FYS4411

Spring 2016

Variational Monte Carlo studies of electronic systems

John-Anders Stende

Date: May 30, 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 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

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

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 ω using the above Hamiltonian. The Hamiltonian includes a standard harmonic oscillator part

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

and the repulsive interaction between two electrons given by

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

with the distance between electrons given by $r_{ij} = |\mathbf{r}_1 - \mathbf{r}_2|$. We define the modulus of the positions of the electrons (for a given electron i) as $r_i = \sqrt{r_{i_x}^2 + r_{i_y}^2}$.

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 Ψ_T is a product of a Slater determinant $\Psi_{SD} = |D|$ and a so-called Jastrow correlation term Ψ_J,

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

where α and β are the variational parameters. The Slater matrix D is defined as

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

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

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

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

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

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 withouth interaction is a well known problem in quantum mechanics. The exact ground state energy for one electron in two dimensions is (in atomic units),

$$\varepsilon_{n_x,n_y} = \omega(n_x + n_y + 1) \tag{8}$$

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 $n = n_x + n_y$),

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

- $n = 2 \quad (2,0) \quad (1,1) \quad (0,2)$
- $n = 1 \quad (1,0) \quad (0,1)$
- $n = 0 \quad (0,0)$

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 (8) we obtain the following energies,

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.

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 $E = 3\,a.u.$. 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 fermonic system. VMC applies the *variational principle* from quantum mechanics

$$E_0 \leq \frac{\langle\Psi_T|H|\Psi_T\rangle}{\langle\Psi_T|\Psi_T\rangle} \tag{9}$$

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

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

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 (10), or in general

$$I = \int_\Omega f(\mathbf{x})d\mathbf{x} \tag{11}$$

Instead of using an explicit integration scheme, we sample points

$$\mathbf{x}_1 \dots \mathbf{x}_N \in \Omega \tag{12}$$

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

$$I \approx \frac{1}{N}\sum_{i=1}^{N} f(\mathbf{x}_i) \tag{13}$$

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

$$\frac{A_{i\to j}}{A_{j\to i}} = \frac{p_i T_{i\to j}}{p_j T_{j\to i}} \tag{14}$$

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

$$P(\mathbf{R},\boldsymbol{\alpha}) = \frac{|\Psi_T(\mathbf{R},\boldsymbol{\alpha})|^2}{\int |\Psi_T(\mathbf{R},\boldsymbol{\alpha})|^2 d\mathbf{R}} \tag{15}$$

and the local energy,

$$E_L(\mathbf{R},\boldsymbol{\alpha}) = \frac{1}{\Psi_T(\mathbf{R},\boldsymbol{\alpha})} H\Psi_T(\mathbf{R},\boldsymbol{\alpha}), \tag{16}$$

the integral (10) can be rewritten as

$$\langle E_L \rangle = \int P(\mathbf{R},\boldsymbol{\alpha}) E_L(\mathbf{R},\boldsymbol{\alpha}) d\mathbf{R} \tag{17}$$

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

$$\langle E_L \rangle \approx \frac{1}{N}\sum_{i=1}^{N} P(\mathbf{R}_i,\boldsymbol{\alpha}) E_L(\mathbf{R}_i,\boldsymbol{\alpha}) \tag{18}$$

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 (14) as

$$\frac{p_j}{p_i} = \frac{T_{i\to j} A_{i\to j}}{T_{j\to i} A_{j\to i}} \tag{19}$$

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 (19) 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

$$w = \frac{|\Psi_T(\mathbf{R}_j)|^2}{|\Psi_T(\mathbf{R}_i)|^2} \tag{20}$$

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 (20). 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

$$\frac{\partial x(t)}{\partial t} = DF(x(t)) + \eta \tag{21}$$

where D is the diffusion constant and η a random variable. The new positions y in coordinate space are the solutions of (21) using Euler's method:

$$y = x + DF(x)\Delta t + \xi\sqrt{\Delta t} \tag{22}$$

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 $\Delta t \in [0.001, 0.01]$. (22) 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

$$\frac{\partial P}{\partial t} = \sum_i D\frac{\partial}{\partial \mathbf{x}_i}\left(\mathbf{x}_i - \mathbf{F}_i\right)P(\mathbf{x}, t) \tag{23}$$

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,

$$\frac{\partial^2 P}{\partial \mathbf{x}_i^2} = P\frac{\partial}{\partial \mathbf{x}_i}\mathbf{F}_i + \mathbf{F}_i\frac{\partial}{\partial \mathbf{x}_i}P \tag{24}$$

The drift vector should have the form $\mathbf{F} = g(\mathbf{x})\frac{\partial P}{\partial \mathbf{x}}$. Inserting this in (24) yields

$$\mathbf{F} = 2\frac{1}{\Psi_T}\nabla\Psi_T \tag{25}$$

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 (23) describes such a diffusion process, our new transition probabilty is thus the solution to this equation, given by the Green's function

$$G(y,x,\Delta t) = \frac{1}{(4\pi D\Delta t)^{3N/2}}\exp(-(y-x-D\Delta tF(x))^2/4D\Delta t) \tag{26}$$

which in turn means that our brute force Metropolis accept/reject ratio (20) is replaced by the so-called Metropolis-Hastings article

$$q(y,x) = \frac{G(x,y,\Delta t)|\Psi_T(y)|^2}{G(y,x,\Delta t)|\Psi_T(x)|^2} \tag{27}$$

The Metropolis Hastings algorithm is the same as the brute force method, now with (20) replaced by (27) and trial positions calculated according to (22).

3.4 Optimization of Slater determinant

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 (25). Moreover, all observables like the local energy (16) 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.

