FYS4411 - Report on Project 2

Christian Fleischer

June 17, 2016

Abstract

In this project we do Variational Monte Carlo (VMC) studies of many-body quantum dot systems. The aim is to calculate the ground state energy of two dimensional closed shell quantum dot systems with $N = 2, 6, 12, 20, \ldots$ electrons. The Metropolis algorithm with importance sampling is used for the Monte Carlo simulation, and a Slater determinant is included in the wave function. We use and manipulate this Slater determinant in order to make the simulations more efficient. The Steepest Descent method is used to optimize the variational parameters. To do statistical analysis of the numerical data and find an estimate to the error in the results we use the blocking method. In order to verify the implementation and benchmark the results we use results from closed form expressions and other research. When excluding the electron-electron repulsion and the correlation (Jastrow) factor the energies given by closed form expressions were reproduced exactly. When including the interactions the results were close to benchmarks from other research. In the two-body case, we found an estimate, $E = 3.00351$, to the ground state energy, which is fairly consistent with the known analytical result $E = 3$. In addition to the ground state energy, we also studied the one-body density and the significance of the correlations induced by the Jastrow factor. Finally we parallelized and vectorized the code, and did a performance analysis. From this we found that when doubling the amount of processors the computation time was halved, as expected, and the vectorization gave a speed-up of about a factor 3.5.

Contents

1 Introduction

The term quantum dots is used for nanoscale semiconductor devices which tightly confine electrons or electron holes. Due to their small size, quantum dots exhibit discrete quantum levels. Electronic properties of quantum dots, such as band gap, are highly tunable as a function of size and shape of the quantum dots. Therefore, quantum dots are of interest in several research applications such as LEDs, transistors and solar cells. (Ref. [5])

In this project we will use the Variational Monte Carlo (VMC) method to investigate several properties of quantum dots with $N = 2$, $N = 6$, $N = 12$ and $N = 20$ electrons, so-called closed shell systems. We will evaluate the ground state energy, one-body densities, expectation values of the kinetic and potential energies, and single-particle energies. To calculate the ground state energy we use a specific trial wave function, and simulate using the VMC method with importance sampling. For more than two electrons we use a trial wave function with a Slater determinant, and by manipulating this Slater determinant we improve the efficiency of the simulation. We have two variational parameters α and β. α is used in the Slater determinant, while β is for the correlation (Jastrow) factor. Both of these are optimized using the Steepest Descent method. We find a proper estimate to the errors in our calculation by using the blocking method. To benchmark our results we use both analytical results and results from other research. Finally we do a performance analysis, where we find the effectiveness of parallelization and vectorization on our code. We check that the efficiency improvements our code gets from parallelization are consistent with what we expect from VMC simulations considering these kind of simulations are well suited for parallel computing.

First, in section 2, we will look at the various methods used in this project, such as the Metropolis algorithm and manipulations of the Slater determinant. Afterwards the results of our work are listed and discussed in section 3. Finally a conclusion is given in section 4. Extensive calculations and an explanation of the code structure are given in the appendix, and we refer to the report from project 1 for explanations of methods that are reused in the same way for this project (e.g. blocking).

2 Methods

The system we are looking at contains electrons confined in a pure two-dimensional isotropic harmonic oscillator potential, with the following idealized total Hamiltonian

$$H = \sum_{i=1}^{N} \left(-\frac{1}{2}\nabla_i^2 + \frac{1}{2}\omega^2 r_i^2 \right) + \sum_{i<j} \frac{1}{r_{ij}}, \tag{1}$$

where we have used natural units ($\hbar = c = e = m_e = 1$) and all energies are in so-called atomic units a.u. Using the above Hamiltonian we will study systems of many electrons N as functions of the oscillator frequency ω. The standard harmonic oscillator part (unperturbed part) of the Hamiltonian is

$$H_0 = \sum_{i=1}^{N} \left(-\frac{1}{2}\nabla_i^2 + \frac{1}{2}\omega^2 r_i^2 \right), \tag{2}$$

while the repulsive interaction between two electrons is given by

$$H_1 = \sum_{i<j} \frac{1}{r_{ij}}, \tag{3}$$

where $r_{ij} = |\mathbf{r}_i - \mathbf{r}_j|$ is the distance between two electrons. The modulus of the positions of the electrons (for a given electron i) is defined as $r_i = \sqrt{r_{i_x}^2 + r_{i_y}^2}$.

2.1 Two-body Quantum Dot

For two electrons in a two-dimensional harmonic oscillator potential without interaction between the electrons, the energy of each electron is given by

$$\epsilon_n = \omega(n_x + n_y + 1). \tag{4}$$

For the ground state we have $n_x = n_y = 0$, and since there are two electrons the total (unperturbed) ground state energy is then $\epsilon_0 = 2\omega$. Since electrons are spin-$\frac{1}{2}$ particles they have two possible spin states, which means that both electrons can be in the ground state ($n_x = n_y = 0$) as long as they have different spin. One electron will have spin $+\frac{1}{2}$ and the other will have spin $-\frac{1}{2}$, so the total spin of the system will be zero. It makes sense for the ground state to have total spin zero, since the total spin contributes to the energy of the state, and the ground state is the state with lowest energy.

The perturbed trial wave function we will use for the two-body quantum dot has the form

$$\psi_T(\mathbf{r_1}, \mathbf{r_2}) = C \exp\left(-\alpha\omega(r_1^2 + r_2^2)/2\right) \exp\left(\frac{ar_{12}}{(1 + \beta r_{12})}\right), \tag{5}$$

where $a = 1$ when the electrons have anti-parallel spins and $a = 1/3$ when they have parallel spins. For the two-body quantum dot we are interested in the ground state for only two electrons, meaning the electrons will always have anti-parallel spins and $a = 1$ in this case. α and β are the variational parameters.

2.2 Many-body Quantum Dot

A closed shell system is a system where all used energy levels are filled. This is the case when our number of electrons equal a so-called magic number, i.e. $N = 2, 6, 12, 20, \ldots$. For a closed shell system with more than two electrons, we use the trial wave function

$$\psi_T(\mathbf{r_1}, \mathbf{r_2}, \ldots, \mathbf{r_N}) = Det\left(\phi_1(\mathbf{r_1}), \phi_2(\mathbf{r_2}), \ldots, \phi_N(\mathbf{r_N})\right) \prod_{i<j}^{N} \exp\left(\frac{ar_{ij}}{(1 + \beta r_{ij})}\right), \tag{6}$$

where Det is a Slater determinant. The single-particle wave functions are harmonic oscillator wave functions with the following form

$$\phi_{n_x,n_y}(x, y) = AH_{n_x}(\sqrt{\omega}x)H_{n_y}(\sqrt{\omega}y) \exp\left(-\omega(x^2 + y^2)/2.\right. \tag{7}$$

The functions $H_{n_x}(\sqrt{\omega}x)$ are Hermite polynomials, and A is a normalization constant. In this case, two chosen electrons can have either anti-parallel or parallel spins, so the value of a depends on which two electrons we are looking at.

2.3 Closed Form Expressions

We want to find the expectation value of the local energy, and having a closed form expression for this energy is convenient. The local energy is given by

$$E_L(\mathbf{R}) = \frac{1}{\Psi_T(\mathbf{R})} H\Psi_T(\mathbf{R}). \tag{8}$$

By finding the local energy using two different methods, one using the closed form expression and the other using a purely numerical approach, we can compare the results of the methods to each other

to verify that the program works properly. Using the closed form expression is also computationally faster than the numerical approach, which is convenient when doing large simulations. We can also use closed form expressions for the drift term in importance sampling

$$F = \frac{2\nabla\Psi_T}{\Psi_T}. \tag{9}$$

The necessary closed form expressions for the two-body quantum dot and the many-body quantum dot are calculated in the first appendix, in section A.1 and section A.2 respectively.

2.4 Benchmarks for Verifying the Implementation

In order to verify that the implementation works correctly we should compare our results with known benchmarks. In the unperturbed case the exact ground state energies are analytically known. We have that the energy for a given electron is given by eq. (4). Table 1 contains the total energy of unperturbed closed shell systems up to $N = 20$ electrons.

N	E
2	2ω
6	10ω
12	28ω
20	60ω

Table 1: Benchmarks of the energy E for unperturbed closed shell systems in two dimensions with N electrons.

