FYS4411

Spring 2016

Variational Monte Carlo studies of electronic systems

John-Anders Stende

Date: June 15, 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

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 ω. 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 reporty

2 Systems

As mentioned in the introduction we want to study electrons confined in a Harmonic Oscillator (HO) trap in two dimensions. The trap we will use is an isotropic HO potential, with an idealized Hamiltonian given by

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

where natural units ($\hbar = c = e = m_e = 1$) are used and all energies are in atomic units a.u. The Hamiltonian includes a standard HO part

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

and a repulsive interaction between two electrons, given by

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

where $r_{ij} = |\mathbf{r}_1 - \mathbf{r}_2|$ is the distance between the electrons. The modulus of the positions of the electrons is defined as $r_i = \sqrt{r_{i_x}^2 + r_{i_y}^2}$.

2.1 Unperturbed system

If we only include the HO part of the Hamiltonian we say that the system is unperturbed since there are no interaction between the electrons. The wave function for one electron in an oscillator potential in two dimensions is

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

The functions $H_{n_x}(\sqrt{\alpha\omega}x)$ are Hermite polynomials while A is a normalization constant. For the lowest lying state, $n_x = n_y = 0$, the energy is given by $\epsilon_{n_x,n_y} = \omega(n_x + n_y + 1) = \omega$. Now, due the Pauli exclusion principle we can not have two electrons in the same state, however each energy level (distinct (n_x, n_y) pair) can contain two electrons since they can have either spin up or spin down. We can visualize this as follows: If we "fill" each energy level with

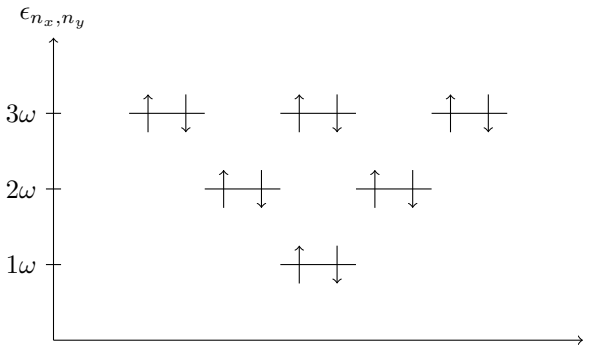


Figure 1: Fig text

electrons we say that we have a closed shell system, where the number of particles needed to fill each energy level constitutes a magic number. The first magic numbers are $N = 2, 6, 12, 20$. In the unperturbed case the ground state energy will just be the sum of each individual particle's energy.

N	E
2	2ω
6	10ω
12	28ω
20	60ω

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

This provides an excellent way to check that our program is working properly when we exclude interaction between the electrons.

3 Methods

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

$$E_0 \leq \frac{\langle \Psi_T | H | \Psi_T \rangle}{\langle \Psi_T | \Psi_T \rangle} \tag{3}$$

which states that the ground state energy is always less or equal than the expectation value of our Hamiltonian H for any trial wavefunction Ψ_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

$$\frac{\langle \Psi_T | H | \Psi_T \rangle}{\langle \Psi_T | \Psi_T \rangle} = \frac{\int d\mathbf{R} \Psi_T^*(\mathbf{R}, \boldsymbol{\alpha}) H(\mathbf{R}) \Psi_T(\mathbf{R}, \boldsymbol{\alpha})}{\int d\mathbf{R} \Psi_T^*(\mathbf{R}, \boldsymbol{\alpha}) \Psi_T(\mathbf{R}, \boldsymbol{\alpha})} \tag{4}$$

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 (4), or in general

$$I = \int_\Omega f(\mathbf{x}) d\mathbf{x} \tag{5}$$

Instead of using an explicit integration scheme, we sample points

$$\mathbf{x}_1 \dots \mathbf{x}_N \in \Omega \tag{6}$$

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

$$I \approx \frac{1}{N} \sum_{i=1}^{N} f(\mathbf{x}_i) \tag{7}$$

This simple approach is however not very efficient, as it samples an equal amount of points in all regions of Ω, 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 $T_{i \to j}$
- Accept the new state with probability $A_{i \to j}$ and use it as the next sample, or recect the new state with probability $1 - A_{i \to j}$ and use state i as sample again

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

$$\frac{A_{i \to j}}{A_{j \to i}} = \frac{p_i T_{i \to j}}{p_j T_{j \to i}} \tag{8}$$

which ensures that $\mathbf{P}_i^{(n \to \infty)} \to p_i$, 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

$$P(\mathbf{R}, \boldsymbol{\alpha}) = \frac{|\Psi_T(\mathbf{R}, \boldsymbol{\alpha})|^2}{\int |\Psi_T(\mathbf{R}, \boldsymbol{\alpha})|^2 d\mathbf{R}} \tag{9}$$

and the local energy,

$$E_L(\mathbf{R}, \boldsymbol{\alpha}) = \frac{1}{\Psi_T(\mathbf{R}, \boldsymbol{\alpha})} H\Psi_T(\mathbf{R}, \boldsymbol{\alpha}), \tag{10}$$

the integral (4) can be rewritten as

$$\langle E_L \rangle = \int P(\mathbf{R}, \boldsymbol{\alpha}) E_L(\mathbf{R}, \boldsymbol{\alpha}) d\mathbf{R} \tag{11}$$

