COMPUTATIONAL PHYSICS 2
Project 1
VMC for calculating the ground state energy of a trapped hard sphere Bose gas

Martin Krokan Hovden

March 31, 2020

Abstract

In this article a Quantum Monte Carlo method known as Variational Monte Carlo (VMC) is used for simulating a trapped hard sphere Bose gas with a specific wave function. The quantity of interest is the ground state energy of the system. Two VMC methods are used, the brute-force Metropolis algorithm and the importance sampling Metropolis algorithm. Two different systems are used to model the trapped Bose gas. The first is a system of particles in a spherical harmonic oscillator potential with a non-interacting Gaussian wave function. The results from running the algorithms are compared to the exact values for the system. It shows that both methods provide good estimates of the ground state energy, but the convergence rate and acceptance ratio is better when using importance sampling. The variance of the estimated ground state energy is also lower using importance sampling. After that, particles in a 3 dimensional elliptical potential trap with interaction between the particles is studied using the brute-force Metropolis algorithm. The Jastrow factor was added to the Gaussian wave function to account for the interactions between the particles. The results are benchmarked against the results in the article by M.L. Markova et. al[7]. The optimal alpha-values was found using the gradient descent method. For the non-interacting system, gradient descent found the exact alpha-value corresponding to the ground state energy, which is $\alpha = 0.5$. The one-body density of the system is also found. This makes visualization of the system easier.

Contents

1 INTRODUCTION

Confined Bose systems have been a major field of interest and research over the last years. This is partly due to demonstrations of Bose-Einstein condensation in gases of alkali atoms confined in magnetic traps[7]. In this project I will use the popular Variational Monte Carlo (VMC) method for calculating the ground state energy of a trapped, hard sphere Bose gas. The VMC method is a powerful method that can be used to simulate various systems in quantum mechanics. The main difference between VMC and other variational methods is the use of Monte Carlo methods for calculating the high dimensional integrals involved in the calculations. This gives us the opportunity to simulate systems that would be hard to simulate using other approaches.

The system simulated is described by a trial wave function and a Hamiltonian. Two different systems are used for modelling the Bose gas. The first system will use particles in a spherical harmonic oscillator with a Gaussian trial wave function. The system is modelled without interactions between the particles. Markov Chain Monte Carlo methods are used to simulate the system. In particular, the Brute-Force Metropolis algorithm and the Importance Sampling Metropolis algorithm are used. For this system, the exact solution is known, and the algorithm can be tested with those as a reference point. When the methods works properly on the simple non-interacting system, the system is expanded so that the interactions between the particles are accounted for. This hopefully results in a better approximation of the actual Bose gas we want to model. This is done by adding the Jastrow factor to the wave function. The exact analytical solution of this system is not possible to derive. However, the results from an article where the same system is studied is used as benchmarks for the calculation. [7]. The gradient descent method is also implemented for finding the alpha that corresponds to the ground state energy of the system. Since the simulations are done in a very high-dimensional space, the one-body densities of the particles are plotted. This is to get a better understanding of how the system behaves with and without the Jastrow factor.

The report is split up into parts. First, a description of the methods used is presented. The focus is mainly on the applications of the methods to this specific system. After that, the systems to be simulated are described, and the various expressions involved in the calculations are shown. For the non-interacting system, the exact solution is derived. This is beneficial when implementing and testing the algorithms. A short description on how the GitHub repository is structured is included. After that the results from running the simulations are presented and discussed. In addition, there is an added appendix where various derivations of expressions can be found.

2 THEORY AND METHODS

The theory and methods part is mainly based on lecture notes written by Morten Hjorth-Jensen for the course FYS4411: Computational Physics 2. For further details and derivations of the methods, see [1].

2.1 Variational Monte Carlo Methods

Variational Monte Carlo (VMC) methods are methods that can be used for calculating the ground state energy of a system in quantum mechanics. The main idea is to minimize the total energy of a system characterized by a wavefuntion depending on a variational parameter α, and a Hamiltonian. More formally, the optimal α can be found by minimizing

$$E[H] = \langle H \rangle = \frac{\int d\boldsymbol{R}\psi_T^*(\boldsymbol{R},\alpha)H(\boldsymbol{R})\psi_T(\boldsymbol{R},\alpha)}{\int d\boldsymbol{R}\psi_T^*(\boldsymbol{R},\alpha)\psi_T(\boldsymbol{R},\alpha)} \tag{1}$$

with respect to α where ψ_T is the trial wave function and H is the Hamiltonian of the system. The variational principle states that the expectation value of H is an upper bound for the ground state energy,

$$E_0 \leq \langle H \rangle.[1] \tag{2}$$

This can be shown that first noting that the trial wave function can be rewritten as a linear combination

$$\psi_T(\boldsymbol{r}) = \sum_i a_i \psi_i(\boldsymbol{R}). \tag{3}$$

where the coefficients a_i are assumed to be normalized. Then

$$\frac{\sum_{nm} a_m^* a_n \int d\boldsymbol{R}\psi_m^*(\boldsymbol{R})H(\boldsymbol{R})\boldsymbol{R}\psi_n(\boldsymbol{R})}{\sum_{nm} a_m^* a_n \int d\boldsymbol{R}\psi_m^*(\boldsymbol{R})\psi_n(\boldsymbol{R})} = \frac{\sum_n a_n^2 E_n}{\sum_n a_n^2} \geq E_0.[1] \tag{4}$$

The only difference between Variational Monte Carlo method and the more traditional variational methods is that the integrals involved are estimated using Monte Carlo methods. When the number of dimensions and particles increase and complicated trial wave functions are used, the expression for the expectation value of the Hamiltonian becomes very complicated and the integral becomes high dimensional when the number of dimensions and particles increase. Analytical methods for integration then becomes hard to use. In addition, quadrature methods scale poorly when we work in a high dimensional space [3]. Therefore, it is typically needed to use Monte Carlo methods for calculating $E[H]$. This is often done with the help of the so-called local energy.

2.2 Local Energy

The local energy of the system is defined by

$$E_L = \frac{1}{\psi_T} H\psi_T.[1] \tag{5}$$

By also noting that the probability distribution function can be written as

$$P(\boldsymbol{R}) = \frac{|\psi_T(\boldsymbol{R})|^2}{\int |\psi_T(\boldsymbol{R})|^2 d\boldsymbol{R}}, \tag{6}$$

where the denominator is the normalization factor, equation 1 can be rewritten to a more convenient form

$$E[H] = \int P(\boldsymbol{R},\alpha)E_L(\boldsymbol{R},\alpha)d\boldsymbol{R} \tag{7}$$

The local energy will be used to calculate the expectation value of the Hamiltonian. Calculating this integral analytically becomes impossible for most wave functions and using quadrature methods scales badly to high dimensions and for complicated systems [3]. Using Monte Carlo integration lets us solve more complicated problems quite easily. The main problem is that the computational time can be high to get good estimates. In addition, finding good analytical expression for the quantities involved in the local energy can be hard.

2.3 Monte Carlo Integration

Monte Carlo integration is a non-deterministic numerical method for solving definite integrals [3]. The main idea is to use a random number generator to draw random numbers where the function value is evaluated. An approximation of the integral value is then found by finding the mean of all the random function values. In general, the more random samples we use, the better the approximation becomes. There also exist more advanced methods that converges to the exact solution more rapidly by drawing the random numbers from a more appropriate distribution. One of those methods are presented later, and is called importance sampling.

More formally the expected value of a function can be approximated by

$$E[f(\boldsymbol{x})] = \int f(x)p(x)dx \approx \frac{1}{N}\sum_{i=1}^{N} f(X_i) = \hat{\mu}, \tag{8}$$

where X_i is drawn from the probability distribution p [3]. By adjusting N it is possible to increase the accuracy of the Monte Carlo integration, and it can be shown that

$$\lim_{n\to\infty} \frac{1}{N}\sum_{i=1}^{N} f(x_i) = E[f(\boldsymbol{x})] \tag{9}$$

by the law of large numbers [3].

When working with random numbers and Monte Carlo methods it is interesting to see how good the estimate of a parameter is. This can be done by calculating the variance of the estimator. The sampling variance of $\hat{\mu}$ can then be calculated using

$$\hat{\text{Var}}(\hat{\mu}) = \frac{1}{N-1}\sum_{i=1}^{N}(f(X_i) - \hat{\mu})^2. \tag{10}$$

where we assume that the samples are uncorrelated [3]. Using equations 8 and 10 and adapting them to the quantum system, the energy estimate and the variance of the energy estimate can be calculated using

$$E[H(\alpha)] = \frac{1}{N}\sum_{i=1}^{N} E_L(\boldsymbol{r_i}) \tag{11}$$

and

$$\text{Var}(E[H(\alpha)]) = \frac{1}{N-1}\sum_{i=1}^{N}(E_L(\boldsymbol{r_i}) - E[H(\alpha)]^2) = \frac{1}{N-1}\text{Var}(E_L) \tag{12}$$

where $\boldsymbol{r_i}$ is the coordinates of the i'th particle and N is the number of samples. The problem is that the samples of the local energy is not uncorrelated. This is because the current position is dependent on the previous position. A better method for estimating the variance is called blocking and will be presented in a later section.

2.4 Summary of Variational Monte Carlo Methods

In VMC the goal is to adjust the variational parameter until the minimum energy of the system is reached [1]. VMC methods can be summarized as follows:

- Contruct a trial wavefuntion ψ_T that is dependent on the variational parameter α (1 parameter in this case) and the position of the particles $\boldsymbol{R} = (\boldsymbol{R_1}, ..., \boldsymbol{R_N})$.
- Calculate the expected value of the hamiltionian, $\langle H \rangle$.
- Then use a numerical optimization algorithm, like the gradient descent algorithm, to adjust the parameter α until a minimum is reached.

Two methods for calculating the integral will be discussed. Since the shape of the wave function varies over the domain, using a clever method called importance sampling can often lead to better results compared to a brute force method.

2.5 Markov Chain Monte Carlo

The main problem with the Monte Carlo integration method is that the probability density function can be hard to sample from. Also, the normalization factor can be very tedious to calculate. One way to draw positions from such a distribution is to use the Markov Chain Monte Carlo Methods [3]. Two variations of Markov Chain Monte Carlo methods will be used; the brute-force Metropolis algorithm and the importance sampling Metropolis algorithm. The brute-force method does not take into consideration the shape of the distribution, and the samples are drawn from a uniform distribution. The importance sampling method on the other hand accounts for the shape of the function, and will typically lead to better results. Importance sampling is a method used to reduce the variance of the estimates. For a theoretical background on the methods, see [1], [3] or my articles for project 3 and 4 from FYS4150: Computational Physics 1. The latter can be found at the GitHub-address.

2.5.1 Metropolis Algorithm

The Metropolis algorithm is a Markov chain Monte Carlo method. It can be used to sample from a normalized probability distribution with the help of a stochastic process. It is often used to obtain samples from complicated distributions that normally would be hard to sample from [3]. In this article it will be used to sample particle configurations from the trial wave function. This is something that would be hard doing without a Markov chain Monte Carlo method.

The main idea behind using Markov chain Monte Carlo methods for sampling is to construct a Markov Chain where the sequence of states converge towards the actual distribution we want to sample from [3]. We can then use the samples to calculate different properties of the system. We typically use a so-called burn-in period before we start recording the properties. This is so that the system is close to the limiting sequence when the samples are recorded [3]. We will then typically get more accurate results. Markov Chain Monte Carlo method works especially well in high-dimensional problems where more traditional methods will suffer from the curse of dimensionality [3].

The Brute-force Metropolis algorithm generates the samples as follows [1]:

- Initialize the system with number of Monte Carlo cycles, choose an initial system configuration $\boldsymbol{R}$ and variational parameter α, and then calculate the initial value $|\psi_T^\alpha(\boldsymbol{R})|^2$. $\psi_T^\alpha(\boldsymbol{R})$ is the trial wave function dependent on the variational parameter α and the position of the particles. Initialize the energy and other properties of interest.
- For each metropolis step:
  - Sample a new trial position, $\boldsymbol{R_p} = \boldsymbol{R} + r \cdot step$ where r is random vector from a uniform distribution.
  - Calculate the acceptance ratio $w = \frac{P(\boldsymbol{R_p})}{P(\boldsymbol{R})}$.
  - Sample a new random number u from an uniform distribution, $u \in [0, 1]$.
    - If $w \geq u$, the new trial position is accepted.
    - Else we reject the new trial position and continue the iterations with the old position.
  - Sample the quantities of interest. For this case the local energy.
- When the sampling is done, the statistics of interest can be calculated.

