FYS4411 - Computational Physics II

Project 2

Dorthea Gjestvang
Even Marius Nordhagen

April 18, 2018

- Github repository containing programs and results:
  `https://github.com/evenmn/FYS4411/tree/master/Project%202`

Abstract

Abstract

Contents

1 Introduction

2 Theory

2.1 Presentation of potential and trial wavefunction

We study a system of N bosons trapped in a harmonic oscillator with the Hamiltonian given by

$$\hat{H} = \sum_i^N \Big(-\frac{\hbar^2}{2m}\nabla_i^2 + V_{ext}(\vec{r}_i)\Big) + \sum_{i<j}^N V_{int}(\vec{r}_i, \vec{r}_j) \tag{1}$$

with V_{ext} as the external potential, which is the harmonic oscillator potential defined in equation 2, and V_{int} as the interaction term, defined in equation 3, which ensures the particles to be separated by a distance a. $\hbar$ is the reduced Plancks constant and m is the mass of the particles in question. In this project we study the gas of ^{87}Rb alkali atoms, which have a scattering length $a_{\text{Rb}} = 0.0043$ given in units of a typical trap size a_{ho}. We will consider a harmonic oscillator which can either be spherical, where all dimensions have the same scales and the harmonic oscillator frequency is ω_{ho}, or elliptical, where the z dimension has a different frequency ω_z.

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

$$V_{int}(\vec{r}_i, \vec{r}_j) = \begin{cases} 0 & \text{if} \quad |\vec{r}_i - \vec{r}_j| \geq a \\ \infty & \text{if} \quad |\vec{r}_i - \vec{r}_j| < a \end{cases} \tag{3}$$

The trial wavefunction is on the form

$$\Psi_T(\vec{r}_1, \vec{r}_2, ..., \vec{r}_N, \alpha, \beta) = \prod_i^N g(\alpha, \beta, \vec{r}_i) \prod_{i<j} f(a, r_{ij}) \tag{4}$$

where $r_{ij} = |\vec{r}_i - \vec{r}_j|$ and g is assumed to be a gaussian function,

$$g(\alpha, \beta, \vec{r}_i) = \exp[-\alpha(x_i^2 + y_i^2 + \beta z_i^2)], \tag{5}$$

which is practical since

$$\prod_i^N g(\alpha, \beta, \vec{r}_i) = \exp\Big[-\alpha \sum_{i=1}^N (x_i^2 + y_i^2 + \beta z_i^2)\Big]. \tag{6}$$

α is a variational parameter that we later use to find the energy minimum, and β is a constant. The trial wave function used is the exact ground state wave function for a harmonic oscillator with no interaction, so by choosing the correct α, we will find the exact ground state energy. The α for the ground state wave function in this case is known to be 0.5. It is sufficient to study the ground state wave function only, since all particles occupy this state in a Bose-Einstein condensation.

The f presented above is the correlation wave function, which is

$$f(a, r_{ij}) = \begin{cases} 0 & r_{ij} \leq a \\ \left(1 - \frac{a}{r_{ij}}\right) & r_{ij} > a. \end{cases} \tag{7}$$

where a is a parameter describing the minimum distance allowed between particles in the trap. This is also denoted as the Jastrow factor, and it sets the wave function to zero when the distance between two particles is smaller than a, such that we get a hard-sphere repulsive potential.

2.2 Energy calculation

We want to find the ground state energy of our system for a given wave function Ψ_T. The energy is given as the expectation value of the Hamiltonian, given by equation 8.

$$E = \langle H \rangle = \frac{\int \Psi^*(\vec{r}) H \Psi(\vec{r}) d\vec{r}}{\int \Psi^*(\vec{r}) \Psi(\vec{r}) d\vec{r}} \tag{8}$$

By defining a quantity called the local energy E_L given by equation 9

$$E_L(\vec{r}) = \frac{1}{\Psi_T(\vec{r})} \hat{H} \Psi_T(\vec{r}). \tag{9}$$

the energy can be written as

$$E = \int |\Psi_T^2| E_L d\vec{r} \tag{10}$$

This equation can be solved by calculating the local energy $E_{L,i}$ for each iteration in the Monte Carlo loop, summing them, and then divide by the number of Monte Carlo cycles:

$$E = \frac{\sum_i^M E_{L,i}}{M} \tag{11}$$

We want to calculate the energy E as a function of α for the trial wave function given in equation 4 by using Variational Monte Carlo (VMC) described in section 3.1.

