Project1 Fys4411

Håkon Emil Kristiansen and Stian Goplen

March 19, 2016

Contents

The source code associated with this project is found at the following github adress: https://github.com/hakii27/Prosjekt1

Abstract

We have written a Variational Monte Carlo program that employs the Metropolis-Hastings- or the importance sampled Metropolis-Hastings algorithm to evaluate the ground state energy of a quantum mechanical system. As an ansatz for the real wave function a trial wave function with one variational parameter is used. In the process the implementation has been thoroughly tested against a special case with known analytical results and is shown to agree perfectly in this case. We present results on the ground state energy of a trapped, hard sphere Bose gas in three dimensions with $N = \{10, 50, 100\}$ particles. These results are referenced against a previous study with good agreement for $N = 10$. However for $N = 100$ the result deviates on the order of $\sim 3\%$. In order to minimize the energy we have implemented the steepest descent method which is shown to work correctly under certain criteria. Blocking is used to achieve a better estimate of the error in the computation of the ground-state energies. Finally we have computed the one-body density for $N = 10$ and $N = 50$ which shows expected characteristics of the system.

1 Introduction

The aim of this project is to write an object-oriented program that employs the Variational Monte Carlo (VMC) method to evaluate the ground state energy of a Quantum many-body system. The ground state energy is given by

$$E_0 = \langle \hat{H} \rangle = \int \Psi^* \hat{H} \Psi d\mathbf{R}, \tag{1}$$

where $\hat{H}$ is the hamiltonian of the system under consideration and Ψ is the wavefunction of the ground state. The main idea of VMC for quantum systems is to use a trial wave function, $\Psi_T(\mathbf{R}; \alpha_1, ..., \alpha_n)$, where the set of parameters $\{\alpha_1, ..., \alpha_n\}$ are so-called variational parameters and minimize E_0 with respect to this set of parameters. This strategy is made possible by the variational principle. In this project we will only consider problems with one variational parameter α. Specifically we will study a Bose gas trapped in a harmonic oscillator potential.

In order to evaluate the integral we will use Monte Carlo integration combined with either the Metropolis-Hastings or importance sampled Metropolis-Hastings algorithm. For analysis of the numerical data we will use the technique of blocking. The method of Steepest descent will be implemented in order to obtain the best possible parameter α. The programming language of choice is C++ since it is a highly efficient, low level language which supports object-orientation.

2 Variational Monte Carlo

2.1 The variational principle

The starting point of VMC is the variational principle which states that [1],

$$E_0 \leq \langle \Phi | \hat{H} | \Phi \rangle \tag{2}$$

where Φ is any normalized function. This implies that if we choose some trial wavefunction $\Psi_T(\mathbf{R}; \boldsymbol{\alpha})$ (presumably incorret) where $\boldsymbol{\alpha} = \{\alpha_1, ..., \alpha_n\}$ is a set of variational parameters and computes $\langle \Psi_T | \hat{H} | \Psi_T \rangle$ we have an upper bound on E_0. If we then minimize $\Psi_T(\mathbf{R}; \boldsymbol{\alpha})$ with respect to $\boldsymbol{\alpha}$ we have the smallest upper bound we can get with a specific choice of a trial wave function. Since $\langle \Psi_T | \hat{H} | \Psi_T \rangle$ is an expectation value, we can also compute the variance,

$$\sigma_H^2 = \langle \Psi_T | \hat{H}^2 | \Psi_T \rangle - (\langle \Psi_T | \hat{H} | \Psi_T \rangle)^2 \tag{3}$$

[1] David J. Griffiths, Introduction to Quantum Mechanics 2nd edition, Pearson - Prentice Hall, New Jersey (2005), Chapter7 page 293.

If you should happen to stumble upon a Ψ_T whith $\sigma_H^2 = 0$ this would imply that we have the correct wave function for the ground state.

However, the success of VMC depends heavily on the choice of a good trial wave functions which is not a trivial matter. One also have to compute a dN-dimensional integral where d is the the number of spatial dimensions and N the number of particles in the system. Consider a system with $N = 10$ and $d = 3$ then we have to compute

$$\langle \Psi_T | \hat{H} | \Psi_T \rangle = \int \Psi_T^* \hat{H} \Psi_T dx_1 dy_1 dz_1 ... dx_{10} dy_{10} dz_{10} \tag{4}$$

which is an integral in 30-dimensions. Hence, armed with the variational principle we need a method for evaluating such integrals numerically. For this we turn to Monte Carlo integration and the Metropolis-Hastings algorithm.

We also need a way of minimizing the energy once we have chosen a trial wave function. For this we have chosen the method of Steepest descent which is explained in more detail in section 2.5

2.2 Monte Carlo integration

Consider the integral,

$$I = \int_0^1 f(x) dx \tag{5}$$

and let

$$\langle f \rangle = \frac{1}{N} \sum_{i=1}^{N} f(x_i) p(x_i) \tag{6}$$

where $p(x)$ is a probability distribution function (PDF). Hence $\langle f \rangle$ is the expectation value of f with respect to $p(x)$. Note that for each i we call $f(x_i)$ a sample of f and likewise for p. Then the main idea of Monte Carlo integration (MCi) is to approximate the integral with the expectation value,

$$I = \int_0^1 f(x) dx \approx \langle f \rangle. \tag{7}$$

Another important quantity when performing MCi is the variance,

$$\sigma_f^2 = \langle f^2 \rangle - \langle f \rangle^2 \tag{8}$$

which is a measure of the extent to which f deviates from its average over the region of integration. The aim of MCi is to have σ_f^2 as small as possible after N samples. A more detailed discussion can be found in Ref. [2]

If we now could rewrite $\langle \Psi_T | \hat{H} | \Psi_T \rangle$ in such a way that it contains some PDF then we can use MCi. We know that the interpretation of the wave function in quantum mechanics is that $|\Psi(\mathbf{R})|^2$ is the probability of finding your system in configuration $\mathbf{R}$. We can then rewrite our original integral as follows,

$$\begin{aligned}
\langle \Psi_T | \hat{H} | \Psi_T \rangle &= \int d\mathbf{R} \Psi_T(\mathbf{R})^* \hat{H} \Psi_T(\mathbf{R}) \\
&= \int d\mathbf{R} \Psi_T^* \left(\frac{\Psi_T}{\Psi_T} \right) \hat{H} \Psi_T \\
&= \int d\mathbf{R} |\Psi_T|^2 \frac{1}{\Psi_T} (\hat{H} \Psi_T) \\
&= \int d\mathbf{R} P(\mathbf{R}) E_L(\mathbf{R}) \\
&\approx \frac{1}{N} \sum_{i=1}^{N} E_L(\mathbf{R}_i),
\end{aligned}$$

[2] Morten Hjorth-Jensen, Computational Physics Lecture Notes, Oslo (2015), Section 11.1.2.

where $E_L(\mathbf{R}) = \frac{1}{\Psi_T}(\hat{H}\Psi_T)$ is referred to as the local energy and $P(\mathbf{R}) = |\Psi_T(\mathbf{R})|^2$.

If we now could sample $\mathbf{R}$ according to $P(\mathbf{R})$ we can approximate the integral as a sum over the local energies. This is where we need the Metropolis-Hastings algorithm.

