potentially reveal flaws in the code. On the other hand, a code that efficiently reproduces the experimental data to a significant degree is a confirmation of both the theory involved and the accuracy of the algorithmic implementation. We have therefore performed a simple experiment to find the most energetically favourable configuration of a H_2O molecule.

7.2 Investigating the molecular structure

Our experimental setup is described in Figure 14. For a number of separations between the hydrogen atoms, a number of equal distanced locations was calculated for the oxygen. As our basis, we chose the STO-3G basis, since it occurs frequently in the literature for comparison. (See for example [11]). We then analyzed the data in python and created a contour plot in Figure 10, with half the separation of the hydrogens along the x-axis and the displacement of the oxygen along the y-axis.

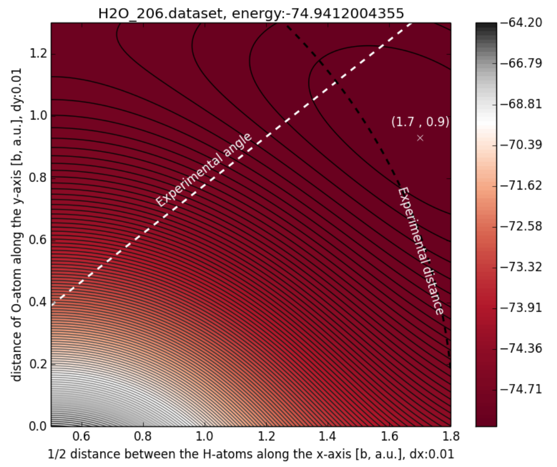


Figure 10. Resulting energy from a variety of configurations of the two H-atoms and one O-atom. Half the separation of the H-atoms is given on the x-axis, while the displacement of the O-atom is given on the y-axis. The system has an isoscale geometry. The black dotted curve marks configurations which fulfills the experimental bondlength, while the white dotted line marks configurations which fulfills the experimental angle. The energy minimum of -74.9412 occurs at $x = 1.7, y = 0.93$, resulting in an angle of 122.64 degrees.

As Figure 10 shows, the general features of the plot is as one would expect for the system. For very short distances, the nucleon-nucleon interaction combined with the electrons coulomb interaction with eachother creates a potential inversely proportional to the distance, resulting in a strong repulsion for smaller bond lengths. There exists an energy minimum somewhere along the way as the atoms is separated. This minimum does not occur along any of the axes, as it would for a molecule with 180 degrees angle between the bonds.

There is some discrepancies in our calculations from the experimental data. The resulting bondlength for our energy-minimized angle is about 1.93 bohr (a.u.), overestimated with about 7 %, while the angle is overestimated with just above 17 %. The energy of -74.9412 a.u. converged within 1.5% of the experimental value.

Somewhat unexpected is however the discrepancy from the literature regarding the STO-3G basis, where our implementation actually overestimated the angle by about 22 %. The bond length as aquired by other solvers using the STO-3G basis, is 1.871 bohr (a.u.) [11], meaning that our solver reproduces this result within 3.6 % of the value. Comparing our energy to other similar solvers using STO-3G [10], we find that we converge within 0.3 % of the value - a quite pleasing result.

7.3 The results for H_2O

We used the results from the calculations on H_2O to create a electron density plot, shown in fig. 11. The plot shows the (relative) probability of finding an eletron in the plane spanned by the molecule. The plot is not normalized.

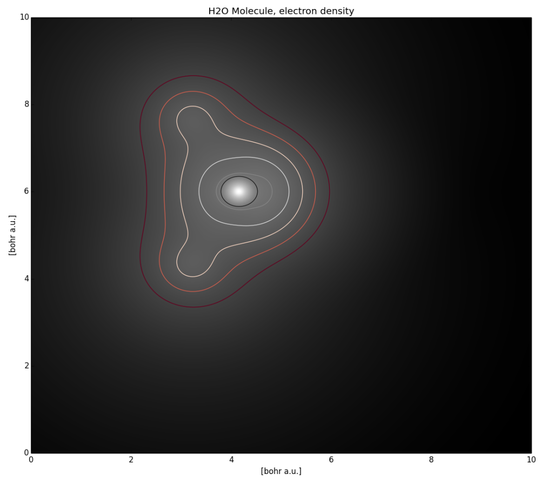


Figure 11. The electron density distribution in the plane spanned by the H_2O molecule. The plot is not normalized and for best visual result we plottet the logarithm of the density. A number of contours shows equal valued regions. The electrons are highly localized on the oxygen (with the highest number of protons), and the electron-electron repulsion creates very visible distortions to the otherwise circular distribution around the cores. This is especially visible for the contours around the oxygen molecule, where the electron is more likely to be found on the side furthest away from the hydrogens.