When the repulsive interaction is ignored ($a = 0$), it can be shown that the local energy for a system of N particles and dim spatial dimensions is given by

$$E_L = dim \cdot N \cdot \alpha + \Big(\frac{1}{2} - 2\alpha^2\Big) \sum_i \vec{r}_i^2, \tag{12}$$

which is proven in appendix A. This is only true for the a spherical harmonic oscillator trap, for the elliptical trap we need to add β in front of the z-component, which is also given in appendix A. You may also notice that this equation is scaled, more about that in section 2.4.

For the a spherical harmonic oscillator where particle interactions are ignored, the analytical expression for the energy is well-known and reads $E = \hbar\omega(n + dim/2)$ where n is the energy level and dim is number of spatial dimensions. In this project we will study the ground state only, such that $n = 0$, and for N particles and dim spatial dimensions we therefore obtain the expression for the ground state energy shown in equation 13.

$$E = \frac{1}{2} N \cdot dim \cdot \hbar\omega_{ho}. \tag{13}$$

For $a \neq 0$ it gets rather more complicated, because the Jastrow factor from equation 7 is now different from 1. We also need to add the interaction term V_{int}, the hard-sphere potential from equation 3. We are now ready to find an analytical expression for the local energy. By defining

$$f(a, r_{ij}) = \exp\bigg(\sum_{i<j} u(r_{ij})\bigg) \tag{14}$$

and doing a change of variables

$$\frac{\partial}{\partial \vec{r}_k} = \frac{\partial}{\partial \vec{r}_k}\frac{\partial r_{kj}}{\partial r_{kj}} = \frac{\partial r_{kj}}{\partial \vec{r}_k}\frac{\partial}{\partial r_{kj}} = \frac{(\vec{r}_k - \vec{r}_j)}{r_{kj}}\frac{\partial}{\partial r_{kj}} \tag{15}$$

one will end up with

$$E_L = \sum_k \Bigg(-\frac{1}{2}\Bigg(4\alpha^2\Big(x_k^2 + y_k^2 + \beta^2 z_k^2 - \frac{1}{\alpha} - \frac{\beta}{2\alpha}\Big) - 4\alpha \sum_{j\neq k}(x_k, y_k, \beta z_k)\frac{(\vec{r}_k - \vec{r}_j)}{r_{kj}} u'(r_{kj})$$

$$+ \sum_{ij\neq k} \frac{(\vec{r}_k - \vec{r}_j)(\vec{r}_k - \vec{r}_i)}{r_{ki} r_{kj}} u'(r_{ki}) u'(r_{kj}) + \sum_{j\neq k}\Big(u''(r_{kj}) + \frac{2}{r_{kj}} u'(r_{kj})\Big)\Bigg) + V_{ext}(\vec{r}_k)\Bigg) + V_{int}.$$

which is also shown in appendix A. This is not a pretty expression, but will yield correct results for both the interacting and non-inteacting case.

2.2.1 Numerical calculation of E_L

Another approach when calculating E_L is to split up the local energy expression as shown in equation 16, and calculate the local energy with a numerical approach where the second derivative needed to find the kinetic energy can be approximated by the three-point formula, see equation 17.

$$E_{L,i} = -\frac{\hbar^2}{2m}\frac{\nabla_i^2 \Psi_T}{\Psi_T} + V_{ext}(\vec{r}_i) = E_{k,i} + E_{p,i} \tag{16}$$

$$f''(x) \simeq \frac{f(x+h) - 2f(x) + f(x-h)}{h^2}. \tag{17}$$

In our case the position is a three dimensional vector, so we need to handle each dimension separately. Both the analytical and the numerical local energies are implemented, and in section **??**, the CPU time for the analytical and numerical approach are compared for a various number of particles.