2.3 Metropolis-Hastings algorithm

The purpose of the Metropolis algorithm is to generate a collection of states according to a desired distribution $P(\mathbf{R})$. The algorithm uses a markov process, which is uniquely defined by its transition probabilities, $P_{i\to j}$, to reach a unique stationary distribution $\pi(\mathbf{R})$ such that $\pi(\mathbf{R}) = P(\mathbf{R})$. But for $\pi(\mathbf{R})$ to be a unique stationary distribution two conditions must be met. To show the existence of the stationary distribution we use the condition of detailed balance. It requires that each transition $i \to j$ is reversible

$$\pi_i P_{i\to j} = \pi_j P_{j\to i} \tag{9}$$

where π_i is the probability of being in state i and $P_{i\to j}$ is the probability of the transition from state i to j. The uniqueness of the stationary distribution is guaranteed by the ergodicity of the markov process. This means that the the process must be aperiodic and positive recurring.

We need to design a markov process that fulfills these conditions such that $\pi(\mathbf{R}) = P(\mathbf{R})$ and we start off with detailed balance

$$\pi_i P_{i\to j} = \pi_j P_{j\to i} \tag{10}$$

which can be written as

$$\frac{P_{i\to j}}{P_{j\to i}} = \frac{\pi_j}{\pi_i}. \tag{11}$$

What we will do now is to seperate the transition into two steps. The acceptance distribution $A_{i\to j}$ is the probability of accepting state j given i and the proposal distribution $T_{i\to j}$ is the probability of proposing state j given i. Meaning that we can write

$$P_{i\to j} = T_{i\to j} A_{i\to j}. \tag{12}$$

We insert this into equation 11

$$\frac{A_{i\to j}}{A_{j\to i}} = \frac{\pi_j}{\pi_i}\frac{T_{j\to i}}{T_{i\to j}}. \tag{13}$$

We want to choose A and T such that no matter what $\pi(\mathbf{R})$ we choose it will converge to $P(\mathbf{R})$. In principle this means that the states we draw are from the distribution $P(\mathbf{R})$.

We now want to choose an acceptance such that this criteria is fulfilled and we will use the metropolis choice

$$A_{i\to j} = \min\left\{1, \frac{P_j T_{j\to i}}{P_i T_{i\to j}}\right\}. \tag{14}$$

In our case the last term takes the form,

$$\frac{P_j T_{j\to i}}{P_i T_{i\to j}} = \frac{|\Psi_T(\mathbf{y})|^2}{|\Psi_T(\mathbf{x})|^2}. \tag{15}$$

I.e we will make an initial state for the system and pick a new state according to $T_{i\to j}$. If the state is accepted according to $A_{i\to j}$ the transition takes place and the system is updated. If its not accepted we draw a new state.

2.4 Importance sampled Metropolis-Hastings

The main idea of importance sampling is to improve upon the way one proposes new states. The solution of the so-called Langevin equation

$$\mathbf{y} = \mathbf{x} + D\mathbf{F}(\mathbf{x})\Delta t + \boldsymbol{\xi}\sqrt{\Delta t}, \tag{16}$$

is used to propose a new state. $\mathbf{x}$ is the current state, $D = 1/2$ in atomic units, $\boldsymbol{\xi}$ is a vector of random numbers and Δt a chosen time step. $\mathbf{F}$ is known as the quantum force which is given by

$$\mathbf{F} = 2\frac{1}{\Psi_T}\nabla\Psi_T. \tag{17}$$

This force makes it a more efficient method than the brute force monte carlo since the gradient $\nabla\Psi_T$ push the walkers towards regions of configuration space where the trial wave-function is large. In contrast to the brute-force algorithm we now have an acceptance probability given by

$$A(y,x) = \min(1, \frac{P_j T_{j\to i}}{P_i T_{i\to j}}), \tag{18}$$

where this time the last term is given by

$$\frac{P_j T_{j\to i}}{P_i T_{i\to j}} = \frac{G(x,y,\Delta t)|\Psi_T(y)|^2}{G(y,x,\Delta t)|\Psi_T(x)|^2} \tag{19}$$

instead of the expression in Eq. 15, where

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

is the solution of the Fokker-Planck equation. For a more detailed discussion on the Langevin and Fokker-Plack equations see Ref. [3]

2.5 Steepest descent

We seek to minimize expectation value of the local energy with respect to the variational parameters $\boldsymbol{\alpha} = \{\alpha_1, ..., \alpha_n\}$ numerically. That is to say we want to find $\boldsymbol{\alpha}$ such that,

$$\nabla_{\boldsymbol{\alpha}}\langle E_L(\mathbf{R};\boldsymbol{\alpha})\rangle = 0. \tag{21}$$

One way to this is by the method of steepest descent. The rationale behind this approach is the well known fact that if a function $F(\boldsymbol{\alpha})$ is differentiable in a neighborhood of $\boldsymbol{\alpha}$, then $F(\boldsymbol{\alpha})$ decreases fastest if one goes in the direction of $-\nabla_{\boldsymbol{\alpha}}F(\boldsymbol{\alpha})$. Hence we can generate a new set of parameters $\boldsymbol{\alpha}_{k+1}$ by the iterative scheme,

$$\boldsymbol{\alpha}_{k+1} = \boldsymbol{\alpha}_k - \gamma_i\nabla_{\boldsymbol{\alpha}_k}F(\boldsymbol{\alpha}_k). \tag{22}$$

If γ_i is sufficiently small [4] we have $F(\boldsymbol{\alpha}_k) \geq F(\boldsymbol{\alpha}_{k+1})$ for all k and hopefully the sequence $\{\boldsymbol{\alpha}_k\}_{k=0}^N$ converges to the desired minimum. The subtraction of $-\gamma_i\nabla_{\boldsymbol{\alpha}_k}F(\boldsymbol{\alpha}_k)$ is due to the fact that we want to move against the gradient towards a local minimum.

In its crudest form γ_i is a constant step size, while it is also possible to adapt γ_i for each iteration to improve upon convergence. We use a very simple adaption scheme for γ_i where we halve the stepsize if $|\nabla_{\boldsymbol{\alpha}_k}F(\boldsymbol{\alpha}_{k+1})| \geq |\nabla_{\boldsymbol{\alpha}_k}F(\boldsymbol{\alpha}_k)|$ and keeps the previous suggestion of $\boldsymbol{\alpha}$.

Note also that we have to supply a initial guess $\boldsymbol{\alpha}_0$ and naturally the better the initial guess the more we can hope for convergence.

We are only concerned with one variational parameter so for our specific problem we know have that,

$$\alpha_{k+1} = \alpha_k - \gamma_i\frac{d}{d_{\alpha_k}}\langle E_L(\mathbf{R};\alpha_k)\rangle. \tag{23}$$

The derivative of the local energy with respect to α is given by,

$$\frac{dE_L}{d\alpha} = 2\left(\left\langle\frac{1}{\Psi_T}\frac{d\Psi_T}{d\alpha}E_L\right\rangle - \left\langle\frac{1}{\Psi_T}\frac{d\Psi_T}{d\alpha}\right\rangle\langle E_L\rangle\right) \tag{24}$$

[3] Morten Hjorth-Jensen, Computational Physics Lecture Notes, Oslo (2015), Section 12.6.
[4] https://en.wikipedia.org/wiki/Gradient_descent

2.6 Blocking

When doing numerical computations we have to provide an estimate of the error in our results. If all our samples are assumed to be uncorrelated our best estimate of the standard deviation of $\langle E_L \rangle$ is,

$$\sigma = \sqrt{\frac{1}{N}(\langle E_L^2 \rangle - \langle E_L \rangle^2)} \tag{25}$$

with N being the number of samples. However, if the samples are correlated one can show [5] that

$$\sigma = \sqrt{\frac{1 + 2\tau/\Delta t}{N}(\langle E_L^2 \rangle - \langle E_L \rangle^2))} \tag{26}$$

where τ is the time between a sample and the next uncorrelated sample (also known as the correlation time) and Δt is the time between each sample. The problem is that we do not know τ or it is expensive to compute. What we want to do is to estimate τ using the blocking technique.

