

FYS4411  
SPRING 2016

---

# Variational Monte Carlo studies of electronic systems

---

John-Anders Stende

Date: June 13, 2016

## Abstract

The aim of this project is to use the Variational Monte Carlo (VMC) method to evaluate the ground state energy, onebody densities, expectation values of the kinetic and potential energies and single-particle energies of quantum dots with  $N = 2$ ,  $N = 6$ ,  $N = 12$  and  $N = 20$  electrons, i.e. closed-shell systems. A performance analysis of the code is also made.

\*\*\*Main findings\*\*\*

## Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Theory</b>	<b>2</b>
2.1	Physical system . . . . .	2
2.2	Trial wavefunction . . . . .	2
2.3	Benchmarks . . . . .	2
<b>3</b>	<b>Methods</b>	<b>3</b>
3.1	Monte Carlo integration . . . . .	3
3.2	Metropolis algorithm . . . . .	4
3.3	Importance sampling . . . . .	5
3.4	Optimization of trial wave function and its derivatives . . . . .	6
3.4.1	Optimization of Metropolis ratio . . . . .	6
3.4.2	Optimization of inverse Slater matrix . . . . .	7
3.4.3	Splitting the Slater determinant . . . . .	7
3.4.4	Optimization of gradient . . . . .	8
3.5	Steepest descent method . . . . .	8
3.6	Blocking . . . . .	10
3.7	Implementation . . . . .	11
<b>4</b>	<b>Results</b>	<b>12</b>
4.1	Results for two electrons - method 1 . . . . .	12
<b>5</b>	<b>Conclusions</b>	<b>12</b>
<b>6</b>	<b>Appendix</b>	<b>12</b>
6.1	Closed-form expressions . . . . .	12
6.2	Analytic energy for two-body quantum dot . . . . .	14

## 1 Introduction

Quantum dots are nano-scale semiconductor devices that contain strongly confined electrons. They exhibit discrete quantum levels due to their small size, including shell structures and magic numbers for the ground states, as in atoms and nuclei. The electronic properties of these materials can be tuned by applying external fields, and are thus of interest in many research applications such as transistors, solar cells, LEDs etc. Studies of quantum dots containing several electrons require reliable many-body methods that also incorporate uncertainty quantifications.

In this project we compute the ground state energy for  $N = 2$ ,  $N = 6$ ,  $N = 12$  and  $N = 20$  electrons confined in a two-dimensional harmonic oscillator trap with different oscillator frequencies  $\omega$ . These values of  $N$  are magic numbers for the system. For the two-body case we find the energy both with and without the use of Slater determinants for benchmarking purposes. The one-body density is computed for all  $N$ , both with and without correlations in the trial wave function. For  $N = 2$  we also compute the mean distance between the electrons and the expectation values of the potential and kinetic energies. We also perform a timing analysis by comparing a serial and parallel code both with and without vectorization.

\*\*\*structure of report\*\*\*y

## 2 Theory

### 2.1 Physical system

We consider a system of electrons confined in a pure two-dimensional isotropic harmonic oscillator potential, with an idealized total Hamiltonian given by

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

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

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

and the repulsive interaction between two electrons given by

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

with the distance between electrons given by  $r_{ij} = |\mathbf{r}_i - \mathbf{r}_j|$ . We define the modulus of the positions of the electrons (for a given electron  $i$ ) as  $r_i = \sqrt{r_{ix}^2 + r_{iy}^2}$ .

### 2.2 Trial wavefunction

We study a system of electrons, which are fermions that obey the Pauli exclusion principle. This means that we have to approximate the exact wave function for the system with a trial wave function that is antisymmetric. Our antisymmetric ansatz  $\Psi_T$  is a product of a Slater determinant  $\Psi_{SD} = |D|$  and a so-called Jastrow correlation term  $\Psi_J$ ,

$$\psi_T(\mathbf{r}_1, \mathbf{r}_2, \dots, \mathbf{r}_N, \alpha, \beta) = \Psi_{SD} \Psi_J = |D(\mathbf{r}_1, \mathbf{r}_2, \dots, \mathbf{r}_N, \alpha)| \prod_{i<j}^N \exp \left( \frac{ar_{ij}}{(1 + \beta r_{ij})} \right), \quad (4)$$

where  $\alpha$  and  $\beta$  are the variational parameters. The Slater matrix  $D$  is defined as

$$d_{ij} = \phi_j(\mathbf{r}_i) \quad (5)$$

where  $\phi_j(\mathbf{r}_i)$  are single-particle wave functions, i.e. eigenfunctions of the one-body Hamiltonian (2). The rows correspond to the position of a given particle, while the columns stand for the various quantum numbers. The single-particle wave functions in a two-dimensional harmonic oscillator are

$$\phi_{n_x, n_y}(x, y) = A H_{n_x}(\sqrt{\alpha\omega}x) H_{n_y}(\sqrt{\alpha\omega}y) \exp[-\alpha\omega(x^2 + y^2)/2]. \quad (6)$$

The functions  $H_{n_x}(\sqrt{\alpha\omega}x)$  are so-called Hermite polynomials (given in appendix), while  $A$  is a normalization constant. For  $N = 2$ , the trial wave function (4) reduces to

$$\psi_T(\mathbf{r}_1, \mathbf{r}_2, \alpha, \beta) = C \exp(-\alpha\omega(r_1^2 + r_2^2)/2) \exp \left( \frac{ar_{12}}{(1 + \beta r_{12})} \right), \quad (7)$$

For the pure harmonic oscillator (2),  $\Psi_T$  does not have the correlation term:

$$\psi_T(\mathbf{r}_1, \mathbf{r}_2, \alpha, \beta) = C \exp(-\alpha\omega(r_1^2 + r_2^2)/2) \quad (8)$$

### 2.3 Benchmarks

We need to compare our results to exact theoretical closed-form expressions or other research to validate our code.  $N$  fermions in a harmonic oscillator potential without interaction is a well known problem in quantum mechanics. The exact ground state energy for one electron in two dimensions is (in atomic units),

$$\varepsilon_{n_x, n_y} = \omega(n_x + n_y + 1) \quad (9)$$

Each energy state  $(n_x, n_y)$  can be occupied by a maximum of two electrons, one with spin up and one with spin down. This gives the following shell structure (with  $n = n_x + n_y$ ),

- $n = 3$  (3, 0) (2, 1) (1, 2) (0, 3)
- $n = 2$  (2, 0) (1, 1) (0, 2)
- $n = 1$  (1, 0) (0, 1)
- $n = 0$  (0, 0)

We see that for  $N = 2, 6, 12, 20$ , all the states up to shell 0, 1, 2, 3 are filled respectively. These values of  $N$  are thus magic numbers. From (9) we obtain the following energies,

$N$	$E$
2	$2\omega$
6	$10\omega$
12	$28\omega$
20	$60\omega$

Table 1: Exact ground state energies  $E$  in atomic units for  $N$  electrons in a pure two-dimensional harmonic oscillator with oscillator frequency  $\omega$ . The values of  $N$  are magic numbers for the system.