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

$$\langle E_L \rangle \approx \frac{1}{N} \sum_{i=1}^{N} P(\mathbf{R}_i, \boldsymbol{\alpha}) E_L(\mathbf{R}_i, \boldsymbol{\alpha}) \tag{12}$$

where N is the number of Monte Carlo cycles and $\mathbf{R}_i$ is the position of the particles at step i. The integral $\int |\Psi_T(\mathbf{R}, \boldsymbol{\alpha})|^2 d\mathbf{R}$ 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 (8) as

$$\frac{p_j}{p_i} = \frac{T_{i \to j} A_{i \to j}}{T_{j \to i} A_{j \to i}} \tag{13}$$

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 $T_{i \to j}$ is symmetric, implying that $T_{i \to j} = T_{j \to i}$; the ratio of probabilities (13) 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

$$w = \frac{|\Psi_T(\mathbf{R}_j)|^2}{|\Psi_T(\mathbf{R}_i)|^2} \tag{14}$$

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 $\boldsymbol{\alpha}$. 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 (14). 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

$$\frac{\partial x(t)}{\partial t} = DF(x(t)) + \eta \tag{15}$$

where D is the diffusion constant and η a random variable. The new positions y in coordinate space are the solutions of (15) using Euler's method:

$$y = x + DF(x)\Delta t + \xi\sqrt{\Delta t} \tag{16}$$

where ξ is a gaussian random variable and Δt is a chosen time step. D is equal to $1/2$ which comes from the factor $1/2$ in the kinetic energy operator. Δt is to be viewed as a parameter which yields stable values of the ground state energy for values $\Delta t \in [0.001, 0.01]$. (16) 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

$$\frac{\partial P}{\partial t} = \sum_i D\frac{\partial}{\partial \mathbf{x}_i}\left(\mathbf{x}_i - \mathbf{F}_i\right)P(\mathbf{x}, t) \tag{17}$$

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,

$$\frac{\partial^2 P}{\partial \mathbf{x}_i^2} = P\frac{\partial}{\partial \mathbf{x}_i}\mathbf{F}_i + \mathbf{F}_i\frac{\partial}{\partial \mathbf{x}_i}P \tag{18}$$

The drift vector should have the form $\mathbf{F} = g(\mathbf{x})\frac{\partial P}{\partial \mathbf{x}}$. Inserting this in (18) yields

$$\mathbf{F} = 2\frac{1}{\Psi_T}\nabla\Psi_T \tag{19}$$

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 samling respectively. The Fokker-Planck equation (17) describes such a diffusion process, our new transition probabilty is thus the solution to this equation, given by the Green's function

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

which in turn means that our brute force Metropolis accept/reject ratio (14) is replaced by the so-called Metropolis-Hastings article

$$q(y,x) = \frac{G(x,y,\Delta t)|\Psi_T(y)|^2}{G(y,x,\Delta t)|\Psi_T(x)|^2} \tag{21}$$

The Metropolis Hastings algorithm is the same as the brute force method, now with (14) replaced by (21) and trial positions calculated according to (16).

3.4 Optimization of trial wave function and its deriviatives

The trial wave function (??) plays a central role in our VMC simulation. It is needed in the Metropolis algorithm and in the evaluation of the quantum force (19). Moreover, all observables like the local energy (10) 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 determinant-to-determinant ratio

The ratio (14) used in the Metropolis algorithm is the following for our trial wave function (not squared):

$$R = \frac{|D|^{\text{new}}}{|D|^{\text{old}}} \frac{\Psi_C^{\text{new}}}{\Psi_C^{\text{old}}} \tag{22}$$

where we label the Slater determinant-to-determinant ratio as R_{SD} and the correlation-to-correlation ratio R_C. It turns out that we can compute R_{SD} using an algorithm that requires to keep track of the *inverse* of the Slater matrix (??). The inverse of D can be expressed in terms of its cofactors C_{ij} and its determinant $|D|$,

$$D_{ij}^{-1} = \frac{C_{ji}}{|D|} \tag{23}$$

where C_{ji} is the transposed cofactor matrix. The Slater part R_{SD} of the ratio (14) can thus be written,

$$R_{SD} = \frac{|D(\mathbf{R}^{\text{new}})|}{|D(\mathbf{R}^{\text{old}})|} = \frac{\sum_{j=1}^{N} D_{ij}(\mathbf{R}^{\text{new}}) C_{ij}(\mathbf{R}^{\text{new}})}{\sum_{j=1}^{N} D_{ij}(\mathbf{R}^{\text{old}}) C_{ij}(\mathbf{R}^{\text{old}})} \tag{24}$$

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

$$C_{ij}(\mathbf{R}^{\text{new}}) = C_{ij}(\mathbf{R}^{\text{old}}) \quad j \in \{1, \ldots, N\} \tag{25}$$

and

$$R_{SD} = \frac{\sum_{j=1}^{N} D_{ij}(\mathbf{R}^{\text{new}}) D_{ji}^{-1}(\mathbf{R}^{\text{old}})}{\sum_{j=1}^{N} D_{ij}(\mathbf{R}^{\text{old}}) D_{ji}^{-1}(\mathbf{R}^{\text{old}})} \tag{26}$$

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

$$R_{SD} = \sum_{j=1}^{N} D_{ij}(\mathbf{R}^{\text{new}}) D_{ji}^{-1}(\mathbf{R}^{\text{old}}) = \sum_{j=1}^{N} \phi_j(\mathbf{R}_i^{\text{new}}) D_{ji}^{-1}(\mathbf{R}^{\text{old}}) \tag{27}$$

where the last equality follows from the definition of the Slater matrix (??). 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 (27) 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:

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

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

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

This means that we only need to invert the Slater matrix with conventional methods like Gaussian elimination or LU-decomposition once, after we have initialized them. For the subesequent steps, we use the above algorithm to obtain the inverse matrix.