The main idea of blocking is as follows: Let $\mathbf{E} = \{E_1, ..., E_N\}$ be a list of N samples. Then divide $\mathbf{E}$ into blocks of size M with $n = N/M$ elements in each block. Then compute $\mathbf{E}^{\text{new}} = \{E_1^{\text{new}}, ..., E_M^{\text{new}}\}$, where E_k^{new} is the mean of the values in block M. This reduces the number of samples to M. We now assume that these new samples are uncorrelated and compute the standard deviation according to 25. Notice that if we let N and Δt in 26 be fixed, the error will increase with increasing τ. If we make a plot of the standard deviation vs. the number of elements in each block the std. will stop increasing at some point and this suggests that the samples are uncorrelated. Then we have an estimate of the correlation time, $\tau \approx n$.

However there are som pitfalls to be aware of. When the number of blocks is small we have very few samples which gives us a very unreliable value of the std. This tells us that we have to be careful if the std. do not stop increasing until the number of blocks is small. In that case we would need a larger set of original samples $\mathbf{E}$ in order to have a reliable estimate of τ.

In practice, we write all our samples of the quantity we want to compute the std. of to file and make the blocking analysis in a separate script. For this purpose we have chosen to write a short python script called Blocking.py which is found at the provided github adress.

Algorithm 1 Blocking algorithm

```
procedure BLOCKING
    E = array of original samples;
    M = list of different number of blocks;
    N = length(E);
    var = array of variance of element k ∈ M
    k = 0, index counter
    for m ∈ M do
        n = N/m, number of elements in each block.
        E_new = empty array of length m.
        for j = 1, ..., m do
            E_new[j] = sum(E[jn : (j + 1)n])/n;
        ⟨E_new⟩ = sum(E_new)/m;
        ⟨E²_new⟩ = sum(E²_new)/m;
        var[k] = (⟨E²_new⟩ − ⟨E_new⟩²)/m
        k = k + 1
```

[5] Morten Hjorth-Jensen, Computational Physics Lecture Notes, Oslo (2015), Section 11.2.3.

2.7 One-body density

As a way to better visualize the characteristics of a many-body system it is desirable to compute the so-called one-body density of the system. By definition the one-body density is given by

$$\rho(\mathbf{r}, \mathbf{r}') = \int d\mathbf{r}' |\Psi_T(\mathbf{R})|^2. \tag{27}$$

The interpretation of ρ is the probability of finding some particle at position $\mathbf{r}$. The integrand $d\mathbf{r}'$ signifies that we integrate over all possible positions other than $\mathbf{r}$.

Consider the simple example where we can write Ψ_T as a product of normalized single-particle functions,$\phi(\mathbf{r}_k)$, then

$$\begin{aligned}
\rho(\mathbf{r}, \mathbf{r}') &= \int d\mathbf{r}' |\phi(\mathbf{r})|^2 | \prod_{r \in \mathbf{r}'} \phi(r)|^2 \\
&= |\phi(\mathbf{r})|^2 \int d\mathbf{r}' | \prod_{r \in \mathbf{r}'} \phi(r)|^2 \\
&= |\phi(\mathbf{r})|^2
\end{aligned}$$

which is just the single-particle probability of finding some particle at position $\mathbf{r}$.

3 Systems

As mentioned in the introduction we want to study a trapped Bose gas. When writing large programs it is important to test the code properly. Hence we want to start with a simple approximation to the full problem and gradually develop towards the most general problem.

The trap we will use is a spherical (S) or an elliptical (E) Harmonic Oscillator (HO) trap in one, two and three dimensions, given by

$$\hat{V}_{\text{ext}}(\mathbf{r}) = \begin{cases} \frac{1}{2} m \omega_{\text{ho}}^2 r^2 & (S) \\ \frac{1}{2} m [\omega_{\text{ho}}^2 (x^2 + y^2) + \omega_z^2 z^2] & (E) \end{cases} \tag{28}$$

where ω_{ho}^2 defines the strength of the trap. In the case of an elliptical trap frequency in the xy-plane while ω_z is the frequency in the z-direction.

3.1 Hamiltonians

In general the hamiltonian of the system is on the form,

$$\hat{H} = \sum_i^N \left(-\frac{\hbar}{2m} \nabla_i^2 + \hat{V}_{\text{ext}}(\mathbf{r}_i) \right) + \sum_{i<j} \hat{V}_{\text{int}}(\mathbf{r}_i, \mathbf{r}_j) \tag{29}$$

where the first sum is over one-body operators, namely the kinetic energy and the trap potential. The last term is the inter-boson interaction, which will be represented by a pairwise, repulsive potential

$$\hat{V}_{\text{int}}(|\mathbf{r}_i - \mathbf{r}_j|) = \begin{cases} \infty & |\mathbf{r}_i - \mathbf{r}_j| \leq a \\ 0 & |\mathbf{r}_i - \mathbf{r}_j| > a \end{cases} \tag{30}$$

where a is the hard-core diameter of the bosons. To begin with we will consider non-interacting systems and then generalize to interacting systems.

3.2 Trial wave functions

The trial wave function we will use for the ground state with N-atoms is,

$$\psi_T(\mathbf{R}) = \prod_i g(\alpha, \beta, \mathbf{r}_i) \prod_{i<j} f(a, |\mathbf{r}_i - \mathbf{r}_j|) \tag{31}$$

where α and β are variational parameters. The single-particle wave function is given by

$$g(\alpha, \beta, \mathbf{r}_i) = \exp[-\alpha(x_i^2 + y_i^2 + \beta z_i^2)]. \tag{32}$$

For spherical traps $\beta = 1$ and for non-interacting bosons $a = 0$. The correlation wave function is

$$f(a, |\mathbf{r}_i - \mathbf{r}_j|) = \begin{cases} 0 & |\mathbf{r}_i - \mathbf{r}_j| \leq a \\ (1 - \frac{a}{|\mathbf{r}_i - \mathbf{r}_j|}) & |\mathbf{r}_i - \mathbf{r}_j| > a \end{cases} \tag{33}$$

3.3 Simplified system

During the implementation we consider a simplified system with trial wave function

$$\Psi_T(\mathbf{R}) = \prod_{i=1}^{N} g(\alpha, \beta = 1, \mathbf{r}_i). \tag{34}$$

The hamiltonian of this system is given by,

$$\hat{H} = \sum_{i=1}^{N} \left(-\frac{\hbar}{2m} \nabla_i + \hat{V}_{\text{ext}}(\mathbf{r}_i) \right) \tag{35}$$

where $\hat{V}_{\text{ext}}(\mathbf{r}_i)$ is the spherical harmonic oscillator potential. In Appendix A.1 we show the expectation value of the ground-state energy for this system is,

$$\langle \hat{H} \rangle = dN\alpha \tag{36}$$

where d is the number of spatial dimensions, N the number of particles and α is the variational parameter. This result holds provided that $\alpha = \omega_{ho}/2$. We will use this result to verify our program and frequently refer to this as the non-interacting system in the following sections.

4 Implementation and validation

4.1 Implementation

4.1.1 Overall structure of the VMC code

We have tried to write a program that is as general as possible, using an object-oriented approach. This is highly desirable since we want to use the program to study other types of systems, in specific fermionic systems, in future projects. An approach like this makes it easy to change options and implement new features. Objects that have similar properties are wrapped into classes, typically with a parent class which is abstract, meaning that we can not create an instance of this class. Typical examples are different kind of trial wavefunctions which all inherits from a parent class wavefunction, and different kind of hamiltonians which is distinguished by their potential energy operator. Another important class is the system class which contains all information about the system under consideration and also has the solver function, called metropolisStep() or metropolisStepImportanceSampling(), which performs the Monte Carlo integration using either the brute force or importance sampled Metropolis-Hastings algorithm.