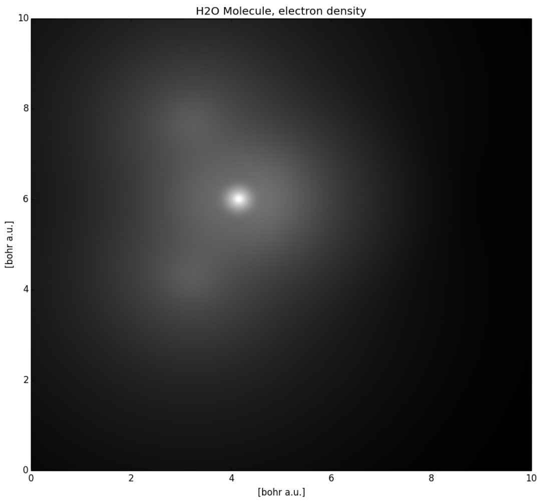


Figure 12. This figure shows the same density distribution as in Figure 11, but without the contours.

While the energy converged very close to other similar calculations, the resulting geometry seems quite odd. In our opinion, this may imply one or more of the following:

- There might be an error in the solver itself.
- The basis may not be properly set up.
- The experimental setup may be flawed.
- We might be interpreting the data wrong.

To possibly exclude the two latter points, a new crude calculation was performed. The two H-atoms was held fixed in a distance of 2.856 bohr (a.u.), while the energy field was probed with a O-atom. This resulted in Figure 13. The angle resulting from this plot is 98.74 degrees, very close to the STO-3G calculations referred to by Foresman & Frisch [11]. The energy for this calculation converged at -74.9410 a.u., within 0.03 % of the value obtained in other similar experiments. A final comparative test was performed by choosing the angle to be 100 degrees and the bond length to be 1.871, to exatly replicate the minimum described in the literature. For this calculation we obtained an energy of -74.4164 a.u., only within 0.8 % of the target value.

These results suggests that the inaccuracy (or possibly an error) lies within the solver itself, since there is no deviation from other similar solvers in the parameters used to call the solver. The two latter points in the checklist above may therefore be removed from our list of suspects.

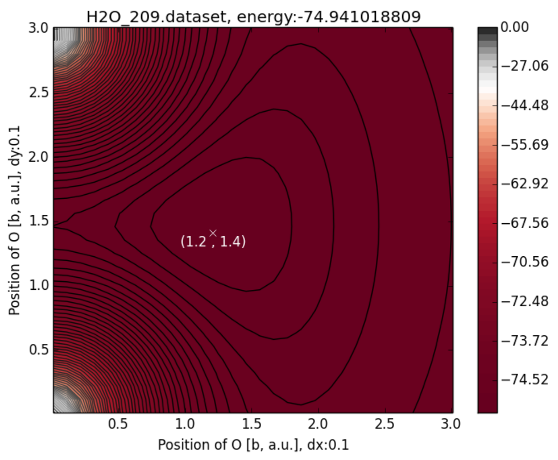


Figure 13. Two hydrogens are held fixed while an oxygen probes the energy field. The resolution is quite crude, using a mesh of 30 x 30 points, but we see a minimum occuring at x=1.21 and y = 1.41, marked with an X.

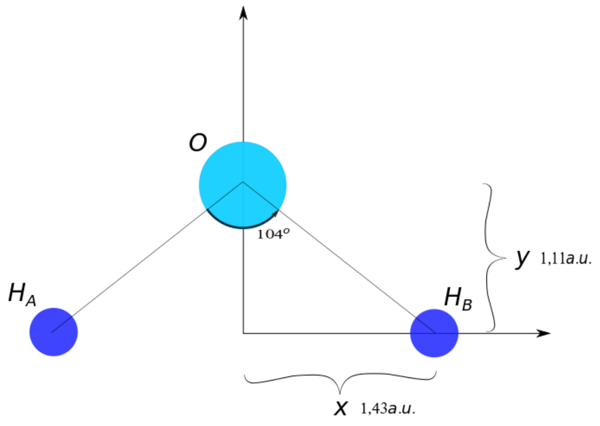


Figure 14. The setup of the H_2O molecule. The oxygene atom placed in a distance y along the y-axis, and the hydrogen atoms placed at distances x and $-x$ respectively along the x-axis. Approximate values of lengths and angle is shown in the figure.