2.2.2 Gross-Pitaevskii equation

Since we study a dilute bose gas, we can use the Gross-Pitaevskii (GP) equation to estimate the energy density, also when including particle-particle interaction [9], [10]. The energy is then given by an integral over all the positions [3] as follows:

$$E_{\mathrm{GP}}[\Psi] = \int d\vec{r}\left[\frac{\hbar^2}{2m}|\nabla\Psi(\vec{r})|^2 + V_{ext}(\vec{r})|\Psi|^2 + \frac{2\pi\hbar^2 a}{m}|\Psi|^4\right]. \tag{18}$$

This integral can easily be solved analytically when we ignore the interaction, and for three dimentions we get

$$E_{\mathrm{GP}}(\alpha, \beta, N) = N \cdot \left(1 + \frac{\beta}{2}\right)\left(\frac{1}{4\alpha} + \alpha\right) \tag{19}$$

which is derived in appendix D.

This expression is valid for few particles and small hard-sphere diameter. As the average energy per particle increases as we increase N, the expression is no longer valid for a higher number of particles. For the interacting case, we could solve the integral using a numerical method, for instance Monte Carlo integration. We will not implement this, but rather benchmark our results against the solution of GP when ignoring interaction.

2.3 Onebody density

In many cases it is convenient to know the positions of the particles, but when the number of particles increases, the set of positions turns into a messy collection of numbers which is not really informative. Instead of presenting the positions, the density of particles can give us a good overview of where the particles are located. With N particles, the one-body density with respect to a particle i is an integral over all particles but particle i:

$$\rho_i = \int_{-\infty}^{\infty} d\vec{r}_1 \dots d\vec{r}_{i-1} d\vec{r}_{i+1} \dots d\vec{r}_N |\Psi(\vec{r}_1, \dots \vec{r}_N)|^2. \tag{20}$$

For the non-interacting case this integral can be solved analytically, as shown in equation 21. Using Monte Carlo integration, the one-body density can be solved for any case. Anyhow, the interesting part is the radial density, so we either have to solve the integral in spherical coordinates or convert to spherical coordinates afterwards.

Alternatively, the onebody radial density can be found in a more intuitive way. Imagine we divide the volume around particle i into bins, where bin j is located at a distance $j \cdot r_1$, as shown in figure 1. The radii are thus quantized. By counting the number of particles in a bin and dividing by the surface area, we find the average density of particles in the bin. If we decrease the initial radius r_1 of the innermost bin such that we have a large number of bins, this method can be used to find the onebody density.

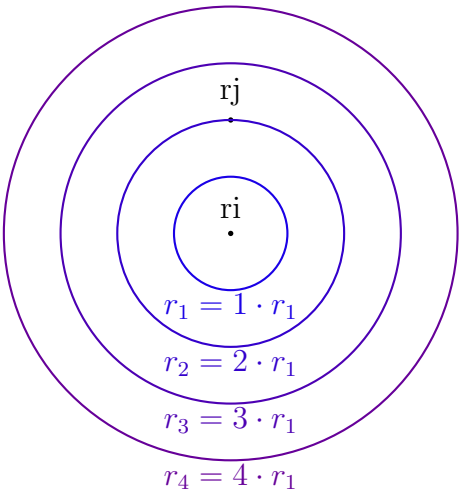

Figure 1: One can find the onebody density by dividing the volume around a particle with coordinates $\vec{r}_i$ into bins and then count the number of particles, here illustrated in two dimensions.

The analytical expression without the Jastrow factor is found from [2], and can be modified for our particular system, presented for the three dimensional case in equation 21.

$$\rho_i(x_i, y_i, z_i) = \left(\frac{2\alpha}{\pi}\right)^{3/2} \beta^{1/2} e^{-2\alpha(x_i^2+y_i^2+\beta z_i^2)}. \tag{21}$$

Since the particles are identical the choice of the particle i is of no consequence.