Note that for verification purposes, all of the potentials implemented this far has two possible ways of computing the laplacian, either using a analytic expression which is derived by its action on the specific trial wavefunction under consideration or by numerical differentiation. The gradient which is needed in order to compute the quantum force when we use importance sampling is implemented as a function in the wavefunctions family of classes.

4.1.2 Program flow

The flow of the program is in general quite simple. A system is intialized by specifying all parameters such as the number of particles, the number of spatial dimensions, intial guesses for the variational parameters, the number of Monte Carlo cycles and a steplength in time. One also chooses a type of trial wavefunction, a hamiltonian and then all particles get an initial position according to a random uniform distribution. Finally a call to systems runMetropolisStep() starts up the whole VMC machinery. A set of different systems is setup in the Examples class which also demonstrates how one performs optimization using the steepest descent method.

4.1.3 Remarks on the proposal of new positions

We want to make a remark on the way we propose new positions with the brute-force and importance sampled algorithm respectively. In the brute-force case we pick a particle at random and a spatial dimension at random and then we draw a random number $c \in (-1, 1)$. Then this particles position in the randomly chosen spatial dimension is changed by an amount $c\Delta t$.

However, in the importance sampled case we have chosen to make a change to all particles according to,

$$\mathbf{r}^{\text{new}} = \mathbf{r}^{old} + D\mathbf{F}(\mathbf{r}^{old})\Delta t + \boldsymbol{\xi}\sqrt{\Delta t}, \tag{37}$$

where $\mathbf{F}(\mathbf{r}^{old})$ is the quantum force at the old position and $\boldsymbol{\xi}$ is a vector of random numbers. The point of this anecdote is that it makes the computation of the acceptance probability given by Eq. 19 very time consuming, due to the fact that we have to compute a large dot product every Monte Carlo cycle.

4.2 Validating the code

A very important aspect when writing large programs is to test the implementation of different parts of the code. We use the system described in section 3.3 to validate the implementation of brute-force and importance sampled Metropolis-Hastings algorithm. Recall that when we use the importance sampled Metropolis-Hastings algorithm we need to compute the quantum force,

$$\mathbf{F} = 2\frac{\nabla\Psi_T}{\Psi_T}$$

which is a $3dN$-dimensional vector. Also when computing the local energy we have to evaluate

$$\sum_{i=1}^{N} \nabla_i^2 \Psi_T.$$

These are the most time consuming quantities to compute in the program, hence for CPU efficiency and in order to reduce numerical errors it is highly desirable to derive analytical expressions for $\nabla\psi_T$ (see Appendix A.2) and $\nabla^2\psi_T$ (see Appendix A.3) rather than using numerical differentiation. However, we have made it possible to use numerical differentiation in order to verify that we have implemented the analytical expressions efficiently.

Note that when performing these tests we have used simpler expressions for the gradient and the laplacian than those derived in the Appendix. We have used the gradient and laplacian of Eq. 34 which is straight-forward to derive since it is just a product of exponential functions. In the last paragraph of this section we discuss what we have done in order to validate the implementation of the general expressions for the gradient and laplacian.

The results of these tests are indeed very comforting as we can clearly see in table 1 and table 2. Both the brute force and the importance sampled algorithm produces the correct values for the energy (equation 36) with variance exactly zero and we can see that when we do not use numerical differentiation the program is significantly faster. In general it seems like if we have $\Delta t \in [1.5, 2.5]$ for the brute force algorithm we have an acceptance rate around 50% as is common

Table 1: Test of brute-force Metropolis-Hastings. The calculations have been run for $d = 3$, $N_{\text{cycles}} = 10^6$, $\Delta t = 2.0$, $\alpha = 0.5$ and $\omega_{ho} = 1$. The subscript A and N denotes that we have used the analytic expression or numerical differentiation for computing the laplacian of Ψ_T respectively. We get an acceptance rate of approximately 55% for all the different number of particles. Note that "$-$" signifies that the program did not finish in these cases.

Number of particles	$\langle E \rangle_A / \hbar\omega$	σ_A^2	Cpu_A[s]	$\langle E \rangle_N / \hbar\omega$	σ_N^2	Cpu_N[s]
1	1.5	0	0.60	1.5	$2.54 \cdot 10^{-13}$	1.71
10	15	0	2.30	14.9999	$2.54 \cdot 10^{-10}$	55.69
100	150	0	20.09	$-$	$-$	$-$
500	750	0	100.52	$-$	$-$	$-$

Table 2: Test of importance sampled Metropolis-Hastings. Again we have used $d = 3$, $\alpha = 0.5$ and $\omega_{ho} = 1$. However, now $\Delta t = 0.01$ and we have limited ourselves to $N_{\text{cycles}} = 10^4$ due to the high numerical cost described in section 4.1.3.

Number of particles	$\langle E \rangle_A / \hbar\omega$	σ_A^2	Cpu_A[s]	$\langle E \rangle_N / \hbar\omega$	σ_N^2	Cpu_N[s]
1	1.5	0	0.02	1.4999	$2.22 \cdot 10^{-11}$	0.03
10	15	0	0.66	14.9995	$2.82 \cdot 10^{-8}$	1.20
100	150	0	56.96	149.947	$2.85 \cdot 10^{-5}$	108.51
500	750	0	1438.72	$-$	$-$	$-$

practice. When we use importance sampling we want the acceptance rate to be as close to 100% as possible. This is fulfilled if we choose $\Delta t \in [0.01, 0.0001]$.

Another important part of the program to test is the steepest descent method. For the non-interacting case we know that if $\omega = 1$, $\alpha = 1/2$ is the optimal parameter. Then one way to check that the steepest descent method runs correctly is to start the program with an incorrect α and see if it converges to $\alpha = 1/2$. We claim convergence if,

$$\frac{dE_L}{d\alpha} \leq \epsilon, \tag{38}$$

where ϵ is a chosen precision. Performing this test we have used $\epsilon = 10^{-12}$. We have convergence if the initial guess $\alpha_0 \geq 0.35$ with approximately the same number of iterations until Eq. 38 is satisfied. However, we have discovered that the method fails to converge if the initial guess is below this value even for this simple test case. This is thought to be due to the way we adapt the step length γ_i in equation 23 since a small α_0 gives a large value for $dE_L/d\alpha_0$. This is something we will have to investigate further in the future. In the meantime we note that we have to be cautious when we use this part of the program. This test is easily repeated by returning "Examples::nonInteractingHOSteepestDescent(initial_guess)" which is already set up in "main.cpp" provided in the source code.

The program is written in such a way that the implementation of the hamiltonian and trial wave function of the full problem is a separate class from those of the simple test case. Note that the implementation of the full system only works for three spatial dimensions. It is in this part that we implement the full expression derived in Appendix A.3 and it is very difficult not to make small errors. One way to test this implementation is to let $\beta = 1$ in Eq. 31 and turn off the Jastrow-factor which in effect is to run the simplified system. Hence, we would expect that we get $dN\alpha$ for the ground-state energy. When we do this we get a value which is slightly lower than the exact result. This is thought to be a numerical error due to the fact that this computation is dependent on subtractions of the positions, which are very close to each other in magnitude due to the fact that we only move one particle each Monte Carlo cycle when we use the brute-force algorithm (see section 4.1.3). Subtraction of almost equal numbers is known to cause numerical round-off errors which may be a contributing factor to the error in the value of the energy. Another

possibility is naturally that we have just mistyped some part of the implementation. Debugging possible error like this is a very time-consuming task and this should be investigated as soon as time allows.