For the full Hamiltonian (1) there are only analytic solutions for two electrons. According to [1], the ground state energy for two electrons with  $\omega = 1$  is  $E = 3 \text{ a.u.}$ . For  $N = 6, 12, 20$  we benchmark our results with [2].

### 3 Methods

We use the *Variational Monte Carlo* (VMC) method in this project to obtain the ground state energy for our fermionic system. VMC applies the *variational principle* from quantum mechanics

$$E_0 \leq \frac{\langle \Psi_T | H | \Psi_T \rangle}{\langle \Psi_T | \Psi_T \rangle} \quad (10)$$

which states that the ground state energy is always less or equal than the expectation value of our Hamiltonian  $H$  for any trial wavefunction  $\Psi_T$ . VMC consists in choosing a trial wavefunction depending on one or more variational parameters, and finding the values of these parameters for which the expectation value of the energy is the lowest possible. The main challenge is to compute the multidimensional integral

$$\frac{\langle \Psi_T | H | \Psi_T \rangle}{\langle \Psi_T | \Psi_T \rangle} = \frac{\int d\mathbf{R} \Psi_T^*(\mathbf{R}, \boldsymbol{\alpha}) H(\mathbf{R}) \Psi_T(\mathbf{R}, \boldsymbol{\alpha})}{\int d\mathbf{R} \Psi_T^*(\mathbf{R}, \boldsymbol{\alpha}) \Psi_T(\mathbf{R}, \boldsymbol{\alpha})} \quad (11)$$

where  $\mathbf{R}$  is the positions of all the particles and  $\boldsymbol{\alpha}$  is the set of variational parameters. Traditional integration methods like Gauss-Legendre methods are too computationally expensive, therefore other methods are needed.

#### 3.1 Monte Carlo integration

Monte Carlo integration employs a non-deterministic approach to evaluate multidimensional integrals like (11), or in general

$$I = \int_{\Omega} f(\mathbf{x}) d\mathbf{x} \quad (12)$$

Instead of using an explicit integration scheme, we sample points

$$\mathbf{x}_1 \dots \mathbf{x}_N \in \Omega \quad (13)$$

according to some rule. The most naive approach is to use  $N$  uniform samples. The integral can then be approximated as the average of the function values at these points

$$I \approx \frac{1}{N} \sum_{i=1}^N f(\mathbf{x}_i) \quad (14)$$

This simple approach is however not very efficient, as it samples an equal amount of points in all regions of  $\Omega$ , including those where  $f$  is zero.

### 3.2 Metropolis algorithm

A more clever approach is to sample points according to the probability distribution (PDF) defined by  $f$ . Such a PDF is in general difficult to obtain, thus we can't sample directly from it. Instead we use the Metropolis algorithm, which is a method to obtain random samples from a PDF for which direct sampling is difficult. These sample values are produced iteratively, with the distribution of the next sample being dependent only on the current sample value, thus making the sequence of samples into a Markov chain. We define  $\mathbf{P}_i^{(n)}$  to be the probability for finding the system in state  $i$  at step  $n$ . The Metropolis algorithm is as follows:

- Sample a possible new state  $j$  with some probability  $T_{i \rightarrow j}$
- Accept the new state with probability  $A_{i \rightarrow j}$  and use it as the next sample, or reject the new state with probability  $1 - A_{i \rightarrow j}$  and use state  $i$  as sample again

The transition probability  $T$  and the acceptance probability  $A$  must fulfill the principle of detailed balance

$$\frac{A_{i \rightarrow j}}{A_{j \rightarrow i}} = \frac{p_i T_{i \rightarrow j}}{p_j T_{j \rightarrow i}} \quad (15)$$

which ensures that  $\mathbf{P}_i^{(n \rightarrow \infty)} \rightarrow p_i$ , i.e. we end up at the correct distribution regardless of what we begin with.

The particles undergo a random walk under the guidance of the Metropolis algorithm. Defining the PDF

$$P(\mathbf{R}, \alpha) = \frac{|\Psi_T(\mathbf{R}, \alpha)|^2}{\int |\Psi_T(\mathbf{R}, \alpha)|^2 d\mathbf{R}} \quad (16)$$

and the local energy,

$$E_L(\mathbf{R}, \alpha) = \frac{1}{\Psi_T(\mathbf{R}, \alpha)} H \Psi_T(\mathbf{R}, \alpha), \quad (17)$$

the integral (11) can be rewritten as

$$\langle E_L \rangle = \int P(\mathbf{R}, \alpha) E_L(\mathbf{R}, \alpha) d\mathbf{R} \quad (18)$$

and we see that our problem amounts to finding the expectation value of the local energy  $E_L$  on the PDF  $P$ . Using Monte Carlo integration, we approximate this integral as

$$\langle E_L \rangle \approx \frac{1}{N} \sum_{i=1}^N P(\mathbf{R}_i, \alpha) E_L(\mathbf{R}_i, \alpha) \quad (19)$$

where  $N$  is the number of Monte Carlo cycles and  $\mathbf{R}_i$  is the position of the particles at step  $i$ . The integral  $\int |\Psi_T(\mathbf{R}, \alpha)|^2 d\mathbf{R}$  is in general very difficult to compute, but the Metropolis algorithm only needs a *ratio* of probabilities to decide if a move is accepted or not. This can be seen if we rewrite (15) as

$$\frac{p_j}{p_i} = \frac{T_{i \rightarrow j} A_{i \rightarrow j}}{T_{j \rightarrow i} A_{j \rightarrow i}} \quad (20)$$

In our case  $p_j = P(\mathbf{R}_j)$  and  $p_i = P(\mathbf{R}_i)$ . The simplest form of the Metropolis algorithm, called brute force Metropolis, is to assume that the transition probability  $T_{i \rightarrow j}$  is symmetric, implying that  $T_{i \rightarrow j} = T_{j \rightarrow i}$ ; the ratio of probabilities (20) thus equals the ratio of acceptance probabilities. This leads to a description of the Metropolis algorithm where we accept or reject a new move by calculating the ratio

$$w = \frac{|\Psi_T(\mathbf{R}_j)|^2}{|\Psi_T(\mathbf{R}_i)|^2} \quad (21)$$

If  $w \geq s$ , where  $s$  is a random number  $s \in [0, 1]$ , the new position is accepted, else we stay at the same place. We now have the full machinery of the Monte Carlo approach to obtain the ground state energy of our bosonic system:

- Fix the number of Monte Carlo steps and choose the initial positions  $\mathbf{R}$  and variational parameters  $\alpha$ . Also set the step size  $\Delta \mathbf{R}$  to be used when moving from  $\mathbf{R}_i$  to  $\mathbf{R}_j$ .
- Initialize the local energy
- Choose a random particle
- Calculate a trial position  $\mathbf{R}_j = \mathbf{R}_i + r \Delta \mathbf{R}$  where  $r$  is a random variable  $r \in [0, 1]$

- Use the Metropolis algorithm to accept or reject this move by calculating the ratio (21). If  $w \geq s$ , where  $s$  is a random number  $s \in [0, 1]$ , the new position is accepted, else we stay at the same place.
- If the step is accepted, set  $\mathbf{R} = \mathbf{R}_j$  for the chosen particle
- Sample the local energy

When the Monte Carlo sampling is finished, we calculate the mean local energy, which is our approximation of the ground state energy of the system. The Metropolis algorithm is implemented as follows:

Listing 1: Brute Force Metropolis algorithm

```
int particle = Random::nextInt(m_numberOfParticles); // choose random particle
int dimension = Random::nextInt(m_numberOfDimensions); // choose random dimension
double change = (Random::nextDouble()*2-1)*m_stepLength; // propose change

// get old wavefunction
double waveFuncOld = m_waveFunction->evaluate(m_particles);

// adjust position
m_particles[particle]->adjustPosition(change, dimension);

// get new wavefunction
double waveFuncNew = m_waveFunction->evaluate(m_particles);

// accept/reject new position using Metropolis algorithm
double ratio = pow(waveFuncNew, 2) / pow(waveFuncOld, 2);

if (ratio >= Random::nextDouble()) {
    //cout << m_particles[particle]->getPosition()[0] << endl;
    return true;
}
else {
    // correct position change
    m_particles[particle]->adjustPosition(-change, dimension);
    return false;
}
```

### 3.3 Importance sampling

A more efficient way to do Monte Carlo sampling is to replace the brute force Metropolis algorithm with a walk in coordinate space biased by the trial wavefunction. This approach is based on the Fokker-Planck equation and the Langevin equation for generating a trajectory in coordinate space.

The Langevin equation is a stochastic differential equation

$$\frac{\partial x(t)}{\partial t} = DF(x(t)) + \eta \quad (22)$$

where  $D$  is the diffusion constant and  $\eta$  a random variable. The new positions  $y$  in coordinate space are the solutions of (22) using Euler's method:

$$y = x + DF(x)\Delta t + \xi\sqrt{\Delta t} \quad (23)$$

where  $\xi$  is a gaussian random variable and  $\Delta t$  is a chosen time step.  $D$  is equal to  $1/2$  which comes from the factor  $1/2$  in the kinetic energy operator.  $\Delta t$  is to be viewed as a parameter which yields stable values of the ground state energy for values  $\Delta t \in [0.001, 0.01]$ . (23) is similar to the brute force Metropolis equation for updating positions except for the term containing  $F(x)$ . This is the function that pushes the particles towards regions of configuration space where the wavefunction is large, in contrast to the brute force method where all regions are equally probable. In three dimension  $F(x)$  is called the *drift vector*  $\mathbf{F}(\mathbf{x})$ . The drift vector can be found from the Fokker-Planck equation

$$\frac{\partial P}{\partial t} = \sum_i D \frac{\partial}{\partial \mathbf{x}_i} (\mathbf{x}_i - \mathbf{F}_i) P(\mathbf{x}, t) \quad (24)$$

The convergence to a stationary probability density can be obtained by setting the left hand side to zero. The resulting equation is only satisfied if all terms of the sum are equal to zero,

$$\frac{\partial^2 P}{\partial \mathbf{x}_i^2} = P \frac{\partial}{\partial \mathbf{x}_i} \mathbf{F}_i + \mathbf{F}_i \frac{\partial}{\partial \mathbf{x}_i} P \quad (25)$$

The drift vector should have the form  $\mathbf{F} = g(\mathbf{x}) \frac{\partial P}{\partial \mathbf{x}}$ . Inserting this in (25) yields

$$\mathbf{F} = 2 \frac{1}{\Psi_T} \nabla \Psi_T \quad (26)$$

which is known as the *quantum force*.

The Monte Carlo method with the Metropolis algorithm can be seen as isotropic diffusion process by a time-dependent probability density, with or without a drift, corresponding to brute force and importance sampling respectively. The Fokker-Planck equation (24) describes such a diffusion process, our new transition probability is thus the solution to this equation, given by the Green's function

$$G(y, x, \Delta t) = \frac{1}{(4\pi D \Delta t)^{3N/2}} \exp(-(y - x - D \Delta t F(x))^2 / 4D \Delta t) \quad (27)$$

which in turn means that our brute force Metropolis accept/reject ratio (21) is replaced by the so-called Metropolis-Hastings article

$$q(y, x) = \frac{G(x, y, \Delta t) |\Psi_T(y)|^2}{G(y, x, \Delta t) |\Psi_T(x)|^2} \quad (28)$$

The Metropolis Hastings algorithm is the same as the brute force method, now with (21) replaced by (28) and trial positions calculated according to (23).

### 3.4 Optimization of trial wave function and its derivatives

The trial wave function (4) plays a central role in our VMC simulation. It is needed in the Metropolis algorithm and in the evaluation of the quantum force (26). Moreover, all observables like the local energy (17) is computed w.r.t. it. The most time-consuming part of the evaluation of the wave function is the computation of the Slater determinant. Computing a determinant of an  $N \times N$  matrix by standard Gaussian elimination is of the order of  $\mathcal{O}(N^3)$  calculations. As there are  $N \cdot d$  independent coordinates we need to evaluate  $Nd$  Slater determinants for the gradient (quantum force and kinetic energy) and  $Nd$  for the Laplacian (kinetic energy). Therefore, it is imperative to find alternative ways of computing the quantities related to the trial wave function to improve performance.