For the perturbed case we use the benchmarks given in table 2 provided by Ref. [2]. From Ref. [1] we also have an analytical result for the ground state energy, $E = 3$, for the two-body case with $\omega = 1$.

N	E
2	3.000
6	20.1597
12	65.700
20	155.868

Table 2: Benchmarks of the energy E for perturbed closed shell systems in two dimensions with N electrons and $\omega = 1$.

2.5 Variational Monte Carlo

As described in project 1 (Ref. [7]), the Variational Monte Carlo method, which we use in this project as well, is based on the variational principle in quantum mechanics. The variational principle states that, given a Hamiltonian H and a trial wave function ψ_T, the expectation value $\langle H \rangle$ defined as

$$E[H] = \langle H \rangle = \frac{\int d\mathbf{R}\Psi_T^*(\mathbf{R},\boldsymbol{\alpha})H(\mathbf{R})\Psi_T(\mathbf{R},\boldsymbol{\alpha})}{\int d\mathbf{R}\Psi_T^*(\mathbf{R},\boldsymbol{\alpha})\Psi_T(\mathbf{R},\boldsymbol{\alpha})} \tag{10}$$

is an upper bound to the ground state energy E_0 of the Hamiltonian, i.e.

$$E_0 \leq \langle H \rangle. \tag{11}$$

Here our trial wave function is dependant on some variational parameters $\boldsymbol{\alpha}$, and the goal of the VMC method is to vary these parameters until we find the lowest possible value of $\langle H \rangle$ in order to get an estimate for the ground state energy E_0. In general, the integrals we have to compute to find $\langle H \rangle$ are multi-dimensional ones, so using traditional integration methods like Gauss-Legendre is too computationally expensive. Therefore we turn to Monte Carlo methods.

2.5.1 Metropolis Sampling

The general explanation of Metropolis sampling is given in the report from project 1 (Ref. [7]), so it will not be repeated here. However, when our trial wave function contains a Slater determinant we can manipulate it to improve the performance of the code by avoiding calculating the entire determinant at every Metropolis step. The ratio used in Metropolis sampling is given by (Ref. [3])

$$R = \frac{|\hat{D}(\mathbf{r}^{\text{new}})|}{|\hat{D}(\mathbf{r}^{\text{old}})|} \frac{\Psi_C^{\text{new}}}{\Psi_C^{\text{old}}}, \tag{12}$$

where $|\hat{D}|$ is the Slater determinant and Ψ_C is the correlation part of the wave function, while "new"and "old" refer to the position before and after a proposed move. If we move only one electron at the time, only a single row in the Slater determinant changes. By doing the calculations in section A.2, we can calculate the Slater determinant part of R using the following formula

$$R_{SD} = \sum_{j=1}^{N} \phi_j(\mathbf{r}_i^{\text{new}}) d_{ji}^{-1}(\mathbf{r}^{\text{old}}), \tag{13}$$

where $\phi_j(\mathbf{r}_i^{\text{new}})$ are the single particle wave functions evaluated at the new position, and $d_{ji}^{-1}(\mathbf{r}^{\text{old}})$ are the elements on the i-th column of the inverse Slater matrix $\hat{D}^{-1}$. In addition we need to maintain the inverse matrix by using an updating algorithm whenever a move is accepted. This updating algorithm is also covered in section A.2, and the equations used are

$$d_{kj}^{-1}(\mathbf{r}^{\text{new}}) = \begin{cases} d_{kj}^{-1}(\mathbf{r}^{\text{old}}) - \frac{d_{ki}^{-1}(\mathbf{r}^{\text{old}})}{R} \sum_{l=1}^{N} d_{il}(\mathbf{r}^{\text{new}}) d_{lj}^{-1}(\mathbf{r}^{\text{old}}) & \text{if } j \neq i \\ \frac{d_{ki}^{-1}(\mathbf{r}^{\text{old}})}{R} \sum_{l=1}^{N} d_{il}(\mathbf{r}^{\text{old}}) d_{lj}^{-1}(\mathbf{r}^{\text{old}}) & \text{if } j = i \end{cases}$$

For the correlation part of R we simply have

$$R_C = \frac{\Psi_C^{\text{new}}}{\Psi_C^{\text{old}}} = \prod_{i<j}^{N} \exp\left(f_{ij}^{\text{new}} - f_{ij}^{\text{old}}\right) \tag{14}$$

$$= \exp\left(\sum_{i<j}^{N} f_{ij}^{\text{new}} - f_{ij}^{\text{old}}\right), \tag{15}$$

where

$$f_{ij} = \frac{a r_{ij}}{(1 + \beta r_{ij})}. \tag{16}$$

For $i, j \neq k$, where k is the index of the moved electron, we get $f_{ij}^{\text{new}} - f_{ij}^{\text{old}} = 0$. Therefore we can simplify the expression to

$$\begin{aligned} R_C &= \exp\left(\sum_{i=1}^{k-1} f_{ik}^{\text{new}} - f_{ik}^{\text{old}} + \sum_{j=k+1}^{N} f_{kj}^{\text{new}} - f_{kj}^{\text{old}}\right) \\ &= \exp\left(\sum_{i=1, i\neq k}^{N} f_{ik}^{\text{new}} - f_{ik}^{\text{old}}\right). \end{aligned} \tag{17}$$

2.5.2 Metropolis-Hastings Algorithm (Importance Sampling)

As described in the report from project 1 (Ref. [7]), when using importance sampling, the new position for a suggested move is given by

$$y = x + DF(x)\Delta t + \xi\sqrt{\Delta t}, \tag{18}$$

where x is the old position, ξ is gaussian random variable and Δt is a chosen time step. D is the diffusion coefficient and is equal to $1/2$. The ratio used to accept or reject the move is given by

$$q(y, x) = \frac{G(x, y, \Delta t)|\Psi_T(y)|^2}{G(y, x, \Delta t)|\Psi_T(x)|^2}, \tag{19}$$

where G is the Green's function

$$G(y, x, \Delta t) = \frac{1}{(4\pi D\Delta t)^{3N/2}} \exp(-(y - x - D\Delta t F(x))^2/4D\Delta t). \tag{20}$$

For both of these calculations we need to find the *quantum force* F, which is given by

$$\mathbf{F} = 2\frac{1}{\Psi_T}\nabla\Psi_T. \tag{21}$$

We therefore need the gradient of the wave function. For the Slater determinant part, we can again use that moving only one particle changes only one row in the determinant, in order to reduce computation time. As described in section A.2, we then get

$$\frac{\vec{\nabla}_i|\hat{D}(\mathbf{r})|}{|\hat{D}(\mathbf{r})|} = \sum_{j=1}^{N} \vec{\nabla}_i \phi_j(\mathbf{r}_i) d_{ji}^{-1}(\mathbf{r}), \tag{22}$$

Following the calculations in A.2 we also get that the gradient for the correlation part is given by

$$\frac{\nabla_k \Psi_C}{\Psi_C} = \sum_{j\neq k} \frac{\mathbf{r}_{kj}}{r_{kj}} \frac{a}{(1 + \beta r_{kj})^2}. \tag{23}$$

2.5.3 Splitting the Slater Determinant

This section on splitting the Slater determinant follows Ref. [3]. The Slater determinant is on the form

$$\Phi(\mathbf{r}_1, \mathbf{r}_2, , \mathbf{r}_3, \mathbf{r}_4, \alpha, \beta, \gamma, \delta) = \frac{1}{\sqrt{4!}} \begin{vmatrix} \psi_{100\uparrow}(\mathbf{r}_1) & \psi_{100\uparrow}(\mathbf{r}_2) & \psi_{100\uparrow}(\mathbf{r}_3) & \psi_{100\uparrow}(\mathbf{r}_4) \\ \psi_{100\downarrow}(\mathbf{r}_1) & \psi_{100\downarrow}(\mathbf{r}_2) & \psi_{100\downarrow}(\mathbf{r}_3) & \psi_{100\downarrow}(\mathbf{r}_4) \\ \psi_{200\uparrow}(\mathbf{r}_1) & \psi_{200\uparrow}(\mathbf{r}_2) & \psi_{200\uparrow}(\mathbf{r}_3) & \psi_{200\uparrow}(\mathbf{r}_4) \\ \psi_{200\downarrow}(\mathbf{r}_1) & \psi_{200\downarrow}(\mathbf{r}_2) & \psi_{200\downarrow}(\mathbf{r}_3) & \psi_{200\downarrow}(\mathbf{r}_4) \end{vmatrix}, \tag{24}$$