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,

$$|D| = |D|_{\uparrow} \cdot |D|_{\downarrow} \tag{28}$$

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

$$|D| = \frac{1}{\sqrt{4!}} \begin{vmatrix} \phi_{00\uparrow}(\mathbf{r}_1) & \phi_{00\downarrow}(\mathbf{r}_1) & \phi_{10\uparrow}(\mathbf{r}_1) & \phi_{10\downarrow}(\mathbf{r}_1) \\ \phi_{00\uparrow}(\mathbf{r}_2) & \phi_{00\downarrow}(\mathbf{r}_2) & \phi_{10\uparrow}(\mathbf{r}_2) & \phi_{10\downarrow}(\mathbf{r}_2) \\ \phi_{00\uparrow}(\mathbf{r}_3) & \phi_{00\downarrow}(\mathbf{r}_3) & \phi_{10\uparrow}(\mathbf{r}_3) & \phi_{10\downarrow}(\mathbf{r}_3) \\ \phi_{00\uparrow}(\mathbf{r}_4) & \phi_{00\downarrow}(\mathbf{r}_4) & \phi_{10\uparrow}(\mathbf{r}_4) & \phi_{10\downarrow}(\mathbf{r}_4) \end{vmatrix} \tag{29}$$

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,

$$|D| = \frac{1}{\sqrt{2}} \begin{vmatrix} \phi_{00\uparrow}(\mathbf{r}_1) & \phi_{10\uparrow}(\mathbf{r}_1) \\ \phi_{00\uparrow}(\mathbf{r}_2) & \phi_{10\uparrow}(\mathbf{r}_2) \end{vmatrix} \cdot \frac{1}{\sqrt{2}} \begin{vmatrix} \phi_{00\downarrow}(\mathbf{r}_3) & \phi_{10\downarrow}(\mathbf{r}_3) \\ \phi_{00\downarrow}(\mathbf{r}_4) & \phi_{10\downarrow}(\mathbf{r}_4) \end{vmatrix} \tag{30}$$

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

$$\Psi_T = |D|_{\uparrow} |D|_{\downarrow} \Psi_C \tag{31}$$

The factorization above makes it possible to perform the calculation of the Slater ratio (27) and the updating of the inverse Slater matrix seperately for $|D|_{\uparrow}$ and $|D|_{\downarrow}$:

$$\frac{|D|^{\text{new}}}{|D|^{\text{old}}} = \frac{|D|_{\uparrow}^{\text{new}}}{|D|_{\downarrow}^{\text{old}}} \cdot \frac{|D|_{\uparrow}^{\text{new}}}{|D|_{\downarrow}^{\text{old}}} \tag{32}$$

We see from (30) 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 (19) used in importance sampling. This ratio can be written as

$$\frac{\nabla\Psi_T}{\Psi_T} = \frac{\nabla(\Psi_{SD}\Psi_C)}{\Psi_{SD}\Psi_C} = \frac{\Psi_C\nabla\Psi_{SD} + \Psi_{SD}\nabla\Psi_C}{\Psi_{SD}\Psi_C} = \frac{\nabla\Psi_D}{\Psi_D} + \frac{\nabla\Psi_C}{\Psi_C} \tag{33}$$

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

$$\frac{\nabla\Psi_T}{\Psi_T} = \frac{\nabla|D|_\uparrow}{|D|_\uparrow} + \frac{\nabla|D|_\downarrow}{|D|_\downarrow} + \frac{\nabla\Psi_C}{\Psi_C} \tag{34}$$

The process to obtain the gradient-determinant ratio needed in this expression is analogous to the one used in deriving (27). The result is

$$\frac{\nabla_i|D|}{|D|} = \sum_{j=1}^{N/2}\nabla_i D_{ij}(\mathbf{R}^{\text{old}})D^{-1}_{ji}(\mathbf{R}^{\text{old}}) = \sum_{j=1}^{N/2}\nabla_i\phi_j(\mathbf{R}_i^{\text{old}})D^{-1}_{ji}(\mathbf{R}^{\text{old}}) \tag{35}$$

where the sum runs up to $N/2$ for both $|D|_\uparrow$ and $|D|_\downarrow$. This expression is used to calculate the quantum force before the ratio (14) is computed. After the position of one particle is altered, the expression is modified to

$$\frac{\nabla_i|D|}{|D|} = \frac{1}{R_{SD}}\sum_{j=1}^{N/2}\nabla_i\phi_j(\mathbf{R}_i^{\text{new}})D^{-1}_{ji}(\mathbf{R}^{\text{old}}) \tag{36}$$

Note that the old inverse Slater matrix is used in both cases because this is not updated until after a step has been accepted. We see from (34) that we can compute the gradients of the spin-up and spin-down determinants seperately. Furthermore, according to the factorization (30), one of the Slater gradients is always zero for a given particle i:

- $\nabla_i|D|_\downarrow = 0$ for $i \leq N/2$
- $\nabla_i|D|_\uparrow = 0$ for $i > N/2$

thus we only need to compute one of them for each particle. This is implemented as follows:

Listing 2: Computation of the Slater gradient ratio for particle i. The spin-up matrix only contains the single-particle wave functions as functions of the first half of the particles. The spin-down matrix is a function of the second half. One of them is thus zero for a given particle i.

```
// get position of particle i
  double x = particles[i]->getPosition()[0];
  double y = particles[i]->getPosition()[1];

  // spin-up slater
  if (i < m_numberOfParticlesHalf) {
      for (int j=0; j < m_numberOfParticlesHalf; j++) {
          int nx = m_quantumNumbers(j,0);
          int ny = m_quantumNumbers(j,1);
          std::vector<double> grad = singleParticleWFGradient(nx, ny, x, y);
          m_gradientUp[0] += grad[0] * m_slaterSpinUpInverse(j,i);
          m_gradientUp[1] += grad[1] * m_slaterSpinUpInverse(j,i);
      }
  }
  // spin-down slater
  else {
      for (int j=0; j < m_numberOfParticlesHalf; j++) {
          int nx = m_quantumNumbers(j,0);
          int ny = m_quantumNumbers(j,1);
          std::vector<double> grad = singleParticleWFGradient(nx, ny, x, y);
          m_gradientDown[0] += grad[0] *
          m_slaterSpinDownInverse(j,i-m_numberOfParticlesHalf);
          m_gradientDown[1] += grad[1] *
          m_slaterSpinDownInverse(j,i-m_numberOfParticlesHalf);
      }
  }
```

3.4.5 Optimization of the Laplacian

We need the Laplacian of the trial wave function to compute the expectation value of the kinetic energy K. For electron i this is

$$\langle K_i \rangle = -\frac{1}{2}\frac{\langle \Psi_T | \nabla_i^2 | \Psi_T \rangle}{\langle \Psi_T | \Psi \rangle_T} \tag{37}$$

It can be shown that the Laplacian of the trial wave function can be written

$$\frac{\nabla^2 \Psi_T}{\Psi_T} = \frac{\nabla^2 |D|_\uparrow}{|D|_\uparrow} + \frac{\nabla^2 |D|_\downarrow}{|D|_\downarrow} + \frac{\nabla^2 \Psi_C}{\Psi_C} + 2\left[\frac{\nabla |D|_\uparrow}{|D|_\uparrow} + \frac{\nabla |D|_\downarrow}{|D|_\downarrow}\right] \cdot \frac{\nabla \Psi_C}{\Psi_C} \tag{38}$$

where the Laplace-determinant-to-determinant ratio is given by

$$\frac{\nabla_i^2 |D|}{|D|} = \sum_{j=1}^{N/2} \nabla_i^2 D_{ij}(\mathbf{R}^{\text{new}}) D_{ji}^{-1}(\mathbf{R}^{\text{new}}) = \sum_{j=1}^{N/2} \nabla_i^2 \phi_j(\mathbf{R}_i^{\text{new}}) D_{ji}^{-1}(\mathbf{R}^{\text{new}}) \tag{39}$$

This expression must be computed for all electrons to obtain the total Laplacian for the whole system, which is needed in the local energy (10). To save computation time, we only recalculate E_L when a step is accepted.

3.4.6 Optimization of the correlation-to-correlation ratio

The correlation function Ψ_C (**??**) can be written

$$\Psi_C = \prod_{i<j}^{N} \exp(f_{ij}) = \exp(U) \tag{40}$$

where

$$f_{ij} = \frac{a_{ij} r_{ij}}{1 + \beta r_{ij}} \tag{41}$$

and

$$U = \sum_{i<j}^{N} f_{ij} \tag{42}$$

We see that Ψ_C depends on the relative distances r_{ij} between the electrons. The total number of different relative distances is $N(N-1)/2$, computing the correlation-to-correeation ratio R_C therefore scales as $\mathcal{O}(N^2)$. However, when moving only one electron at a time, say the k-th electron, only the $N-1$ distances having k as one of their indicies are changed. The rest of the factors thus cancel, and we have

$$R_C = \frac{\Psi_C^{\text{new}}}{\Psi_C^{\text{old}}} = \frac{\exp(U^{\text{new}})}{\exp(U^{\text{old}})} = \exp(\Delta U) \tag{43}$$

where

$$\Delta U = \sum_{i \neq k}^{N} (f_{ki}^{\text{new}} - f_{ki}^{\text{old}}) \tag{44}$$

3.4.7 Optimization of the $\nabla \Psi_C / \Psi_C$ ratio

The expression to be derived in the following is of interest when computing the quantum force (19) and the kinetic energy (37). From the discussion in the last section, only $N-1$ terms survive when we differentiate w.r.t. particle k. The ratio $\nabla \Psi_C / \Psi_C$ for particle i in one dimension is thus

$$\frac{1}{\Psi_C}\frac{\partial \Psi_C}{\partial x_k} = \frac{1}{\exp U}\frac{\partial \exp U}{\partial x_k} = \frac{\partial U}{\partial x_k} = \sum_{i \neq k}^{N} \frac{\partial f_{ki}}{\partial x_k} \tag{45}$$

where

$$\frac{\partial f_{ki}}{\partial x_k} = \frac{\partial f_{ki}}{\partial r_{ki}}\frac{\partial r_{ki}}{\partial x_k} = \frac{a_{ki}}{(1 + \beta r_{ki})^2}\frac{x_k - x_i}{r_{ki}} \tag{46}$$

with similar expressions for y_k.