We note that when the new position has a higher probability we always accept the new step. By doing so we will in theory stay in the areas with a high probability value. However, we can sometimes accept a less favorable step. To calculate the acceptance ratio, it is not necessary to calculate the complicated normalization factor of the probability, since they cancel out in the division.

2.5.2 Importance Sampling

The main problem with the Brute-force metropolis algorithm is that the new steps are proposed according to a uniform distribution and does not account for the shape of the wave function. By introducing importance sampling the algorithm proposes more physically relevant steps, which means that the Markov Chain typically will converge to the equilibrium state faster. This can reduce the number of iterations needed for a certain accuracy drastically [3]. The idea is to choose samples that are more relevant to the system, i.e find positions where the particles are more likely to be according to the trial wave function. The main difference from the Metropolis algorithm is how we choose the new step, and the expression for the acceptance/rejection-ratio.

The new method for sampling the proposed position comes from the Fokker-Planck equation, the Langevin equation and the idea about moving in the direction of the gradient of the wave function [1]. The derivations are taken directly from the lecture notes by M. Hjorth-Jensen[1].

The Fokker-Planck equation is given by

$$\frac{\partial P}{\partial t} = D\frac{\partial}{\partial x}\left(\frac{\partial}{\partial x} - F\right)P(x,t), \tag{13}$$

where F is the drift force and D is the diffusion constant [1]. The Fokker-Planck equation describes the development over time of a probability density function P(x,t). By setting the left side equal to zero we can show that

$$\frac{\partial^2 P}{\partial \boldsymbol{x}_i^2} = P\frac{\partial}{\partial \boldsymbol{x_i}}\boldsymbol{F_i}\frac{\partial}{\partial \boldsymbol{x_i}}P \tag{14}$$

for all i[1].

By letting the drift force be on the form $g(\boldsymbol{x})\frac{\partial P}{\partial \boldsymbol{x}}$, it can be shown that the drift force can be defined as

$$\boldsymbol{F} = 2\frac{1}{\psi_T}\nabla\psi_T[1]. \tag{15}$$

It is responsible for pushing the random walk into configurations where the particles actually have a higher probability of being. By using this drift-force the particles are moved towards a more desirable position [1]. This is better compared to the uniform sampling in the brute-force metropolis algorithm since the system will tend to stay in more physically relevant positions.

The new method for finding the next system configuration is found using the Langevin equation. The Langevin equation is given by

$$\frac{\partial x(t)}{\partial t} = DF(x(t)) + \eta \tag{16}$$

where η is a random variable. Solving this equation using Euler's method gives us the new update formula

$$y = x + DF(x)\Delta t + \epsilon\sqrt{\Delta t} \tag{17}$$

where Δt is the time-step, ϵ is a random variable and x is the current position [1]. D is set to 0.5. When simulating the system Δt have to chosen, typically in the intercal [0.001, 0.01] when calculating ground state energies [1]. This have to be tested using trial and error.

Solving the Fokker-Planck equation gives Greens function

$$G(x, y, \Delta t) = \frac{1}{(4\pi D\Delta t)^{\frac{3N}{2}}}\exp(-(y - x - D\Delta t F(x))^2/4D\Delta t). \tag{18}$$

[1] which can be used in the probabiliby ratio. As we will be working with multidimensional arrays, the squared in the exponential can be interpreted as the dot-product with itself. The interesting quantity is the ratio between the squared wave function of the new and the old position, but now the greens function ratio is also added to the equation. This gives the new transition probability ratio

$$q(y, x) = \frac{G(x, y, \Delta t)|\psi_T(y)|^2}{G(y, x, \Delta t)|\psi_T(x)|^2}.[1] \tag{19}$$

By changing the update step for the position and the acceptance step in the brute-force Metropolis algorithm to the new ones, the importance sampling Metropolis algorithm samples as follows

- Initialize the system with number of Monte Carlo cycles, choose an initial system configuration $\boldsymbol{R}$ and variational parameter α, and then calculate the initial value $|\psi_T^{\alpha}(\boldsymbol{R})|^2$. $\psi_T^{\alpha}(\boldsymbol{R})$ is the trial wave function dependent on the variational parameter α and the position of the particles.
- Initialize energies and variance, and for number of metropolis steps:
  - Sample a new trial position, $\boldsymbol{R_p} = \boldsymbol{R} + D\boldsymbol{F(x)}\Delta t + \boldsymbol{\epsilon}\sqrt{\Delta t}$.
  - Calculate q($\boldsymbol{R_p}$, $\boldsymbol{R}$) as described above.
  - Sample a new random number u from an uniform distribution, $u \in [0,1]$.
    * If $w \geq u$, the new trial position is accepted.
    * Else we reject the new trial position and continue the iterations with the old position.
  - Sample the quantities of interest. In our case the local energy.
- Then calculate the final total energy of the system and other desired values.

2.6 Description of the System

In this article we study Variational Monte Carlo methods for calculating the ground state energy of a hard sphere Bode gas in various potential traps. The system is studied for various numbers of particles and dimensions for two different trial wave functions.

The Hamiltonian used is given

$$H = \sum_{i}^{N} \left(\frac{\hbar^2}{2m} \nabla_i^2 + V_{\text{ext}}(\boldsymbol{r_i}) \right) + \sum_{i<j}^{N} V_{\text{int}}(\boldsymbol{r_i}, \boldsymbol{r_j}), \tag{20}$$

where

$$V_{\text{ext}} = \begin{cases} \frac{1}{2} m \omega_{ho}^2 r^2 & (S) \\ \frac{1}{2} m [\omega_{ho}^2 (x^2 + y^2) + \omega_z^2 z^2] & (E) \end{cases} \tag{21}$$

where (S) stands for spherical and (E) stands for elliptical.

We will study two different system. The first part will focus on the non-interacting case using the spherical potential. After that we will extend our methods to a system with interactions using the elliptical potential. The interaction potential V_{int} is given by

$$V_{\text{int}}(r_{ij}) = \begin{cases} \infty & r_{ij} \leq a \\ 0 & r_{ij} > a \end{cases} \tag{22}$$

where $r_{ij} = |\boldsymbol{r_i} - \boldsymbol{r_j}|$, the distance between particles i and j, and a is the hard-core diameter of the bosons. This potential is to prevent the particles from positioning themselves on top of each other.

2.7 Trial Wave Function

To get good estimates of the energy of the system, the trial wave function should be chosen so that it resembles the actual wave function of the system as close as possible [1]. However, this is not necessarily easy, and some intuition about the system is required to make a good choice. For this system we can use the trial wave function given by

$$\psi(\boldsymbol{r}) = \psi(\boldsymbol{r_1}, \boldsymbol{r_2}, ..., \boldsymbol{r_N}, \alpha, \beta) = \left[\prod_{i}^{N} g(\alpha, \beta, \boldsymbol{r_i}) \right] \left[\prod_{j<k}^{N} f(a, |\boldsymbol{r}_j - \boldsymbol{r}_k|) \right] \tag{23}$$

where N is the number of parameters, and α and β are variational parameters. The first part of the expression is given by $g(\alpha, \beta, \boldsymbol{r_i}) = \exp[-\alpha(x_i^2 + y_i^2 + \beta z_i^2)]$ and $\boldsymbol{r_i} = [x_i, y_i, z_i]^T$. The correlation wave function f is given by

$$f(a, |\boldsymbol{r_j} - \boldsymbol{r_k}|) = \begin{cases} 0 & |\boldsymbol{r}_j - \boldsymbol{r}_k| \leq a \\ (1 - \frac{a}{|\boldsymbol{r}_j - \boldsymbol{r}_k|}) & |\boldsymbol{r}_j - \boldsymbol{r}_k| > a, \end{cases} \tag{24}$$

2.7.1 Non-interacting case in spherical trap

For the non-interacting wave function in a spherical trap, a = 0 and $\beta = 1$. The trial wave function then reduces to

$$\psi(\boldsymbol{r}) = \psi(\boldsymbol{r_1}, \boldsymbol{r_2}, ..., \boldsymbol{r_N}, \alpha, \beta) = \prod_i^N g(\alpha, \beta, \boldsymbol{r_i}). \tag{25}$$

The Hamiltonian also simplifies to

$$H = \sum_i^N \Big(\frac{\hbar^2}{2m} \nabla_i^2 + V_{\text{ext}}(\boldsymbol{r_i}) \Big) \tag{26}$$

since the case where $r_{ij} < a$ never occurs. The expression for the local energy for the non-interacting case with a spherical trap ($\beta = 1$) can easily be found analytically by finding the expression for the Laplacian of the trial wave function. Doing so gives the local energy,

$$E_L(\boldsymbol{r}) = \frac{1}{\psi_T(\boldsymbol{r})} H \psi_T(\boldsymbol{r}) = \frac{\hbar^2}{2m} \Big(-2\alpha N d + 4\alpha^2 \sum_{i=1}^N \sum_{j=1}^d (x_i)_j^2 \Big) + \sum_i^N V_{\text{ext}}(\boldsymbol{r_i}), \tag{27}$$

where N is the number of particles in the system and d is the number of dimensions. Here $(x_j)_i$ is the j'th coordinate of the i'th particle and we will use $\hbar = m = 1$. The expression then becomes

$$E_L(\boldsymbol{r}) = \frac{1}{\psi_T(\boldsymbol{r})} H \psi_T(\boldsymbol{r}) = \Big(\alpha N d + 4\alpha^2 \sum_{i=1}^N \sum_{j=1}^d (x_i)_j^2 \Big) + \sum_i^N \frac{1}{2} \omega^2 \boldsymbol{r_i}^2, \tag{28}$$

For a full derivation of the result, see the appendix. It is also possible to compute the local energy by using numerical derivatives for the first part of the Hamiltonian. For a discussion on numerical derivatives, see for example [1]. In general, the first derivative can be found from

$$f'(x) = \frac{f(x+h) - f(x)}{h} \tag{29}$$

and the second derivative

$$f''(x) = \frac{f(x-h) - 2f(x) + f(x+h)}{h^2}. \tag{30}$$

The expression for the drift-force can similarly be derived with analytical techniques, which gives that

$$F = \frac{2\nabla_k \psi_T}{\psi_T} = 2\frac{\nabla_k \phi(\boldsymbol{r}_k)}{\phi(\boldsymbol{r}_k)} \tag{31}$$

where $\nabla_k \phi(\boldsymbol{r_k}) = -2\alpha \boldsymbol{r_k} \phi(\boldsymbol{r_k})$ so the drift force with respect to the k'th particle is given by

$$F = -4\alpha \boldsymbol{r_k}. \tag{32}$$

For the non-interacting system in an spherical trap it is possible to find an analytical expression for the energy of the system. This is beneficial since the numerical methods can be tested and fine-tuned on those system before tackling more complicated system. The analytical expression is given by

$$E(\alpha) = \text{dim} \cdot N \cdot \frac{4\alpha^2 + 1}{8\alpha}. \tag{33}$$

where dim is the number of dimensions, N is the number of particles, where $\hbar = m = 1$ [6]. The derivation of the expression can be found in the appendix.

To find the ground state energy, we can minimize this expression. First finding the derivatie, which gives

$$E'(\alpha) = \text{dim} \cdot N \cdot \frac{8^2\alpha^2 - (4\alpha^2 + 1)8}{8^2\alpha^2} = N \cdot \text{dim} \Big(1 - \frac{32\alpha^2}{64\alpha^2} + \frac{1}{8\alpha^2} \Big). \tag{34}$$

Setting this equal to 0 and solving for α gives

$$\alpha_{\text{min}} = 0.5, \tag{35}$$

which is independent on the size of the system. Inserting this gives the ground state energy

$$E(\alpha_{\text{min}}) = \text{dim} \cdot N \cdot 0.5. \tag{36}$$

The energy is a linear function of N and dim.

The problem with using the system which do not include interactions is that the trial wave function don't include the interactions between particles. By including a correlation term the simulations can produce better results.