which is zero because the spatial wave functions for the spin up and spin down states are equal. We rewrite the Slater determinant as the product of a spin up Slater determinant and a spin down Salter determinant and get

$$\begin{aligned}\Phi(\mathbf{r}_1,\mathbf{r}_2,,\mathbf{r}_3,\mathbf{r}_4,\alpha,\beta,\gamma,\delta) =& \det\uparrow(1,2)\det\downarrow(3,4) - \det\uparrow(1,3)\det\downarrow(2,4)\\ &- \det\uparrow(1,4)\det\downarrow(3,2) + \det\uparrow(2,3)\det\downarrow(1,4)\\ &- \det\uparrow(2,4)\det\downarrow(1,3) + \det\uparrow(3,4)\det\downarrow(1,2),\end{aligned} \tag{25}$$

where

$$\det\uparrow(1,2) = \frac{1}{\sqrt{2}}\begin{vmatrix}\psi_{100\uparrow}(\mathbf{r}_1) & \psi_{100\uparrow}(\mathbf{r}_2)\\ \psi_{200\uparrow}(\mathbf{r}_1) & \psi_{200\uparrow}(\mathbf{r}_2)\end{vmatrix}, \tag{26}$$

and

$$\det\downarrow(3,4) = \frac{1}{\sqrt{2}}\begin{vmatrix}\psi_{100\downarrow}(\mathbf{r}_3) & \psi_{100\downarrow}(\mathbf{r}_4)\\ \psi_{200\downarrow}(\mathbf{r}_3) & \psi_{200\downarrow}(\mathbf{r}_4)\end{vmatrix}. \tag{27}$$

This still gives a total determinant equal to zero. However, we want to avoid summing over spin variables when the interaction is independent of spin. With regards to the variational energy we can use the following approximation to this Slater determinant

$$\Phi(\mathbf{r}_1,\mathbf{r}_2,,\mathbf{r}_3,\mathbf{r}_4,\alpha,\beta,\gamma,\delta) \propto \det\uparrow(1,2)\det\downarrow(3,4), \tag{28}$$

and in general we can use the approximation

$$\Phi(\mathbf{r}_1,\mathbf{r}_2,\dots\mathbf{r}_N) \propto \det\uparrow\det\downarrow, \tag{29}$$

where the approximation to the Slater determinant is the product of a spin up part with the electrons with spin up only and a spin down part with the electrons with spin down. This ansatz is not antisymmetric when exchanging electrons with opposite spins, however the expectation value we get for the energy is the same as we get for the full Slater determinant, provided that the Hamiltonian is independent of spin. By factorizing the full determinant $|\hat{D}|$ into two smaller ones we can reduce the computation time. We identify the two determinants with an $\uparrow$ and a $\downarrow$

$$|\hat{D}| = |\hat{D}|_\uparrow \cdot |\hat{D}|_\downarrow \tag{30}$$

Combining the dimensionality of the smaller determinants yields the dimensionality of the full determinant. Doing the factorization allows us to calculate the ratio R as well as update the inverse Slater matrix separately for the two determinants

$$\frac{|\hat{D}|^{\text{new}}}{|\hat{D}|^{\text{old}}} = \frac{|\hat{D}|^{\text{new}}_\uparrow}{|\hat{D}|^{\text{old}}_\uparrow} \cdot \frac{|\hat{D}|^{\text{new}}_\downarrow}{|\hat{D}|^{\text{old}}_\downarrow}, \tag{31}$$

which reduces the computation time by a constant factor. The time reduction is greatest when the system has an equal amount of spin up and spin down electrons, so that both of the factorized determinants are half the size of the full determinant. This is the case for the ground state of closed-shell systems (i.e. systems with $2, 6, 12, 20, \dots$ electrons filling up the $1, 2, 3, 4, \dots$ lowest shells).

Setting up the (Split) Slater Determinant in C++

```
void ManyElectrons::setUpSlaterDet() {
    // Function for setting up the Slater determinant at the begining of the
        simulation.
    int n = 0;
    int nx = 0;
    int ny = 0;
    m_quantumNumbers = zeros<mat>(m_halfNumberOfParticles, 2);
    for (int p=0; p < m_halfNumberOfParticles; p++) {
        m_quantumNumbers(p, 0) = nx;      m_quantumNumbers(p, 1) = ny;
        if (ny == n) {
            n++;
            nx = n;
            ny = 0;
        }
        else {
            nx--;
            ny++;
        }
    }

    m_a = zeros<mat>(m_numberOfParticles, m_numberOfParticles);
    int half = m_halfNumberOfParticles;
    for (int i=0; i < m_numberOfParticles; i++) {
        for (int j=0; j < m_numberOfParticles; j++) {
            if ( ((i < half) && (j < half)) || ((i >= half) && (j >= half)) )
                { m_a(i,j) = 1./3; }
            else { m_a(i,j) = 1.; }
        }
    }

    m_spinUpSlater = zeros<mat>(m_halfNumberOfParticles,
        m_halfNumberOfParticles);
    m_spinDownSlater = zeros<mat>(m_halfNumberOfParticles,
        m_halfNumberOfParticles);

    for (int i=0; i < m_halfNumberOfParticles; i++) {
        for (int j=0; j < m_halfNumberOfParticles; j++) {
            nx = m_quantumNumbers(j, 0);
            ny = m_quantumNumbers(j, 1);
            double xSpinUp = m_system->getInitialState()->getParticles()[i]->
                getPosition()[0];
            double ySpinUp = m_system->getInitialState()->getParticles()[i]->
                getPosition()[1];
            double xSpinDown = m_system->getInitialState()->getParticles()[i+
                m_halfNumberOfParticles]->getPosition()[0];
            double ySpinDown = m_system->getInitialState()->getParticles()[i+
                m_halfNumberOfParticles]->getPosition()[1];
            m_spinUpSlater(i,j) = evaluateSingleParticleWF(nx, ny, xSpinUp,
                ySpinUp);
            m_spinDownSlater(i,j) = evaluateSingleParticleWF(nx, ny, xSpinDown
                , ySpinDown);
        }
```

```
    }

    m_spinUpSlaterInverse = m_spinUpSlater.i();
    m_spinDownSlaterInverse = m_spinDownSlater.i();
}
```

2.5.4 Blocking

For this project we use blocking to find an estimated error in our calculations of the expectation value of the ground state energies. This is done in the same way as for project 1 and the explanation of this method can be found in the report from project 1 (Ref. [7]).

2.5.5 The Steepest Descent Method (Gradient Descent)

In order to find approximate optimal values for the variational parameters α and β we use the steepest descent method similarly to what we did in project 1 (Ref. [7]). The difference from project 1 is that now we have two variational parameters instead of just one. We optimize each of the parameters as described in the report from project 1. In project 1 we stopped the optimization when the absolute value of the difference between a new value and an old value of the variational parameter was below a chosen tolerance, or when a set number of maximum iterations had been reached. Here we calculate that difference individually for each parameter and stop the optimization when the sum of these differences is below a chosen tolerance (or at max iterations).

2.6 Optimizing Performance

We use parallelization and vectorization of code to optimize the performance of our program. To parallelize our program we use MPI, and for vectorization we use the $O3$ compiler flag of the GNU C++ compiler. The $O3$ flag tells the compiler to make optimizations to the code where it is possible, in order to reduce computation time. By parallelizing our program, we make it possible to run the program on multiple processors at once. We can run the program on several processors on a single computer (typically 4 or 8 processors) to get some speedup. However, we can also run it on a super computer cluster in order to utilize even more processors (we'll use up to 64 processors in this project) and decrease computation time substantially. If the program is parallelized well we should see a speed-up of about a factor 2 when doubling the amount of processors. In general, Variational Monte Carlo simulations are simple to parallelize, so we should expect to achieve that amount of speed-up when running the code in parallel. When doing VMC calculation in parallel we have to be careful not to use too many processors for a small amount of MC cycles. If we have too many processors the amount of MC cycles for each processor might be too low to give good results. In addition the amount of MC cycles could end up being a non-integer number, which would be bad.

3 Results and Discussion

3.1 Unperturbed Ground State Energy

We calculate the ground state energies for $N = 2$, $N = 6$, $N = 12$ and $N = 20$ when excluding the Jastrow factor and the electron-electron repulsion. Using $\alpha = 1$ we get the results listed in table 3. (β is irrelevant since it only appears in the Jastrow factor.)