3.4.8 Optimization of the $\nabla^2\Psi_C/\Psi_C$ ratio

The following ratio is needed to compute the Laplacian $\nabla^2\Psi_T/\Psi_T$. For particle k in one dimension this is

$$\frac{1}{\Psi_C}\frac{\partial^2\Psi_C}{\partial x_k^2} = \frac{1}{\exp U}\frac{\partial}{\partial x_k}\left(\frac{\partial \exp U}{\partial x_k}\right) = \frac{1}{\exp U}\frac{\partial}{\partial x_k}\left(\exp U\frac{\partial \exp U}{\partial x_k}\right) \tag{47}$$

$$= \left(\frac{\partial U}{\partial x_k}\right)^2 + \frac{\partial^2 U}{\partial x_k^2} = \left(\frac{1}{\Psi_C}\frac{\partial \Psi_C}{\partial x_k}\right)^2 + \sum_{i\neq k}^{N}\frac{\partial^2 f_{ki}}{\partial x_k^2} \tag{48}$$

The total Laplacian for particle k is thus

$$\frac{\nabla_k^2\Psi_C}{\Psi_C} = \left(\frac{\nabla_k\Psi_C}{\Psi_C}\right)^2 + \sum_{i\neq k}^{N}\nabla_i^2 f_{ki} \tag{49}$$

where

$$\nabla_i^2 f_{ki} = \frac{1}{r_{ki}}\frac{\partial f_{ki}}{\partial r_{ki}} + \frac{\partial^2 f_{ki}}{\partial r_{ki}^2} \tag{50}$$

and

$$\frac{\partial^2 f_{ki}}{\partial r_{ki}^2} = -\frac{2a_{ki}\beta}{(1+\beta r_{ki})^3} \tag{51}$$

We must sum over all particles k and both dimensions to obtain the full Laplacian for the system.

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 α and β. 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 γ_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 $|\nabla_{\boldsymbol{\alpha}_i}\langle E_L(\mathbf{R},\boldsymbol{\alpha}_i)\rangle|^2 \geq |\nabla_{\boldsymbol{\alpha}_{i+1}}\langle E_L(\mathbf{R},\boldsymbol{\alpha}_{i+1})\rangle|^2 \geq \ldots$. If this is not the case, we reject the new step and decrease the step size to obtain a more accurate value. $\langle E_L(\mathbf{R},\boldsymbol{\alpha})\rangle$ is as we have seen a multidimensional integral (11), and the gradient w.r.t. $\boldsymbol{\alpha}$ is not easily computed. Let us define

$$\bar{E}_{c_n} = \frac{\partial\langle E_L(\boldsymbol{\alpha})\rangle}{\partial c_n} \tag{52}$$

where c_n is a variational parameter. Using the chain rule and the hermicity of the Hamiltonian, it can be shown that

$$\bar{E}_{c_n} = 2\left(\left\langle\frac{\partial \ln\Psi_T}{\partial c_n}E_L\right\rangle - \left\langle\frac{\partial \ln\Psi_T}{\partial c_n}\right\rangle\langle E_L\rangle\right) \tag{53}$$

thus we need the expectation values of

$$\frac{\partial \ln\Psi_T}{\partial c_n}E_L \tag{54}$$

and

$$\frac{\partial \ln\Psi_T}{\partial c_n} \tag{55}$$

For our trial wave function (**??**), we have

$$\frac{\partial \ln\Psi_T}{\partial c_n} = \frac{\partial \ln|D|_\uparrow}{\partial c_n} + \frac{\partial \ln|D|_\downarrow}{\partial c_n} + \frac{\partial \ln\Psi_C}{\partial c_n} \tag{56}$$

Inserting our variational parameters α and β, we observe that

$$\frac{\partial \ln\Psi_C}{\partial \alpha} = 0 \tag{57}$$

and

$$\frac{\partial \ln |D|_\uparrow}{\partial \beta} = \frac{\partial \ln |D|_\downarrow}{\partial \beta} = 0 \tag{58}$$

so we end up with

$$\frac{\partial \ln \Psi_T}{\partial \alpha} = \frac{\partial \ln |D|_\uparrow}{\partial \alpha} + \frac{\partial \ln |D|_\downarrow}{\partial \alpha} \tag{59}$$

and

$$\frac{\partial \ln \Psi_T}{\partial \beta} = \frac{\partial \ln \Psi_C}{\partial \beta} \tag{60}$$

We can calculate the above derivatives with the help of the following linear algebra identity: If A is an invertible matrix which depends on a real parameter t, and if dA/dt exists, then

$$\frac{d}{dt}|A| = |A| \mathrm{tr}\left(A^{-1}\frac{dA}{dt}\right) \tag{61}$$

thus we have

$$\frac{d}{dt}\ln|A|(t) = \mathrm{tr}\left(A^{-1}\frac{dA}{dt}\right) = \sum_{i=1}^{N}\sum_{j=1}^{N}\frac{dA_{ij}}{dt}A^{-1}_{ji} \tag{62}$$

For the Slater determinants, this expression is analogous to the gradient-ratio (35), only now we differentiate the single-particle wave functions w.r.t. α:

$$\frac{\partial \ln |D|}{\partial \alpha} = \sum_{i=1}^{N/2}\sum_{j=1}^{N/2}\frac{\partial \phi_j(\mathbf{R}_i)}{\partial \alpha}D^{-1}_{ji}(\mathbf{R}_i) \tag{63}$$

The corresponding expression for the correlation function is

$$\frac{\partial \ln \Psi_C}{\partial \beta} = \sum_{i=1}^{N}\sum_{j>i}^{N}\frac{\partial}{\partial \beta}\left(\frac{a_{ij}r_{ij}}{1+\beta r_{ij}}\right) = -\sum_{i=1}^{N}\sum_{j>i}^{N}\frac{a_{ij}r_{ij}^2}{(1+\beta r_{ij})^2} \tag{64}$$

The complete VMC method then amounts to the following:

- Make initial guess $\boldsymbol{\alpha}_0$
- Run 10^4-10^5 Metropolis steps, sample (??) and (??)
- Compute (53)
- Calculate new $\boldsymbol{\alpha}$ 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 3: 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 << "newAlhpa = " << 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

$$\{x_1, x_2 \dots x_k \dots x_N\} \tag{65}$$

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

$$\mathrm{var}(x) = \frac{1}{N}\sum_{k=1}^{n}(x_k - \bar{x}_N) \tag{66}$$

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 err_X of a correlated sample is

$$\mathrm{err}_X = \frac{1}{N}\mathrm{cov}(x) \tag{67}$$

where cov(x) is the sample covariance

$$\mathrm{cov}(x) \equiv \frac{1}{N}\sum_{kl}(x_k - \bar{x}_N)(x_l - \bar{x}_N) \tag{68}$$

which is a measure of the sequential correlation between succeding measurements of a sample. (Note that (66) and (68) 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

$$\mathrm{err}_X = \frac{\tau}{N}\mathrm{var}(x) \tag{69}$$

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

$$n_{\mathrm{eff}} = \frac{N}{\tau} \tag{70}$$

Neglecting τ 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,

$$\sigma = \left(\frac{1 + 2\tau/\Delta t}{N}\left(\langle E_L^2\rangle - \langle E_L\rangle^2\right)\right)^{1/2} \tag{71}$$

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 $[in_b, (i+1)n_b], \ldots$
  - Calculate total mean and variance of all block means and store
- Plot total variance for all block sizes.
- Extract τ and compute (71)

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 evalute the analytical expressions for Ψ_T, $\nabla\Psi_T$, $\nabla^2\Psi_T$ and $d\Psi_T/d\alpha$.
  - *SimpleGaussian*: Implements the above quantities for Ψ_T used in system 1
  - *InteractingGaussian*: Implements the above quantities for Ψ_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 recieves 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 and discussion

4.1 Some text

4.2 More than two particles

4.3 Performance analysis

Table 3: Optimal α and β values for given ω's with jastrow factor.

ω	α	β
1	0.992067	0.400016
0.5	0.952981	0.354743
0.1	0.952833	0.354292
0.05	0.913976	0.235196
0.01	0.911692	0.203919

Table 4: Optimal α values for given ω's without jastrow factor.

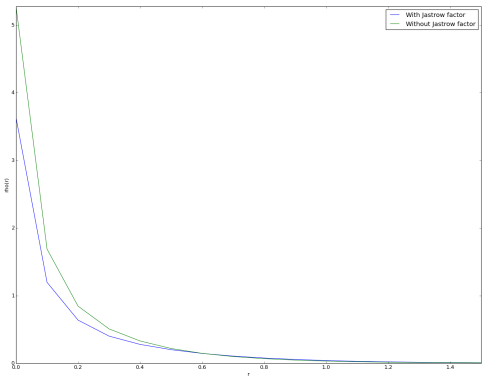
ω	α
1	0.686717
0.5	0.669455
0.1	0.655921
0.05	0.526488
0.01	0.477602

Table 5: Optimal α and β values for given ω's with jastrow factor and Slater determinant.

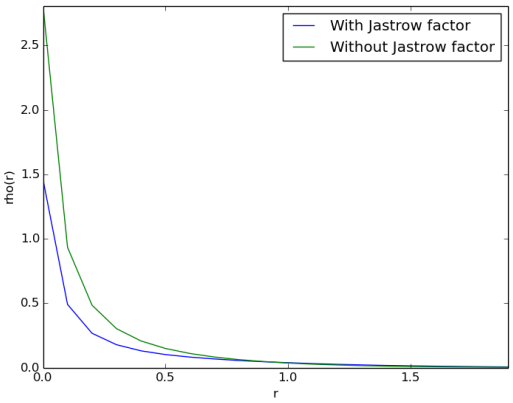
N	ω	α	β
6	1	1.03741	0.472513
	0.5	0.931202	0.395044
	0.1	0.831104	0.211443
	0.05	0.849742	0.155566
	0.01	0.842683	0.10789
12	1	1.10364	0.468861
	0.5	0.936306	0.414534
	0.1	0.84105	0.208143
	0.05	0.842746	0.153921
	0.01	0.835467	0.0923274
20	1	1.06019	0.474467
	0.5	0.944121	0.419367
	0.1	0.856981	0.200372
	0.05	0.842341	0.153017
	0.01	0.835301	0.0922157

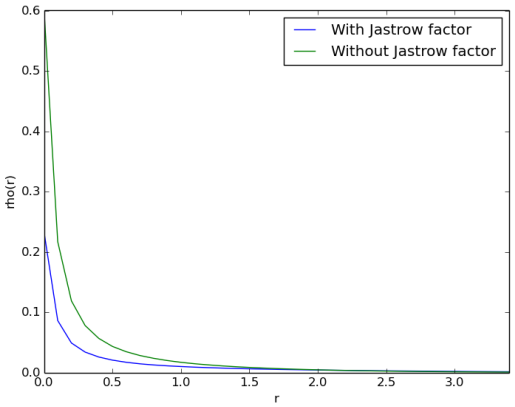
Table 6: Optimal α values for given ω's with Slater determinant and without jastrow factor

N	ω	α
6	1	1.03741
	0.5	0.931202
	0.1	0.831104
	0.05	0.849742
	0.01	0.842683
12	1	1.10364
	0.5	0.936306
	0.1	0.84105
	0.05	0.842746
	0.01	0.835467
20	1	1.06019
	0.5	0.944121
	0.1	0.856981
	0.05	0.842341
	0.01	0.835301