2.7.2 Interacting case in elliptical trap

For the interacting case in a elliptical trap the trial wave function is the full expression in 23. The energy can no longer be solved analytically for this trial wave function, so Monte Carlo integration is needed. The expression for the local energy is a bit harder to derive. The local energy is given by

$$\begin{aligned} E_L = \frac{1}{\psi_T(\boldsymbol{r})} H\psi_T(\boldsymbol{r}) = \sum_k^N \left(\frac{-\hbar^2}{2m} \frac{\nabla_k^2 \psi_T}{\psi_T} + V_{\text{ext}}(\boldsymbol{r_k}) \right) + \sum_{i<j}^N V_{\text{int}}(\boldsymbol{r_i}, \boldsymbol{r_j}) = \\ \sum_k^N \left(\frac{-\hbar^2}{2m} \left[\frac{\nabla_k^2 \phi(\boldsymbol{r_k})}{\phi(\boldsymbol{r_k})} + 2 \frac{\nabla_k \phi(\boldsymbol{r_k})}{\phi(\boldsymbol{r_k})} \left(\sum_{j \neq k} u'(r_{kj}) \frac{\boldsymbol{r_k} - \boldsymbol{r_j}}{r_{kj}} \right) + \right. \right. \\ \left. \left. \sum_{j \neq k} \left(u''(r_{kj}) + \frac{\text{dim - 1}}{r_{kj}} u'(r_{kj}) \right) + \left(\sum_{j \neq k} u'(r_{kj}) \frac{\boldsymbol{r_k} - \boldsymbol{r_j}}{r_{kj}} \right)^2 \right] + V_{\text{ext}}(\boldsymbol{r_k}) \right) + \\ \sum_{i<j}^N V_{\text{int}}(\boldsymbol{r_i}, \boldsymbol{r_j}) \end{aligned} \tag{37}$$

Using the chain-rule it can be shown that

$$u'(r_{jk}) = \frac{a}{(1 - \frac{a}{r_{jk}}) r_{jk}^2} \tag{38}$$

and

$$u''(r_{jk}) = \frac{-2r_{jk}a + a^2}{(r_{jk} - a r_{jk})^2} \tag{39}$$

where the standard rule for derivatives of logarithms is used.

The drift force with respect to the k'th particle for the interacting case is given by

$$\boldsymbol{F} = \frac{\nabla_k \phi(\boldsymbol{r_k})}{\phi(\boldsymbol{r_k})} + \nabla_k \left(\sum_{j \neq k} u(r_{kj}) \right). \tag{40}$$

For a full derivation of the expressions, see the appendix.

When working with the elliptical system, the first term in the drift-force and the local energy needs to account for the beta factor in front of the third dimension. The expression for the gradient in 3 dimensions

$$\nabla_k \phi(\boldsymbol{r_k}) = -2\alpha [x_k, y_k, \beta z_k]^T \phi(\boldsymbol{r_k}) \tag{41}$$

and the laplacian is given by

$$\nabla_k \cdot \nabla_k \phi(\boldsymbol{r_k}) = -2\alpha \cdot (2 + \beta) \phi(\boldsymbol{r_k}) + 4\alpha^2 \left(x_i^2 + y_i^2 + \beta^2 z_i \right) \phi(\boldsymbol{r_k}) \tag{42}$$

2.8 Scaling of the Hamiltonian

In order to make the Hamiltonian easier to implement and to avoid all the constants, the Hamiltonian can be scaled. Introducing the scaled variable

$$r^* = \frac{r}{a_{ho}} \rightarrow r = r^* a_{ho} \tag{43}$$

the Nabla operator (in 3D) can be rewritten as

$$\nabla = \frac{1}{a_{ho}}\left[\frac{\partial}{\partial x^*}, \frac{\partial}{\partial y^*}, \frac{\partial}{\partial z^*}\right]. \tag{44}$$

where the dot product of the Nabla operator with itself can be found to be

$$\nabla^2 = \frac{1}{a_{ho}^2}\left(\frac{\partial}{\partial x^*} + \frac{\partial}{\partial y^*} + \frac{\partial}{\partial z^*}\right) \tag{45}$$

Inserting this into the original Hamiltonian gives

$$H = \sum_i^N \left(-\frac{\hbar^2}{2m}\frac{m\omega}{\hbar}(\nabla^*)^2 + \frac{1}{2}m\omega_{ho}^2\frac{\hbar}{m\omega_{ho}}\big((x_i^*)^2 + (y_i^*)^2 + \omega_z^2/\omega_{ho}^2(z^*)^2\big)\right) + \sum_{i<j} V_{\text{int}} \tag{46}$$

where $a_{ho} = \left(\frac{\hbar}{m\omega_{ho}}\right)^{0.5}$ and $\nabla^* = [\partial/\partial x^*, \partial/\partial y^*, \partial/\partial z^*]$. By simplifying the expression it becomes

$$H = \hbar\omega_{ho}\sum_i^N \left(-\frac{1}{2}(\nabla^*)^2 + \frac{1}{2}\big((x_i^*)^2 + (y_i^*)^2 + \omega_z^2/\omega_{ho}^2(z^*)^2\big)\right) + \sum_{i<j} V_{\text{int}} \tag{47}$$

By letting the energy be in units of $\hbar\omega_{ho}$ and set $\gamma^2 = \frac{w_z^2}{w_{ho}^2} = 2.82843$, the final expression is

$$H = \sum_i^N \left(-\frac{1}{2}\nabla^2 + \frac{1}{2}\big(x_i^2 + y_i^2 + \gamma^2 z^2\big)\right) + \sum_{i<j} V_{\text{int}} \tag{48}$$

where the markings are removed.

2.9 Optimization Methods

Finding the optimal variational parameters analytically can be a challenge when the number of particles and dimensions increase or when using complicated Hamiltonians and wave functions. Numerical optimization methods can in these cases be a good choice.

A popular method that often work well is the gradient descent method. The idea behind the gradient descent method is that we move in the opposite direction of the gradient. The gradient is the direction where a function increases fastest, so moving in the opposite direction will lead to the fastest decrease in function value. More formally, this can be written as

$$x_{t+1} = x_t - \gamma\frac{df(x)}{dx}, \tag{49}$$

where γ is the step-length and x_{t+1} is the updated estimate of the minimum [2].

The gradient descent algorithm runs until some stopping criteria is reached. This can for example be that the norm of the gradient is smaller than some tolerance, or that the number of iterations have exceeded some predetermined number of iterations. In this article the predetermined number of iterations is used since the size of the gradient depends heavily on the number of particle in the system. In addition, the efficiency of the algorithm is very dependent on the choice of step-length [2]. By choosing it to small, the algorithm will take very long to converge. If the step-length is to large, the algorithm might not converge at all, and it can bounce around the solution or in some cases diverge away from the solution [2]. Therefore, it is often useful to test for different values of γ.

For the problem of finding the optimal alpha, we want to minimize the expected local energy with respect to alpha. The derivative of the local energy with respect to α is given by

$$\frac{\partial\langle E_L\rangle}{\partial\alpha} = 2\left(\left\langle E_L\frac{1}{\psi_T}\frac{\partial\psi_T}{\partial\alpha}\right\rangle - \langle E_L\rangle\left\langle\frac{1}{\psi_T}\frac{\partial\psi_T}{\partial\alpha}\right\rangle\right) \tag{50}$$

[1]. By showing that

$$\frac{1}{\psi_T}\frac{\partial\psi_T}{\partial\alpha} = \sum_{i=1}^N\frac{1}{\psi_T}\frac{\partial}{\partial\alpha}\left[-\alpha\sum_{d=1}^{\text{nDims}}(x_i)_d^2\right] = -\sum_{i=1}^N\sum_{d=1}^{\text{nDims}}(x_i)_d^2. \tag{51}$$

the expression for the local energy derivative with respect to α can be calculated using Monte Carlo methods. For each step in the Metropolis algorithm we sample $E_L\frac{1}{\psi_T}\frac{\partial\psi_T}{\partial\alpha}$, E_L and $\frac{1}{\psi_T}\frac{\partial\psi_T}{\partial\alpha}$, and at the end we use Monte Carlo methods for calculating the mean of the samples. Then the derivative of the local energy with respect to alpha can easily be calculated from equation 50.

This results in that the for each iterations of the gradient descent method, we have to run the Metropolis algorithm for a number of iterations to calculate the gradient. This can become quite expensive, since the Metropolis algorithm is time consuming. One idea is to use fewer Metropolis steps each time, and instead calculate a proper estimate of the energy with more Metropolis steps when the optimal alpha is found. It is important to reset the system between each iteration in the gradient descent algorithm. The gradient descent method updates the estimate as follows

$$\alpha_{t+1} = \alpha_t - \gamma\frac{\partial\langle E_L\rangle}{\partial\alpha} \tag{52}$$

where γ is the step-length.

2.10 One-body densities

The one-body density of a system is defined as

$$\rho(x_i) = \int |\psi(x_1, x_2, .., x_N)|^2 dx_2 dx_3 ... dx_N \tag{53}$$

which means that we are interested in the marginal density of particle i. In this article I will focus on how to estimate this numerically. The method for doing this is described in the implementation part.

2.11 Statistical Analysis

It is possible to get better estimates of the expected values and variance. This can be done by using bootstrapping and blocking. There is a problem with the Markov chain Monte Carlo method that the samples that are generated can be quite correlated. This is because the sampling of the next position comes from a probability distribution that are dependent on the previous position. The standard method for calculating the variance is therefore not as accurate. For uncorrelated samples,

$$\sigma^2 = \frac{1}{N-1}(\langle E_L^2\rangle - \langle E_L\rangle^2) \tag{54}$$

can be used. However, it can be shown that when the samples are correlated, the variance is actually given by

$$\sigma^2 = \frac{1+2\tau/\delta t}{N-1}(\langle E_L^2\rangle - \langle E_L\rangle^2) \tag{55}$$

where τ is the correlation time and δt is the time between each sample [4]. In a lot of cases the correlation time is larger than the time between each sample. This means that the actual variance is much higher compared to what the standard estimate of the variance of uncorrelated samples says. Instead, we will use blocking. Bootstrapping is also discussed for calculating the mean of the samples. The parts about bootstrapping and blocking is based on the slides written by M. Hjort-Jensen for the lecture about resampling methods found at `http://compphysics.github.io/ComputationalPhysics2/doc/pub/statanalysis/html/statanalysis.html`.

2.11.1 Bootstrapping

Bootstrapping is a popular resampling method that can be used to compute various statistics, like the mean and variance, of the data. In this article the focus will be on estimating the mean (blocking is used for calculating the variance). The main idea is to draw random sets of samples with replacement from a data set, and then get a number of replicas of a statistic of interest. Then we can to inference on this estimator and find the mean and variance of the estimator. The method can be summarized as follows:

- Run a simulation and record the desired samples of the system for each step of the simulation. For this case the local energies will be sampled.

- Draw n values from the recorded samples with replacement, and calculate the desired properties for the sample. For example the estimator $\hat{\theta}$, and call the estimator for the i'th iteration $\hat{\theta}_i$
- Repeat the previous step n times, and get n estimators $\hat{\theta}_i$.
- The last step is then to calculate some statistic on the recorded estimator, $\hat{\theta}_i's$, for example the mean.

The main problem with the bootstrapping method is that it is computationally expensive. The bootstrapping method will be compared to the standard way of estimating the mean. We will see that for this problem, the extra time is not worth it.

2.11.2 Blocking

Blocking is a popular method for estimating the variance of the mean of a population, $V[\hat{\theta}]$, where $\hat{\theta} = \bar{X}$. It has the advantage over bootstrapping that it gets more accurate when the number of samples is large. The main idea is to group together samples with high correlation and then calculate the estimates on theese groups.