8. Conclusions

In this report, we have shown how the Hartree-Fock Rothaan equations applied on gaussian basis sets may be used to reproduce quite accurate approximations to the groundstate of paired electron systems. Although the code seemingly run fine, we found some strong indications that there is a bug or algorithmic error somewhere inside the code. As the deviation seems to be very small, the bug would possibly not have been indicated had we not used our code to calculate more complex systems.

The calculations on H_2 shows, on the other hand, that both the spatial properties and ground state energy of molecules can be reasonable calculated by the Hartree-Fock method, as long as the system is set up properly.

We conclude that the method may potentially solve a multitude of quantum mechanical many-body systems, and serve as a starting point for more complex calculations.

9. Perspectives

The code has a great potential for improvement. There exists an extensive variety of physical systems to explore, and for each system there exists a number of properties that may be sought. A main goal for the code is to be flexible enough to investigate diverse systems, efficient to such a degree that a lot of interesting calculations may be performed on laptop or desktop computers, and finally above all that the results produced by the code should be exactly comparable to similar results in the literature.

9.1 Functionality

As the process of developing the code has given a great overview of the full code, we see a certain potential of object-orienting the code even more. Currently, a limited set of basis functions belonging to the STO-3G class are available hardcoded inside the basis class. The primitives making up these sets was downloaded from the Basis Set Exchange in a turbomole format, so we needed to write a normalization routine for the turbomole weighting.

A possible extension of the programs functionality would be to download all (or an decent selection) of the available sets in a file, and make this file accesible to the basis. This way, the basis could easily set up any atom or molecule and calculate its integrals. For this to work we would have to write a parser to import the turbomole data into the solver.

Another possible extension of the codes funtionality is to improve upon the electron density plotter. This part of the program may easily be made able to calculate volumetric plots that may be accessed by other programs, such as Python or Blender. More complex molecules may result in neat volumetric plots in 3D.

An even more flexible solver would have to implement the Unrestricted Hartree-Fock algorithm. This would increase the number of possible systems to investigate, and it would make it possible to calculate open shell atoms. The algorithm is briefly described in Thijssen [9], and for a more in-depth review the reader is referred to Szabo [10].

9.2 Optimizations

Even when doing rather small calculations with a basis of 5 contracted STO-3Gs (15 primitives), the code started to take up a lot of CPU time. No actual timing of the different parts of the code was performed, but it seems reasonable to assume that most of the time was spent setting up the intergals. (Diagonalizing a 15 by 15 matrix is not a very heavy operation). By optimizing the integration we may therefore speed up calculations significantly.

As mentioned when discussing the properties of GTOs, there is some interesting symmetries in the particle-particle integrals due to the gaussians being real. When integrating expectation values in quantum mechanics, one usually must take into account the complex nature of the wavefunctions. When dealing with the gaussian primitives, we may however use this fact to limit the number of integrations needed.

In practice, this means that we set it up as 45

$$\begin{aligned}
\langle pr|v|qs\rangle &= \langle qs|v|pr\rangle = \\
\langle qr|v|ps\rangle &= \langle ps|v|qr\rangle = \\
\langle rp|v|sq\rangle &= \langle sp|v|rq\rangle = \\
\langle rq|v|sp\rangle &= \langle sq|v|rp\rangle =
\end{aligned} \tag{45}$$

thus dividing the number of calculations needed for the particle-particle interaction by 8.

9.3 Debugging

The most pressing concern with the code in its present state is the odd results we found in the experiments. As we discussed previously, this is most likely caused by a bug or erraneous equation. To find this error we will need to perform more unit testing, with basis functions both located at the same core and on different cores. We have also discovered some discontinuities in the energy plots for molecules separating, and we are doing some work to isolate the parts of the code that causes this behaviour.