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 $|D|$,

$$d_{ij}^{-1} = \frac{C_{ji}}{|D|} \tag{28}$$

where C_{ji} is the transposed cofactor matrix. The Slater part R_{SD} of the ratio (20) can thus be written,

$$R_{SD} = \frac{|D(\mathbf{R}^{\text{new}})|}{|D(\mathbf{R}^{\text{old}})|} = \frac{\sum_{j=1}^{N} d_{ij}(\mathbf{R}^{\text{new}})C_{ij}(\mathbf{R}^{\text{new}})}{\sum_{j=1}^{N} d_{ij}(\mathbf{R}^{\text{old}})C_{ij}(\mathbf{R}^{\text{old}})} \tag{29}$$

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

$$C_{ij}(\mathbf{R}^{\text{new}}) = C_{ij}(\mathbf{R}^{\text{old}}) \quad j \in \{1,\dots,N\} \tag{30}$$

and

$$R_{SD} = \frac{\sum_{j=1}^{N} d_{ij}(\mathbf{R}^{\text{new}})d_{ji}^{-1}(\mathbf{R}^{\text{old}})}{\sum_{j=1}^{N} d_{ij}(\mathbf{R}^{\text{old}})d_{ji}^{-1}(\mathbf{R}^{\text{old}})} \tag{31}$$

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

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

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 $\mathcal{O}(N)$.

3.4.1 Optimization of inverse Slater matrix

The operation (32) 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

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

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

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

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

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

3.4.2 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,

$$|D| = |D|_{\uparrow} \cdot |D|_{\downarrow} \tag{33}$$

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

$$|D| = \frac{1}{\sqrt{4!}} \begin{vmatrix} \phi_{00\uparrow}(\mathbf{r}_1) & \phi_{00\downarrow}(\mathbf{r}_1) & \phi_{10\uparrow}(\mathbf{r}_1) & \phi_{10\downarrow}(\mathbf{r}_1) \\ \phi_{00\uparrow}(\mathbf{r}_2) & \phi_{00\downarrow}(\mathbf{r}_2) & \phi_{10\uparrow}(\mathbf{r}_2) & \phi_{10\downarrow}(\mathbf{r}_2) \\ \phi_{00\uparrow}(\mathbf{r}_3) & \phi_{00\downarrow}(\mathbf{r}_3) & \phi_{10\uparrow}(\mathbf{r}_3) & \phi_{10\downarrow}(\mathbf{r}_3) \\ \phi_{00\uparrow}(\mathbf{r}_4) & \phi_{00\downarrow}(\mathbf{r}_4) & \phi_{10\uparrow}(\mathbf{r}_4) & \phi_{10\downarrow}(\mathbf{r}_4) \end{vmatrix} \tag{34}$$

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,

$$|D| = \frac{1}{\sqrt{2}} \begin{vmatrix} \phi_{00\uparrow}(\mathbf{r}_1) & \phi_{10\uparrow}(\mathbf{r}_1) \\ \phi_{00\uparrow}(\mathbf{r}_2) & \phi_{10\uparrow}(\mathbf{r}_2) \end{vmatrix} \cdot \frac{1}{\sqrt{2}} \begin{vmatrix} \phi_{00\downarrow}(\mathbf{r}_3) & \phi_{10\downarrow}(\mathbf{r}_3) \\ \phi_{00\downarrow}(\mathbf{r}_4) & \phi_{10\downarrow}(\mathbf{r}_4) \end{vmatrix} \tag{35}$$

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

$$\Psi_T = |D|_{\uparrow} |D|_{\downarrow} \Psi_C \tag{36}$$

The factorization above makes it possible to perform the calculation of the Slater ratio (32) and the updating of the inverse Slater matrix seperately for $|D|_{\uparrow}$ and $|D|_{\downarrow}$:

$$\frac{|D|^{\text{new}}}{|D|^{\text{old}}} = \frac{|D|^{\text{new}}_{\uparrow}}{|D|^{\text{old}}_{\downarrow}} \cdot \frac{|D|^{\text{new}}_{\uparrow}}{|D|^{\text{old}}_{\downarrow}} \tag{37}$$

We see from (35) 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 (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

3.5 Optimization of gradient and Laplacian

We need the ratio $\frac{\nabla\Psi_T}{\Psi_T}$ to compute the quantum force (25) used in importance sampling. This ratio can be written as

$$\frac{\nabla\Psi_T}{\Psi_T} = \frac{\nabla(\Psi_{SD}\Psi_C)}{\Psi_{SD}\Psi_C} = \frac{\Psi_C\nabla\Psi_{SD} + \Psi_{SD}\nabla\Psi_C}{\Psi_{SD}\Psi_C} = \frac{\nabla\Psi_D}{\Psi_D} + \frac{\nabla\Psi_C}{\Psi_C} \tag{38}$$

3.6 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 (17), and the gradient w.r.t. $\boldsymbol{\alpha}$ is not easily computed. Let us define

$$\bar{E}_\alpha = \frac{\partial\langle E_L(\boldsymbol{\alpha})\rangle}{\partial\alpha} \tag{39}$$

and

$$\bar{E}_\beta = \frac{\partial\langle E_L(\boldsymbol{\alpha})\rangle}{\partial\beta} \tag{40}$$

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

$$\bar{E}_\alpha = 2\left(\left\langle\frac{\partial\ln\Psi_T}{\partial\alpha}E_L\right\rangle - \left\langle\frac{\partial\ln\Psi_T}{\partial\alpha}\right\rangle\langle E_L\rangle\right) \tag{41}$$

and

$$\bar{E}_\beta = 2\left(\left\langle\frac{\partial\ln\Psi_T}{\partial\beta}E_L\right\rangle - \left\langle\frac{\partial\ln\Psi_T}{\partial\beta}\right\rangle\langle E_L\rangle\right) \tag{42}$$

thus we need the expectation values of

$$\frac{\partial\ln\Psi_T}{\partial\alpha}E_L \quad \frac{\partial\ln\Psi_T}{\partial\alpha} \tag{43}$$

and

$$\frac{\partial\ln\Psi_T}{\partial\beta}E_L \quad \frac{\partial\ln\Psi_T}{\partial\beta} \tag{44}$$