When using blocking we need the number of samples to be on the form $n = 2^d$ where d is an integer larger than one. Take a vector of samples $\boldsymbol{X}$. The idea is to iteratively make new vectors that consist of the mean of subsequent pairs of the previous vector. The i'th vector can be defined by

$$(\boldsymbol{X_0})_k = (\boldsymbol{X})_k \tag{56}$$

$$(\boldsymbol{X_{i+1}})_k = \frac{1}{2}\bigg((\boldsymbol{X_i})_{2k-1} + (\boldsymbol{X_i})_{2k}\bigg) \tag{57}$$

where $(\boldsymbol{X}_i)_j$ is the j'th component of the i'th blocking vector, and is the result of i blocking transformations. This means that if we start with an vector $\boldsymbol{X} = [X_1, X_2, X_3, X_4]^T$, then $\boldsymbol{X_0} = \boldsymbol{X}$, $\boldsymbol{X_1} = \begin{bmatrix}\frac{(X_0)_1+(X_0)_2}{2}, \frac{(X_0)_3+(X_0)_4)}{2}\end{bmatrix}$ and $\boldsymbol{X_2} = \begin{bmatrix}\frac{(X_1)_1+(X_1)_2}{2}\end{bmatrix}$. Each step is called a blocking transformation, so the vector $\boldsymbol{X_k}$ have gone through k blocking transformations. The k'th vector has $n/2^i$ elements, so for a vector of size 2^d we can do the transformation for a maximum of d-1 times. For each vector we can calculate the statistics of interest, and call for example the variance of the k'th vector σ_k^2.

We can now define the covariance between $(\boldsymbol{X}_{k+1})_i$ and $(\boldsymbol{X}_{k+1})_j$ as

$$\gamma_{k+1}(h) = cov((\boldsymbol{X}_{k+1})_i, (\boldsymbol{X}_{k+1})_j) \tag{58}$$

where $h = |i - j|$. By introducing

$$e_k = \frac{2}{n_k}\sum_{h=1}^{n_k-1}\bigg(1 - \frac{h}{n_k}\bigg)\gamma_k(h) \tag{59}$$

we can write the variance of the mean of the samples as

$$\mathrm{Var}(\bar{X_k}) = \frac{\sigma_k^2}{n_k} + e_k. \tag{60}$$

where σ_k^2 is the variance of the samples in $\boldsymbol{X_k}$. It can then be shown that this expression actually holds for all $0 \leq k \leq d-1$ by using that $n_{j+1}\bar{\boldsymbol{X}}_{j+1} = n_{j+1}\bar{\boldsymbol{X}}_j$. By doing enough blocking transformations, we can actually get e_k to go to zero, since it can be shown that the sequence of e_k decreases [5], and the variance can then be estimated by $\mathrm{Var}(\bar{X}) = \frac{\sigma_k^2}{n_k}$. Here $\hat{\sigma_k^2}$ is estimated using the standard method for finding the sample variance. The implementation used in this article is implemented by the teaching staff and the implementation follows the description in [5].

3 IMPLEMENTATION

All code used can be found at the GitHub-adress:
https://github.com/MartinKHovden/ComputationalPhysics2/tree/master/Project1_. The code for implementing most of the functions is quite spacious, so I will avoid adding most of them in the article. However, the code in the repository is thoroughly commented, so matching the various steps in the code with the corresponding theory should be doable.

The main-function is located in the file `main.cpp`. This is where the system is initialized with the desired parameters, wave function, and Hamiltonian. This is also where one can choose what type of simulation to run. The main function starts by setting up a new system. The system object comes from the file `system.cpp` which is a class that acts as a container for objects for Hamiltonians, the wave functions and other parameters. This is the "brain" of the program.

The first step is to choose a trial wave function for the system. This is done by creating a new instance of one of the sub-classes of the wave function class and using the function `setWavefunction` in the system class. The wave function objects includes functions for evaluating the wave function value and for calculating the laplacian, the gradient, the drift force, and the alpha-derivative of the wave function. There are classes for both the system with and without Jastrow factor.

Then an object of one of the sub-classes for the Hamiltonian is created and added to the system. The Hamiltonian classes implements function for calculating the local energy of the system and calls the functions from the wave function class. This is done according to the equations in the theory part.

The last step before simulation is to set the initial state of the system. This is done according to a gaussian distribution, since this resembles the actual shape of the wave function. In C++ this can be implemented with the following code snippet

```
for (int i=0; i < numberOfParticles; i++) {
        std::vector<double> position = std::vector<double>();

        for (int j=0; j < numberOfDimensions; j++) {
            //The particles are placed according to a gaussian function, since this
            //resembles the wavefunction of the system.
            position.push\_back(Normaldistribution2(gen2)*m_system->getStepLength());
        }

        m_particles.push\_back(new Particle());
        m_particles.at(i)->setNumberOfDimensions(numberOfDimensions);
        m_particles.at(i)->setPosition(position);
    }
```

where m_particles is a vector that contains instances of the particle class found in the file `particle.cpp` and getStepLength() returns the step-length in the Metropolis algorithm. There might be a good idea to implement a test to avoid particles being stacked on top of each other (Didn't think of this as a problem before it was to late to go back and fix).

The user can then choose what type of simulation to run. There are four choices; metropolis brute-force, metropolis importance sampling, gradient descent for finding the optimal alpha, and the computation of the one-body densities.

The Metropolis brute-force algorithm is implemented as described in the theory part. For full details on the implementation in c++, see `system.cpp`. The new position proposal for the brute-force variant can be implemented with the following code-snippet

```
for(int i = 0; i < old_position.size(); i++)
{
    int temp_ran = rand() % 10;

    if(temp_ran < 4.5)
    {
        new_position.push_back(old_position[i] - 1*getStepLength());
    }
    else
    {
        new_position.push_back(old_position[i] + 1*getStepLength());
    }
}
```

For the importance sampling variant the new step proposal can be implemented using

```
for(int i = 0; i < m_numberOfDimensions; i++)
{
    double random_int_1 = Normaldistribution(gen);

    new_position.push_back(old_position[i] + random_int_1*sqrt(timestep) +
        prev_drift_force[i]*m_D*timestep);
}
```

Greens-function used in the acceptance ratio can be calculated using the following code snippet

```
double greens_function_argument = 0.0;
// Calculate the greens function argument of the ratio for this particle.
for(int j = 0; j < m_numberOfDimensions; j++)
{
    greens_function_argument += -(old_position[j] - new_position[j] -
        m_D*timestep*new_drift_force[j])*(old_position[j] - new_position[j] -
        m_D*timestep*new_drift_force[j]);
    greens_function_argument -= -(new_position[j] - old_position[j] -
        m_D*timestep*prev_drift_force[j])*(new_position[j] - old_position[j] -
        m_D*timestep*prev_drift_force[j]);
}
greens_function_argument /= 4*m_D*timestep;
double greens_function = exp(greens_function_argument);
```

When running the Metropolis algorithms, one should often use a burn-in period before the samples are being recorded [3]. This is because the system takes some time to reach the equilibrium state. However, for this article, this is not done. I did not notice before it was to late, and I did not have the time to run all calculations again. Adding the burn-in period would probably give a bit better estimates, and would be an idea for further work. However, I don't think the difference would be drastic in this case after doing some testing.

The gradient descent algorithm is implemented as follows. Set an initial guess for alpha. The algorithm runs through a predefined number of iterations, and for each iteration the alpha-derivative of the expected energy is calculated using the Metropolis algorithm according to the theory and then the next alpha estimate is updated using

```
current_alpha_value = initialGuess //Set an initial guess for alpha.

for(int i = 0; i < numberOfGradientDescentSteps; i++)
{
    /*
    Set a new initial configuration of the system.
    Calculate the alpha-gradient by calling the sampler object using the metropolis
        algorithm for getting samples.
    */
    gradient = 2*(m_sampler->getEnergyTimesAlphaDerivative() -
        (m_sampler->getEnergy()*m_sampler->getAlphaDerivative()));

    current_alpha_value = current_alpha_value - stepLength*gradient;

}
return current_alpha_value
```

For the calculation of the one-body density we run a normal brute-force Metropolis algorithm for a given number of iterations. For each iteration, the distance from origo for each particle is sampled to file. The distances are then added to bins according to the distance from origo. Each bin takes distances in some interval $[r_i, r_{i+1})$ where $r_{i+1} - r_i = h$. The number of distances that fits into each bin is counted. The probability density is plotted by a scatter plot of the resulting frequencies. To get a better feeling for how the one-body density is shaped, a polynomial regression is done to make a smooth estimate. This will also make it easier to compare the densitites with and without the Jastrow factor. To transform the plot into a pdf, the bins have to be normalized. This can be done by dividing by requiring that $\sum_i^N hx_i/k = 1$, where x_i is the number of samples with distance from origo in the interval $[r_i, r_{i+1})$ and k is the normalization factor. Since we in total have N samples, we see that

$$Nh/k = 1 \tag{61}$$

so the normalization factor is Nh. Dividing each bin by this normalization factor and plotting the resulting values gives us an pdf. each by the normalization factor

To run tests on the code, see the information in the Readme-file at the GitHub-address. The tests checks if the Metropolis algorithm (brute-force and importance sampling) gives the correct results for the non-interacting system where the exact values are known. A test is also implemented to test the gradient descent algorithm which checks if it find the minimum alpha-value of 0.5 for the non-interacting system.

4 RESULTS AND DISCUSSION

All plots can be found in the various jupyter notebooks. The file-name of the notebooks describes the problem they solve. All simulation used the compiler flag -03 to speed up the calculations. The energies presented in this section is in units of $\hbar\omega_{ho}$.

4.1 Spherical harmonic oscillator with no interactions

4.1.1 Brute-force Metropolis

We will start by discussing the results from running the standard Metropolis algorithm on a system in the spherical harmonic oscillator for particles with no interactions. The trial wave function is given by

$$\psi_T(\boldsymbol{r}) = \prod_{k=1}^{N} \exp(-\alpha(x_i^2 + y_i^2 + z_i^2)) \tag{62}$$

and the Hamiltonian is the scaled version with natural units presented in the theory part

$$H = \sum_{k=1}^{N} \left[\frac{-\nabla_i^2}{2} + x_i^2 + y_i^2 + z_i^2 \right] \tag{63}$$

where the spherical potential trap is used and $\beta = m = \hbar = \omega = 1$ and $a = 0$. The Metropolis algorithm is run for 2^{19} steps with a step-length equal to 0.5. This seems to give the overall best results without the computing time getting to high. The variance of the estimates are calculated using the blocking method presented in the theory part. The estimates found using blocking is also compared to the estimate using the standard method for computing variance in a later section. However, since the standard method often gives to optimistic estimates for correlated data, the main analysis will use blocking. The first part of the discussion will focus on the simulation using the analytical expressions for the wave function derivatives. This method is later compared to the numerical implementation.

From the theory on the non-interacting system, we know the expression for the energy as a function of alpha, and that the optimal alpha value should be 0.5 for all combinations of particles and dimensions. The performance of the Metropolis algorithm can then easily by measured by comparison to the exact values.

The energy will first be calculated using the Metropolis algorithm for alpha-values 0.4, 0.5 and 0.6. In table 1 - 6 the results are shown for the non-interactive case with 1, 10, 100 and 500 particles in 1 and 3 dimensions. The local energy and its derivatives are calculated using the analytical expressions. The tables shows the total energy of the system and its relative error in parenthesis, the energy per particle, the variance calculated using blocking, the acceptance ratio and the computational time. The error is calculated using the exact values, and is the absolute difference between the Metropolis estimate and the exact value. The total energy of the system increases linearly with the number of particles and the number of dimensions. From this it also follows that for the non-interacting case, the energy per particle is approximately independent of the number of particles. This is also what was expected from the expression for the energy. From the tabular values it can be seen that the minimum is found for $\alpha = 0.5$, where the variance is zero, which is in agreement with the theoretical minimum. Also note that the acceptance ratio decreases when the number of dimensions increase. However, it keeps stable as a function of particles. As expected, the computing time increases when the number of particles and dimensions increase. When the number of particles increase, the variance also increase. For all combinations of particles and dimensions up to 500, the absolute error is smaller than 0.02. However, for 500 particles in 3 dimensions, the error is significantly larger at above 5.5. Looking at the relative error, it is always below 0.7, but still we see that the largest relatvie error occurs for the system with 500 particles. This indicates that the method can lead to poorer results when the size of the system increases.