2.4 Scaling

For big numerical projects, working with dimensionless quantities is a great advantage. Not only does it improve the code structure and performance, but it also avoids truncation errors due to small constants. For this project a natural scaling parameter for the energy is $\hbar\omega_{ho}$, which appears in the analytical energy expression in equation 13. The equivalent dimensionless equation can then be written as

$$E' = \frac{N \cdot dim}{2} \tag{22}$$

where $E' = E/\hbar\omega_{ho}$. Additionally, we can scale the position with respect to the length of the spherical trap, a_{ho}, such that

$$r_i' = \frac{r_i}{a_{ho}} = r_i \cdot \sqrt{\frac{m\omega_{ho}}{\hbar}}, \tag{23}$$

and the Hamiltonian turns into

$$H = \frac{1}{2}\sum_i \Big(-\nabla^2 + \vec{r}_i^{\,2}\Big) + \sum_{i<j} V_{int}(\vec{r}_i, \vec{r}_j) \tag{24}$$

A watchful eye will see that this corresponds to setting $\hbar = \omega_{ho} = m = 1$, which is the natural units.

For the spherical trap situation we are left with the variational parameters α and β only, but when we study an elliptical trap we still want to get rid of ω_z. Since β^2 should be the factor in front of the z-coordinate when the Hamiltonian is dimensionless, it can be proven that $\beta = \omega_z/\omega_{ho}$, see appendix C. We end up with the Hamiltonian

$$H = \frac{1}{2}\sum_i \Big(-\nabla^2 + x_i^2 + y_i^2 + \beta^2 z_i^2\Big) + \sum_{i<j} V_{int}(\vec{r}_i, \vec{r}_j) \tag{25}$$

where β is chosen to be $\sqrt{8} \cong 2.82843$, which was used in the experiment of Anderson et.al. [7].

2.5 Error estimation

When presenting data from an experiment, one should always know the errors in the answer. Experimental data, including data from numerical experiments, are never determined beyond any doubt, and an estimate of this error should therefore be presented alongside the data.

There are two kinds of errors. Statistical errors originate from how much statistics one has; when 10^6 measured points give approximately the same answer, one can be more sure that the actual value is close to those points, more so than if one only has 1 data point. Estimating the statistical error is easily done. The systematic error, however, is harder to handle. It arises for example from calculations being based on faulty theory, or defect measurement devices. Here, we will present how to get an estimate of the statistical error in a numerical experiment.

When conducting an experiment θ with n measured points, $\{x_1, ..., x_n\}$, the sample mean $\langle x_\theta \rangle$ of the experiment is defined as shown in equation 26.

$$\langle x_\theta \rangle = \frac{1}{n} \sum_{k=1}^{n} x_{\theta,k} \tag{26}$$

The corresponding sample variance σ_θ is then defined as

$$\sigma_\theta^2 = \frac{1}{n} \sum_{k=1}^{n} (x_{\theta,k} - \langle x_\theta \rangle)^2 \tag{27}$$

This gives us the error in the given experiment θ. If we repeat this experiment m times, the mean after all the experiments are

$$\langle x_m \rangle = \frac{1}{n} \sum_{k=1}^{n} \langle x_\theta \rangle. \tag{28}$$

The total variance is then

$$\sigma_m^2 = \frac{1}{m} \sum_{\theta=1}^{m} (\langle x_\theta \rangle - \langle x_m \rangle)^2. \tag{29}$$

This can be reduced to

$$\sigma_m^2 = \frac{\sigma^2}{n} + \text{covariance term}, \tag{30}$$

where σ is the sample variance over all the experiments, defined as

$$\sigma^2 = \frac{1}{mn}\sum_{\theta=1}^{m}\sum_{k=1}^{n}(x_{\theta,k} - \langle x_m \rangle)^2 \tag{31}$$

and the covariance is the linear correlation between the measured points. The definition of the covariance is shown in equation 32.

$$cov(x, y) = \frac{1}{n^2}\sum_{i}\sum_{j>i}(x_i - x_j)(y_i - y_j). \tag{32}$$

A common simplification is to reduce equation 30 to the following:

$$\sigma^2 \approx \langle x^2 \rangle - \langle x \rangle^2 \tag{33}$$

This equation, however, does not take into account the covariance term from equation 30, and as the covariance term is added to the expression for the variance, equation 33 will underestimate the uncertainty σ for positive covariances.

A direct implementation of equation 30 including the covariance term is not suitable, as the expression for the covariance includes a double sum, and for a large number of iterations, this will turn into an extremely time consuming process for a large number of Monte Carlo iterations. Luckily, there are methods for calculating an accurate estimation of the variance without including a double loop in the Monte Carlo program. One of these methods is the blocking method, which is presented in section 3.4.