5 Results and discussion

Finally we apply the program to the full system described in section 3. We use only the brute-force algorithm due to the fact that it is faster. The only variational parameter is α and β is fixed to $\beta = 2.82843$ (exercise text). We use $\Delta t = 1.5$ since previous tests of the program indicates that this will give an acceptance rate of approximately 50%, which is considered reasonable according to common practice. The hard-core diameter of the bosons are set to $a = 0.0043$. We first seek an optimal variational parameter α^* using the method of steepest descent with initial guess $\alpha_0 = 0.52$ and number of Monte Carlo cycles $N_{\text{cycles}} = 10^4$ for each iteration. This gives $\alpha^* = 0.499719$. Then we perform a simulation to compute the ground-state energy with $N_{\text{cycles}} = 10^6$ for $N_p = \{10, 50, 100\}$ number of particles and $\alpha = \alpha^*$. We use the trial wave function with the Jastrow factor given by Eq. 31. Blocking is used to give a better estimate of the variance for $N_p = 10$ and $N_p = 50$. At last we plot the one-body densities for $N_p = 10$ and $N_p = 50$ with and without the Jastrow-factor.

5.1 Ground-state energies

In table 3 we present our results of the ground-state energies and compare them with the results of a previous study. We note that in the case of $N_p = 100$ our result deviate on the order of $\sim 3\%$ which is significant. In the two other cases the deviation is lower. An explanation of this might be to the problems described in the last paragraph of section 4.2.

As explained in section 2.6 the estimate of the variance is too optimistic if we consider all samples to be uncorrelated. In Fig. 1 we find that for $N_p = 10$ a more accurate estimate of the variance is $\sigma^2 \approx 1.1 \cdot 10^{-7}$ while for $N_p = 50$ we find that $\sigma^2 \approx 4 \cdot 10^{-6}$. No information was given on the variance achieved in the previous study so it is not possible to compare these values.

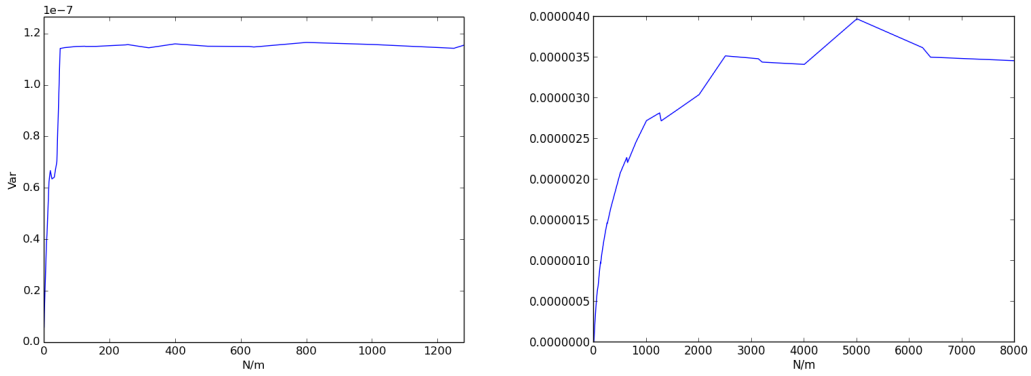
We claimed earlier that a step length $\Delta t = 1.5$ would yield an acceptance rate of approximately 50% which is clearly demonstrated by the results.

Table 3: In the following we present the estimated ground-state energies for systems with $N_p = \{10, 50, 100\}$ number of particles. Also included is the results of a previous which we recieved on mail (from Morten Hjorth-Jensen). We have used the brute-force Metropolis-Hastings algorithm with step length $\Delta t = 1.5$, number of Monte Carlo cycles $N_{\text{cycles}} = 10^6$, $\beta = 2.82843$ and with optimal parameter $\alpha^* = 0.499719$ in all cases. The referenced results indicates that $\alpha^* = 0.5$. Note that the estimate of the variance is very optimistic since it is assumed that all samples are uncorrelated. In Figure 1 we give a better estimate of the variance for $N_p = 10$ and $N_p = 50$ using blocking. If we compute the ground-state energies without the Jastrow factor we get slightly lower values which indicates that the system is weakly interacting.

Number of particles	$\langle E \rangle / \hbar\omega$	σ^2	Cpu-time[s]	Acceptance rate	$\langle E \rangle_{\text{ref}} / \hbar\omega$	Deviation[%]
10	24.141	$5.88 \cdot 10^{-9}$	24.39	0.55	24.2	0.24
50	120.484	$1.36 \cdot 10^{-8}$	1491.98	0.56	122	1.24
100	239.757	$1.99 \cdot 10^{-7}$	20737.13	0.57	247	2.93

5.2 One-body densities

The one-body density is interpreted as the probability of finding some particle at a distance r from the center of the system. If we exclude the Jastrow-factor the bosons do not repel each other and we expect that the probability is higher closer to the center of the system than if we include

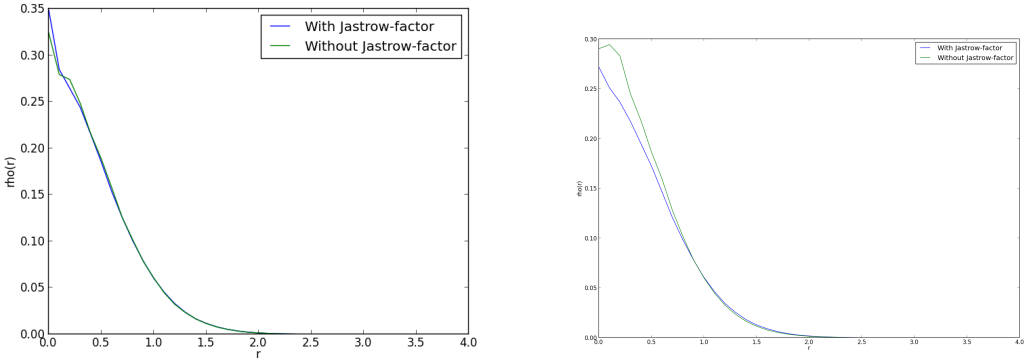


(a) Plot of blocking applied to the interacting system with $N_p = 10$.

(b) Plot of blocking applied to the interacting system with $N_p = 50$.

Figure 1: Blocking applied to systems with $N_p = 10$ and $N_p = 50$. This gives an estimate of the correlation time, $\tau \approx 800$ and the variance $\sigma^2 \approx 1.1 \cdot 10^{-7}$ for $N_p = 10$. For $N_p = 50$, $\tau \approx 5000$ and $\sigma^2 \approx 4 \cdot 10^{-6}$. We notice that the correlation time is larger when $N_p = 50$. This is expected since we only change the position of one particle for every time we propose a new state as explained in section 4.1.3. It is clear that if we change the position of 1 out of 10 particles as much as we change the position of 1 out of 50 particles the system with 10 particles will be less correlated. We would then expect the correlation time to be even larger for $N_p = 100$ which also indicates that σ^2 would be larger meaning that the estimate of the variance in table 3 is far too optimistic. The input parameters is described in table 3.

the Jastrow-factor. Also with repulsion included we would expect a higher probability of finding a boson at a larger distance from the center. It is also expected that this effect is more apparent with more particles in the system since it is affected by every other boson. All of these features is apparent in the results presented in Fig. 2.



(a) Plot of the one-body density with and without the Jastrow-factor for $N_p = 10$.

(b) Plot of the one-body density with and without the Jastrow-factor for $N_p = 50$.