N	$\langle E \rangle$	σ^2
2	2	0
6	10	0
12	28	0
20	60	0

Table 3: The ground state energies, $\langle E \rangle$, when excluding the Jastrow factor and the electron-electron repulsion. N is the number of electrons and σ^2 is the variance. We see that the unperturbed single-particle energies are reproduced by our program, so our Slater determinant code should be correct.

From table 3 we see that our program reproduces the benchmarks from table 1. This indicates that we have implemented the Slater determinant correctly.

3.2 Optimizing the Variational Parameters with Steepest Descent

To find the optimal variational parameters we used the steepest descent method for the two-body quantum dot system. The optimization ended by reaching maximum iterations (at 100 iterations) instead of reaching the set tolerance, so the parameters could be optimized further by letting the optimization run longer. However, the change in the variational parameters was small for each iteration close to the end of the optimization, and the values provided by the optimization gave satisfactory results when used to calculate the ground state energy (as can be seen in the results below). The values of the variational parameters obtained by the optimization are listed in table 4.

α	β
0.98456	0.40691

Table 4: Estimates to the optimal variational parameters, obtained by the steepest descent method.

The variational parameters listed in table 4 are used for all of the following results.

3.3 Perturbed Ground State Energy

Using importance sampling with the analytical expression for the local energy, including blocking and optimization of variational parameters, we estimate the ground state energy as well as the expectation value of the kinetic energy and potential energy. This is done using $\omega = 0.01$, $\omega = 0.05$, $\omega = 0.1$, $\omega = 0.5$ and $\omega = 1.0$. The results are given in table 5.

From table 5 we see that for $\omega = 1$ the results are consistent with the benchmarks listed in table 2. The ground state energy for the two-body case with $\omega = 1$ is also close to the analytical solution $E = 3$ provided by Ref. [1].

N	ω	$\langle E \rangle$	$\langle KE \rangle$	$\langle PE \rangle$	σ_B
2	0.01	0.10606	0.01138	0.09468	4.2e-3
	0.05	0.28352	0.04845	0.23508	3.8e-3
	0.10	0.45829	0.08821	0.37008	3.2e-3
	0.50	1.66218	0.38479	1.27739	3.2e-3
	1.00	3.00351	0.77336	2.23016	3.9e-4
6	0.01	1.08344	0.06367	1.01977	2.5e-2
	0.05	2.59538	0.28086	2.31452	1.5e-2
	0.10	3.92962	0.53007	3.39955	1.3e-2
	0.50	11.8763	2.30907	9.56723	5.8e-3
	1.00	20.1550	4.36484	15.7902	8.7e-2
12	0.01	4.15725	0.17184	3.98541	5.1e-2
	0.05	9.11192	0.69518	8.41674	3.7e-2
	0.10	13.5882	1.27034	12.3178	3.5e-2
	0.50	39.3276	5.28910	34.0385	1.3e-2
	1.00	65.8395	9.90634	55.9332	2.6e-2
20	0.01	10.7243	0.33982	10.3845	2.0e-1
	0.05	22.3500	1.37373	20.9763	6.9e-1
	0.10	33.0928	2.48747	30.6053	6.6e-1
	0.50	94.1764	9.96665	84.2098	3.6e-2
	1.00	157.096	18.2104	138.886	6.1e-2

Table 5: Estimates to the ground state energies and expectation values of kinetic energy and potential energy. N is the number of electrons, ω is the harmonic oscillator frequency, $\langle E \rangle$ is the estimate to the ground state energy, $\langle KE \rangle$ is the expectation value of the kinetic energy and $\langle PE \rangle$ is the expectation value of the potential energy. σ_B is an estimation of the error in $\langle E \rangle$, provided by the blocking method. The number of Monte Carlo cycles used was 1e6. The results are consistent with the benchmarks provided by Ref. [2].

The quantum virial theorem is given as (Ref. [6]),

$$2\langle T \rangle = \sum_n \left\langle X_n \frac{dV}{dX_n} \right\rangle, \tag{32}$$

where X_n is the position operator of particle n, and for the pure harmonic oscillator potential it reduces to $2\langle T \rangle = \langle V \rangle$, where T is the kinetic energy and V is the potential energy. When comparing this to our results for $N = 2$, we see from table 5 that when $\omega = 1$ we have $3\langle T \rangle \sim \langle V \rangle$ and as we decrease ω the potential energy starts to dominate more and more. For $\omega = 0.01$ we end up with $9\langle T \rangle \sim \langle V \rangle$. When we decrease the effect of the harmonic oscillator by decreasing ω, the effects of the Coloumb interaction will start to dominate the energy of the system.

Figure 1 shows an example of the blocking plots that σ_B was read from. The σ_B used is the value of σ where the curve reaches a plateau.

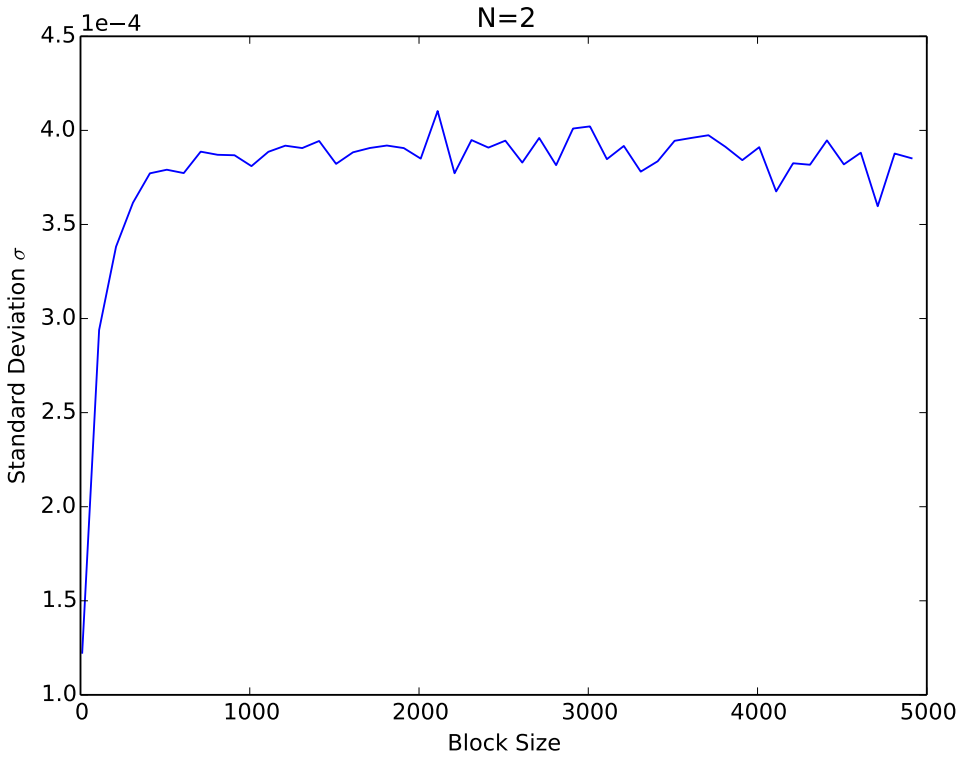
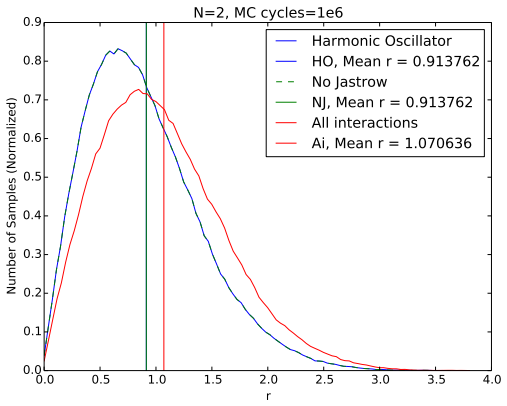


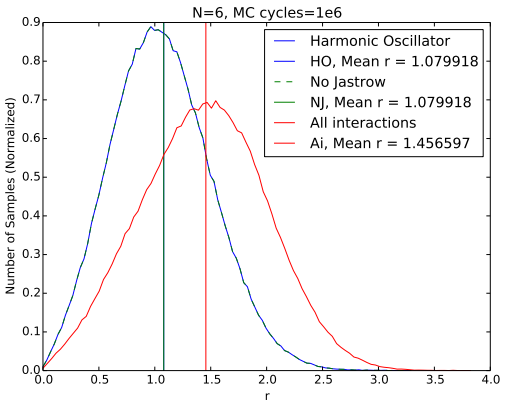
Figure 1: Blocking plot for $N = 2$ electrons when $\omega = 1$, when using 1e6 Monte Carlo cycles. The curve reaches a plateau at about $\sigma = 3.9\text{e-}4$ indicating that this is a good approximation of the error in the calculation of the ground state energy.