For our trial wave function (4), we have

$$\ln\Psi_T = \tag{45}$$

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 (41)
- 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 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 << "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);
}
```

.

3.7 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

$$\{x_1, x_2 \dots x_k \dots x_N\} \tag{46}$$

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

$$\mathrm{var}(x) = \frac{1}{N}\sum_{k=1}^{n}(x_k - \bar{x}_N) \tag{47}$$

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

$$\mathrm{err}_X = \frac{1}{N}\mathrm{cov}(x) \tag{48}$$

where cov(x) is the sample covariance

$$\mathrm{cov}(x) \equiv \frac{1}{N}\sum_{kl}(x_k - \bar{x}_N)(x_l - \bar{x}_N) \tag{49}$$

which is a measure of the sequential correlation between succeding measurements of a sample. (Note that (47) and (49) 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

$$\mathrm{err}_X = \frac{\tau}{N}\mathrm{var}(x) \tag{50}$$

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

$$n_{\mathrm{eff}} = \frac{N}{\tau} \tag{51}$$

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,

$$\sigma = \left(\frac{1 + 2\tau/\Delta t}{N}\left(\langle E_L^2\rangle - \langle E_L\rangle^2\right)\right)^{1/2} \tag{52}$$

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 τ and compute (52)

3.8 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

We calculate the ground state energy for system 1 both with and without importance sampling. Only the brute force Metropolis algorithm is applied for system 2. The kinetic energy is calculated both numerically and analytically for system 1, referred to as the *numerical method* and the *analytical method* respectively. Only the analytical method is applied for system 2.
We use blocking to analyze the error of the statistical data for system 2. The steepest descent method is applied to optimize α. In addition to this, the one-body density is computed for both systems.

4.1 System 1 - brute force Metropolis

The parameters used to produce the below results are as follows

Listing 3: Parameters brute force Metropolis system 1

```
    int numberOfSteps       = (int) 1e4;
    double omega            = 1.0;          // oscillator frequency
    double alpha            = 0.5;          // variational parameter 1
    double stepLength       = 1.5;          // metropolis step length
    double equilibration    = 0.1;
```

We want to valide our code using the benchmark (8) and compare the CPU time difference for computing kinetic energy numerically vs analytically. The Metropolis step length is set so that the acceptance rate equals about 0.5. The number of Metropolis steps is only 1e4 because setting $\alpha = 0.5$ is equivalent to having a trial wavefunction that is exact for this system, according to (**??**). Thus, a few steps should suffice to obtain the exact energy.

N	d	⟨E⟩	σ	CPU time
1	1	0.5	0	5.53e-3
1	2	1	0	5.66e-3
1	3	1.5	0	5.70e-3
10	1	5	0	7.20e-3
10	2	10	0	7.36e-3
10	3	15	0	1.00-3
100	1	50	0	0.0215
100	2	100	0	0.223
100	3	150	0	0.274
500	1	250	0	0.0843
500	2	500	0	0.110
500	3	750	0	0.131

Table 3: Analytical kinetic energy

N	d	$\langle E_L \rangle$	σ	CPU time
1	1	0.5	9.87e-8	6.96e-3
1	2	1	1.873-7	0.0151
1	3	1.5	1.88e-7	0.0157
10	1	5	4.42e-7	0.0314
10	2	10	6.78e-7	0.0598
10	3	15	1.11e-6	0.0955
100	1	50	2.78e-6	0.743
100	2	100	6.18e-6	1.78
100	3	150	8.95e-6	3.16
500	1	200	1.32e-5	15.1
500	2	500	2.52e-5	41.5
500	3	750	6.65e-5	74.5

Table 4: Numerical kinetic energy

As expected, the CPU time increases with N and d (number of dimensions). Calcuating the kinetic energy analytically speeds up the simulation a great deal and the CPU time difference increases exponentially with N.

The analytical method reproduces the exact energies with zero error. The numerical method results in a small deviation due to the inherent error of the differentiation algorithm that increases with N. This is not a statistical error, thus we do not perform an error analysis with the blocking method for system 1.

4.2 System 1 - importance sampling

With the following parameters,

Listing 4: Parameters importance sampling Metropolis system 1

```
    int numberOfSteps       = (int) 1e4;
    double omega            = 1.0;          // oscillator frequency
    double alpha            = 0.5;          // variational parameter 1
    double stepLength       = 1.5;          // metropolis step length
    double equilibration    = 0.1;          // amount of the total steps
    double timeStep         = 0.001;        // importance sampling
```

we obtain

N	d	$\langle E_L \rangle$	σ	Time
1	1	0.5	0	0.0183
1	2	1	0	0.0338
1	3	1.5	0	0.0501
10	1	5	0	0.0479
10	2	10	0	0.0488
10	3	15	0	0.0812
100	1	50	0	0.0605
100	2	100	0	0.1212
100	3	150	0	0.218
500	1	200	0	0.171
500	2	500	0	0.451
500	3	750	0	0.941

Table 5: Analytical kinetic energy

N	D	$\langle E_L \rangle$	σ	CPU time
1	1	0.5	9.41e-8	0.0170
1	2	1	1.48e-7	0.0352
1	3	1.5	1.58e-7	0.0618
10	1	5	3.42e-7	0.0413
10	2	10	3.50e-7	0.104
10	3	15	9.23e-7	0.163
100	1	50	3.02e-6	0.737
100	2	100	5.72e-6	1.889
100	3	150	6.88e-6	3.319
500	1	200	1.01e-5	15.509
500	2	500	4.94e-5	41.047
500	3	750	2.42e-5	75.584

Table 6: Numerical kinetic energy

The analytic method again reproduces the exact energies, while the numerical method has a small error. However,there are two differences between the results produced by the brute force algorithm and Metropolis with importance sampling. Firstly, the CPU time increases because we have to calculate the drift vector and Green's function for each Monte Carlo cycle. Secondly, the error (for numerical kinetic energy) *decreases* for the same number of Monte Carlo cycles as we should expect considering the more efficient way of sampling positions compared to brute force sampling.

We also want to investigate how the results depend on the chosen time step. For $N = 100$ and $d = 3$ we get,

Time step	Acceptance rate	σ
0.001	0.871	0
0.005	0.831	0
0.01	0.801	0

Table 7: Analytical kinetic energy

Time step	Acceptance rate	σ
0.001	0.871	6.88e-6
0.005	0.831	7.39e-6
0.01	0.801	9.91e-6

Table 8: Numerical kinetic energy

We see that the acceptance rate decreases for increasing time step. This is because the probability of accepting a new state is larger for small Δt; the ratio (27) yields a value near 1 when the wave function at the proposed new position is very close to the old one.

The error increases slightly with the time step, caused by the fact that the decreasing acceptance rate leads to more inaccurate sampling.

4.3 System 2

Before running the simulation for system 2, we must optimize the variational parameter α using the steepest descent method. We run $1e5$ number of Metropolis steps for each iteration. The set of parameters

Listing 5: Parameters system 2