3 Method

3.1 Variational Monte Carlo

Variational Monte Carlo (VMC) is a widely used method for approximating the ground state of a quantum system. The method is based on Markov chains, where one particle or a set of particles are moved one step for each cycle, i.e.

$$\vec{R}_{new} = \vec{R} + r \cdot \text{step}. \tag{34}$$

Both the direction and particles moves are randomly chosen, so with a plain VMC implementation the particles will move at random, independent from each other. We are going to use the Metropolis algorithm in addition to the VMC, where the Metropolis algorithm accepts or rejects moves based on the probability ratio between the old and the new position. This makes the system approach the most likely state, and the idea is that after a certain number of cycles the system will be in the most likely state.

3.2 Metropolis Algorithm

As mentioned above, the task of the Metropolis algorithm is to move the system towards the most likely state. The standard algorithm, here named brute force, is the simplest one, and does not deal with the transition probabilities. The modified Metropolis-Hastings algorithm includes, on the other hand, the transition probabilities and will be slightly more time consuming per cycle. We expect the latter to converge faster to the most likely state.

The foundation of the Metropolis algorithm is that the probability for a system to undergo a transition from state i to state j is given by the transition probability multiplied by the acceptance probability

$$W_{i\to j} = T_{i\to j} \cdot A_{i\to j} \tag{35}$$

where $T_{i\to j}$ is the transition probability and $A_{i\to j}$ is the acceptance probability. Built on this, the probability for being in a state i at time (step) n is

$$P_i^{(n)} = \sum_j \left[P_j^{(n-1)} T_{j\to i} A_{j\to i} + P_i^{(n-1)} T_{i\to j}(1 - A_{i\to j}) \right] \tag{36}$$

since this can happen in two different ways. One can start in this state i at time $n-1$ and be rejected or one can start in another state j at time $n-1$ and complete an accepted move to state i. In fact $\sum_j T_{i\to j} = 1$, so we can rewrite this as

$$P_i^{(n)} = P_i^{(n-1)} + \sum_j \left[P_j^{(n-1)} T_{j\to i} A_{j\to i} - P_i^{(n-1)} T_{i\to j} A_{i\to j} \right]. \tag{37}$$

When the times goes to infinity, the system approaches the most likely state and we will have $P_i^{(n)} = p_i$, which requires

$$\sum_j \left[p_j T_{j\to i} A_{j\to i} - p_i T_{i\to j} A_{i\to j} \right] = 0. \tag{38}$$

Rearranging, we obtain the useful result

$$\frac{A_{j\to i}}{A_{i\to j}} = \frac{p_i T_{i\to j}}{p_j T_{j\to i}} \tag{39}$$

which will be used in the acceptance criteria.

3.2.1 Brute force

In the brute force Metropolis algorithm we want to check if the new position is more likely than the current position, and for that we calculate the probabilities $P(\vec{R}) = |\Psi_T(\vec{R})|^2$ for both positions. We get rid off the transition probabilities setting $T_{i\to j} = T_{j\to i}$, and then end up with the plain ratio

$$w = \frac{P(\vec{R}_{new})}{P(\vec{R})} = \frac{|\Psi_T(\vec{R}_{new})|^2}{|\Psi_T(\vec{R})|^2}. \tag{40}$$

w will be larger than one if the new position is more likely than the current, and smaller than one if the current position is more likely than the new one. Metropolis handle this by accepting the move if the ratio w is larger than a random number r in the interval $[0, 1]$, and rejecting if not:

$$\text{New position: } \begin{cases} \text{accept} & \text{if} \quad w > r \\ \text{reject} & \text{if} \quad w \leq r. \end{cases} \tag{41}$$

3.2.2 Importance sampling

The importance sampling technique is often refered to as Metropolis-Hastings algorithm. The approach is the same as for the brute force Metropolis algorithm, but we will end ut with a slightly more complicated acceptation criteria. To understand the details, we need to begin with the Fokker-Planck equation, which describes the time-evolution of the probability density function $P(R, t)$. In one dimension it reads

$$\frac{\partial P(R,t)}{\partial t} = D\frac{\partial}{\partial R}\left(\frac{\partial}{\partial R} - F\right)P(R,t). \tag{42}$$

where F is the drift force given by equation 43 and D is the diffusion coefficient, in this case equal to 0.5. Calculations of the analytical expression of the drift force F for a spherical harmonic oscillator and $a = 0$ can be found in appendix B.