#### 3.4.1 Optimization of Metropolis ratio

It turns out that the solution is an algorithm that requires to keep track of the *inverse* of the Slater matrix (5). The inverse of  $D$  can be expressed in terms of its cofactors  $C_{ij}$  and its determinant  $|D|$ ,

$$d_{ij}^{-1} = \frac{C_{ji}}{|D|} \quad (29)$$

where  $C_{ji}$  is the transposed cofactor matrix. The Slater part  $R_{SD}$  of the ratio (21) can thus be written,

$$R_{SD} = \frac{|D(\mathbf{R}^{\text{new}})|}{|D(\mathbf{R}^{\text{old}})|} = \frac{\sum_{j=1}^N d_{ij}(\mathbf{R}^{\text{new}}) C_{ij}(\mathbf{R}^{\text{new}})}{\sum_{j=1}^N d_{ij}(\mathbf{R}^{\text{old}}) C_{ij}(\mathbf{R}^{\text{old}})} \quad (30)$$

When moving *one* particle for each Monte Carlo cycle,  $\mathbf{R}^{\text{new}}$  differs from  $\mathbf{R}^{\text{old}}$  by the position of only one, say the  $i$ -th particle. This means that only the  $i$ -th row of  $D(\mathbf{R}^{\text{new}})$  and  $D(\mathbf{R}^{\text{old}})$  will be different. Taking into account that the  $i$ -th row of a cofactor matrix  $C$  is independent of the entries of the  $i$ -th row of its corresponding matrix  $D$ , we have that

$$C_{ij}(\mathbf{R}^{\text{new}}) = C_{ij}(\mathbf{R}^{\text{old}}) \quad j \in \{1, \dots, N\} \quad (31)$$

and

$$R_{SD} = \frac{\sum_{j=1}^N d_{ij}(\mathbf{R}^{\text{new}}) d_{ji}^{-1}(\mathbf{R}^{\text{old}})}{\sum_{j=1}^N d_{ij}(\mathbf{R}^{\text{old}}) d_{ji}^{-1}(\mathbf{R}^{\text{old}})} \quad (32)$$

By definition, the denominator of this expression is unity, thus we obtain for the ratio,

$$R_{SD} = \sum_{j=1}^N d_{ij}(\mathbf{R}^{\text{new}}) d_{ji}^{-1}(\mathbf{R}^{\text{old}}) = \sum_{j=1}^N \phi_j(\mathbf{R}_i^{\text{new}}) d_{ji}^{-1}(\mathbf{R}^{\text{old}}) \quad (33)$$

where the last equality follows from the definition of the Slater matrix (5). This operation is simply a dot product of a vector of single-particle wave functions evaluated at the new position with the  $i$ -th column of the inverse matrix  $D^{-1}$  evaluated at the original position, and has a time scaling of  $\mathcal{O}(N)$ .

### 3.4.2 Optimization of inverse Slater matrix

The operation (33) demands that we maintain the inverse matrix  $D^{-1}$  for each MC cycle. Getting the inverse of an  $N \times N$ -matrix på Gaussian elimination has a complexicty of order  $\mathcal{O}(N^3)$  operations, which we cannot afford. An alternative way of updating the inverse of a matrix when only a row/column is changed was suggested by Sherman and Morris (REFERENCE?). This algorithm has a time scaling of  $\mathcal{O}(N^2)$  and is as follows:

- Update all but the  $i$ -th column of  $D^{-1}$ . For each column  $j \neq i$ , calculate the quantity

$$S_j = \sum_{l=1}^N d_{il}(\mathbf{R}^{\text{new}}) d_{lj}^{-1}(\mathbf{R}^{\text{old}})$$

- The new elements of the  $j$ -th column of  $D^{-1}$  is then given by:

$$d_{kj}^{-1}(\mathbf{R}^{\text{new}}) = d_{kj}^{-1}(\mathbf{R}^{\text{old}}) - \frac{S_j}{R_{SD}} d_{ki}^{-1}(\mathbf{R}^{\text{old}}) \quad k = \{1, \dots, N\}, \quad j \neq i$$

- Finally the  $i$ -th column of  $D^{-1}$  is updated simply as follows:

$$d_{ki}^{-1}(\mathbf{R}^{\text{new}}) = \frac{1}{R_{SD}} d_{ki}^{-1}(\mathbf{R}^{\text{old}}) \quad k = \{1, \dots, N\}$$

### 3.4.3 Splitting the Slater determinant

It can be shown, see for example Moskowitz and Kalos [3] that we can approximate the Slater determinant  $|D|$  as a product of two smaller ones, where each can be identified with spin-up  $\uparrow$  and spin-down  $\downarrow$  respectively,

$$|D| = |D|_{\uparrow} \cdot |D|_{\downarrow} \quad (34)$$

To illustrate, we write out the Slater determinant (5) for  $N = 4$ :

$$|D| = \frac{1}{\sqrt{4!}} \begin{vmatrix} \phi_{00\uparrow}(\mathbf{r}_1) & \phi_{00\downarrow}(\mathbf{r}_1) & \phi_{10\uparrow}(\mathbf{r}_1) & \phi_{10\downarrow}(\mathbf{r}_1) \\ \phi_{00\uparrow}(\mathbf{r}_2) & \phi_{00\downarrow}(\mathbf{r}_2) & \phi_{10\uparrow}(\mathbf{r}_2) & \phi_{10\downarrow}(\mathbf{r}_2) \\ \phi_{00\uparrow}(\mathbf{r}_3) & \phi_{00\downarrow}(\mathbf{r}_3) & \phi_{10\uparrow}(\mathbf{r}_3) & \phi_{10\downarrow}(\mathbf{r}_3) \\ \phi_{00\uparrow}(\mathbf{r}_4) & \phi_{00\downarrow}(\mathbf{r}_4) & \phi_{10\uparrow}(\mathbf{r}_4) & \phi_{10\downarrow}(\mathbf{r}_4) \end{vmatrix} \quad (35)$$

The Slater determinant as written is zero since the spatial wave functions for the spin-up and spin-down states are equal. However, we can now factorize this determinant in the following way,

$$|D| = \frac{1}{\sqrt{2}} \begin{vmatrix} \phi_{00\uparrow}(\mathbf{r}_1) & \phi_{10\uparrow}(\mathbf{r}_1) \\ \phi_{00\uparrow}(\mathbf{r}_2) & \phi_{10\uparrow}(\mathbf{r}_2) \end{vmatrix} \cdot \frac{1}{\sqrt{2}} \begin{vmatrix} \phi_{00\downarrow}(\mathbf{r}_3) & \phi_{10\downarrow}(\mathbf{r}_3) \\ \phi_{00\downarrow}(\mathbf{r}_4) & \phi_{10\downarrow}(\mathbf{r}_4) \end{vmatrix} \quad (36)$$

This ansatz is not antisymmetric under exchange of two electrons with opposite spins, but it can be shown that it gives the same expectation value for the energy as the full Slater determinant. The above is correct only for spin-independent Hamiltonians. Our trial wave function can now be written as

$$\Psi_T = |D|_{\uparrow} |D|_{\downarrow} \Psi_C(??) \quad (37)$$

The factorization above makes it possible to perform the calculation of the Slater ratio (33) and the updating of the inverse Slater matrix separately for  $|D|_{\uparrow}$  and  $|D|_{\downarrow}$ :

$$\frac{|D|^{\text{new}}}{|D|^{\text{old}}} = \frac{|D|_{\uparrow}^{\text{new}}}{|D|_{\uparrow}^{\text{old}}} \cdot \frac{|D|_{\downarrow}^{\text{new}}}{|D|_{\downarrow}^{\text{old}}} \quad (38)$$



We see from (36) that one of the two determinants is unaffected when we move only one particle at a time, it will therefore cancel from the ratio. This means that we only need to update one determinant of size  $N/2$  when we compute  $R_{SD}$  and update  $|D|^{-1}$ . The efficiency enhancements due to optimization are summed up in Table 2.