3.4 One-Body Density

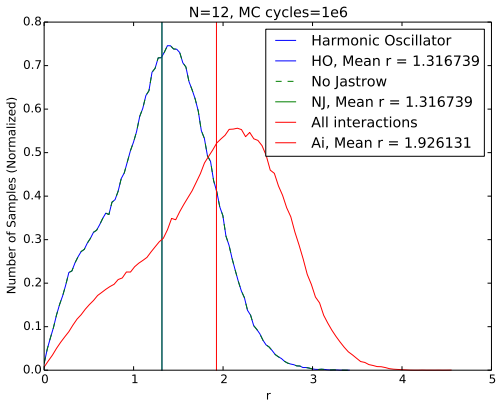
Figure 2 shows the results for the one-body density for $N = 2$, $N = 6$, $N = 12$ and $N = 20$ electrons with and without electron-electron repulsion and the Jastrow factor. We see that including the Jastrow factor shifts the curve to the right, and consequently increases the mean, by a significant amount, so the correlations induced by the Jastrow factor are fairly important. For $N = 12$ and $N = 20$ the curves have a somewhat strange form, possibly due to $1e6$ MC cycles not being enough for simulating that many electrons.



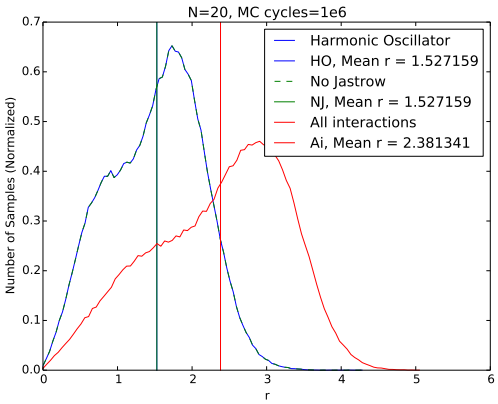
(a) N=2



(b) N=6



(c) N=12



(d) N=20

Figure 2: Plots for one-body density for $N = 2$, $N = 6$, $N = 12$ and $N = 20$ electrons, with the mean marked by a vertical line. The figures include the plots for a pure harmonic oscillator, a harmonic oscillator with electron-electron repulsion, and a harmonic oscillator with both electron-electron repulsion and the Jastrow factor. We see that the two harmonic oscillators without the Jastrow factor, have the exact same curve, but when we include the Jastrow factor the curve is shifted to the right. We also see that including the Jastrow factor increases the mean by a significant amount (about 17% for $N = 2$ and more for larger N), so the correlations induced by the Jastrow factor are fairly important. The plots for $N = 12$ and $N = 20$ are somewhat strange, which could indicate that we should use more than $1e6$ MC cycles when simulating that many electrons.

3.5 Performance Analysis

We want to check if our program is able to utilize vectorization and parallel computing well. For vectorization we use the compiler flag $O3$, while for no vectorization we use the flag $O0$. We run the program in parallel on a super computer cluster (Smaug), and use up to 64 processors. The computation times for various amount of processors with and without vectorization are listed in table 6.

	CP Time [s]	
n	$O0$	$O3$
1	545.218	158.773
2	276.395	80.4636
4	138.610	40.0966
8	69.4340	20.2077
16	35.0710	10.1289
32	17.7141	5.12122
64	8.91754	2.57613

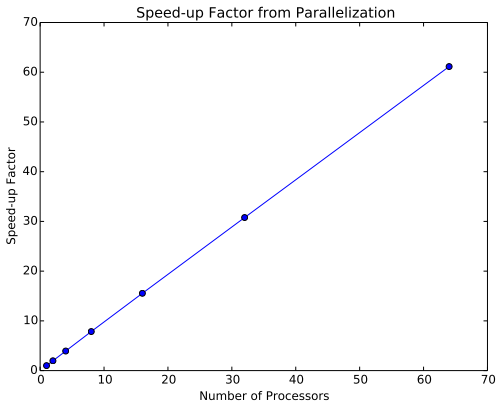
Table 6: The table lists computation times from running the same simulation, of $N = 6$ electrons, with $1e6$ MC cycles and with various forms of optimization. For each simulation the computation time is the average of five runs. n is the number of processors used to run the program in parallel. $O0$ is the compiler flag for no vectorization of code, while $O3$ is one of the compiler flags for vectorization of code. From the results we see that vectorizing the code with $O3$ makes the program run about 3.5 times faster. We also see that when doubling the number of processors, the computation time is about halved, which is what we expect for a properly parallelized VMC program.

From table 6 we see that when doubling the amount of processors, the computation time is indeed about halved. This indicates that our code is parallelized properly. We also see that using the $O3$ vectorization gives a speed-up of about a factor 3.5. The exact speed-up factors are listed in table 7. These are relative to the time used with $O0$ and 1 processor.

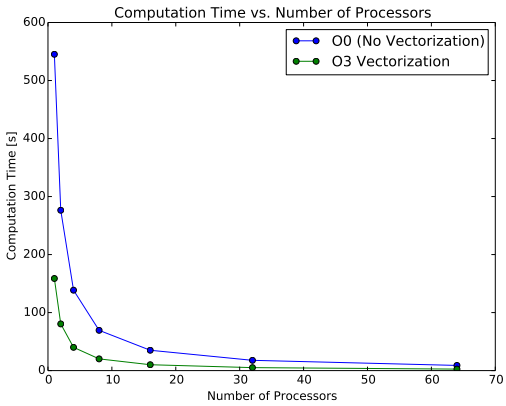
	Speed-up Factor	
n	$O0$	$O3$
1	1	3.43395
2	1.97260	6.77596
4	3.93347	13.5976
8	7.85232	26.9807
16	15.5461	53.8280
32	30.7788	106.463
64	61.1400	211.642

Table 7: Speed-up factors of parallelization and vectorization. n is the number of processes used, $O0$ is a compiler flag for no vectorization, while $O3$ is a flag for vectorization. Using only 1 processor and no vectorization ($O0$) is the initial speed before any optimizations and therefore naturally has the factor 1. All the other speed-up factors are relative to the situation where no optimization is used. We see that parallelization gives a speed-up factor close to the amount of processors used, while using the $O3$ vectorization flag gives an additional speed-up factor of about 3.45 (varies slightly with the amount of processors used). In total, using $O3$ and 64 processors, we see that we can get a speed-up factor of 211.642 which is very convenient when running long simulations.

Figure 3a shows that the speed-up factor from parallelization increases linearly with the amount of processors used, and that the speed-up factor is about equal to the number of processors. This is as we expect when doing Variational Monte Carlo. Figure 3b visualizes the data in table 6. We see that the computation time used decreases as $1/n$ with the amount of processors used n, both with and without vectorization, while vectorization gives some further decrease in computation time.



(a) Speed-up factor of parallelization



(b) Computation time with optimizations

Figure 3: Figure 3a gives the speed-up factor for using multiple processors in parallel, and the speed-up is relative to using only 1 processor. We see that the speed-up factor increases linearly with the number of processors and that the factor is almost equal to the number of processors used. This is as expected, since Variational Monte Carlo simulations in general are well suited for parallelization. Figure 3b shows the computation time when using parallelization both with and without vectorization. We see that the computation time decreases as $1/n$ with the number of processors n, and that using vectorization decreases the computation time further. Again this is as expected for parallelization of Variational Monte Carlo.

4 Conclusion

In this project we have used the Variational Monte Carlo method with importance sampling to estimate the ground state energy for quantum dot systems in two dimensions, with an amount of electrons equal to "magic numbers" ($N = 2, 6, 12, 20, \dots$). To improve efficiency when simulating many electrons, we have used and manipulated a Slater determinant in the trial wave function. The results reproduced known analytical results when excluding electron-electron repulsion and the correlation factor. When including these interactions, the results were close to benchmarks provided by Ref. [2], when using variational parameters $\alpha = 0.98456$ and $\beta = 0.40691$ found with the Steepest Descent method. For the two-body quantum dot system we found the following estimate to the ground state energy; $E = 3.00351$, which is fairly consistent with the known analytical result $E = 3$ from Ref. [1]. To estimate the error in the results we used the blocking method.