```
double omega            = 1.0;        // oscillator frequency
double beta             = 2.82843;    // variational parameter 2
double stepLength       = 1.5;        // metropolis step length
double equilibration    = 0.1;        // amount of the total steps
double a                = 0.0043;     // hard sphere radius
double gamma            = 2.82843;    // trap potential strength z-direction

double initialAlpha = 0.7;
double stepLengthOptimize = 0.01;
```

yields the the optimal variational parameter $\alpha = 0.500605$. We now run simulations for $N = 10$, $N = 50$ and $N = 100$ bosons with this optimal α, computing the Laplacian analytically. The number of Monte Carlo steps is $1e6$ for $N = 10$ and $1e5$ for $N = 50$ and $N = 100$.

N	$\langle E_L \rangle$	σ	CPU time
10	24.143	7.14e-3	9.027
50	120.51	0.121	87.86
100	239.5	1.114	687.66

These energies are quite close to those in **??** from [1]. Reproducing these excactly is not expected, as our set of parameters are not equal to those used in [1]. The energy values are not that different from those of system 1, especially for $N = 10$ and $N = 50$. This tells us that the correlation effects are not very dominant, due to the dilute nature of our system. A denser system would have had larger correlations.

4.3.1 Error analysis

As discussed above, our estimate of error for system 2 is too low because the correlation effects are not included in the error estimate. A proper error analysis is done using the blocking method. First, we plot the standard deviation σ as a function of block size n_b,

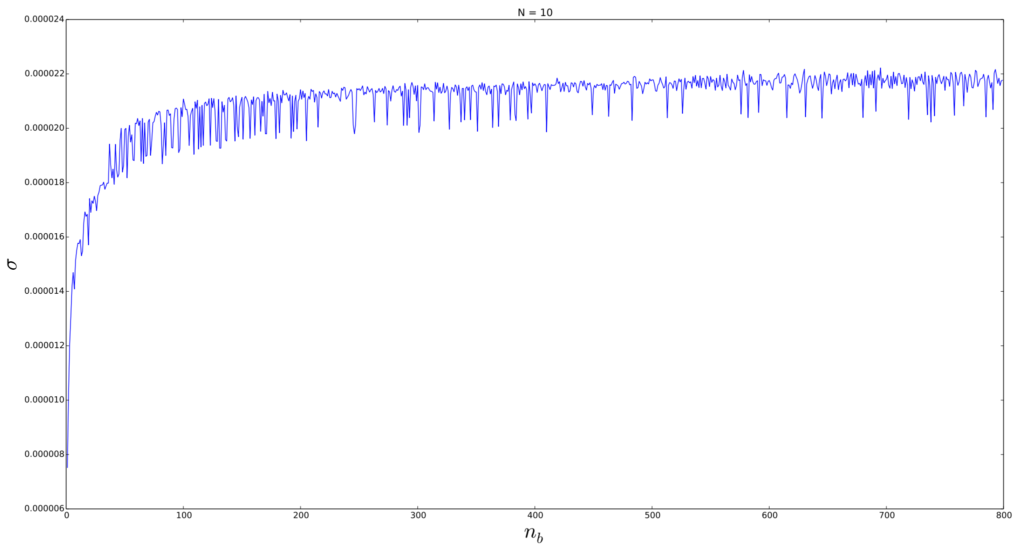

Figure 1: N = 10

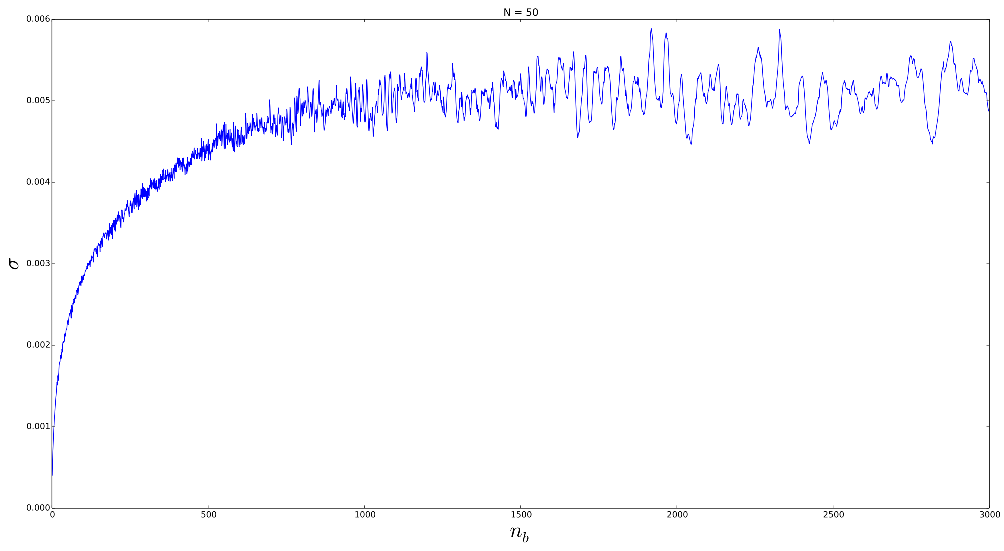

Figure 2: N = 50

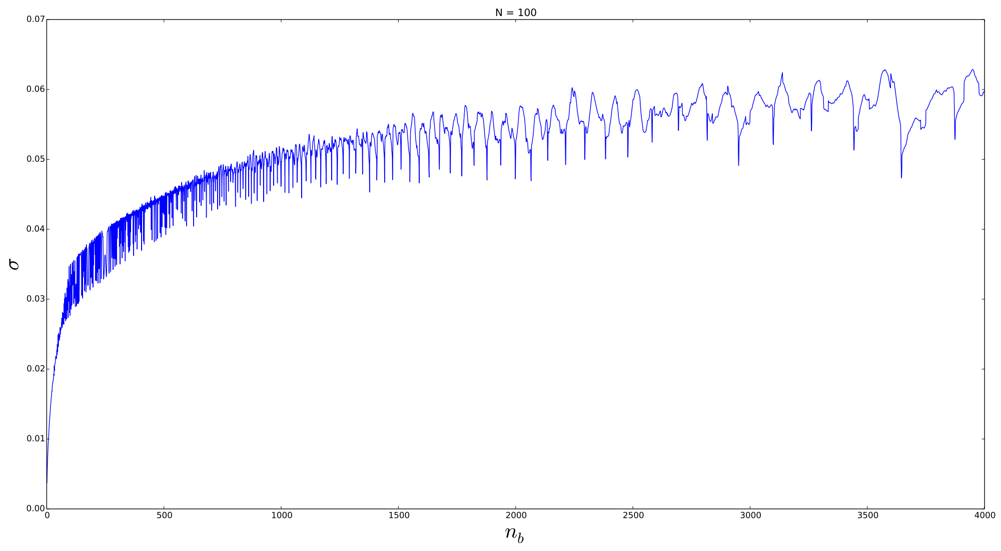

Figure 3: N = 100

We can now estimate τ by looking at which block size n_b the plateau is reached and multiply this value with the above Monte Carlo step size,

- N=10: $n_b \approx 100 \Rightarrow \tau = 100 \cdot 1.5 = 150$
- N=50: $n_b \approx 1000 \Rightarrow \tau = 1000 \cdot 1.5 = 1500$
- N=100: $n_b \approx 1500 \Rightarrow \tau = 1500 \cdot 1.5 = 2250$

The true sample errors are then, according to (52),

- N=10: $\sigma_{true} = 0.032$
- N=50: $\sigma_{true} = 0.77$
- N=100: $\sigma_{true} = 6.1$

We observe that the true sample errors are almost one order of magnitude larger than the uncorrelated error estimates.

4.4 One-body density

The one-body density is defined as

$$\rho(\mathbf{r}) = \int d\mathbf{r}_2 \dots d\mathbf{r}_N |\Psi(\mathbf{r}, \mathbf{r}_2, \dots, \mathbf{r}_N)|^2 \tag{53}$$

This quantity can be visuzalized by making a histogram of the radial distance of the bosons. The following paramters,

Listing 6: Parameters one-body density