Operation	No optimization	With optimization
Evaluation of $R_{SD}$	$\mathcal{O}(N)$	$\mathcal{O}(N/2)$
Updating inverse	$\mathcal{O}(N^2)$	$\mathcal{O}(N^2/4)$
Transition of one particle	$\mathcal{O}(N) + \mathcal{O}(N^2)$	$\mathcal{O}(N/2) + \mathcal{O}(N^2/4)$

Table 2: Comparison of the computational cost involved in the computation of the Slater determinant with and without optimization

#### 3.4.4 Optimization of gradient

We need the ratio  $\frac{\nabla \Psi_T}{\Psi_T}$  to compute the quantum force (26) used in importance sampling. This ratio can be written as

$$\frac{\nabla \Psi_T}{\Psi_T} = \frac{\nabla(\Psi_{SD}\Psi_C)}{\Psi_{SD}\Psi_C} = \frac{\Psi_C \nabla \Psi_{SD} + \Psi_{SD} \nabla \Psi_C}{\Psi_{SD}\Psi_C} = \frac{\nabla \Psi_D}{\Psi_D} + \frac{\nabla \Psi_C}{\Psi_C} \quad (39)$$

Inserting  $\Psi_{SD} = |D|_{\uparrow}|D|_{\downarrow}$  yields

$$\frac{\nabla \Psi_T}{\Psi_T} = \frac{\nabla |D|_{\uparrow}}{|D|_{\uparrow}} + \frac{\nabla |D|_{\downarrow}}{|D|_{\downarrow}} + \frac{\nabla \Psi_C}{\Psi_C} \quad (40)$$

The process to obtain the gradient-determinant ratio needed in this expression is analogous to the one used in deriving (33). The result is

$$\frac{\nabla_i |D|}{|D|} = \sum_{j=1}^N \nabla_i d_{ij}(\mathbf{R}^{\text{old}}) d_{ji}^{-1}(\mathbf{R}^{\text{old}}) = \sum_{j=1}^N \nabla_i \phi_j(\mathbf{R}_i^{\text{old}}) d_{ji}^{-1}(\mathbf{R}^{\text{old}}) \quad (41)$$

This expression is used to calculate the quantum force before the ratio (21) is computed. After the position of one particle is altered, the expression is modified to

$$\frac{\nabla_i |D|}{|D|} = \frac{1}{R_{SD}} \sum_{j=1}^N \nabla_i \phi_j(\mathbf{R}_i^{\text{new}}) d_{ji}^{-1}(\mathbf{R}^{\text{old}}) \quad (42)$$

Note that the old inverse Slater matrix is used in both cases because this is not updated until after a step has been accepted. According to (??), we only need to compute the gradient w.r.t. either  $|D|_{\uparrow}$  or  $|D|_{\downarrow}$ . We use the particle index to decide which matrix we need to update: If  $i < N/2$  we only calculate the spin-up gradient:

$$\frac{\nabla_i |D|_{\uparrow}}{|D|_{\uparrow}} = \sum_{j=0}^{N/2-1} \nabla_i \phi_j(\mathbf{R}_i^{\text{old}}) d_{ji}^{-1}(\mathbf{R}^{\text{old}}) \quad (43)$$

or else,  $i \geq N/2$ , we update the spin-down gradient:

$$\frac{\nabla_i |D|_{\downarrow}}{|D|_{\downarrow}} = \sum_{j=N/2}^N \nabla_i \phi_j(\mathbf{R}_i^{\text{old}}) d_{ji}^{-1}(\mathbf{R}^{\text{old}}) \quad (44)$$

### 3.5 Steepest descent method

We turn now to the problem of finding the variational parameters that minimizes the expectation value of the local energy  $\langle E_L(\mathbf{R}, \boldsymbol{\alpha}) \rangle$ . This project considers a trial wavefunction with two variational parameters  $\alpha$  and  $\beta$ . There are many optimization algorithms to choose from, we have chosen the Steepest descent method due to its simplicity. This method finds a local minimum of a function by taking steps proportional to the negative gradient of the function at a given point, i.e. where the function has the steepest descent. The algorithm is as follows:

- Choose an initial set of parameters  $\boldsymbol{\alpha}_0$  and step length  $\gamma_0$ .
- For  $i \geq 0$ : Compute  $\boldsymbol{\alpha}_{i+1} = \boldsymbol{\alpha}_i - \gamma_i \nabla_{\boldsymbol{\alpha}_i} \langle E_L(\mathbf{R}, \boldsymbol{\alpha}_i) \rangle$
- Continue until a maximum number of steps are performed or  $|\nabla_{\boldsymbol{\alpha}} \langle E_L(\mathbf{R}, \boldsymbol{\alpha}) \rangle|$  is less than some tolerance

We should get  $|\nabla_{\alpha_i}\langle E_L(\mathbf{R}, \alpha_i)\rangle|^2 \geq |\nabla_{\alpha_{i+1}}\langle E_L(\mathbf{R}, \alpha_{i+1})\rangle|^2 \geq \dots$ . If this is not the case, we reject the new step and decrease the step size to obtain a more accurate value.  $\langle E_L(\mathbf{R}, \alpha)\rangle$  is as we have seen a multidimensional integral (18), and the gradient w.r.t.  $\alpha$  is not easily computed. Let us define

$$\bar{E}_\alpha = \frac{\partial \langle E_L(\alpha) \rangle}{\partial \alpha} \quad (45)$$

and

$$\bar{E}_\beta = \frac{\partial \langle E_L(\alpha) \rangle}{\partial \beta} \quad (46)$$