Figure 2: We can see from the figures that without the Jastrow-factor we have a higher probability of finding a boson closer to the center of the system. Notice also that with the Jastrow-factor included we have a slightly higher probability of finding the boson at a larger distance from the center. In the case of $N_p = 10$ this effect is nearly impossible to see, however, for $N_p = 50$ we can clearly see a difference with and without the Jastrow-factor.

6 Conclusion

We have found ground-state energies that agrees quite well with referenced results, especially for $N_p = 10$. However, we note that we have quite a significant deviation for $N_p = 100$. The program has been thoroughly tested against a simplified version of the full problem which agrees perfectly with analytic results.

We have relied upon the brute-force Metropolis algorithm when computing the ground-state energies of the full problem since we found that this implementation allowed us to run far more Monte Carlo cycles. If time had allowed us, it would have been intersting to investigate if the importance-sampled algorithm would give a different (possibly better) estimate of the variance after using blocking with fewer Monte Carlo cycles. We suspect that this is a possibility due to the fact that all particles are moved which should make each state less correlated as explained in section 4.1.3.

Also, we found that the steepest descent method failed to converge if we chose a bad initial guess for the variational parameter. We have hypothesized that this is due to the way we update the step length in Eq. 23. There are several interesting ways to investigate this problem. For example we could try to improve upon the way we adapt the step length, or implement another method such as the Conjugate gradient method.

We have found it rewarding to work with this project and learned alot about how to write large programs in a structured manner using an object-oriented approach. For the first time we have experienced how critical it is to test the program thoroughly throughout the process of the development. Probably the most satisfying experience was when we made the plots of the one-body densities which seemed to agree with our "intuition". A lack of time in the final hours of writing the report made us feel that some parts of the theory section might have been neglected. Even though, we feel content with the final result.

A Appendix

A.1 Closed form expression for the local energy of the non-interacting system

In the following we compute a closed form expression for the local energy of the non-interacting system with the spherical harmonic oscillator potential.With $\hbar = m = 1$ and N being the number of particles, the hamiltonian is given by,

$$\hat{H} = \sum_{i=1}^{N} \left(-\frac{1}{2}\nabla_i^2 + \hat{V}_{\text{ext}}(\mathbf{r}_i) \right) \tag{39}$$

where $\hat{V}_{\text{ext}}(\mathbf{r}_i) = \frac{1}{2}\omega_{ho}^2 r_i^2$ is the spherical harmonic oscillator potential. The corresponding trial wave function is,

$$\Psi_T(\mathbf{R}) = \prod_{i=1}^{N} \exp(-\alpha(x_i^2 + y_i^2 + \beta z_i^2)) \tag{40}$$

with $\beta = 1$. The laplacian of particle k is,

$$\begin{aligned}
\nabla_k^2 \Psi_T(\mathbf{R}) &= \frac{\partial^2}{\partial x_k^2} \exp(-\alpha(x_k^2 + y_k^2 + z_k^2)) + \frac{\partial^2}{\partial y_k^2} \exp(-\alpha(x_k^2 + y_k^2 + z_k^2)) \\
&\quad + \frac{\partial^2}{\partial z_k^2} \exp(-\alpha(x_k^2 + y_k^2 + z_k^2)) \\
&= (-2\alpha + 4\alpha^2 x_k - 2\alpha + 4\alpha^2 y_k - 2\alpha + 4\alpha^2 z_k) \exp(-\alpha(x_k^2 + y_k^2 + z_k^2)).
\end{aligned}$$

We can write this expression in a compact way for $d = \{1, 2, 3\}$ with d being the number of spatial dimensions as follows,

$$\begin{aligned}\nabla_k^2 \Psi_T(\mathbf{R}) &= (-2\alpha + 4\alpha^2 x_k - 2\alpha + 4\alpha^2 y_k - 2\alpha + 4\alpha^2 z_k) \exp(-\alpha(x_k^2 + y_k^2 + z_k^2)) \\ &= (-2d\alpha + 4\alpha^2(x_k^2 + y_k^2 + z_k^2)) \exp(-\alpha(x_k^2 + y_k^2 + z_k^2)) \\ &= (-2d\alpha + 4\alpha^2 r_k^2) \exp(-\alpha r_k^2)\end{aligned}$$

Let $\Phi(\mathbf{r}_k) = \exp(-\alpha(x_k^2 + y_k^2 + z_k^2))$, then we can write

$$\Psi_T(\mathbf{R}) = \prod_{i=1}^{N} \Phi(\mathbf{r}_i) \tag{41}$$

The laplacian of $\Psi_T(\mathbf{R})$ then becomes,

$$\begin{aligned}-\frac{1}{2}\left(\sum_{i=1}^{N} \nabla_i^2 \prod_{i=1}^{N} \Phi(\mathbf{r}_i)\right) &= -\frac{1}{2}\left(\nabla_1^2 \prod_{i=1}^{N} \Phi(\mathbf{r}_i) + \cdots + \nabla_N^2 \prod_{i=1}^{N} \Phi(\mathbf{r}_i)\right) \\ &= -\frac{1}{2}\left(\nabla_1^2 \Phi(\mathbf{r}_1) \prod_{i\neq 1}^{N} \Phi(\mathbf{r}_i) + \cdots + \nabla_N^2 \Phi(\mathbf{r}_N) \prod_{i\neq N}^{N} \Phi(\mathbf{r}_i)\right) \\ &= -\frac{1}{2}\left((-2d\alpha + 4\alpha^2 r_1^2)\Phi(\mathbf{r}_1) \prod_{i\neq 1}^{N} \Phi(\mathbf{r}_i) + \cdots + (-2d\alpha + 4\alpha^2 r_N^2)\Phi(\mathbf{r}_N) \prod_{i\neq N}^{N} \Phi(\mathbf{r}_i)\right) \\ &= -\frac{1}{2}\left((-2d\alpha + 4\alpha^2 r_1^2)\Psi_T(\mathbf{R}) + \cdots + (-2d\alpha + 4\alpha^2 r_N^2)\Psi_T(\mathbf{R})\right) \\ &= -\frac{1}{2}\left(\sum_{i=1}^{N}(-2d\alpha + 4\alpha^2 r_i^2)\right)\Psi_T(\mathbf{R}) \\ &= \left(dN\alpha - 2\alpha^2 \sum_{i=1}^{N} r_i^2\right)\Psi_T(\mathbf{R})\end{aligned}$$

Inserting this into the expression for the local energy we get,

$$\begin{aligned}E_L(\mathbf{R}) &= \frac{1}{\Psi_T(\mathbf{R})}\hat{H}\Psi_T(\mathbf{R}) \\ &= \frac{1}{\Psi_T(\mathbf{R})}\left(\left(dN\alpha - 2\alpha^2 \sum_{i=1}^{N} r_i^2\right)\Psi_T(\mathbf{R}) + \frac{1}{2}\omega_{ho}^2\left(\sum_{i=1}^{N} r_i^2\right)\Psi_T(\mathbf{R})\right) \\ &= dN\alpha - 2\alpha^2 \sum_{i=1}^{N} r_i^2 + \frac{1}{2}\omega_{ho}^2 \sum_{i=1}^{N} r_i^2\end{aligned}$$

Notice now that if $\alpha = \omega_{ho}/2$, the last two terms cancel out are left with,

$$E_L(\mathbf{R}) = dN\alpha. \tag{42}$$

In this case it follows immediately that,

$$\begin{aligned}\langle \hat{H} \rangle &= \int d\mathbf{R} |\Psi_T(\mathbf{R})|^2 E_L(\mathbf{R}) \\ &= \int d\mathbf{R} |\Psi_T(\mathbf{R})|^2 dN\alpha \\ &= dN\alpha\end{aligned}$$