```
    int numberOfDimensions   = 3;
    int numberOfParticles    = 30;
    int numberOfSteps        = (int) 1e6;
    double omega             = 1.0;        // oscillator frequency
    double alpha             = 0.5;        // variational parameter 1
    double beta              = 2.82843;    // variational parameter 2
    double stepLength        = 1.5;        // metropolis step length
    double equilibration     = 0.1;        // amount of the total steps used
    double a                 = 0.0043;     // hard sphere radius
    double gamma             = 2.82843;    // trap potential strength z-direction
```

results in

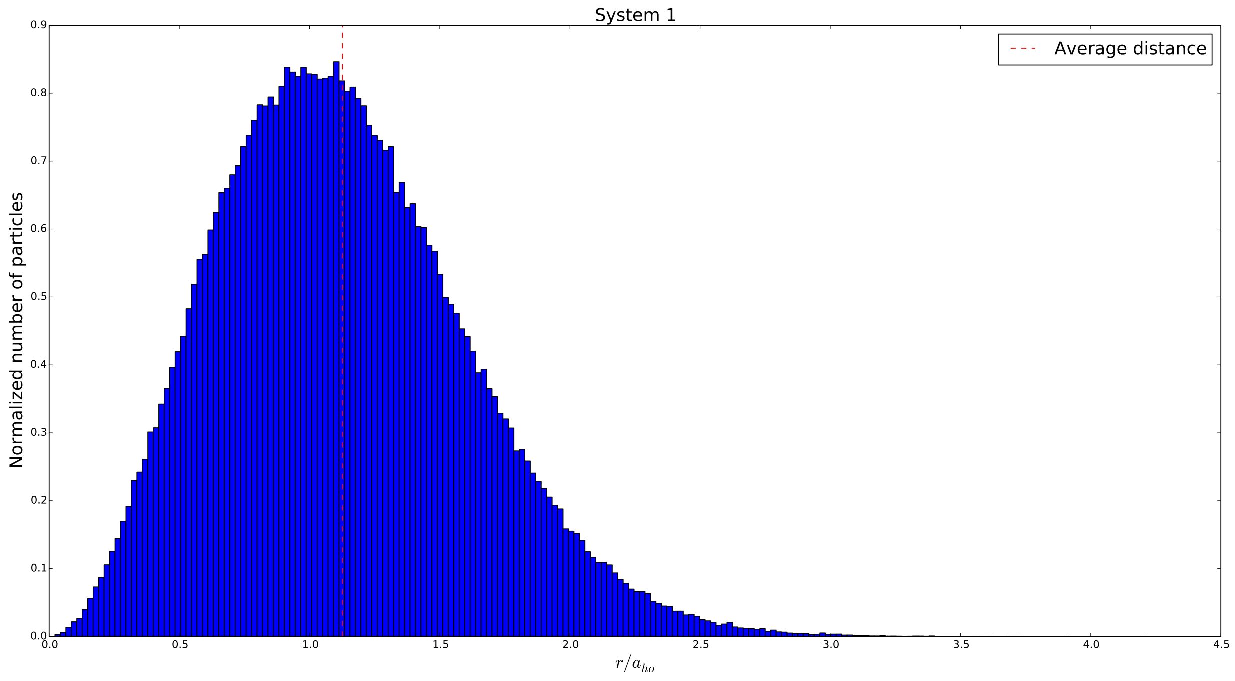

Figure 4: Radial distribution of particles - system 1

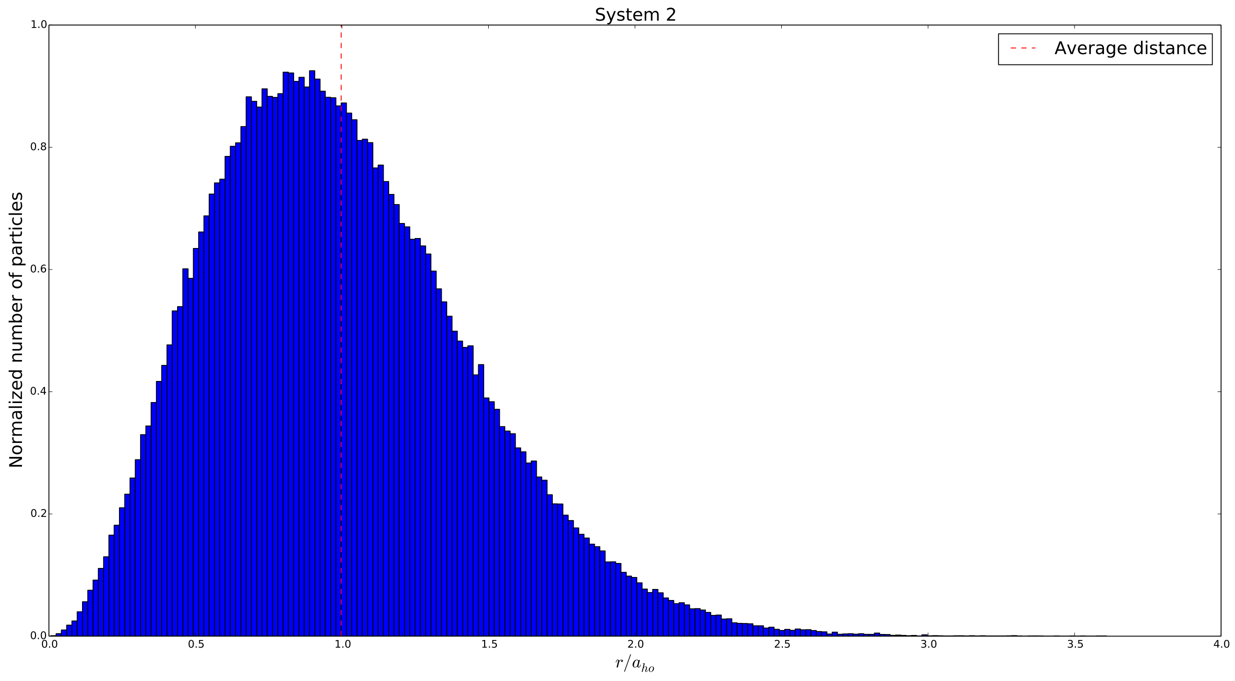

Figure 5: Radial distribution of particles - system 2

The distribution for system 1 is shifted towards the right compared to system 2, thus the average radial distance is larger.

The distribution of particles in the yz-plane is included for visualization purposes,

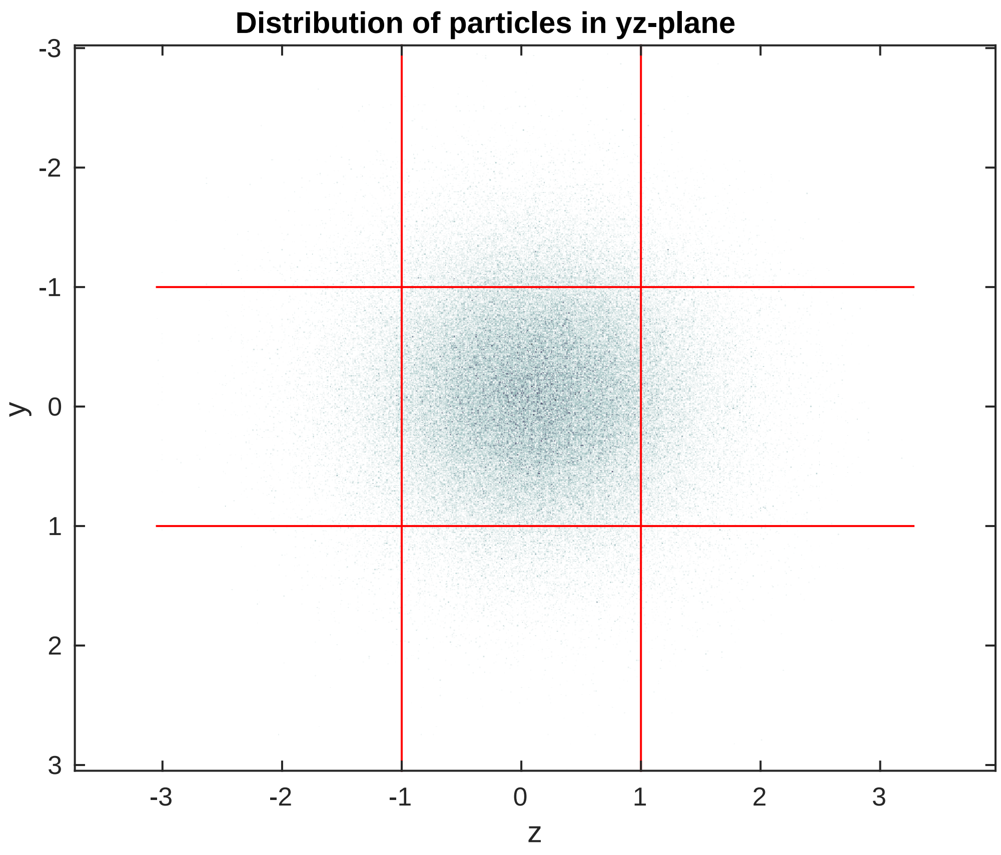

Figure 6: System 1

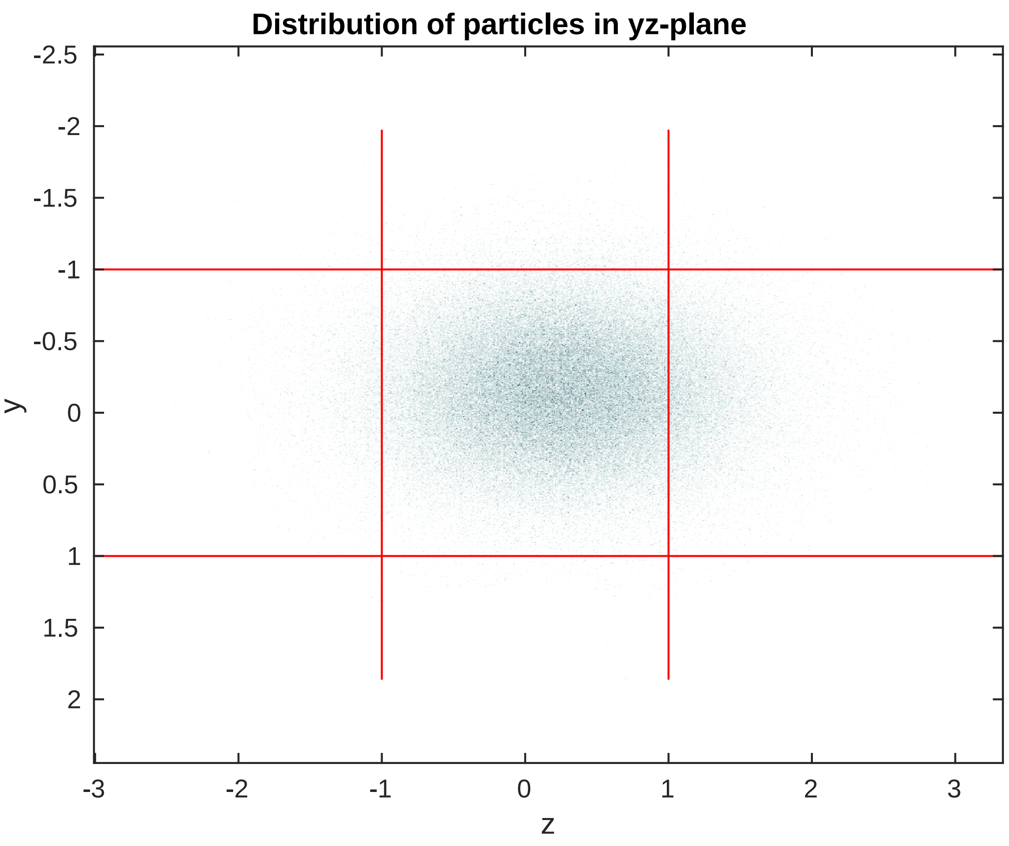

Figure 7: System 2

We observe also here that system 1 is more spread out than system 2. System 1 has a spherical distribution centered around origo. System 2 is however shifted towards positive y- and z-direction and has a slightly elliptical shape with an elongation in the z-direction. The latter is caused by the elliptical harmonic oscillator potential (??) with $\omega_z/\omega_{ho} = 2.82834$.

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 α 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 expectiation value of the so-called local energy

$$E_L(\mathbf{R}) = \frac{1}{\Psi_T(\mathbf{R})} H \Psi_T(\mathbf{R}), \tag{54}$$