We also found the one-body density and used this to look at the significance of the correlations induced by the Jastrow factor. We then found that in the two-body case the Jastrow factor increased the mean by about 17%, and the effects were greater for larger systems. Finally we did a performance analysis where we looked at effects of parallelization and vectoriation on the computation time. For parallelization we found that doubling the amount of processors about halved the computation time, which is what we expect from VMC since it's suitable for parallelization. Using vectorization gave a speed-up of about a factor 3.5.

When optimizing the variational parameters the optimization could be improved by letting the the algorithm run longer. We could also use more advanced methods like the Conjugate Gradient method to optimize the parameters. In this project we simulated systems with up to $N = 20$ electrons, however the program is written in such a way that it can simulate closed shell systems with more electrons as well, without having to modify the program. The program can also be

modified to study other quantum mechanical systems using VMC.

References

[1] M. Taut, Phys. Rev. A **48**, 3561 - 3566 (1993).

[2] M. .L. Pedersen, G. Hagen, M. Hjorth-Jensen, S. Kvaal, and F. Pederiva, Phys. Rev. B **84**, 115302 (2011).

[3] M. Hjorth-Jensen, *Computational Physics 2: Variational Monte Carlo methods*, `https://github.com/CompPhysics/ComputationalPhysics2/blob/master/doc/pub/vmc/pdf/vmc-print.pdf`, Read: 16.06.2016.

[4] M. Hjorth-Jensen, *Computational Physics Lecture Notes Fall 2015*, `https://github.com/CompPhysics/ComputationalPhysics2/blob/master/doc/Literature/lectures2015.pdf`, Chapter 16.10, Read: 16.06.2016.

[5] `https://en.wikipedia.org/wiki/Quantum_dot`, Read: 16.06.2016.

[6] `https://en.wikipedia.org/wiki/Virial_theorem`, Read: 16.06.2016.

[7] C. Fleischer, *FYS4411 - Report on Project 1*, `https://github.com/christianfleischer/FYS4411_Projects/blob/master/Project1/Report/report.pdf`, Read: 16.06.2016

Appendices

URL to github folder with source files etc.
`https://github.com/christianfleischer/FYS4411_Projects/tree/master/Project2`

`SSH URL to github repository for all FYS4150 projects:`
`git@github.com:christianfleischer/FYS4411_Projects.git`

A Calculations of Closed Form Expressions

A.1 Two-body quantum dots

To find an expression for the local energy we need to find the Laplacian of the trial wave function

$$\psi_T(\mathbf{r_1}, \mathbf{r_2}) = C \exp\left(-\alpha\omega(r_1^2 + r_2^2)/2\right) \exp\left(\frac{ar_{12}}{(1+\beta r_{12})}\right). \tag{33}$$

We define

$$u = -\alpha\omega(r_1^2 + r_2^2)/2 \tag{34}$$

$$v = \frac{ar_{12}}{(1+\beta r_{12})} \tag{35}$$

Then we have

$$\nabla_1 \exp(u) = -\alpha\omega\mathbf{r}_1 \exp(u). \tag{36}$$

For $\nabla_1 \exp(v)$ we look at the first component of the gradient

$$\begin{aligned}\frac{\partial}{\partial x_1}\exp(v(r_{12})) =& \frac{\partial r_{12}}{\partial x_1}\frac{\partial}{\partial r_{12}}\exp(v(r_{12})) \\ =& \frac{(x_1 - x_2)}{r_{12}}\frac{\partial}{\partial r_{12}}\exp(v(r_{12})) \\ =& \frac{(x_1 - x_2)}{r_{12}}\frac{a}{(1+\beta r_{12})^2}\exp(v(r_{12})),\end{aligned} \tag{37}$$

which gives

$$\nabla_1 \exp(v) = \frac{(\mathbf{r}_1 - \mathbf{r}_2)}{r_{12}}\frac{a}{(1+\beta r_{12})^2}\exp(v). \tag{38}$$

For the second particle we have

$$\nabla_2 \exp(u) = -\alpha\omega\mathbf{r}_2 \exp(u) \tag{39}$$

$$\nabla_2 \exp(v) = -\nabla_1 \exp(v). \tag{40}$$

The gradients for the full wave function are then

$$\nabla_1\psi_T = \left(-\alpha\omega\mathbf{r}_1 + \frac{(\mathbf{r}_1 - \mathbf{r}_2)}{r_{12}}\frac{a}{(1+\beta r_{12})^2}\right) C\exp(u)\exp(v) \tag{41}$$

$$\nabla_2\psi_T = \left(-\alpha\omega\mathbf{r}_2 - \frac{(\mathbf{r}_1 - \mathbf{r}_2)}{r_{12}}\frac{a}{(1+\beta r_{12})^2}\right) C\exp(u)\exp(v). \tag{42}$$

To find the Laplacian we need

$$\nabla_1(-\alpha\omega\mathbf{r}_1) = -\alpha\omega d = -2\alpha\omega, \tag{43}$$

where d is the number of dimensions, in our case $d = 2$. We also need

$$\begin{aligned}\nabla_1\left(\frac{(\mathbf{r}_1-\mathbf{r}_2)}{r_{12}}\frac{a}{(1+\beta r_{12})^2}\right) =& \frac{a}{(1+\beta r_{12})^2}\nabla_1\frac{(\mathbf{r}_1-\mathbf{r}_2)}{r_{12}} + \frac{(\mathbf{r}_1-\mathbf{r}_2)}{r_{12}}\nabla_1\frac{a}{(1+\beta r_{12})^2}\\ =& \frac{d-1}{r_{12}}\frac{a}{(1+\beta r_{12})^2} + \frac{(\mathbf{r}_1-\mathbf{r}_2)}{r_{12}}\frac{(\mathbf{r}_1-\mathbf{r}_2)}{r_{12}}\frac{\partial}{\partial r_{12}}\frac{a}{(1+\beta r_{12})^2}\\ =& \frac{1}{r_{12}}\frac{a}{(1+\beta r_{12})^2} - \frac{2a\beta}{(1+\beta r_{12})^3}.\end{aligned} \tag{44}$$

We also have

$$\nabla_2(-\alpha\omega\mathbf{r}_2) = \nabla_1(-\alpha\omega\mathbf{r}_1), \tag{45}$$

and

$$\nabla_2\left(-\frac{(\mathbf{r}_1-\mathbf{r}_2)}{r_{12}}\frac{a}{(1+\beta r_{12})^2}\right) = \nabla_1\left(\frac{(\mathbf{r}_1-\mathbf{r}_2)}{r_{12}}\frac{a}{(1+\beta r_{12})^2}\right) \tag{46}$$

We end up with the Laplacians

$$\frac{\nabla_1^2\psi_T}{\psi_T} = \left(-2\alpha\omega + \frac{1}{r_{12}}\frac{a}{(1+\beta r_{12})^2} - \frac{2a\beta}{(1+\beta r_{12})^3} + \left(-\alpha\omega\mathbf{r}_1 + \frac{(\mathbf{r}_1-\mathbf{r}_2)}{r_{12}}\frac{a}{(1+\beta r_{12})^2}\right)^2\right) \tag{47}$$

$$\frac{\nabla_2^2\psi_T}{\psi_T} = \left(-2\alpha\omega + \frac{1}{r_{12}}\frac{a}{(1+\beta r_{12})^2} - \frac{2a\beta}{(1+\beta r_{12})^3} + \left(-\alpha\omega\mathbf{r}_2 - \frac{(\mathbf{r}_1-\mathbf{r}_2)}{r_{12}}\frac{a}{(1+\beta r_{12})^2}\right)^2\right). \tag{48}$$

The sum of the Laplacians in the Hamiltonian is then

$$\begin{aligned}\frac{\nabla_1^2\psi_T}{\psi_T} + \frac{\nabla_2^2\psi_T}{\psi_T} =& -4\alpha\omega + \frac{1}{r_{12}}\frac{a}{(1+\beta r_{12})^2} - \frac{2a\beta}{(1+\beta r_12)^3}\\ &+ \alpha^2\omega^2(r_1^2+r_2^2) + \frac{2a^2}{(1+\beta r_{12})^4} - \alpha\omega r_{12}\frac{2a}{(1+\beta r_{12})^2}.\end{aligned} \tag{49}$$