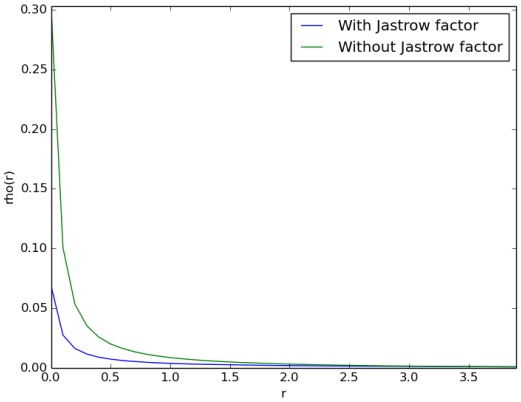


(a) $\omega = 1$ and $N = 2$ without Slater determinant.

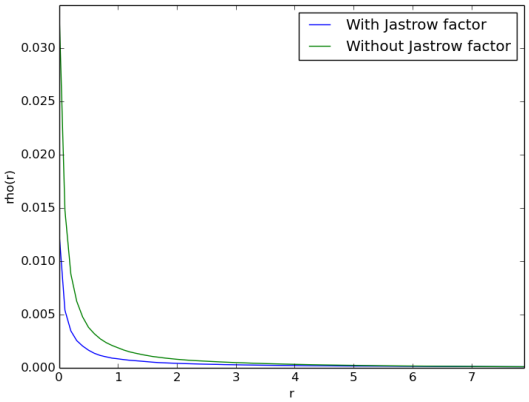




(c) $\omega = 0.1$ and $N = 2$ without Slater determinant.

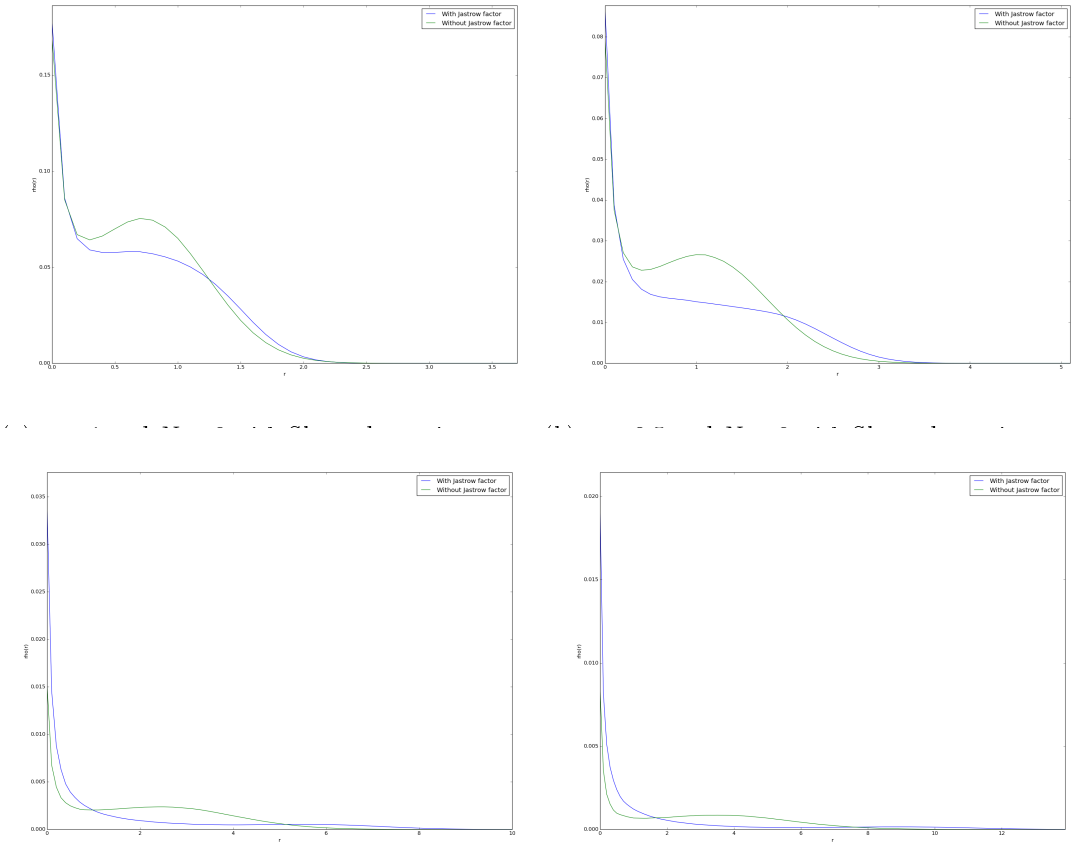


(d) $\omega = 0.05$ and $N = 2$ without Slater determinant.



(e) $\omega = 0.01$ and $N = 2$ without Slater determinant.

Figure 2: blablabla



(c) $\omega = 0.1$ and $N = 6$ with Slater determinant. (d) $\omega = 0.05$ and $N = 6$ with Slater determinant.