$$F(R) = \frac{2\nabla\psi_T}{\psi_T} \tag{43}$$

Even though the probability density function can give a lot of useful information, an equation describing the motion of such a particle would be more appropriate for our purposes. Fortunately this equation exists, and satisfies the Fokker-Planck equation. The Langevin equation can be written as

$$\frac{\partial R(t)}{\partial t} = DF(R(t)) + \eta \tag{44}$$

where η can be considered as a random variable. This differential equation can be solved by applying the forward Euler method and introducing gaussian variables ξ

$$R_{new} = R + DF(R)\Delta t + \xi\sqrt{\Delta t} \tag{45}$$

which will be used to update the position. This is an improved way of choosing the direction in which the particle is moved compared to the brute force algorithm, as the drift force $F(R)$ says something about which direction the particle is pushed in, and the choice of the new proposed position is thus dependent on this.

Moreover we also need to update the acceptance criteria since we no longer ignore the transition probabilities. With the Fokker-Planck equation as base, the transition probabilities are given by Green's function

$$\begin{aligned} G(R_{new}, R, \Delta t) &= \frac{1}{(4\pi D\Delta t)^{3N/2}} \exp[-(R_{new} - R - D\Delta t F(R))^2/4D\Delta t] \\ &= T_{R\to R_{new}} \end{aligned} \tag{46}$$

and the acceptance criteria becomes

$$r < \frac{G(R, R_{new}, \Delta t)|\Psi_T(R_{new})|^2}{G(R_{new}, R, \Delta t)|\Psi_T(R)|^2}. \tag{47}$$

3.3 Minimization methods

When the interaction term is excluded, we know which α that corresponds to the energy minimum, and it is in principle no need to try different α's. However, sometimes we have no idea where to search for the minimum point, and we need to try various α values to determine the lowest energy. If we do not know where to start searching, this can be a time consuming activity. Would it not be nice if the program could do this for us?

In fact there are multiple techniques for doing this, where the most complicated ones obviously also are the best. Anyway, in this project we will have good initial guesses, and are therefore not in need for the most fancy algorithms.

3.3.1 Gradient descent

Perhaps the simplest and most intuitive method for finding the minimum is the gradient descent method (GD), which reads

$$\alpha^+ = \alpha - \eta \cdot \frac{d\langle E(\alpha)\rangle}{d\alpha}. \tag{48}$$

where α^+ is the updated α and η is a step size. The idea is that one finds the gradient of the energy with respect to a certain α, and moves in the direction which minimizes the energy. This is repeated until one has found an energy minimum, where the energy minimum is defined as either where $\frac{d\langle E(\alpha)\rangle}{d\alpha}$ is smaller than a given tolerance, or the energy fluctuates around a value are smaller than a tolerance, and thus changes minimally.

To implement equation 48, we need an expression for the derivative of E with respect to alpha:

$$\bar{E_\alpha} = \frac{d\langle E(\alpha)\rangle}{d\alpha}. \tag{49}$$

By using the expression for the expectation value for the energy $\langle E(\alpha)\rangle$ in equation 50

$$\langle E(\alpha)\rangle = \frac{\langle \psi_T(\alpha)|H|\psi_T(\alpha)\rangle}{\langle \psi_T(\alpha)|\psi_T(\alpha)\rangle} \tag{50}$$

and applying the chain rule of differentiation, it can be shown that equation 49 is equal to equation 51

$$\bar{E_\alpha} = 2\left[\langle E_L(\alpha)\frac{\bar{\psi_\alpha}}{\psi_\alpha}\rangle - \langle E_L(\alpha)\rangle\langle\frac{\bar{\psi_\alpha}}{\psi_\alpha}\rangle\right] \tag{51}$$

where

$$\bar{\psi_\alpha} = \frac{d\psi(\alpha)}{d\alpha}. \tag{52}$$

The algorithm of this minimization method is thus as follows:

```
for(max number of iterations with minimizing)

        do M Monte Carlo cycles
        calculate E and dE/dalpha

        Check if dE/dalpha < eps or alpha fluctuation over the last 5 steps
            ↪ is < eps


                if yes, print optimal alpha and break loop
                if no, continue to next iteration
```

3.4 Blocking method

As described in section 2.5, we need a method to give a proper estimation of the variance σ^2 of the points in our experiment, preferably without cal-

culating the double loop from the expression of the covariance in equation 32.