Using the chain rule and the hermicity of the Hamiltonian, it can be shown that

$$\bar{E}_\alpha = 2 \left( \left\langle \frac{\partial \ln \Psi_T}{\partial \alpha} E_L \right\rangle - \left\langle \frac{\partial \ln \Psi_T}{\partial \alpha} \right\rangle \langle E_L \rangle \right) \quad (47)$$

and

$$\bar{E}_\beta = 2 \left( \left\langle \frac{\partial \ln \Psi_T}{\partial \beta} E_L \right\rangle - \left\langle \frac{\partial \ln \Psi_T}{\partial \beta} \right\rangle \langle E_L \rangle \right) \quad (48)$$

thus we need the expectation values of

$$\frac{\partial \ln \Psi_T}{\partial \alpha} E_L \quad \frac{\partial \ln \Psi_T}{\partial \alpha} \quad (49)$$

and

$$\frac{\partial \ln \Psi_T}{\partial \beta} E_L \quad \frac{\partial \ln \Psi_T}{\partial \beta} \quad (50)$$

For our trial wave function (4), we have

$$\ln \Psi_T = \quad (51)$$

The complete VMC method then amounts to the following:

- Make initial guess  $\alpha_0$
- Run  $10^4$ - $10^5$  Metropolis steps, sample (??) and (??)
- Compute (47)
- Calculate new  $\alpha$  using the Steepest descent method

The above steps are repeated until the above stopping condition is fulfilled, before a new round of Metropolis steps are run, this time with many cycles ( $10^6$ - $10^8$ ). We then obtain our approximation for the ground state energy of the system.

The Steepest descent method is implemented as follows:

Listing 2: The Steepest Descent method

```
void SteepestDescent::optimize(double initialAlpha) {

    int maxNumberOfSteps = 30;
    double tolerance = 1e-6;
    double alpha = initialAlpha;
    double oldEnergy = 1e10;
    for (int i=0; i < maxNumberOfSteps; i++) {

        if (i > 0) {
            oldEnergy = m_system->getSampler()->getEnergy();
        }

        // make initial state
        m_system->getInitialState()->setupInitialState();

        // set value of alpha
        m_system->getWaveFunction()->setAlpha(alpha);

        // run metropolis steps
        m_system->runMetropolisSteps((int) 1e5, false, false, false);
```

```

double newEnergy = m_system->getSampler()->getEnergy ();

// compute derivative of exp. value of local energy w.r.t. alpha
double localEnergyDerivative = 2 *
    ( m_system->getSampler()->getWaveFunctionEnergy () -
      m_system->getSampler()->getWaveFunctionDerivative () *
      newEnergy );

if (newEnergy > oldEnergy) {
    m_stepLengthOptimize /= 2.0;
    cout << "New step length: " << m_stepLengthOptimize << endl;
}
else {
    // compute new alpha
    alpha -= m_stepLengthOptimize*localEnergyDerivative;
}

cout << "newAlpha = " << alpha << endl;

if ( localEnergyDerivative < tolerance ) {
    break;
}

}
cout << "Optimal alpha = " << alpha << endl;

// run many Metropolis steps with the optimal alpha

// make initial state
m_system->getInitialState()->setupInitialState ();

// set value of alpha
m_system->getWaveFunction()->setAlpha(alpha);

// run metropolis steps
m_system->runMetropolisSteps((int) 1e6, false, false, false);
}

```

### 3.6 Blocking

Monte Carlo simulations can be treated as computer experiments. The results can be analyzed with the same statistical tools as we would use analyzing experimental data. We are looking for expectation values of these data, and how accurate they are. A stochastic process like a Monte Carlo experiment produces sequentially a chain of values

$$\{x_1, x_2 \dots x_k \dots x_N\} \quad (52)$$

called a sample. Each value  $x_k$  is called a measurement. The sample variance

$$\text{var}(x) = \frac{1}{N} \sum_{k=1}^n (x_k - \bar{x}_N) \quad (53)$$

where  $\bar{x}_N$  is the sample mean, is a measure of the statistical error of a *uncorrelated* sample. However, a Monte Carlo simulation with interacting particles produces a correlated sample, thus we need another measure of the sample error. It can be shown that an estimate of the error  $\text{err}_X$  of a correlated sample is

$$\text{err}_X = \frac{1}{N} \text{cov}(x) \quad (54)$$

where  $\text{cov}(x)$  is the sample covariance

$$\text{cov}(x) \equiv \frac{1}{N} \sum_{kl} (x_k - \bar{x}_N)(x_l - \bar{x}_N) \quad (55)$$

which is a measure of the sequential correlation between succeeding measurements of a sample. (Note that (53) and (55) are experimental values for the sample, not the *true* properties of the stochastic variables, which we need an infinite number of measurements to calculate). With the help of the *autocorrelation function* from statistical theory we can rewrite this error as

$$\text{err}_X = \frac{\tau}{N} \text{var}(x) \quad (56)$$

where  $\tau$  is the *autocorrelation time* which accounts for the correlation between measurements. In the presence of correlation the effective number of measurements becomes

$$n_{\text{eff}} = \frac{N}{\tau} \quad (57)$$

Neglecting  $\tau$  thus gives an error estimate that is less than the true sample error. The autocorrelation time is however expensive to compute. We can avoid the computation of this quantity by using the technique of blocking. The idea behind this method is to split the sample into blocks, find the mean of each block and then calculate the total mean and variance of all the block means. This is done for increasing block sizes  $n_b$  until the measurements of two sequential blocks are uncorrelated, enabling us to extract the value of  $\tau = n_b \Delta t$ . The true sample error,

$$\sigma = \left( \frac{1 + 2\tau/\Delta t}{N} (\langle E_L^2 \rangle - \langle E_L \rangle^2) \right)^{1/2} \quad (58)$$

can then be calculated.

The blocking algorithm is as follows:

- Do a Monte Carlo simulation, store the local energy for each step to file
- Read the file into an array
- Loop over increasing block sizes:
  - For each block size  $n_b$ , loop over array in steps of  $n_b$  taking the mean of elements  $[in_b, (i+1)n_b], \dots$
  - Calculate total mean and variance of all block means and store
- Plot total variance for all block sizes.
- Extract  $\tau$  and compute (58)

### 3.7 Implementation

We have made an object-oriented code in C++. We give here an overview of the class structure and what the different classes do,

- *Main program*: Sets all the parameters needed to a simulation.
- *System*: Runs the Monte Carlo cycles with/without importance sampling
- *Sampler*: Samples quantities we want to measure for each cycle and computes expectation values
- *Particle*: Sets and adjusts particle positions
- *SteepestDescent*: Runs the steepest descent method
- *InitialState*: Super-class for setting up different initial states
  - *RandomUniform*: Assigns initial positions according to a uniform distribution
- *WaveFunction*: Super-class for different wave functions. Sets the variational parameters. All subclasses must implement functions to evaluate the analytical expressions for  $\Psi_T$ ,  $\nabla \Psi_T$ ,  $\nabla^2 \Psi_T$  and  $d\Psi_T/d\alpha$ .
  - *SimpleGaussian*: Implements the above quantities for  $\Psi_T$  used in system 1
  - *InteractingGaussian*: Implements the above quantities for  $\Psi_T$  used in system 2
- *Hamiltonian*: Super-class for different Hamiltonians. Calculates  $E_L$  by computing potential and kinetic energy, either numerically or analytically for the latter. The analytical Laplacian is obtained from *WaveFunction*.
  - *HarmonicOscillator*: Hamiltonian for system 1
  - *HarmonicOscillatorInteracting*: Hamiltonian for system 2

In addition to these we use class *Random* to generate pseudo-random numbers.