Figure 3: blablabla

5 Conclusions

6 Appendix

6.1 Analytic energy for two-body quantum dot

$$\Psi_T(\mathbf{r}_1, \mathbf{r}_2) = \exp(-\alpha\omega(r_1^2 + r_2^2)/2)\exp\left(\frac{ar_{12}}{1+\beta r_{12}}\right) \tag{72}$$

$$= K_1 K_2 \tag{73}$$

Laplacian of this wave function:

$$\nabla^2\Psi_T(\mathbf{r}_1, \mathbf{r}_2) = \frac{\partial\Psi_T}{\partial x_1^2} + \frac{\partial\Psi_T}{\partial y_1^2} + \frac{\partial\Psi_T}{\partial x_2^2} + \frac{\partial\Psi_T}{\partial y_2^2} \tag{74}$$

Using the following

$$, \frac{\partial r_1}{\partial x_1} = x_1/r_1 \tag{75}$$

and

$$\frac{\partial r_{12}}{\partial x_1} = (x_1 - x_2)/r_1 \tag{76}$$

Gradient of first term:

$$\frac{\partial K_1}{\partial x_1} = -\alpha\omega x_1 K_1 \tag{77}$$

so that

$$\nabla K_1 = -\alpha\omega K_1 \mathbf{r} \tag{78}$$

where

$$\mathbf{r} = (x_1, y_1, x_2, y_2) \tag{79}$$

Gradient of second term:

$$\frac{\partial K_2}{\partial x_1} = K_2 \frac{a(x_1 - x_2)}{r_{12}(1+\beta r_{12})^2} \tag{80}$$

so that

$$\nabla K_2 = K_2 \frac{a}{r_{12}(1+\beta r_{12})^2} \mathbf{r}_{12} \tag{81}$$

where

$$\mathbf{r}_{12} = (x_1 - x_2, y_1 - y_2, x_2 - x_1, y_2 - y_1) \tag{82}$$

The quantum force used in importance sampling is thus

$$F = 2\frac{\nabla \Psi_T}{\Psi_T} = 2\frac{\nabla(K_1 K_2)}{K_1 K_2} = 2\frac{1}{K_1 K_2}(K_2 \nabla K_1 + K_1 \nabla K_2) = -2\alpha\omega\mathbf{r} + \frac{2a}{r_{12}(1+\beta r_{12})^2}\mathbf{r}_{12} \tag{83}$$

Laplacian of first term:

$$\frac{\partial^2 K_1}{\partial x_1^2} = K_1(\alpha^2\omega^2 x_1^2 - \alpha\omega) \tag{84}$$

so that

$$\nabla^2 K_1 = K_1(\alpha^2\omega^2(r_1^2 + r_2^2) - 4\alpha\omega) \tag{85}$$

Laplacian of second term:

$$\frac{\partial^2 K_2}{\partial x_1^2} = K_2 \left[\frac{a^2(x_1 - x_2)^2}{r_{12}^2(1+\beta r_{12})^4} + \frac{a r_{12}(1+\beta r_{12})^2}{r_{12}^2(1+\beta r_{12})^4} \right. \tag{86}$$

$$\left. - \frac{a(x_1 - x_2)[(x_1 - x_2)/r_{12}(1+\beta r_{12})^2 + 2r_{12}(1+\beta r_{12})\beta(x_1 - x_2)/r_{12}]}{r_{12}^2(1+\beta r_{12})^4} \right] \tag{87}$$

$$= K_2 \left[\frac{a^2(x_1 - x_2)^2}{r_{12}^2(1+\beta r_{12})^4} + \frac{a}{r_{12}(1+\beta r_{12})^2} \right. \tag{88}$$

$$\left. - \frac{a(x_1 - x_2)^2}{r_{12}^3(1+\beta r_{12})^2} - \frac{2a\beta(x_1 - x_2)^2}{r_{12}^2(1+\beta r_{12})^3} \right] \tag{89}$$

so that

$$\nabla^2 K_2 = K_2 \left[\frac{2a^2}{(1+\beta r_{12})^4} + \frac{4a}{r_{12}(1+\beta r_{12})^2} - \frac{2a}{r_{12}(1+\beta r_{12})^2} - \frac{2a\beta}{(1+\beta r_{12})^3} \right] \tag{90}$$

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

Have that

$$\nabla^2 \Psi_T = \nabla^2 K_1 K_2 + 2\nabla K_1 \nabla K_2 + K_1 \nabla^2 K_2 \tag{92}$$

and

$$\nabla K_1 \nabla K_2 = -K_1 K_2 \frac{a\alpha\omega}{r_{12}(1+\beta r_{12})^2} \Big[x_1(x_1 - x_2) + y_1(y_1 - y_2) - x_2(x_1 - x_2) - y_2(y_1 - y_2) \Big] \tag{93}$$

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

$$= -K_1 K_2 \frac{a\alpha\omega r_{12}}{(1+\beta r_{12})^2} \tag{95}$$

The analytic expression for the Laplacian ratio is thus

$$\frac{\nabla^2 \Psi_T}{\Psi_T} = 2\alpha^2\omega^2(r_1^2 + r_2^2) - 4\alpha\omega - \frac{2a\alpha\omega r_{12}}{(1+\beta r_{12})^2} \tag{96}$$

$$+ \frac{2a}{(1+\beta r_{12})^2} \left[\frac{a}{(1+\beta r_{12})^2} + \frac{1}{r_{12}} - \frac{2\beta}{1+\beta r_{12}} \right] \tag{97}$$

and we get for the local energy in the Hamiltonian (??)

$$E_L = \frac{1}{\Psi_T} H\Psi_T = -\frac{1}{2}\frac{\nabla^2\Psi_T}{\Psi_T} + \frac{1}{2}\omega^2(r_1^2 + r_2^2) + \frac{1}{r_{12}} \tag{98}$$

6.2 Hermite polynomials and their derivatives

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 inito 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)