One method that can be used, is the blocking method, which is quite fast and can handle large data sets. Say that we have a data set $\{x_1, ..., x_i, ..., x_n\}$ from an experiment, which in our case will be the estimations of the local energies for each Monte Carlo cycle. The mean of this data set is m, and we want to estimate the variance of this data set, $\sigma^2(m)$, including the covariance.

The variance is defined as

$$\sigma^2 = \frac{1}{n-1}\sum_i (x_i - m) \tag{53}$$

However, this does not include the covariance, as the points x_i are correlated.

If we transform the data set $\{x_1, ..., x_i, ..., x_n\}$ by taking the mean of two neighbouring points in the following way

$$x_i' = \frac{1}{2}(x_{2i} + x_{2i+1}) \tag{54}$$

the number of points in the transformed data set $n' = \frac{1}{2}n$. Each pair of points x_i' is referred to as a block, and using the variance formula above, the variance within each block is calculated. An estimate of the total variance $\hat{\sigma}$ is then

$$\hat{\sigma} = \frac{\sigma_1 + ... + \sigma_{n'}}{n'} \tag{55}$$

By repeating this procedure several times, and each time using more measurements for each block, the estimated standard deviation $\hat{\sigma}$ will grow, and eventually reach a plateau. When the estimated standard deviation $\hat{\sigma}$ flats out, it means that the blocks are no longer correlated, and thus the covariance which bugged us in equation 30, is now 0. The covariance is therefore accounted for, and the $\hat{\sigma}$ is now a good estimation of the error in the data set.

4 Code

The focus of this project is developing a code to make VMC run correctly, and in this section we will explain briefly how the code works and which implementation techniques we have used. All quantities used in the implementation are dimensionless.

4.1 Code structure

To keep the program neat, specific parts were placed in specific files. The file wavefunction.cpp contains the class Wavefunction, which both sets up the wave function and calculate the local energy given a wave function. The file metropolis.cpp contains the Variational Monte Carlo loop, and the file gd.cpp handles the calculation of the optimal variational parameter α. We also collected the tools that are used by multiple scripts in its own file to maximize the reuse of the code, in particular Green's function and the quantum force calculations. The user only has to deal with main.cpp, as all choices of parameters are to be specified here. In figure 2 we give an overview of the structure, where the lines indicate which scripts that communicate.

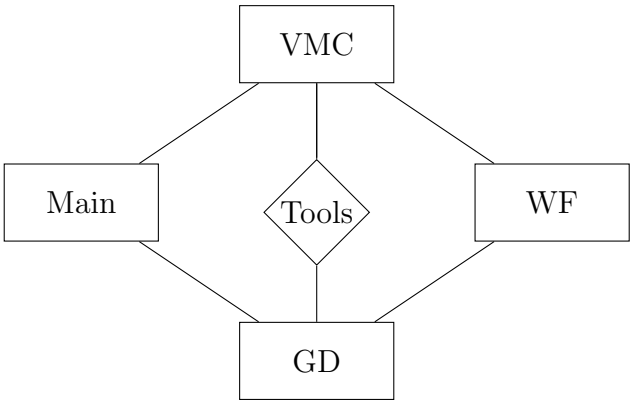

Figure 2: An overview of the code structure. metropolis.cpp is refered to as VMC, gd.cpp is GD, wavefunction.cpp is WF, tools.cpp is Tools and obviously main.cpp is Main. See text for furter description.