α	$\langle E_{\text{total}} \rangle$	Error (relative error)	$\langle E_{\text{particle}} \rangle$	$\text{Var}_{\text{blocking}}$	acceptance ratio	time [s]
0.4	0.51336	0.000992 (0.19%)	0.51336	3.8151e-07	0.74981	0.218
0.5	0.5	0.0 (0%)	0.5	0.0	0.0	0.234
0.6	0.50816	0.000174 (0.03%)	0.50816	1.8075e-07	0.69284	0.234

Table 1: Non-interacting system with spherical trap for N = 1 and d = 1.

α	$\langle E_{\text{total}} \rangle$	Error (relative error)	$\langle E_{\text{particle}} \rangle$	$\text{Var}_{\text{blocking}}$	acceptance ratio	time [s]
0.4	1.5361	0.00136 (0.089%	1.536139541	1.6114e-06	0.58401	0.265
0.5	1.5	0.0 (0%)	1.5	0.0	0.0	0.265
0.6	1.5262	0.0011592 (0.07%)	1.52615922	9.1704e-07	0.50114	0.281

Table 2: Non-interacting system with spherical trap for N = 1 and d = 3.

α	$\langle E_{\text{total}} \rangle$	Error (relative error)	$\langle E_{\text{particle}} \rangle$	$\text{Var}_{\text{blocking}}$	acceptance ratio	time [s]
0.4	5.1138	0.0112 (0.2%)	0.5113794157	3.4108e-05	0.75341	0.859
0.5	5.0	0.0 (0.0%)	0.5	0.0	0.0	0.859
0.6	5.0889 (0.1%)	0.0056	0.508890389	1.8026e-05	0.69802	0.875

Table 3: Non-interacting system with spherical trap for N = 10 and d = 1.

α	$\langle E_{\text{total}} \rangle$	Error (relative error)	$\langle E_{\text{particle}} \rangle$	$\text{Var}_{\text{blocking}}$	acceptance ratio	time [s]
0.4	15.384	0.0086 (0.06%)	1.538360314	0.00016827	0.58384	0.953
0.5	15.0	0.0 (0.0%)	1.5	0.0	0.0	0.953
0.6	15.243	0.0072 (0.04%)	1.524278877	0.00010209	0.50221	0.953

Table 4: Non-interacting system with spherical trap for N = 10 and d = 3.

α	$\langle E_{\text{total}} \rangle$	Error (relative error)	$\langle E_{\text{particle}} \rangle$	$\text{Var}_{\text{blocking}}$	acceptance ratio	time [s]
0.4	51.249	0.0012 (0.0023%)	0.5124877603	0.0031497	0.75192	7.109
0.5	50.0	0.0 (0.0%)	0.5	0.0	0.0	7.031
0.6	50.837	0.0034 (0.006%)	0.5083674338	0.0013695	0.69971	7.078

Table 5: Non-interacting system with spherical trap for N = 100 and d = 1.

In table 8 and 9 the results from using the numerical expression is shown for 10 and 100 particles in 3 dimensions. Simulating a system with 500 particles was to tedious. The step-length in the numerical estimates was set to 1e-8. The results are close to the same as for the analytical method. The error and the variance is actually marginally smaller except for the system with 10 particles in 3 dimensions for $\alpha = 0.4$. For $\alpha = 0.5$, the numerical method also results in zero

α	$\langle E_{\text{total}} \rangle$	Error (relative error)	$\langle E_{\text{particle}} \rangle$	$\text{Var}_{\text{blocking}}$	acceptance ratio	time [s]
0.4	153.77	0.0184 (0.01%)	1.537683917	0.019798	0.58162	7.468
0.5	150.0	0.0	1.5	0.0	0.0	7.515
0.6	152.42	0.084 (0.06%)	1.524161906	0.009019	0.50005	7.468

Table 6: Non-interacting system with spherical trap for N = 100 and d = 3.

α	$\langle E_{\text{total}} \rangle$	Error (relative error)	$\langle E_{\text{particle}} \rangle$	$\text{Var}_{\text{blocking}}$	acceptance ratio	time [s]
0.4	763.23	5.5163 (0.7%)	1.526467315	0.16003	0.57391	41.28
0.5	750.0	0.0	1.5	0.0	0.0	41.26
0.6	766.87	4.3659 (0.5%)	1.533731711	0.04733	0.49194	42.70

Table 7: Non-interacting system with spherical trap for N = 500 and d = 3.

variance. However, when it comes to computational time, the numerical method is significantly more expensive to use. For a system with 100 particles in 3 dimensions it takes 1110 seconds to finish the simulation using the numerical expression. The same simulation using the analytical expression takes only 14 seconds, so the analytical version is close to 80 times faster. The rest of the results will therefore be presented using the analytical expressions.

α	$\langle E_{\text{total}} \rangle$	Error (relative error)	$\langle E_{\text{particle}} \rangle$	$\text{Var}_{\text{blocking}}$	acceptance ratio	time [s]
0.4	15.378	0.0033 (0.01%)	1.537828584	0.00016425	0.58138	13.71
0.5	15.0	0.0	1.5	0.0	0.0	13.67
0.6	15.248	0.0023 (0.015%)	1.524769812	9.5202e-05	0.50167	13.78

Table 8: Numerical non-interacting system with spherical trap for N = 10 and d = 3.

α	$\langle E_{\text{total}} \rangle$	Error (relative error)	$\langle E_{\text{particle}} \rangle$	$\text{Var}_{\text{blocking}}$	acceptance ratio	time [s]
0.4	153.98	0.2303 (0.15%)	1.539803017	0.018943	0.58214	1110.
0.5	150.0	0.0	1.5	0.0	0.0	1111.
0.6	152.49	0.0147 (0.01%)	1.524852888	0.0081995	0.50046	1108.

Table 9: Numerical non-interacting system with spherical trap for N = 100 and d = 3.

4.1.2 Importance sampling Metropolis

We now use the Metropolis algorithm with importance sampling. It is expected that by using importance sampling we will get a much higher acceptance ratio, since the new step is proposed after taking into consideration the shape of the wave function. The variance is also expected to decrease as a result of this. For all the runs with importance sampling, see the jupyter notebook `non_interacting_system_numerical.ipynb`. In table 11 to 15 the results are presented for one particle in 1 dimension and for 100 particles in 3 dimensions. The tables shows the dependence on the time-step in the importance sampling step. We can then look at the main trends in the various quantities as a function of the time-step. First of all, the acceptance ratio is highly dependent on the time-step. For a system of 1 particle in 1 dimension, the acceptance ratio is as low as 0.25 when the time-step is set to 3. By decreasing the time-step to 0.3 or 0.03, the acceptance ratio goes to above 0.97. This is as expected. When using a smaller time-step, the new proposed position is more similar to the previous one. The previous one are probably in a important place in space due to importance sampling, so the next step is also highly probable to be accepted. On the other hand, when the time-step is high, the new position can be far away from the previous step, which means that the new proposed position not necessarily is in a relevant position. For the system with 100 particles in 3 dimensions, the acceptance ratio is just above 0.1 for $\alpha = 0.4$. This shows that by decreasing the time-step, the acceptance ratio increases. However, the problem with choosing the time-step to small is the variance will increase. We note that the variance have a minimum (for the time-step values tested in this article) for $\delta t = 0.3$. Also the error increase when decreasing the the time-step from 0.3 to 0.03 or 3. The best time-step therefore seems to be 0.3, and is the time-step that will be used for the rest of the analysis. We can now compare the importance sampling method to the brute force method.

Both the brute-force Metropolis algorithm and the importance sampling Metropolis computes good estimates of the exact value, and they find the same minimums with zero variance for $\alpha = 0.5$. However, note that the variance of the mean of the local energy is higher for the Brute-force method. This comes from the fact that the importance sampling algorithm stays in the important parts of the space compared to the brute-force method who takes a lot more time to converge. We also see that for the importance sampling algorithm almost all of the proposed steps are accepted; between 90-100% accepted steps. For the Brute-force algorithm the acceptance ratio stays below 75%. The difference in acceptance ratio is as expected. In figure 2 the convergence of the two methods is compared. Importance sampling tends to stabilize faster. When it comes to computational time, both methods use similar time for the same systems.

α	$\langle E_{\text{total}}\rangle$	Error	$\langle E_{\text{particle}}\rangle$	$\text{Var}_{\text{blocking}}$	acceptance ratio	time [s]
0.4	0.51207	0.00043091 (0.08%)	0.5120690945	3.5691e-07	0.41298	0.312
0.5	0.5	0.0	0.5	0.0	0.0	0.312
0.6	0.50678	0.0015501 (0.3%)	0.5067832804	1.7494e-06	0.25061	0.343

Table 10: Importance sampling Metropolis, Non-interacting system with spherical trap for N = 1, d = 1 and $\delta t = 3$

α	$\langle E_{\text{total}}\rangle$	Error (relative error)	$\langle E_{\text{particle}}\rangle$	$\text{Var}_{\text{blocking}}$	acceptance ratio	time [s]
0.4	0.51253	2.59e-05 (0.005%)	0.5125258981	1.6992e-07	0.97366	0.312
0.5	0.5	0.0	0.5	0.0	0.0	0.328
0.6	0.5086	0.00026 (0.05%)	0.5085960186	7.8946e-08	0.95151	0.328

Table 11: Importance sampling Metropolis, Non-interacting system with spherical trap for N = 1, d = 1 and $\delta t = 0.3$

α	$\langle E_{\text{total}}\rangle$	Error (relative error)	$\langle E_{\text{particle}}\rangle$	$\text{Var}_{\text{blocking}}$	acceptance ratio	time [s]
0.4	0.51255	5.0265e-05 (0.01%)	0.5125502651	1.7491e-06	0.99915	0.312
0.5	0.5	0.0	0.5	0.0	0.0	0.312
0.6	0.50861	0.00027846 (0.05%)	0.508611793	7.6959e-07	0.99841	0.312

Table 12: Importance sampling Metropolis, Non-interacting system with spherical trap for N = 1, d = 1 and $\delta t = 0.03$

α	$\langle E_{\text{total}}\rangle$	Error (relative error)	$\langle E_{\text{particle}}\rangle$	$\text{Var}_{\text{blocking}}$	acceptance ratio	time [s]
0.4	153.48	0.26578 (0.17%)	1.534842196	0.047603	0.10736	8.609
0.5	150.0	0.0	1.5	0.0	0.0	8.703
0.6	152.27	0.22527 (0.14%)	1.52274732	0.15038	0.02446	8.640

Table 13: Importance sampling Metropolis, Non-interacting system with spherical trap for N = 100, d = 3 and $\delta t = 3.0$

α	$\langle E_{\text{total}}\rangle$	Error (relative error)	$\langle E_{\text{particle}}\rangle$	$\text{Var}_{\text{blocking}}$	acceptance ratio	time [s]
0.4	153.7	0.0532 (0.03%)	1.535732394	0.0054229	0.94538	7.593
0.5	150.0	0.0	1.5	0.0	0.0	7.578
0.6	152.55	0.0456 (0.03%)	1.525456128	0.0026923	0.90078	7.578

Table 14: Importance sampling Metropolis, Non-interacting system with spherical trap for N = 100, d = 3 and $\delta t = 0.3$

4.1.3 Blocking vs. standard variance estimation

The variances presented in the tables above are calculated using the blocking method described in the theory part. To illustrate the problem with using the standard method for calculating the

α	$\langle E_{\text{total}} \rangle$	Error (relative error)	$\langle E_{\text{particle}} \rangle$	$\text{Var}_{\text{blocking}}$	acceptance ratio	time [s]
0.4	153.53	0.2181 (0.14%)	1.535318955	0.037567	0.9917	8.671
0.5	150.0	0.0	1.5	0.0	0.0	8.671
0.6	152.76	0.25819 (0.16%)	1.527581873	0.020902	0.9901	8.640

Table 15: Importance sampling Metropolis, Non-interacting system with spherical trap for N = 100, d = 3 and $\delta t = 0.03$

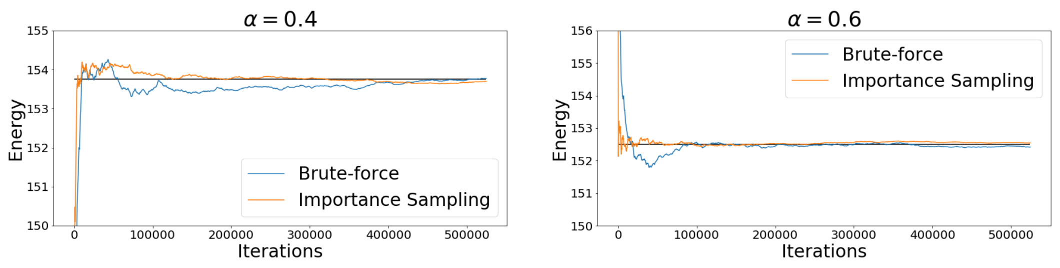

Figure 1: Comparison of convergence rate for Brute-force Metropolis and importance sampling metropolis on the non-interacting system. N=100, d=3 and step-length = 0.5. Time-step = 0.3.