We can find closed-form expressions for the local energy with our specific Hamiltonian H and trial wavefunction Ψ_T. Computing the local energy involves a second derivative of Ψ_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 calcuations we will use natural units, thus $\hbar = m = 1$. For one particle in one dimension we have

$$\Psi_T(x) = e^{-\alpha x^2} \tag{55}$$

and

$$H = -\frac{1}{2}\frac{\partial^2}{\partial x^2} + \frac{1}{2}\omega x^2 \tag{56}$$

The second derivate of the trial wave function is

$$\frac{\partial^2 \Psi_T}{\partial x^2} = 2\alpha e^{-\alpha x^2}(2\alpha x^2 - 1) \tag{57}$$

so that

$$E_L = \frac{1}{\Psi_T} H\Psi_T = \alpha(1 - 2\alpha x^2) + \frac{1}{2}\omega^2 x^2 \tag{58}$$

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

$$\frac{1}{\Psi_T} \nabla^2 \Psi_T = 2\alpha(2\alpha x^2 - 1) + 2\alpha(2\alpha y^2 - 1) + 2\alpha(2\alpha z^2 - 1) \tag{59}$$

$$= 2\alpha(2\alpha r^2 - 3) \tag{60}$$

thus the local energy is

$$E_L = \frac{1}{\Psi_T}\left(-\frac{1}{2} \nabla^2 \Psi_T + V_{ext}\right) = \alpha(3 - 2\alpha r^2) + \frac{1}{2}\omega^2 r^2 \tag{61}$$

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

$$\Psi_T(\mathbf{R}) = \prod_i e^{-\alpha r_i^2} \tag{62}$$

$$H = \sum_i^N \left(-\frac{1}{2}\nabla_i^2 + \frac{1}{2}\omega^2 r_i^2\right) \tag{63}$$

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

$$\nabla_k^2 \prod_i e^{-\alpha r_i^2} = 2\alpha(2\alpha x_k^2 - 1)\prod_i e^{-\alpha r_i^2} \tag{64}$$

and when we divide with Ψ_T to obtain the the local energy we end up with

$$E_L = \sum_i^N \left(\alpha(3 - 2\alpha r_i^2) + \frac{1}{2}\omega^2 r_i^2\right) \tag{65}$$

For one dimension the expression is