The complete expression for the local energy is then

$$\begin{aligned}E_L =& \frac{1}{\psi_T}H\psi_T\\ =& 2\alpha\omega + \frac{1}{r_{12}}\frac{a}{(1+\beta r_{12})^2} - \frac{2a\beta}{(1+\beta r_12)^3} - \frac{1}{2}\alpha^2\omega^2(r_1^2+r_2^2)\\ &- \frac{a^2}{(1+\beta r_{12})^4} + \alpha\omega r_{12}\frac{a}{(1+\beta r_{12})^2} + \frac{1}{2}\omega^2(r_1^2+r_2^2) + \frac{1}{r_{12}}\\ =& 2\alpha\omega + \frac{1}{2}\omega^2(r_1^2+r_2^2)(1-\alpha^2) - \frac{2a\beta}{(1+\beta r_12)^3}\\ &- \frac{a^2}{(1+\beta r_{12})^4} + \left(\alpha\omega r_{12} + \frac{1}{r_{12}}\right)\frac{a}{(1+\beta r_{12})^2} + \frac{1}{r_{12}}.\end{aligned} \tag{50}$$

A.2 Many-body quantum dots

These calculations follow Ref. [3]. In the many-body case we have the trial wave function

$$\psi_T(\mathbf{r_1}, \mathbf{r_2}, \ldots, \mathbf{r_N}) = Det\left(\phi_1(\mathbf{r_1}), \phi_2(\mathbf{r_2}), \ldots, \phi_N(\mathbf{r_N})\right) \prod_{i<j}^{N} \exp\left(\frac{ar_{ij}}{(1+\beta r_{ij})}\right), \tag{51}$$

where Det is a Slater determinant, and the single-particle wave functions are the harmonic oscillator wave functions given by

$$\phi_{n_x,n_y}(x,y) = AH_{n_x}(\sqrt{\omega}x)H_{n_y}(\sqrt{\omega}y)\exp\left(-\omega(x^2+y^2)/2.\right. \tag{52}$$

A is a normalization constant, while the functions $H_{n_x}(\sqrt{\omega}x)$ are Hermite polynomials. For $N = 6$ electrons we need the Hermite polynomials for $n_x = 0, 1$ and $n_y = 0, 1$, for $N = 12$ we need to include the $n_x, n_y = 2$ Hermite polynomials, and for $N = 20$ we also need the Hermite polynomials for $n_x, n_y = 3$. When evaluating the trial wave function, the calculation of the gradient and the Laplacian of an N-particle Slater determinant is likely to be most time-consuming. This is because we have to differentiate with respect to all spatial coordinates of all electrons. We can improve the efficiency of the calculation by moving only one electron at the time. When we then differentiate the Slater determinant with respect to a given coordinate of that electron, only one row in the corresponding Slater matrix is changed. This means that we don't have to recalculate the entire determinant at every Metropolis step. Instead we use an algorithm which requires us to keep track of the inverse of the Slater matrix.

The matrix elements of the Slater matrix $\hat{D}$ are given by

$$d_{ij} = \phi_j(x_i), \tag{53}$$

where $\phi_j(\mathbf{r_i})$ is a single particle wave function. x_i is one of the spatial coordinates of the given particle, while j indicates the quantum numbers (n_x and n_y in our case). The inverse of $\hat{D}$ can be expressed by its determinant $|\hat{D}|$, and its cofactors C_{ij} as follows

$$d_{ij}^{-1} = \frac{C_{ji}}{|\hat{D}|}, \tag{54}$$

where the interchanged indices of C_{ji} means that the cofactor matrix should be transposed. Assuming $\hat{D}$ is invertible we have

$$\sum_{k=1}^{N} d_{ik}d_{kj}^{-1} = \delta_{ij}. \tag{55}$$

We define the ratio R, between $|\hat{D}(\mathbf{r}^{\text{new}})|$ and $|\hat{D}(\mathbf{r}^{\text{old}})|$, which by definition can be expressed as

$$R \equiv \frac{|\hat{D}(\mathbf{r}^{\text{new}})|}{|\hat{D}(\mathbf{r}^{\text{old}})|} = \frac{\sum_{j=1}^{N} d_{ij}(\mathbf{r}^{\text{new}})C_{ij}(\mathbf{r}^{\text{new}})}{\sum_{j=1}^{N} d_{ij}(\mathbf{r}^{\text{old}})C_{ij}(\mathbf{r}^{\text{old}})}. \tag{56}$$

If we move only one electron at a time, $\mathbf{r}^{\text{new}}$ and $\mathbf{r}^{\text{old}}$ differ only by the position of that one, i-th, electron, which means $\hat{D}(\mathbf{r}^{\text{new}})$ and $\hat{D}(\mathbf{r}^{\text{old}})$ differ only by the entries of the i-th row. The i-th row of a cofactor matrix $\hat{C}$ is independent of the entries in the i-th row of the corresponding matrix $\hat{D}$. In our case this means that the i-th row of $\hat{C}(\mathbf{r}^{\text{new}})$ and $\hat{C}(\mathbf{r}^{\text{old}})$ must be equal, so we have

$$C_{ij}(\mathbf{r}^{\text{new}}) = C_{ij}(\mathbf{r}^{\text{old}}) \quad \forall\ j \in \{1, \ldots, N\}. \tag{57}$$

We use eq. (54) and eq. (57) with eq. (56) in order to obtain

$$R = \frac{\sum_{j=1}^{N} d_{ij}(\mathbf{r}^{\text{new}})C_{ij}(\mathbf{r}^{\text{old}})}{\sum_{j=1}^{N} d_{ij}(\mathbf{r}^{\text{old}})C_{ij}(\mathbf{r}^{\text{old}})} = \frac{\sum_{j=1}^{N} d_{ij}(\mathbf{r}^{\text{new}})d_{ji}^{-1}(\mathbf{r}^{\text{old}})}{\sum_{j=1}^{N} d_{ij}(\mathbf{r}^{\text{old}})d_{ji}^{-1}(\mathbf{r}^{\text{old}})}, \tag{58}$$

where, by eq. (55), the denominator of the rightmost expression is unity, and we end up with

$$R = \sum_{j=1}^{N} d_{ij}(\mathbf{r}^{\text{new}})d_{ji}^{-1}(\mathbf{r}^{\text{old}}) = \sum_{j=1}^{N} \phi_j(\mathbf{r}_i^{\text{new}})d_{ji}^{-1}(\mathbf{r}^{\text{old}}). \tag{59}$$

This means that if we only move the i-th electron, the ratio R is given by the dot product between the vector, $(\phi_1(\mathbf{r}_i^{\text{new}}), \ldots, \phi_N(\mathbf{r}_i^{\text{new}}))$, of single particle wave functions evaluated at the new position, and the i-th column of the inverse matrix $\hat{D}^{-1}$ evaluated at the original position.

We need to maintain the inverse matrix, so if the new position $\mathbf{r}^{\text{new}}$ is accepted we need to use an algorithm for updating the inverse matrix. We start by updating all but the i-th column of $\hat{D}^{-1}$. For each column $j \neq i$, we calculate

$$S_j = (\hat{D}(\mathbf{r}^{\text{new}}) \times \hat{D}^{-1}(\mathbf{r}^{\text{old}}))_{ij} = \sum_{l=1}^{N} d_{il}(\mathbf{r}^{\text{new}})d_{lj}^{-1}(\mathbf{r}^{\text{old}}), \tag{60}$$

then we calculate the new elements of the j-th column of $\hat{D}^{-1}$ as follows

$$d_{kj}^{-1}(\mathbf{r}^{\text{new}}) = d_{kj}^{-1}(\mathbf{r}^{\text{old}}) - \frac{S_j}{R} d_{ki}^{-1}(\mathbf{r}^{\text{old}}) \quad \begin{array}{l} \forall \;\; k \in \{1, \ldots, N\} \\ j \neq i \end{array}. \tag{61}$$

The last step is to update the i-th column of $\hat{D}^{-1}$ using the following equation

$$d_{ki}^{-1}(\mathbf{r}^{\text{new}}) = \frac{1}{R}\, d_{ki}^{-1}(\mathbf{r}^{\text{old}}) \quad \forall \;\; k \in \{1, \ldots, N\}. \tag{62}$$

Only the i-th row of the Slater matrix changes when differentiating the Slater determinant with respect to the coordinates of a single particle $\mathbf{r}_i$ as well, which means we can calculate the gradient and the Laplacian as follows

$$\frac{\vec{\nabla}_i |\hat{D}(\mathbf{r})|}{|\hat{D}(\mathbf{r})|} = \sum_{j=1}^{N} \vec{\nabla}_i d_{ij}(\mathbf{r}) d_{ji}^{-1}(\mathbf{r}) = \sum_{j=1}^{N} \vec{\nabla}_i \phi_j(\mathbf{r}_i) d_{ji}^{-1}(\mathbf{r}) \tag{63}$$

and

$$\frac{\nabla_i^2 |\hat{D}(\mathbf{r})|}{|\hat{D}(\mathbf{r})|} = \sum_{j=1}^{N} \nabla_i^2 d_{ij}(\mathbf{r}) d_{ji}^{-1}(\mathbf{r}) = \sum_{j=1}^{N} \nabla_i^2 \phi_j(\mathbf{r}_i)\, d_{ji}^{-1}(\mathbf{r}). \tag{64}$$

Therefore in order to calculate the derivatives of the Slater determinant, we only need the derivatives of the single particle wave functions and the elements of the inverse Slater matrix.