10. Appendix

10.1 Eigenvalue of the groundstate

The integralform of the eigenvalue equation for the groundstate of the many body time independent Schrödinger equation reads:

$$\begin{aligned}
&\varepsilon_0 = \\
&N! \int d\tau \Big(\prod_{i=1}^{N} \psi_i(\mathbf{x_i}) \Big)^* \hat{H} \mathscr{A} \Big(\prod_{i=1}^{N} \psi_i(\mathbf{x_i}) \Big)
\end{aligned}$$

With:

$$\hat{H} = \Big(\sum_i^N \hat{f}(\mathbf{x_i}) + \frac{1}{2} \sum_{i,j \neq i}^{N} \hat{v}(\mathbf{x_i}, \mathbf{x_j}) \Big)$$

10.2 Gaussian Product theorem

The Gaussian product theorem states that the product of two gaussian functions are also a gaussian function. The equations and components are specified in the following equations. First, the product of two gaussian functions in eq. (46), followed by the definitions of the components in eqs. (47) to (49).

$$e^{(-a|\mathbf{r}-\mathbf{A}|^2)} \cdot e^{(-a|\mathbf{r}-\mathbf{B}|^2)} = K_{AB} e^{(-p|\mathbf{r}-\mathbf{P}|^2)} \tag{46}$$

$$K_{AB} = exp\Big(- \frac{ab}{a+b} |\mathbf{A}-\mathbf{B}|^2 \Big) \tag{47}$$

$$\mathbf{P} = \frac{a\mathbf{A}+b\mathbf{B}}{a+b} \tag{48}$$

$$p = a+b \tag{49}$$

10.3 Lebniz integration rule

Diffrentiation of a integrand which is not a integration variable might be done by first diffrentiate the expression in the integration before doing the integration. eq. (50).

$$\frac{d}{dx}\Big(\int_{y0}^{y1} f(x,y)dy \Big) = \int_{y0}^{y1} f_x(x,y)dy \tag{50}$$

References

[1] Wikipedia: Lagrange multipliers
`http://en.wikipedia.org/wiki/Lagrange_multiplier`
(url available 30.04.14)

[2] Hjorth-Jensen, M
Slides from lectures, FYS4411, Spring 2014
`http://www.uio.no/studier/emner/matnat/fys/FYS4411/v14/slides2014.pdf`
(url available 27.03.2014)

[3] Unittest++ - A lightweigth unit testing framework.
`http://unittest-cpp.sourceforge.net`
(url available 31.05.2014)

[4] Gauss-Hermite quadrature
`http://en.wikipedia.org/wiki/Gauss%E2%80%93Hermite_quadrature`

[5] Mamedov, B.A. *On the Evaluation of Boys Function Using Downward Recursion Relation* Journal of Mathematical Chemistry Kluwer Academic Publishers - Plenum Publishers, 2004
`http://link.springer.com/article/10.1023%2FB%3AJOMC.0000044226.49921.f5`

[6] Wikipedia:Basis Sets (chemistry)
`en.wikipedia.org/wiki/Basis_set_(chemistry)`
(url available 31.05.2014)

[7] Griffiths, David J. *Introduction to Quantum Mechanics* Pearson Prentice Hall (Pearson Education, Inc.), 2005

[8] I. Shawitt and R. J. Bartlett. *Many-Body methods in Chemistry and Physics* Cambridge University Press, Cambridge, 2009

[9] Thijssen, J.M. *Computational Physics* Cambridge University Press, Cambridge, 2007

[10] A. Szabo and N.S. Ostlund, *Modern Quantum Chemistry* MacMillan, New York, 1982.

[11] J.B.Foresman and A.Frisch *Exploring Chemistry with Electronic Strucure* Gaussian, Inc., Pittsburg, PA, 1996

[12] Peter M. W. Gill, Benny G. Johnson, and John A. Pople An Investigation of the performance of a Hybrid of Hartree-Fock and Density Functional Theory John Wiley & Sons, Inc, 1992

[13] Armadillo: A *C++* linear algebra library
`http://arma.sourceforge.net/`
(url available 31.05.2014)

[14] Helgaker, T. *Molecular Integral Evaluation*
`http://folk.uio.no/helgaker/talks/SostrupIntegrals_06.pdf`
(url available 29.05.2014)

[15] Wikipedia: Hartree - atomic units of energy
`http://en.wikipedia.org/wiki/Hartree`
`http://physics.nist.gov/cgi-bin/cuu/Value?hr`
(url available 31.05.2014)

[16] Jules W. Moskowitz, M.H.Kalos
A new look at correlations in atomic and molecular systems. International Journal of Quantum Chemistry, John Wiley & Sons, Inc, 2004