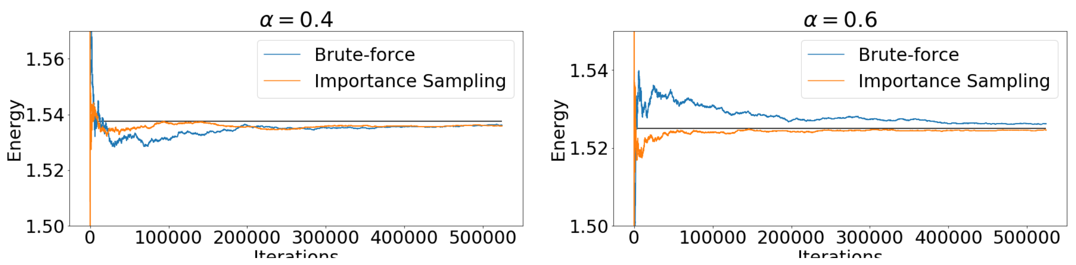

Figure 2: Comparison of convergence rate for Brute-force Metropolis and importance sampling metropolis on the non-interacting system. N=1, d=3 and step-length = 0.5. Time-step = 0.3. X-label: iterations (dissapeared when downloading plot to overleaf.)

sample variance of the sample means we use this method for two different system configurations. The standard sample variance is calculated using the formula

$$Var(\bar{X}) = \left(E(X^2) - E(X)^2\right)/n \tag{64}$$

and the results are presented in table 16. One system with 1 particle in 1 dimension and

	N=1, d=1		N=100, d=3	
Method	$\alpha = 0.4$	α=0.6	$\alpha = 0.4$	$\alpha = 0.6$
$\text{Var}_{\text{blocking}}$	3.81e-07	1.81e-07	0.019798	0.009091
$\text{Var}_{\text{standard}}$	4.89e-08	3.20e-08	1.57e-05	1.0381e-05

Table 16: Comparison between the variance estimate using blocking and the standard method on the non-interacting system for different combinations of number of particles and number of dimensions. The values are calculated using importance sampling algorithm with time-step=0.3

one with 100 particles in 3 dimensions. The problem with the auto-correlation is present in these calculation. The standard method for calculating the variance underestimates the actual variance, and provides lower estimates of the variance compared to using blocking. The rest of the variance estimates will therefore continue to be presented using the blocking method.

Instead of calculating the mean of all the samples to get the energy, we can use bootstrapping for estimating the mean of the samples. In table 17 the results are presented for the same systems as for the blocking test. We see that the bias is small when using the normal method and the estimates are approximately the same. However, the computational time using the bootstrapping method was much higher. Going forward, all energies will be presented by simply taking the standard sample mean of all the samples.

	N=1, d=1		N=100, d=3	
Method	$\alpha = 0.4$	α=0.6	$\alpha = 0.4$	$\alpha = 0.6$
$\langle \mathrm{E}_{\mathrm{total}} \rangle_{\mathrm{bootstrap}}$	0.51346	0.50817	153.78	152.41
$\langle \mathrm{E}_{\mathrm{total}} \rangle_{\mathrm{standard}}$	0.51349	0.50816	153.768	152.416

Table 17: Comparison between the mean estimate using bootstrapping and the standard method on the non-interacting system for different combinations of number of particles and number of dimensions. The values are calculated using importance sampling algorithm with time-step=0.3

4.2 Elliptical harmonic oscillator with interactions

4.2.1 Brute-force Metropolis

The results from using the brute-force metropolis algorithm on the elliptical harmonic oscillator with interactions with a = 0.0043 and $\beta = \gamma = 2.838$ are now studied. The simulations are run for 2^{19} metropolis steps with the step-length = 0.5 and 0.05, for 10 and 50 particles in 3 dimensions. For 100 particles I will limit myself to only simulating the system using the optimal alpha-value found using the gradient descent method in a later section. For this system the step-length is also reduced to 0.01 as an attempt to improve stability (without too good results). Comparing the energies in table 18 and 19 to the non-interacting case we note that the energy per particle increases when the particles interacts. For the system with 10 particles in 3 dimensions, the energy for the non-interacting system is 15 for $\alpha = 0.5$, while the energy is 22.18 for the interacting system. The energy per particle for this case goes from 1.5 to 2.28. We also note that the energy per particle is not the same when the number of particles is increased for the interacting case. The energy per particle is higher for 50 particles compared to for 10 particles. This is in contrast to the non-interacting case where the energy per particle was the same no matter the number of particles. The Jastrow factor have a significant effect on the energy of the system. The variance also increases, and the acceptance ratio gets lower. When it comes to the computational time, the time for the non-interacting case is 7 seconds, while the time is 37 seconds for the interacting case. This is as expected due to the complicated expression for the wave function compared to the non-interacting case. Again, it seems like the optimal alpha value is close to 0.5 from looking at the values in the table, but we would expect a slight change in minimum when the particles interact. In the next section the gradient descent algorithm will be used to get a more accurate estimate of the minimum.

α	$\langle E_{\mathrm{total}} \rangle$	$\langle E_{\mathrm{particle}} \rangle$	Var	acceptance ratio	time [s]
0.4	24.353	2.435329173	0.00052572	0.49011	37.03
0.5	23.188	2.223771731	0.0016673	0.43939	36.78
0.6	23.725	2.372519635	0.00019564	0.3982	37.07

Table 18: Brute-force Metropolis on the interacting system with a spherical trap with $\beta = \gamma = 2.8342$ for N = 10 and d = 3. The step-length = 0.5 and the algorithm is run for 2^{19} steps.

α	$\langle E_{\mathrm{total}} \rangle$	$\langle E_{\mathrm{particle}} \rangle$	Var	acceptance ratio	time [s]
0.4	122.29	2.445789319	0.2875	0.49969	954.4
0.5	118.25	2.407916953	0.82747	0.45207	943.5
0.6	121.91	2.433742972	1.1648	0.41204	944.2

Table 19: Brute-force Metropolis on the interacting system with a spherical trap with $\beta = \gamma = 2.8342$ for N = 50 and d = 3. The step-length = 0.05 and the algorithm is run for 2^{19} steps.

Comparing the energy values to the results in [7], we see that the results obtained here is bit lower than what they got. For 10 particles and $\alpha = 0.5$ I got an energy of 23.188 while Markova et. al. got an energy of 24.288. The same goes for 50 particles woth $\alpha = 0.5$ where my estimate of 118.25 is lower than their estimate of 123.06. However, for both cases, the Gross-Pitaevskii equation gives a value in the middle of the estimates, which might indicate the the actual solution lies somewhere in the middle.

4.3 Finding the optimal α-value using gradient descent

The results from running the gradient descent algorithm are shown in table 20 and 21. The results are presented from running the algorithm on both the non-interacting system (table 20) and the interacting system (table 21). For each step in the gradient descent algorithm the Metropolis algorithm ran for $2^{19}/10$ iterations and the gradient descent algorithm were stopped when number of iterations reached 100.

From the theory part we know that the minimum for the non-interacting system is found for $\alpha_{\text{min}} = 0.5$ for any number of particles and dimensions. Table 20 shows that the gradient descent algorithm are able to locate the minimum for the non-interacting system of varying combinations of number of particles and dimensions. For this system, the algorithm worked very well and converged for all initial values in the range [0.1,0.9].

	N=1, d=1	N=1, d=3	N=10, d=1	N=10, d=3	N=100, d=1	N=100,d=3
α_{min}	0.5	0.5	0.5	0.5	0.5	0.5

Table 20: Optimal alpha value found using the gradient descent algorithm on the non-interacting system. GD-step-length = 0.001, number of GD steps = 100, number of Metropolis steps per GD step = $\frac{2^{19}}{10}$. Metropolis step-length = 0.5

Table 21 shows the results from running gradient descent on the interacting system with a = 0.0042 and $\beta = 2.838$. Since the Metropolis algorithm was a bit unstable for systems with 50 and 100 particles, the Metropolis step-length was set to 0.01. The table shows that the minimum is obtained for a slightly perturbed α-value away from 0.5 when the number of particles increase. For a system with 50 particles, the optimal value was found to be 0.50032 and for a system with 100 particles the optimal value was 0.5017. The perturbation in the alpha-value increases with the number of particles. However, the change is small due to the weak forces between the particles. By using a stronger force, the perturbation would probably be larger.

	N=10, d=3	N=50, d=3	N=100, d=3
α_{min}	0.5	0.50032	0.5017

Table 21: Optimal alpha value found using the gradient descent algorithm on the interacting system. Step-length = 0.000001, number of GD steps = 100, number of Metropolis steps per GD step = $\frac{2^{19}}{10}$. Results are an average of three runs starter from 0.495, 0.5 and 0.505.

N	$\langle E_{\text{total}} \rangle$
10	23.3
50	121.2
100	235.6

Table 22: Energy values for the non-interacting system with the optimal alpha-values from table 21.

The energy found for the optimal alpha values in the interacting system is presented in table 22. The results from Markova et. al. [7] is 24.274, 123.67 and 246.90 for their optimal alpha-value 0.50614, 0.51958 and 0.52981 for 10, 50 and 100 particles. We note that the optimal alpha-values found by Markova et. al. is more pertubed than what was found in this article. The minimum energy values was slightly smaller in our analysis. However, the difference is small and might come down numerical error due to too few Metropolis steps or the choice of step-length.

4.4 One-body densities

In figure 3 the one-body density of a particle in a system with 100 particles in 3 dimensions are shown. The optimal alpha value for this system is chosen according to the minimum found using the gradient descent method. The results are shown for both the non-interacting case and the interacting case. When the particles interact, the particles tends to spread out more due to the hard-core diameter of each particles. This is because multiple particles can not occupy the same space. The curve for the interacting system is therefore a bit flatter than the curve for the non-interacting system. The top of the curve is also slightly more to the right. However, for a interacting system where the forces are so small, the difference is not large, as we can see on the plot. Running the simulation for a system with greater forces between the particles or a larger hard sphere radius would lead to an even flatter curve.

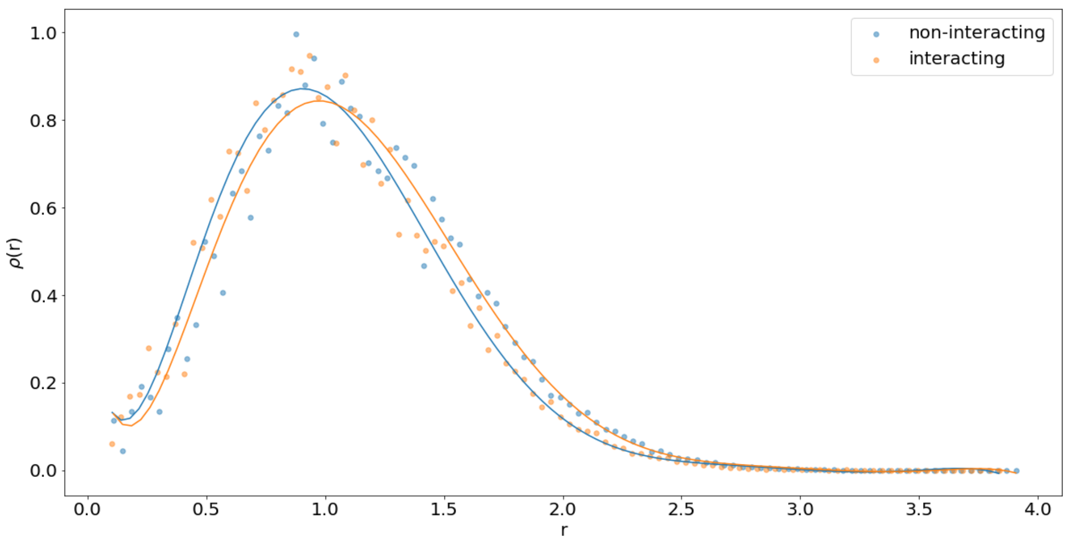

Figure 3: One-body density for a particle in a system with 100 particles in 3 dimensions for the non-interacting (a=0) and the interacting (a = 0.0043) systems for $\alpha = 05017$.