All the information are stored in System in the form of class objects of the other classes, which in turn receives the System object so that they can access this information via setters and getters in the System class. This way of communicating between classes limits the user's capability to alter vital functionality and also makes the program more user-friendly. Object-oriented code is also easy to expand on. We don't need to add new functionality to e.g. implement a new wave function; only the specifics of this new wave function needs to be implemented.

## 4 Results

We calculate the ground-state energies for the magic numbers  $N = 2$ ,  $N = 6$ ,  $N = 12$  and  $N = 20$  and for oscillator frequencies  $\omega = 0.01$ ,  $\omega = 0.05$ ,  $\omega = 0.1$ ,  $\omega = 0.5$  and  $\omega = 1.0$ . The one-body density is computed for all  $N$  and  $\omega$ , both with and without the Jastrow factor. For  $N = 2$  we calculate the ground state energy using two different wave functions:

1. The trial wave function is (??) for which we have an analytic expression (98) for the Laplacian  $\nabla\Psi_T$  (method 1)
2. The trial wave function is (4), i.e. the full Slater determinant (method 2)

These two approaches should yield the same energies and is therefore a good test to see if the Slater determinants are implemented correctly. They are in the following called method 1 and method 2 respectively. We also calculate the mean distance between the two electrons.

We utilize the blocking technique for error analysis of the energy calculations. A performance analysis is made by comparing the time usage of a serial vs. a parallel version of the code, both with and without vectorization.

### 4.1 Results for two electrons - method 1

As a first validation of our code, we calculate the ground state energy of the two-electron system in a pure harmonic oscillator (??) and benchmark the results with Table (1).

## 5 Conclusions

For system 1, the exact ground state energies were reproduced with zero error when using the closed-form expression for the Laplacian of the wave function. Computing this quantity numerically resulted in a small variance caused by the error in the differentiation algorithm. Implementing closed-form expressions saved significant CPU time.

The error in the energies for system 2 were larger, but quite close to those in [1]. This tells us that the optimal  $\alpha$  found by the steepest descent method was a good approximation. Doing a proper error analysis by applying the blocking method was important to estimate the true sample errors, which was almost one order of magnitude larger than the initial suggestions.

The one-body density visualizations showed that system 2 were more dense than system 1. The different traps we used decided how the distribution of particles looked like: The spherical trap resulted in a spherical distribution centered around zero, while the elliptical trap stretched out the distribution in the z-direction.

The framework we have developed can also be used to model fermionic systems if we include the proper symmetry requirements. Further optimization of the code can be done, especially by applying parallelization.

## 6 Appendix

### 6.1 Closed-form expressions

The quantity we are aiming to compute is the expectation value of the so-called local energy

$$E_L(\mathbf{R}) = \frac{1}{\Psi_T(\mathbf{R})} H \Psi_T(\mathbf{R}), \quad (59)$$

We can find closed-form expressions for the local energy with our specific Hamiltonian  $H$  and trial wavefunction  $\Psi_T$ . Computing the local energy involves a second derivative of  $\Psi_T$ , which can be expensive to compute numerically. Analytical expressions are therefore useful, as they can speed up the computations.

First, we find the local energy with only the (spherical) harmonic oscillator potential, that is we set  $a = 0$  and  $\beta = 1$ . During these calculations we will use natural units, thus  $\hbar = m = 1$ . For one particle in one dimension we have

$$\Psi_T(x) = e^{-\alpha x^2} \quad (60)$$

and

$$H = -\frac{1}{2} \frac{\partial^2}{\partial x^2} + \frac{1}{2} \omega^2 x^2 \quad (61)$$

The second derivate of the trial wave function is

$$\frac{\partial^2 \Psi_T}{\partial x^2} = 2\alpha e^{-\alpha x^2} (2\alpha x^2 - 1) \quad (62)$$

so that

$$E_L = \frac{1}{\Psi_T} H \Psi_T = \alpha(1 - 2\alpha x^2) + \frac{1}{2} \omega^2 x^2 \quad (63)$$

In three dimensions the double derivative is replaced by the Laplacian when computing the kinetic energy

$$\frac{1}{\Psi_T} \nabla^2 \Psi_T = 2\alpha(2\alpha x^2 - 1) + 2\alpha(2\alpha y^2 - 1) + 2\alpha(2\alpha z^2 - 1) \quad (64)$$

$$= 2\alpha(2\alpha r^2 - 3) \quad (65)$$

thus the local energy is

$$E_L = \frac{1}{\Psi_T} \left( -\frac{1}{2} \nabla^2 \Psi_T + V_{ext} \right) = \alpha(3 - 2\alpha r^2) + \frac{1}{2} \omega^2 r^2 \quad (66)$$

We now turn our attention to  $N$  particles, with the following wavefunction and Hamiltonian

$$\Psi_T(\mathbf{R}) = \prod_i e^{-\alpha r_i^2} \quad (67)$$

$$H = \sum_i^N \left( -\frac{1}{2} \nabla_i^2 + \frac{1}{2} \omega^2 r_i^2 \right) \quad (68)$$

The first term of the k-th Laplacian of this wavefunction is

$$\nabla_k^2 \prod_i e^{-\alpha r_i^2} = 2\alpha(2\alpha x_k^2 - 1) \prod_i e^{-\alpha r_i^2} \quad (69)$$

and when we divide with  $\Psi_T$  to obtain the the local energy we end up with

$$E_L = \sum_i^N \left( \alpha(3 - 2\alpha r_i^2) + \frac{1}{2} \omega^2 r_i^2 \right) \quad (70)$$

For one dimension the expression is

$$E_L = \sum_i^N \left( \alpha(1 - 2\alpha x_i^2) + \frac{1}{2} \omega^2 x_i^2 \right) \quad (71)$$