A.2 Closed form of the gradient of the trial wave function

We will compute $\nabla\Psi_T$ and $\nabla^2\Psi_T$ for

$$\Psi_T(\mathbf{R}) = \prod_i g(\alpha, \beta, \mathbf{r}_i) \prod_{i<j} f(a, |\mathbf{r}_i - \mathbf{r}_j|) \tag{43}$$

where

$$g(\alpha, \beta, \mathbf{r}_i) = \exp(-\alpha(x_i^2 + y_i^2 + \beta z_i^2)). \tag{44}$$

Let

$$r_{ij} = |\mathbf{r}_i - \mathbf{r}_j| \tag{45}$$

$$f(r_{ij}) = \exp\left(\sum_{i<j} u(r_{ij})\right) \tag{46}$$

$$u(r_{ij}) = \ln(f(r_{ij})) \tag{47}$$

$$\Phi(\mathbf{r}_i) = g(\alpha, \beta, \mathbf{r}_i). \tag{48}$$

and write $\Psi_T(\mathbf{R})$ as

$$\Psi_T(\mathbf{R}) = \prod_i \Phi(\mathbf{r}_i) \exp\left(\sum_{i<j} u(r_{ij})\right). \tag{49}$$

We now want to compute the gradient of particle k of $\Psi_T(\mathbf{R})$.

$$\nabla\Psi_T(\mathbf{R}) = \nabla_k \left[\prod_i \Phi(\mathbf{r}_i) \exp\left(\sum_{i<j} u(r_{ij})\right)\right] \tag{50}$$

which by the product rule is

$$\left(\nabla_k \prod_i \Phi(\mathbf{r}_i)\right) \exp\left(\sum_{i<j} u(r_{ij})\right) + \prod_i \Phi(\mathbf{r}_i) \left(\nabla_k \left(\sum_{i<j} u(r_{ij})\right)\right). \tag{51}$$

We notice that

$$\begin{aligned}
\nabla_k \prod_i \Phi(\mathbf{r}_i) &= \nabla_k \left(\Phi(\mathbf{r}_1)...\Phi(\mathbf{r}_k)...\Phi(\mathbf{r}_N)\right) \\
&= \nabla_k\Phi(\mathbf{r}_k) \left(\Phi(\mathbf{r}_1)...\Phi(\mathbf{r}_k)...\Phi(\mathbf{r}_N)\right) \\
&= \nabla_k\Phi(\mathbf{r}_k) \prod_{i\neq k} \Phi(\mathbf{r}_i), \text{ since } \nabla_k \text{ acts only on } \Phi(\mathbf{r}_k).
\end{aligned}$$

By the chain rule we have that

$$\nabla_k \exp\left(\sum_{i<j} u(r_{ij})\right) = \exp\left(\sum_{i<j} u(r_{ij})\right) \nabla_k \sum_{i<j} u(r_{ij}). \tag{52}$$

Writing out the last term,

$$\begin{aligned}
\nabla_k \sum_{i<j} u(r_{ij}) = \nabla_k(&u(r_{12}) + u(r_{13}) + u(r_{14}) + ... + u(r_{1N}) \\
&+u(r_{23}) + u(r_{24}) + ... + u(r_{2N}) \\
&\ddots \quad \vdots \\
&+ u(r_{N-1N}),
\end{aligned}$$

we notice that for any particular $i, j \neq k$

$$\nabla_k u(r_{ij}) = 0, \tag{53}$$

which gives

$$\nabla_k \sum_{i<j} u(r_{ij}) = \sum_{j \neq k} \nabla_k u(r_{kj}). \tag{54}$$

Next we rewrite $\nabla_k u(r_{kj})$ as follows

$$\begin{aligned}
\nabla_k u(r_{kj}) &= \left(\frac{\partial}{\partial x_k} \hat{i} + \frac{\partial}{\partial y_k} \hat{j} + \frac{\partial}{\partial z_k} \right) u(r_{kj}) \\
&= \left(\frac{\partial r_{kj}}{\partial x_k} \hat{i} + \frac{\partial r_{kj}}{\partial y_k} \hat{j} + \frac{\partial r_{kj}}{\partial z_k} \right) \frac{\partial u(r_{kj})}{\partial r_{kj}}.
\end{aligned}$$

By the chain rule we have that

$$\begin{aligned}
\frac{\partial}{\partial x_k} r_{kj} &= \frac{\partial}{\partial x_k} \sqrt{(x_k - x_j)^2 + (y_k - y_j)^2 + (z_k - z_j)^2} \\
&= \frac{2(x_k - x_j)}{2 r_{kj}} \\
&= \frac{x_k - x_j}{r_{kj}}.
\end{aligned}$$

In the same way we have that

$$\frac{\partial}{\partial y_k} r_{kj} = \frac{y_k - y_j}{r_{kj}}; \;\; \frac{\partial}{\partial z_k} r_{kj} = \frac{z_k - z_j}{r_{kj}}. \tag{55}$$

Introducing $u'(r_{kj}) = \frac{\partial u(r_{kj})}{\partial r_{kj}}$ we arrive at

$$\begin{aligned}
\nabla_k u(r_{kj}) &= [(x_k - x_j)\hat{i} + (y_k - y_j)\hat{j} + (z_k - z_j)\hat{k}] \frac{u'(r_{kj})}{r_{kj}} \\
&= \frac{\mathbf{r}_k - \mathbf{r}_j}{r_{kj}} u'(r_{kj}).
\end{aligned} \tag{56}$$

Finally we have that

$$\begin{aligned}
\nabla_k \Psi_T(\mathbf{R}) = \nabla_k \Phi(\mathbf{r}_k) \left[\prod_{i \neq k} \Phi(\mathbf{r}_i) \right] \exp\left(\sum_{i<j} u(r_{ij}) \right) \\
+ \prod_i \Phi(\mathbf{r}_i) \exp\left(\sum_{i<j} u(r_{ij}) \right) \sum_{j \neq k} \frac{(\mathbf{r}_k - \mathbf{r}_j)}{r_{kj}} u'(r_{kj}).
\end{aligned} \tag{57}$$

A.3 Closed form of the laplacian of the trial wave function

Next we want to compute the laplacian of paricle k. We notice that we can write

$$\begin{aligned}
\Psi_T(\mathbf{R}) &= \prod_i \Phi(\mathbf{r}_i) \exp\left(\sum_{i<j} u(r_{ij}) \right) \\
&= \Phi(\mathbf{r}_k) \prod_{i \neq k} \Phi(\mathbf{r}_i) \exp\left(\sum_{i<j} u(r_{ij}) \right)
\end{aligned}$$