5 CONCLUSION

In this article VMC methods was used for calculating the ground state energy of a trapped Bose gas. The trapped Bose gas was modelled using two system with different complexity. The methods was tested against analytical values which showed that the implementations worked well. The first system used for modelling the Bose gas was particles in a spherical harmonic oscillator with a Gaussian wave function. The exact values for this system was derived, and the Metropolis algorithm was tested against the exact values. Both the Brute-force Metropolis algorithm and the importance sampling algorithm was able to calculate the ground state energy effectively for $\alpha = 0.5$ with no variance. For other α-values the estimates were good for both methods for particle number up to 100, with relative error below 1%. The estimates using VMC matched the exact values. However, the importance sampling metropolis was better in every other aspect except for the computational time which was approximately the same for both methods on the non-interacting system. The variance was in general smaller and the acceptance ratio was much higher, above 90% compared to below 60% for the brute-force method. Using the gradient descent method on the non-interacting system gave good results, and it was able to find the true minimum for $\alpha = 0.5$ for all combinations of particles tested in this article. Another interesting thing was the comparison between using blocking and the standard estimate for variance. The standard estimate tends to underestimate the variance when the samples are correlated. This was also the case here since the standard method generally gave a smaller estimate of the variance compared to using blocking. Using the brute-force Metropolis algorithm on the interacting system with the elliptical potential trap and the added Jastrow factor gave good results up to 50 particles, and the results was close to results from other studies [7]. However, the algorithm started to get problems for 100 particles, and the estimates was very unstable. This might have been due to a bad choice of step-length or a bug in the implementation. Using the gradient descent algorithm for the systems of 10 and 50 particles, we saw a small perturbation in α compared to the non-interacting case. The perturbation grew with the number of particles. The gradient descent algorithm was not as stable for 100 particles. However, for some runs I managed to get convergence to $\alpha = 0.5012$. Plotting the one-body density showed that the particles tended to spread out more when the interactions between the particles was included. This was because the particles could no longer be stacked on top of each other.

For further work I would suggest to run the Metropolis steps for more steps to get better estimates. I did not have the computing power to do this in the time I had available. It might also be interesting to run the simulations for a higher number of particles. In addition, there might be some parameters in the implementations that deserved some more testing. First off all it might be a good idea to implement a test that assures that no particles are on top of each other in the initial state. There might also be a good idea to use a burn in period before the sampling start, to get better estimates of the quantities of interest. The work in this report could also be used as a base for studies of more complicated systems, which would be interesting to try out in the future.

6 APPENDIX

The main goal of this appendix is to derive the different analytical expression used in the article. The first part will focus on deriving the expression for the local energies for the interacting and the non-interacting case. After that the remaining expression will be derived with the help of the intermediate steps in the derivation of the local energies.

6.1 Analytical expressions for the local energy

6.1.1 Interacting case

For the interacting case the trial wave function is given by

$$\psi_T(\boldsymbol{r}) = \left[\prod_i g(\alpha, \boldsymbol{r_i})\right] \exp\left(\sum_{j<k} u(\boldsymbol{r_{jk}})\right) = \left[\prod_i \phi(\boldsymbol{r_i})\right] f(\boldsymbol{r_{jk}}) \tag{65}$$

where f is called the Jastrow factor, $\boldsymbol{r}_{ij} = |\boldsymbol{r_i} - \boldsymbol{r_j}|$ and $u(\boldsymbol{r_{jk}}) = \ln(f(\boldsymbol{r_{jk}}))$.

The first step is to find the gradient of this expression with respect to particle k. Using the product rule gives

$$\nabla_k \psi_T = \nabla_k \left[\prod_i \phi(\boldsymbol{r_i})\right] f(\boldsymbol{r_{jk}}) + \left[\prod_i \phi(\boldsymbol{r_i})\right] \nabla_k f(\boldsymbol{r_{jk}}) \tag{66}$$

Focusing on the first term, which will also be used in the non-interacting case, gives

$$\nabla_k \left[\prod_i \phi(\boldsymbol{r_i})\right] f(\boldsymbol{r_{jk}}) = \nabla_k \phi(\boldsymbol{r_k}) \left[\prod_{i\neq k} \phi(\boldsymbol{r_i})\right] f(\boldsymbol{r_{jk}}). \tag{67}$$

Taking the gradient of the Jastrow factor in the second term gives

$$\begin{aligned}\nabla_k f(\boldsymbol{r_{ij}}) = \nabla_k \sum_{i=1}^{N} \sum_{j=i+1}^{N} u(r_{ij}) &= \nabla_k \left(\sum_{m=k+1}^{N} u(\boldsymbol{r_{km}}) + \sum_{n=1}^{k-1} u(\boldsymbol{r_{nk}}) \right) \exp\left(\sum_{i=1}^{N} \sum_{j=i+1}^{N} u(\boldsymbol{r_{ij}}) \right) \\ &= \nabla_k \left(\sum_{m\neq k} u(\boldsymbol{r_{km}}) \right) \exp\left(\sum_{i=1}^{N} \sum_{j=i+1}^{N} u(\boldsymbol{r_{ij}}) \right) \end{aligned} \tag{68}$$

where the last step comes from the fact that $\boldsymbol{r_{km}} = \boldsymbol{r_{mk}}$ and that $\nabla \sum f(x) = \sum \nabla f(x)$. The complete expression for the gradient of the trial wave function with respect to the k'th particle becomes the vector given by

$$\begin{aligned}\nabla_k \psi_T = &\underbrace{\nabla_k \phi(\boldsymbol{r_k}) \left[\prod_{i\neq k} \phi(\boldsymbol{r_i})\right] \exp\left(\sum_{j<m} u(\boldsymbol{r_{jm}})\right)}_{1} + \\ &\underbrace{\left[\prod_i \phi(\boldsymbol{r_i})\right] \nabla_k \left(\sum_{m\neq k} u(\boldsymbol{r_{km}}) \right) \exp\left(\sum_{j=1}^{N} \sum_{m=i+1}^{N} u(\boldsymbol{r_{jm}}) \right)}_{2} \end{aligned} \tag{69}$$

where the names of some of the indexes are changed. We will do the calculations term by term.

The complicated part is finding the Laplacian of the trial wave function with respect to the k'th particle that is needed in the local energy, $\nabla_k^2 \psi_T = \nabla_k \cdot \nabla_k \psi_T$. Taking the gradient of part 1 in equation 69 and then using the chain rule gives

$$\nabla_k \cdot \left[\nabla_k \phi(\boldsymbol{r_k}) \left[\prod_{i \neq k} \phi(\boldsymbol{r_i}) \right] \exp\left(\sum_{j<m} u(\boldsymbol{r_{jm}}) \right) \right] = \nabla_k \cdot \nabla_k \phi(\boldsymbol{r_k}) \left[\prod_{i \neq k} \phi(\boldsymbol{r_i}) \right] \exp\left(\sum_{j<m} u(\boldsymbol{r_{jm}}) \right) + \\ \nabla_k \phi(\boldsymbol{r_k}) \left[\prod_{i \neq k} \phi(\boldsymbol{r_i}) \right] \nabla_k \left(\sum_{m \neq k} u(\boldsymbol{r_{km}}) \right) \exp\left(\sum_{i<j} u(\boldsymbol{r_{ij}}) \right) \quad (70)$$

The gradient of term 2 in equation 69 can also be found using the product rule, which gives

$$\nabla_k \cdot \left[\left[\prod_i \phi(\boldsymbol{r_i}) \right] \nabla_k \left(\sum_{m \neq k} u(\boldsymbol{r_{km}}) \right) \exp\left(\sum_{j=1}^{N} \sum_{m=i+1}^{N} u(\boldsymbol{r_{jm}}) \right) \right] = \\ \nabla_k \phi(\boldsymbol{r_k}) \left[\prod_{i \neq k} \phi(\boldsymbol{r_i}) \right] \nabla_k \left(\sum_{m \neq k} u(\boldsymbol{r_{km}}) \right) \exp\left(\sum_{i<j} u(\boldsymbol{r_{ij}}) \right) + \\ \prod_i \phi(\boldsymbol{r_i}) \left[\nabla_k \cdot \nabla_k \left(\sum_m u(\boldsymbol{r_{km}}) \right) \right] \exp\left(\sum_{i<j} u(\boldsymbol{r_{ij}} \right) + \\ \prod_i \phi(\boldsymbol{r_i}) \left[\nabla_k \left(\sum_m u(\boldsymbol{r_{km}}) \right) \right]^2 \exp\left(\sum_{i<j} u(\boldsymbol{r_{ij}} \right) \quad (71)$$

Combining the two previous steps gives that

$$\boxed{\begin{aligned} \nabla_k^2 \psi_T = \nabla_k \cdot \nabla_k \psi_T = \nabla_k \cdot \nabla_k \phi(\boldsymbol{r_k}) \left[\prod_{i \neq k} \phi(\boldsymbol{r_i}) \right] \exp\left(\sum_{j<m} u(\boldsymbol{r_{jm}}) \right) + \\ 2 \nabla_k \phi(\boldsymbol{r_k}) \left[\prod_{i \neq k} \phi(\boldsymbol{r_i}) \right] \nabla_k \left(\sum_{m \neq k} u(\boldsymbol{r_{km}}) \right) \exp\left(\sum_{i<j} u(\boldsymbol{r_{ij}}) \right) + \\ \prod_i \phi(\boldsymbol{r_i}) \left[\nabla_k \cdot \nabla_k \left(\sum_m u(\boldsymbol{r_{km}}) \right) \right] \exp\left(\sum_{i<j} u(\boldsymbol{r_{ij}} \right) + \\ \prod_i \phi(\boldsymbol{r_i}) \left[\nabla_k \left(\sum_m u(\boldsymbol{r_{km}}) \right) \right]^2 \exp\left(\sum_{i<j} u(\boldsymbol{r_{ij}} \right) \quad (72) \end{aligned}}$$

The term required in the local energy is

$$\boxed{\frac{1}{\psi_T} \nabla_k^2 \psi_T = \frac{\nabla_k^2 \phi(\boldsymbol{r_k})}{\phi(\boldsymbol{r_k})} + 2 \frac{\nabla_k \phi(\boldsymbol{r_k})}{\phi(\boldsymbol{r_k})} \left(\sum_{j \neq k} \nabla_k u(\boldsymbol{r_{kj}}) \right) + \sum_{m \neq k} \nabla_k^2 u(\boldsymbol{r_{km}}) + \left[\sum_{j \neq k} \nabla_k u(\boldsymbol{r_{kj}}) \right]^2 \quad (73)}$$

after most of the terms cancels out by division with the trial wave function. Noting that

$$\nabla_k u(\boldsymbol{r_{kj}}) = \frac{\partial u(\boldsymbol{r_{kj}})}{\partial \boldsymbol{r_{kj}}} \frac{\partial \boldsymbol{r_{kj}}}{\partial \boldsymbol{r_k}} = u'(\boldsymbol{r_{kj}}) \frac{\boldsymbol{r_k} - \boldsymbol{r_j}}{\boldsymbol{r_{kj}}} \quad (74)$$

by using the chain rule. To find the laplacian it is important to remember that the Nabla operator returns a scalar if it is acting on a vector function, and returns the gradient, which is a vector if it is acting on a scalar function. Using this results in

$$\nabla_k^2 u(\boldsymbol{r_{kj}}) = \nabla_k u'(r_{jk}) \left[\frac{\boldsymbol{r_k} - \boldsymbol{r_j}}{r_{kj}} \right] + u'(r_{kj}) \nabla_k \left[\frac{\boldsymbol{r_k} - \boldsymbol{r_j}}{r_{kj}} \right] = \\ \frac{\partial u'(r_{kj})}{\partial r_{kj}} \frac{\partial r_{kj}}{\partial \boldsymbol{r_k}} \left[\frac{\boldsymbol{r_k} - \boldsymbol{r_j}}{r_{kj}} \right] + u'(r_{kj}) \underbrace{\nabla_k \left[\frac{\boldsymbol{r_k} - \boldsymbol{r_j}}{r_{kj}} \right]}_{1} \quad (75)$$