$$E_L = \sum_i^N \left(\alpha(1 - 2\alpha x_i^2) + \frac{1}{2}\omega^2 x_i^2\right) \tag{66}$$

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

$$F = \frac{2\nabla\Psi_T}{\Psi_T}. \tag{67}$$

The gradient of Ψ_T is

$$\nabla\Psi_T = (-2\alpha x, -2\alpha y, -2\alpha z)e^{-\alpha r^2} \tag{68}$$

$$= -2\alpha e^{-\alpha r^2}\mathbf{r} \tag{69}$$

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

$$F = -4\alpha\mathbf{r} \tag{70}$$

6.2 Analytic energy for two-body quantum dot

$$\Psi_T(\mathbf{r_1}, \mathbf{r_2}) = \exp(-\alpha\omega(r_1^2 + r_2^2)/2)\exp\left(\frac{ar_{12}}{1 + \beta r_{12}}\right) \tag{71}$$

$$= K_1 K_2 \tag{72}$$

Laplacian of this wave function:

$$\nabla^2\Psi_T(\mathbf{r_1}, \mathbf{r_2}) = \frac{\partial\Psi_T}{\partial x_1^2} + \frac{\partial\Psi_T}{\partial y_1^2} + \frac{\partial\Psi_T}{\partial x_2^2} + \frac{\partial\Psi_T}{\partial y_2^2} \tag{73}$$

Using the following

$$, \frac{\partial r_1}{\partial x_1} = x_1/r_1 \tag{74}$$

and

$$\frac{\partial r_{12}}{\partial x_1} = (x_1 - x_2)/r_1 \tag{75}$$

Gradient of first term:

$$\frac{\partial K_1}{\partial x_1} = -\alpha\omega x_1 K_1 \tag{76}$$

so that

$$\nabla K_1 = -\alpha\omega K_1(x_1, y_1, x_2, y_2) \tag{77}$$

Gradient of second term:

$$\frac{\partial K_2}{\partial x_1} = K_2\frac{a(x_1 - x_2)}{r_{12}(1 + \beta r_{12})^2} \tag{78}$$

so that

$$\nabla K_2 = K_2\frac{a}{r_{12}(1 + \beta r_{12})^2}(x_1 - x_2, y_1 - y_2, x_2 - x_1, y_2 - y_2) \tag{79}$$

Laplacian of first term:

$$\frac{\partial^2 K_1}{\partial x_1^2} = K_1(\alpha^2\omega^2 x_1^2 - \alpha\omega) \tag{80}$$

so that

$$\nabla^2 K_1 = K_1(\alpha^2\omega^2(r_1^2 + r_2^2) - 4\alpha\omega) \tag{81}$$

Laplacian of second term:

$$\frac{\partial^2 K_2}{\partial x_1^2} = K_2\left[\frac{a^2(x_1 - x_2)^2}{r_{12}^2(1+\beta r_{12})^4} + \frac{ar_{12}(1+\beta r_{12})^2}{r_{12}^2(1+\beta r_{12})^4}\right. \tag{82}$$

$$\left. - \frac{a(x_1 - x_2)[(x_1 - x_2)/r_{12}(1+\beta r_{12})^2 + 2r_{12}(1+\beta r_{12})\beta(x_1 - x_2)/r_{12}]}{r_{12}^2(1+\beta r_{12})^4}\right] \tag{83}$$

$$= K_2\left[\frac{a^2(x_1 - x_2)^2}{r_{12}^2(1+\beta r_{12})^4} + \frac{a}{r_{12}(1+\beta r_{12})^2}\right. \tag{84}$$

$$\left. - \frac{a(x_1 - x_2)^2}{r_{12}^3(1+\beta r_{12})^2} - \frac{2a\beta(x_1 - x_2)^2}{r_{12}^2(1+\beta r_{12})^3}\right] \tag{85}$$

so that

$$\nabla^2 K_2 = K_2\left[\frac{2a^2}{(1+\beta r_{12})^4} + \frac{4a}{r_{12}(1+\beta r_{12})^2} - \frac{2a}{r_{12}(1+\beta r_{12})^2} - \frac{2a\beta}{(1+\beta r_{12})^3}\right] \tag{86}$$

$$= K_2\frac{2a}{(1+\beta r_{12})^2}\left[\frac{a}{(1+\beta r_{12})^2} + \frac{1}{r_{12}} - \frac{2\beta}{1+\beta r_{12}}\right] \tag{87}$$

Have that

$$\nabla^2\Psi_T = \nabla^2 K_1 K_2 + 2\nabla K_1 \nabla K_2 + K_1 \nabla^2 K_2 \tag{88}$$

and

$$\nabla K_1 \nabla K_2 = -K_1 K_2 \frac{a\alpha\omega}{r_{12}(1+\beta r_{12})^2}\Big[x_1(x_1 - x_2) + y_1(y_1 - y_2) - x_2(x_1 - x_2) - y_2(y_1 - y_2)\Big] \tag{89}$$

$$= -K_1 K_2 \frac{a\alpha\omega}{r_{12}(1+\beta r_{12})^2}\Big[(x_1 - x_2)(x_1 - x_2) + (y_1 - y_2)(y_1 - y_2)\Big] \tag{90}$$

$$= -K_1 K_2 \frac{a\alpha\omega r_{12}}{(1+\beta r_{12})^2} \tag{91}$$

The analytic expression for the local energy is thus

$$E_L = \frac{\nabla^2\Psi_T}{\Psi_T} = 2\alpha^2\omega^2(r_1^2 + r_2^2) - 4\alpha\omega - \frac{2a\alpha\omega r_{12}}{(1+\beta r_{12})^2} \tag{92}$$

$$+ \frac{2a}{(1+\beta r_{12})^2}\left[\frac{a}{(1+\beta r_{12})^2} + \frac{1}{r_{12}} - \frac{2\beta}{1+\beta r_{12}}\right] \tag{93}$$

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)