The expectation value of the kinetic energy for electron i expressed in atomic units is

$$\langle \hat{K}_i \rangle = -\frac{1}{2} \frac{\langle \Psi | \nabla_i^2 | \Psi \rangle}{\langle \Psi | \Psi \rangle}, \tag{65}$$

and we have that

$$K_i = -\frac{1}{2}\frac{\nabla_i^2\Psi}{\Psi}. \tag{66}$$

To find the kinetic energy we need the Laplacian of the wave function. We define the Slater determinant part of the wave function as Ψ_D and define the correlation part (Jastrow factor) as Ψ_C. The Laplacian is then

$$\begin{aligned}\frac{\nabla^2\Psi}{\Psi} &= \frac{\nabla^2(\Psi_D\,\Psi_C)}{\Psi_D\,\Psi_C} = \frac{\nabla\cdot[\nabla(\Psi_D\,\Psi_C)]}{\Psi_D\,\Psi_C} = \frac{\nabla\cdot[\Psi_C\nabla\Psi_D + \Psi_D\nabla\Psi_C]}{\Psi_D\,\Psi_C} \\ &= \frac{\nabla\Psi_C\cdot\nabla\Psi_D + \Psi_C\nabla^2\Psi_D + \nabla\Psi_D\cdot\nabla\Psi_C + \Psi_D\nabla^2\Psi_C}{\Psi_D\,\Psi_C} \\ &= \frac{\nabla^2\Psi_D}{\Psi_D} + \frac{\nabla^2\Psi_C}{\Psi_C} + 2\frac{\nabla\Psi_D}{\Psi_D}\cdot\frac{\nabla\Psi_C}{\Psi_C}.\end{aligned} \tag{67}$$

From eq. (67) we see that we need the gradient and Laplacian of both Ψ_D and Ψ_C. For Ψ_D the necessary expression are given by eq. (63) and eq. (64). We have that

$$\Psi_C = \prod_{i<j}\exp f(r_{ij}) = \exp\left\{\sum_{i<j}\frac{ar_{ij}}{1+\beta r_{ij}}\right\}, \tag{68}$$

and by differentiating with the chain rule similarly to what we did in eq. (37), we find that the gradient is

$$\frac{\nabla_k\Psi_C}{\Psi_C} = \sum_{j\neq k}\frac{\mathbf{r}_{kj}}{r_{kj}}\frac{\partial f(r_{kj})}{\partial r_{kj}} = \sum_{j\neq k}\frac{\mathbf{r}_{kj}}{r_{kj}}f'(r_{kj}) = \sum_{j\neq k}\frac{\mathbf{r}_{kj}}{r_{kj}}\frac{a}{(1+\beta r_{kj})^2}, \tag{69}$$

where

$$f'(r_{kj}) = \frac{\partial}{\partial r_{kj}}f(r_{kj}). \tag{70}$$

To find the Laplacian we need to calculate

$$\frac{\nabla_k^2\Psi_C}{\Psi_C} = \frac{1}{\Psi_C}\nabla_k\sum_{j\neq k}\frac{\mathbf{r}_{kj}}{r_{kj}}f'(r_{kj})\Psi_C. \tag{71}$$

We follow the calculations from Ref. [3] and Ref. [4]. Using the product rule we get

$$\frac{\nabla_k^2\Psi_C}{\Psi_C} = \frac{1}{\Psi_C}\sum_{j\neq k}\left(\frac{\mathbf{r}_{kj}}{r_{kj}}f'(r_{kj})\nabla_k\Psi_C + \frac{\mathbf{r}_{kj}}{r_{kj}}\Psi_C\nabla_k f'(r_{kj}) + f'(r_{kj})\Psi_C\nabla_k\frac{\mathbf{r}_{kj}}{r_{kj}}\right) \tag{72}$$

$$= \sum_{ij\neq k}\frac{(\mathbf{r}_k-\mathbf{r}_i)(\mathbf{r}_k-\mathbf{r}_j)}{r_{ki}r_{kj}}f'(r_{ki})f'(r_{kj}) + \sum_{j\neq k}\left(f''(r_{kj}) + \frac{d-1}{r_{kj}}f'(r_{kj})\right), \tag{73}$$

where d is the number of dimensions, $d=2$ in our case. We end up with the following expression

$$\frac{\nabla_k^2\Psi_C}{\Psi_C} = \sum_{ij\neq k}\frac{(\mathbf{r}_k-\mathbf{r}_i)(\mathbf{r}_k-\mathbf{r}_j)}{r_{ki}r_{kj}}\frac{a}{(1+\beta r_{ki})^2}\frac{a}{(1+\beta r_{kj})^2} + \sum_{j\neq k}\left(\frac{a}{r_{kj}(1+\beta r_{kj})^2} - \frac{2a\beta}{(1+\beta r_{kj})^3}\right). \tag{74}$$

B Program Structure

The methods were implemented with a C++ program using object-orientation. The program consists of several classes responsible for different parts of the simulations. The classes are:

- **Hamiltonian:** A super-class for different Hamiltonians. The subclasses calculate the local energy for their specific Hamiltonian. Since calculating the kinetic energy with numerical differentiation is done the same for all Hamiltonians, this super-class is responsible for that. The subclasses are:
  - **HarmonicOscillator:** Calculates the local energy in the non-interacting case. [Project1]
  - **HarmonicOscillatorRepulsive:** Calculates the local energy in the interacting case. [Project1]
  - **HarmonicOscillatorElectrons:** Calculates the local energy for quantum dots. [Project2]
- **WaveFunction:** A super-class for different wave functions. The subclasses evaluate their specific wave function and also calculate the gradient, the Laplacian and the derivative w.r.t. the variational parameter(s) α (and β) using the analytical expressions. The subclasses are:
  - **SimpleGaussian:** WaveFunction subclass for the non-interacting case. [Project1]
  - **RepulsiveGaussian:** WaveFunction subclass for the interacting case. [Project1]
  - **TwoElectrons:** WaveFunction subclass for the two-body quantum dot case. [Project2]
  - **ManyElectrons:** WaveFunction subclass for the many-body quantum dot case. This includes all Slater determinant functionality. [Project 2]
- **InitialState:** A super-class for different initial states. The subclasses set up the initial state. The subclasses are:
  - **RandomUniform:** Sets up an initial state with uniformly distributed particle positions.
- **Particle:** Responsible for creating particles and adjusting their positions.
- **System:** Responsible for running the Monte Carlo simulation. It performs the Metropolis and Metropolis-Hastings algorithms.
- **Sampler:** Responsible for sampling interesting quantities and computing averages. It is also responsible for providing the data to the user, both by printing to terminal and saving to file.
- **SteepestDescent:** Responsible for optimizing variational parameters using the Steepest Descent method. Once the optimal parameter has been found, the System class is tasked with running a large Monte Carlo simulation.
- **Random:** Responsible for generating pseudo-random numbers according to different distributions.

There is also a main program which sets the necessary parameters and makes calls to the classes to start the simulation. The code is also fully parallelized with MPI. An advantage to using an implementation like this is that it makes it easy to add functionality, like more wave functions, Hamiltonians and initial states, or alternative methods for optimizing variational parameters (e.g. the Conjugate Gradient method). The data analysis is done in Python, with the programs **blocking.py**, **density.py** and **performance.py**.