It is also useful to compute the analytical expression for the drift force  $F$  to be used in importance sampling

$$F = \frac{2\nabla \Psi_T}{\Psi_T}. \quad (72)$$

The gradient of  $\Psi_T$  is

$$\nabla \Psi_T = (-2\alpha x, -2\alpha y, -2\alpha z) e^{-\alpha r^2} \quad (73)$$

$$= -2\alpha e^{-\alpha r^2} \mathbf{r} \quad (74)$$

where  $\mathbf{r} = (x, y, z)$ . Dividing by the wavefunction and multiplying with 2

$$F = -4\alpha \mathbf{r} \quad (75)$$

## 6.2 Analytic energy for two-body quantum dot

$$\Psi_T(\mathbf{r}_1, \mathbf{r}_2) = \exp(-\alpha\omega(r_1^2 + r_2^2)/2) \exp\left(\frac{ar_{12}}{1 + \beta r_{12}}\right) \quad (76)$$

$$= K_1 K_2 \quad (77)$$

Laplacian of this wave function:

$$\nabla^2 \Psi_T(\mathbf{r}_1, \mathbf{r}_2) = \frac{\partial \Psi_T}{\partial x_1^2} + \frac{\partial \Psi_T}{\partial y_1^2} + \frac{\partial \Psi_T}{\partial x_2^2} + \frac{\partial \Psi_T}{\partial y_2^2} \quad (78)$$

Using the following

$$\frac{\partial r_1}{\partial x_1} = x_1/r_1 \quad (79)$$

and

$$\frac{\partial r_{12}}{\partial x_1} = (x_1 - x_2)/r_1 \quad (80)$$

Gradient of first term:

$$\frac{\partial K_1}{\partial x_1} = -\alpha\omega x_1 K_1 \quad (81)$$

so that

$$\nabla K_1 = -\alpha\omega K_1(x_1, y_1, x_2, y_2) \quad (82)$$

Gradient of second term:

$$\frac{\partial K_2}{\partial x_1} = K_2 \frac{a(x_1 - x_2)}{r_{12}(1 + \beta r_{12})^2} \quad (83)$$

so that

$$\nabla K_2 = K_2 \frac{a}{r_{12}(1 + \beta r_{12})^2} (x_1 - x_2, y_1 - y_2, x_2 - x_1, y_2 - y_1) \quad (84)$$

Laplacian of first term:

$$\frac{\partial^2 K_1}{\partial x_1^2} = K_1(\alpha^2 \omega^2 x_1^2 - \alpha\omega) \quad (85)$$

so that

$$\nabla^2 K_1 = K_1(\alpha^2 \omega^2 (r_1^2 + r_2^2) - 4\alpha\omega) \quad (86)$$

Laplacian of second term:

$$\frac{\partial^2 K_2}{\partial x_1^2} = K_2 \left[ \frac{a^2(x_1 - x_2)^2}{r_{12}^2(1 + \beta r_{12})^4} + \frac{ar_{12}(1 + \beta r_{12})^2}{r_{12}^2(1 + \beta r_{12})^4} \right] \quad (87)$$

$$- \frac{a(x_1 - x_2)[(x_1 - x_2)/r_{12}(1 + \beta r_{12})^2 + 2r_{12}(1 + \beta r_{12})\beta(x_1 - x_2)/r_{12}]}{r_{12}^2(1 + \beta r_{12})^4} \quad (88)$$

$$= K_2 \left[ \frac{a^2(x_1 - x_2)^2}{r_{12}^2(1 + \beta r_{12})^4} + \frac{a}{r_{12}(1 + \beta r_{12})^2} \right] \quad (89)$$

$$- \frac{a(x_1 - x_2)^2}{r_{12}^3(1 + \beta r_{12})^2} - \frac{2a\beta(x_1 - x_2)^2}{r_{12}^2(1 + \beta r_{12})^3} \quad (90)$$

so that

$$\nabla^2 K_2 = K_2 \left[ \frac{2a^2}{(1 + \beta r_{12})^4} + \frac{4a}{r_{12}(1 + \beta r_{12})^2} - \frac{2a}{r_{12}(1 + \beta r_{12})^2} - \frac{2a\beta}{(1 + \beta r_{12})^3} \right] \quad (91)$$

$$= K_2 \frac{2a}{(1 + \beta r_{12})^2} \left[ \frac{a}{(1 + \beta r_{12})^2} + \frac{1}{r_{12}} - \frac{2\beta}{1 + \beta r_{12}} \right] \quad (92)$$

Have that

$$\nabla^2 \Psi_T = \nabla^2 K_1 K_2 + 2\nabla K_1 \nabla K_2 + K_1 \nabla^2 K_2 \quad (93)$$

and

$$\nabla K_1 \nabla K_2 = -K_1 K_2 \frac{a\alpha\omega}{r_{12}(1+\beta r_{12})^2} \left[ x_1(x_1 - x_2) + y_1(y_1 - y_2) - x_2(x_1 - x_2) - y_2(y_1 - y_2) \right] \quad (94)$$

$$= -K_1 K_2 \frac{a\alpha\omega}{r_{12}(1+\beta r_{12})^2} \left[ (x_1 - x_2)(x_1 - x_2) + (y_1 - y_2)(y_1 - y_2) \right] \quad (95)$$

$$= -K_1 K_2 \frac{a\alpha\omega r_{12}}{(1+\beta r_{12})^2} \quad (96)$$

The analytic expression for the local energy is thus

$$E_L = \frac{\nabla^2 \Psi_T}{\Psi_T} = 2\alpha^2 \omega^2 (r_1^2 + r_2^2) - 4\alpha\omega - \frac{2a\alpha\omega r_{12}}{(1+\beta r_{12})^2} \quad (97)$$

$$+ \frac{2a}{(1+\beta r_{12})^2} \left[ \frac{a}{(1+\beta r_{12})^2} + \frac{1}{r_{12}} - \frac{2\beta}{1+\beta r_{12}} \right] \quad (98)$$

## References

- [1] M. Taut, *Two electrons in an external oscillator potential: Particular analytic solutions of a Coulomb interaction problem* Phys. Rev. A **48**, 3561 - 3566 (1993).
- [2] M. L. Pedersen, G. Hagen, M. Hjorth-Jensen, S. Kvaal, and F. Pederiva, *Ab initio computation of the energies of circular quantum dots* Phys. Rev. B **84**, 115302 (2011)
- [3] Jules W. Moskowitz, M. H. Kalos, *A new look at correlations in atomic and molecular systems. I. Application of fermion monte carlo variational method.* Int. J. Quantum Chem. **20** 1107 (1981)