In 1 it is important to be aware of how the Nabla operator works on scalar and vector functions. Using what was mentioned above and the standard rule for derivatives of division gives

$$\nabla_k \left[\frac{\boldsymbol{r_k} - \boldsymbol{r_j}}{r_{kj}} \right] = \frac{\nabla_k (\boldsymbol{r_k} - \boldsymbol{r_j}) r_{kj} - (\boldsymbol{r_k} - \boldsymbol{r_j}) \frac{\partial r_{kj}}{\partial \boldsymbol{r_j}}}{r_{kj}^2} = \frac{\text{dim} \cdot r_{kj} - (\boldsymbol{r_k} - \boldsymbol{r_j})^2 / r_{kj}}{r_{kj}^2} \tag{76}$$

since $\nabla_k r_{kj} = \frac{\partial r_{kj}}{\partial \boldsymbol{r_k}}$ and dim is the dimension of the system and comes from taking the divergence of the position vector. Rearranging gives that

$$\nabla_k \left[\frac{\boldsymbol{r_k} - \boldsymbol{r_j}}{r_{kj}} \right] = \frac{\text{dim}}{r_{kj}} - \frac{(\boldsymbol{r_k} - \boldsymbol{r_j})^2 / r_{kj}^2}{r_{kj}} = \frac{\text{dim} - 1}{r_{kj}}, \tag{77}$$

where

$$(\boldsymbol{r_k} - \boldsymbol{r_j})^2 / r_{kj}^2 = 1, \tag{78}$$

because of the unit length. The final expression for the laplacian of u is then given by

$$\nabla_k^2 u(\boldsymbol{r_{kj}}) = u''(r_{kj}) + \frac{\text{dim - 1}}{r_{kj}} u'(r_{kj}) \tag{79}$$

Equation 73 can then be rewritten using the new expression for the derivatives of u

$$\boxed{\begin{aligned} \frac{1}{\psi_T} \nabla_k^2 \psi_T = \frac{\nabla_k^2 \phi(\boldsymbol{r_k})}{\phi(\boldsymbol{r_k})} + 2 \frac{\nabla_k \phi(\boldsymbol{r_k})}{\phi(\boldsymbol{r_k})} \left(\sum_{j \neq k} u'(r_{kj}) \frac{\boldsymbol{r_k} - \boldsymbol{r_j}}{r_{kj}} \right) + \\ \sum_{m \neq k} \left[u''(r_{kj}) + \frac{\text{dim - 1}}{r_{kj}} u'(r_{kj}) \right] + \left[\sum_{j \neq k} u'(r_{kj}) \frac{\boldsymbol{r_k} - \boldsymbol{r_j}}{r_{kj}} \right]^2 \quad (80) \end{aligned}}$$

The local energy can now be found by using this complicated expression. The Hamiltonian for the interacting case is given by

$$H = \sum_k^N \left(\frac{-\hbar^2}{2m} \nabla_k^2 + V_{\text{ext}}(\boldsymbol{r_k}) \right) + \sum_{i<j}^N V_{\text{int}}(\boldsymbol{r_i}, \boldsymbol{r_j}). \tag{81}$$

The local energy is then found using the laplacian of the trial wave function and the Hamiltonian

$$\boxed{\begin{aligned} E_L = \frac{1}{\psi_T(\boldsymbol{r})} H \psi_T(\boldsymbol{r}) = \sum_k^N \left(\frac{-\hbar^2}{2m} \frac{\nabla_k^2 \psi_T}{\psi_T} + V_{\text{ext}}(\boldsymbol{r_k}) \right) + \sum_{i<j}^N V_{\text{int}}(\boldsymbol{r_i}, \boldsymbol{r_j}) = \\ \sum_k^N \left(\frac{-\hbar^2}{2m} \left[\frac{\nabla_k^2 \phi(\boldsymbol{r_k})}{\phi(\boldsymbol{r_k})} + 2 \frac{\nabla_k \phi(\boldsymbol{r_k})}{\phi(\boldsymbol{r_k})} \left(\sum_{j \neq k} u'(r_{kj}) \frac{\boldsymbol{r_k} - \boldsymbol{r_j}}{r_{kj}} \right) + \right.\right. \\ \left.\left. \sum_{j \neq k} \left(u''(r_{kj}) + \frac{\text{dim - 1}}{r_{kj}} u'(r_{kj}) \right) + \left(\sum_{j \neq k} u'(r_{kj}) \frac{\boldsymbol{r_k} - \boldsymbol{r_j}}{r_{kj}} \right)^2 \right] + V_{\text{ext}}(\boldsymbol{r_k}) \right) + \\ \sum_{i<j}^N V_{\text{int}}(\boldsymbol{r_i}, \boldsymbol{r_j}) \quad (82) \end{aligned}}$$

where the trial wave function term cancels everywhere expect for in the terms involving the laplacian operator.

6.1.2 Non-interacting case

The laplacian of for the non-interacting system is much easier, and most of the term from the interacting case disappears.

The trial wave function is given by

$$\psi_T(\boldsymbol{r}) = \left[\prod_i \phi(\alpha, \boldsymbol{r_i}) \right] \tag{83}$$

where

$$g(\alpha, \boldsymbol{r_i}) = \exp[-\alpha(x_i^2 + y_i^2 + \beta z_i^2)]. \tag{84}$$

This is in 3D and for the non-interactin case, $\beta = 1$. The Hamiltonian used for the non.interacting case is given by

$$H = \sum_k^N \left(\frac{-\hbar^2}{2m} \nabla_k^2 + V_{\text{ext}}(\boldsymbol{r_k}) \right), \tag{85}$$

where

$$V_{\text{ext}}(\boldsymbol{r}) = \frac{1}{2} m \omega_{\text{ho}}^2 r^2 \tag{86}$$

is the spherical potential trap.

Since the Jastrow factor is abscent the gradient of the trial wave function can be written as

$$\boxed{\nabla_k \psi_T = \nabla_k \left[\prod_i \phi(\boldsymbol{r_i}) \right] = \nabla_k \phi(\boldsymbol{r_k}) \left[\prod_{i \neq k} \phi(\boldsymbol{r_i}) \right]} \tag{87}$$

which gives that

$$\nabla_k^2 \psi_T = \nabla_k^2 \phi(\boldsymbol{r_k}) \left[\prod_{i \neq k} \phi(\boldsymbol{r_i}) \right]. \tag{88}$$

$$\boxed{\frac{1}{\psi_T} \nabla_k^2 \psi_T = \frac{\nabla_k^2 \phi(\boldsymbol{r_k})}{\phi(\boldsymbol{r_k})}} \tag{89}$$

where the product of the ϕ's cancel out when dividing with the trial wave function.

The local energy of the system can then be found using the expression for the laplacian of the trial wave function and the Hamiltonian.

$$\boxed{E_L = \frac{1}{\psi_T} H \psi_T = \sum_k^N \left(\frac{-\hbar^2}{2m} \frac{\nabla_k^2 \phi(\boldsymbol{r_k})}{\phi(\boldsymbol{r_k})} + V_{\text{ext}}(\boldsymbol{r_k}) \right)} \tag{90}$$

For a system with 3 dimensions with $\phi(\boldsymbol{r_k}) = \exp(-\alpha(x_k^2 + y_k^2 + z_k^2))$, the laplacian can be found by first showing that the gradient is

$$\nabla_k \phi(\boldsymbol{r_k}) = -2\alpha [x_k, y_k, z_k]^T \phi(\boldsymbol{r_k}). \tag{91}$$

The laplacian is given by

$$\begin{aligned} \nabla_k \cdot \nabla_k \phi(\boldsymbol{r_k}) = -2\alpha \nabla_k [x_k, y_k, z_k]^T \phi(\boldsymbol{r_k}) - 2\alpha [x_k, y_k, z_k]^T \nabla_k \phi(\boldsymbol{r_k}) = \\ - 2\alpha \cdot 3 \cdot \phi(\boldsymbol{r_k}) + 4\alpha^2 (x_k^2 + y_k^2 + z_k^2) \phi(\boldsymbol{r_k}). \end{aligned} \tag{92}$$

so

$$\frac{\nabla_k^2 \phi(\boldsymbol{r_k})}{\phi(\boldsymbol{r_k})} = -2\alpha \cdot 3 + 4\alpha^2 (x_k^2 + y_k^2 + z_k^2) \tag{93}$$

which can be rewritten to the more general expression for arbitrary number of dimensions

$$\frac{\nabla_k^2 \phi(\boldsymbol{r_k})}{\phi(\boldsymbol{r_k})} = -2\alpha \cdot \text{dim} + 4\alpha^2 (\sum_{m=1}^{\text{dim}} (x_k)_m^2) \tag{94}$$

where $(x_k)_m$ is the m'th coordinate of the k'th particle. From this it follows that

$$\begin{aligned} E_L = \sum_{k=1}^N \left(\frac{-\hbar^2}{2m} \left[4\alpha^2 (\sum_{m=1}^{\text{dim}} (x_k)_m^2) - 2\alpha \cdot \text{dim} \right] + V_{\text{ext}}(\boldsymbol{r_k}) \right) = \\ - 2\alpha \cdot \text{dim} \cdot N + \sum_{k=1}^N \left(\frac{-\hbar^2}{2m} \left[4\alpha^2 (\sum_{m=1}^{\text{dim}} (x_k)_m^2) \right] + V_{\text{ext}}(\boldsymbol{r_k}) \right) \end{aligned} \tag{95}$$

6.2 Analytical expressions for the drift force

In this section the analytical expression for the drift force used in importance sampling is derived. The drift force is given by

$$F = \frac{2\nabla\psi_T}{\psi_T}. \tag{96}$$

6.2.1 Interacting case

Using the expression for the gradient of the trial wave function 69, the drift force can be written as

$$\boxed{F = \frac{\nabla_k \phi(\boldsymbol{r_k})}{\phi(\boldsymbol{r_k})} + \nabla_k \left(\sum_{j \neq k} u(r_{kj}) \right)} \tag{97}$$

after dividing by the trial wave function.

6.2.2 Non-interacting case

The drift force for the non-interacting system is equal to the first part of the drift force for the interacting case

$$\boxed{F = \frac{\nabla_k \phi(\boldsymbol{r_k})}{\phi(\boldsymbol{r_k})}} \tag{98}$$

Inserting the trial wave function for the non-interacting case, we can write out the expression in 3 dimensions. Since $\phi(\boldsymbol{r_k}) = \exp(-\alpha(x_k^2 + y_k^2 + z_k^2))$ the gradient becomes

$$\nabla_k \phi(\boldsymbol{r_k}) = -2\alpha[x_k, y_k, z_k]^T \phi(\boldsymbol{r_k}) \tag{99}$$

so the drift force is given by

$$F = -2\alpha[x_k, y_k, z_k]^T. \tag{100}$$

A similar expression can be found in 1 and 2 dimensions as well.

References

[1] M. Hjorth-Jensen, *Computational Physics, lecture notes 2015*, Oslo, 2015.

[2] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization.* Springer Series in Operating Research and Financial Engineering, 2006.

[3] G. H. Givens, J. A. Hoeting, *Computational Statistics*, Wiley, 2013.

[4] J. K. Nilsen, *Data Blocking [Powerpoint slides]*. Retrived from `https://www.uio.no/studier/emner/matnat/fys/nedlagte-emner/FYS4410/v08/undervisningsmateriale/Material%20for%20Part%20I%20by%20Morten%20HJ/Slides%20from%20Lectures/blocking.pdf`

[5] M. Jonsson, *Standard Error Estimation by an Automated Blocking Method*, Phys. Rev. E **98**, 043304 (2018).

[6] D. Griffitsh, *Introduction to Quantum Mechanics*, Pearson Education, 2004.

[7] M. L. Markova, V. M. Valsdottir *Simulation and investigation of bose einstein condensate using variational Monte Carlo method* (2019)