A standard VMC run goes as follows: the desired simulation case is set up in main.cpp (Main) by specifying the given variables. metropolis.cpp (VMC) is called from Main, which calls wavefunction.cpp (WF) to set up the class WaveFunction. The Monte Carlo loop begins, where the move of one random particle is proposed for each iteration. Depending of whether brute force or importance sampling have been chosen, either WF or tools.cpp (Tools) is called to obtain the transition probability, and the move is either accepted or rejected. VMC calls WF to obtain the local energy of the new position. After M iterations, the estimation of the energy is printed.

When running the gradient decent method, then Main calls gd.cpp (GD). A start value for α is given in the program, and the Monte Carlo loop is conducted as described above. After the loop, the estimated E-value is printed. Based on the parameter $\frac{dE}{d\alpha}$, a new value for α is chosen. After several MC

loops for different α, the optimal α is chosen.

4.1.1 Test

To make sure the program behaves as expected, we have implemented tests. All tests are functions, and can be found in the test.cpp file.

The test "test EL" checks the output estimated energy after the Monte Carlo loops for a given α in the no interaction case, and compares it to the analytical energy obtrained from the Gross-Pitaevskii equation from equation 19. If the difference between the two are larger than a given tolerance, an error is raised.

4.2 Implementation

We use the vector package to create a 2D-array where the positions are stored. For the exact implementation, see the github repository linked on page one. We always struggle for better performance, and to achieve that we try to avoid heavy loops and repetition of calculations. A second priority is to make the code as general as possible, such that the code can be reused for other purposes. As an example we always loop over an arbitrary given number of dimensions, and similar operations are done throughout the programs.

4.2.1 VMC

```
Set up position matrix of N paricles in dim dimentions
Initialize class WaveFunction

for M Monte Carlo iterations
        move random particle
        accept or reject move based on transition probability
        calculate energy of new position
        update sum of E_L

print estimated E
```

4.2.2 GD

```
first choice of alpha, number of gradientdecent-iteartions T and tolerance
    ↪ is specified

for T iterations
        for M Monte Carlo cycles
                move random particle
                accept or reject move based on transition probability
                calculate energy of new position
                update sum of E_L and dE/dalpha
```

```
        print E and dE/dalpha

        test if optimal alpha -> if optimal, break loop and print optimal
            ↪ alpha

        based on dE/d alpha, suggest new alpha
```

5 Results

6 Discussion

7 Conclusion

8 Appendix A

9 References

[1] Morten Hjorth-Jensen. Computational Physics 2: Variational Monte Carlo methods, Lecture Notes Spring 2018. Department of Physics, University of Oslo, (2018).

[2] J. L. DuBois and H. R. Glyde, H. R., *Bose-Einstein condensation in trapped bosons: A variational Monte Carlo analysis*, Phys. Rev. A **63**, 023602 (2001).

[3] J. K. Nilsen, J. Mur-Petit, M. Guilleumas, M. Hjorth-Jensen, and A. Polls, *Vortices in atomic Bose-Einstein condensates in the large-gas-parameter region*, Phys. Rev. A **71**, 053610, (2005).

[4] F. Dalfovo, S. Stringari, *Bosons in anisotropic traps: ground state and vortices* Phys. Rev. A **53**, 2477, (1996).

[5] J. Emspak, *States of Matter: Bose-Einstein Condensate*, LiveScience, (2016). `https://www.livescience.com/54667-bose-einstein-condensate.html` Downloaded March 15th 2018.

[6] S. Perkowitz *Bose-Einstein condensate* Encyclopaedia Britannica `https://www.britannica.com/science/Bose-Einstein-condensate` Downloaded March 15th 2018.

[7] M. H. Anderson, J. R. Ensher, M. R. Matthews, C. E. Wieman, E. A. Cornell, *Observation of Bose-Einstein Condensation in a Dilute Atomic Vapor*, Science **269**, (1995).

[8] J. K. Nilsen *Bose-Einstein condensation in trapped bosons: A quantum Monte Carlo analysis*, Master thesis 2004, Department of Physics, University of Oslo, (2004).

[9] E. P. Gross, *Structure of a quantized vortex in boson systems*, Il Nuovo Cimento, **20** (3): 454–457, (1961).

[10] L. P. Pitaevskii, *Vortex lines in an imperfect Bose gas*, Sov. Phys. JETP. **13** (2): 451–454, (1961).