and

$$\nabla_k \prod_{i \neq k} \Phi(\mathbf{r}_i) = 0. \tag{58}$$

Then by the product rule

$$\begin{aligned}
\frac{1}{\Psi_T(\mathbf{R})} \nabla_k^2 \Psi_T(\mathbf{R}) &= \frac{1}{\Psi_T(\mathbf{R})} \nabla_k \cdot (\nabla_k \Psi_T(\mathbf{R})) \\
&= \frac{1}{\Psi_T(\mathbf{R})} \nabla_k \cdot (\nabla_k \Phi(\mathbf{r}_k) \left[\prod_{i \neq k} \Phi(\mathbf{r}_i) \right] \exp\left(\sum_{i<j} u(r_{ij}) \right) \\
&+ \prod_i \Phi(\mathbf{r}_i) \exp\left(\sum_{i<j} u(r_{ij}) \right) \sum_{j \neq k} \frac{\mathbf{r}_k - \mathbf{r}_j}{r_{kj}} u'(r_{kj})) \\
&= \frac{\nabla_k^2 \Phi(\mathbf{r}_k)}{\Phi(\mathbf{r}_k)} \\
&+ \frac{\nabla_k \Phi(\mathbf{r}_k) \cdot \nabla_k \exp\left(\sum_{i<j} u(r_{ij}) \right)}{\Phi(\mathbf{r}_k) \exp\left(\sum_{i<j} u(r_{ij}) \right)} \\
&+ \frac{\nabla_k \prod_i \Phi(\mathbf{r}_i) \sum_{j \neq k} \frac{\mathbf{r}_k - \mathbf{r}_j}{r_{kj}} u'(r_{kj})}{\Phi(\mathbf{r}_k) \prod_{i \neq k} \Phi(\mathbf{r}_i)} \\
&+ \frac{\nabla_k \exp\left(\sum_{i<j} u(r_{ij}) \right) \sum_{j \neq k} \frac{\mathbf{r}_k - \mathbf{r}_j}{r_{kj}} u'(r_{kj})}{\exp\left(\sum_{i<j} u(r_{ij}) \right)} \\
&+ \nabla_k \sum_{j \neq k} \frac{\mathbf{r}_k - \mathbf{r}_j}{r_{kj}} u'(r_{kj}).
\end{aligned} \tag{59}$$

Now we want to evaluate each term separately.

$$\begin{aligned}
\frac{\nabla_k \Phi(\mathbf{r}_k) \cdot \nabla_k \exp\left(\sum_{i<j} u(r_{ij}) \right)}{\Phi(\mathbf{r}_k) \exp\left(\sum_{i<j} u(r_{ij}) \right)} &= \frac{\nabla_k \Phi(\mathbf{r}_k) \cdot \left(\exp\left(\sum_{i<j} u(r_{ij}) \right) \sum_{j \neq k} \frac{\mathbf{r}_k - \mathbf{r}_j}{r_{kj}} u'(r_{kj}) \right)}{\Phi(\mathbf{r}_k) \exp\left(\sum_{i<j} u(r_{ij}) \right)} \\
&= \frac{\nabla_k \Phi(\mathbf{r}_k) \cdot \sum_{j \neq k} \frac{\mathbf{r}_k - \mathbf{r}_j}{r_{kj}} u'(r_{kj})}{\Phi(\mathbf{r}_k)}
\end{aligned} \tag{60}$$

$$\begin{aligned}
\frac{\nabla_k \prod_i \Phi(\mathbf{r}_i) \sum_{j \neq k} \frac{\mathbf{r}_k - \mathbf{r}_j}{r_{kj}} u'(r_{kj})}{\Phi(\mathbf{r}_k) \prod_{i \neq k} \Phi(\mathbf{r}_i)} &= \frac{\nabla_k \Phi(\mathbf{r}_k) \prod_{i \neq k} \Phi(\mathbf{r}_i) \sum_{j \neq k} \frac{\mathbf{r}_k - \mathbf{r}_j}{r_{kj}} u'(r_{kj})}{\Phi(\mathbf{r}_k) \prod_{i \neq k} \Phi(\mathbf{r}_i)} \\
&= \frac{\nabla_k \Phi(\mathbf{r}_k) \cdot \sum_{j \neq k} \frac{\mathbf{r}_k - \mathbf{r}_j}{r_{kj}} u'(r_{kj})}{\Phi(\mathbf{r}_k)}
\end{aligned} \tag{61}$$

$$\begin{aligned}
\frac{\nabla_k \exp\left(\sum_{i<j} u(r_{ij}) \right) \sum_{j \neq k} \frac{\mathbf{r}_k - \mathbf{r}_j}{r_{kj}} u'(r_{kj})}{\exp\left(\sum_{i<j} u(r_{ij}) \right)} &= \frac{\exp\left(\sum_{i<j} u(r_{ij}) \right) \sum_{j \neq k} \frac{\mathbf{r}_k - \mathbf{r}_j}{r_{kj}} u'(r_{kj}) \sum_{i \neq k} \frac{\mathbf{r}_k - \mathbf{r}_i}{r_{kj}} u'(r_{ki})}{\exp\left(\sum_{i<j} u(r_{ij}) \right)} \\
&= \sum_{i,j \neq k} \frac{(\mathbf{r}_k - \mathbf{r}_i)(\mathbf{r}_k - \mathbf{r}_j)}{r_{ki} r_{kj}} u'(r_{ki}) u'(r_{kj}).
\end{aligned} \tag{62}$$

And finally the last term

$$\nabla_k \sum_{j\neq k} \frac{\mathbf{r}_k - \mathbf{r}_j}{r_{kj}} u'(r_{kj}) = \sum_{j\neq k} \nabla_k \cdot \left((\mathbf{r}_k - \mathbf{r}_j) \frac{u'(r_{kj})}{r_{kj}} \right)$$

looking at the inside of the sum

$$\nabla_k \cdot \left((\mathbf{r}_k - \mathbf{r}_j) \frac{u'(r_{kj})}{r_{kj}} \right) = \left(\frac{\partial}{\partial x_k}\hat{i} + \frac{\partial}{\partial y_k}\hat{j} + \frac{\partial}{\partial z_k}\hat{k} \right) \cdot \left[\left((x_k - x_j)\hat{i} + (y_k - y_j)\hat{j} + (z_k - z_j)\hat{k} \right) \frac{u'(r_{kj})}{r_{kj}} \right] \tag{63}$$

Looking at the ith component

$$\begin{aligned} \frac{\partial}{\partial x_k} \frac{(x_k - x_j)}{r_{kj}} u'(r_{kj}) &= \frac{\partial}{\partial x_k} \left(\frac{x_k - x_j}{r_{kj}} u'(r_{kj}) \right) \\ &= \frac{\partial}{\partial x_k} \left(\frac{x_k - x_j}{r_{kj}} \right) u'(r_{kj}) + \frac{x_k - x_j}{r_{kj}} \left(\frac{\partial}{\partial x_k} u'(r_{kj}) \right) \end{aligned} \tag{64}$$

$$\begin{aligned} \frac{\partial}{\partial x_k} u'(r_{kj}) &= \frac{\partial r_{kj}}{x_k} \frac{u'(r_{kj})}{\partial r_{kj}} \\ &= \frac{x_k - x_j}{r_{kj}} u''(r_{kj}) \end{aligned} \tag{65}$$

Using wolfram alpha we have that

$$\frac{\partial}{\partial x_k} \left(\frac{x_k - x_j}{r_{kj}} \right) = \frac{1}{r_{kj}} - \frac{(x_k - x_j)^2}{r_{kj}^3} \tag{66}$$

Equation 64 then reads

$$\frac{\partial}{\partial x_k} \frac{(x_k - x_j)}{r_{kj}} u'(r_{kj}) = \left(\frac{1}{r_{kj}} - \frac{(x_k - x_j)^2}{r_{kj}^3} \right) u'(r_{kj}) + \frac{(x_k - x_j)^2}{r_{kj}^2} u''(r_{kj}) \tag{67}$$

By using this procedure we find the expression for the $\hat{j}$ and $\hat{k}$ components. Summing up equation

$$\begin{aligned} \sum_{j\neq k} \nabla_k \cdot \left((\mathbf{r}_k - \mathbf{r}_j) \frac{u'(r_{kj})}{r_{kj}} \right) &= \sum_{j\neq k} \left(\frac{1}{r_{kj}} - \frac{(x_k - x_j)^2}{r_{kj}^3} + \frac{1}{r_{kj}} - \frac{(y_k - y_j)^2}{r_{kj}^3} + \frac{1}{r_{kj}} - \frac{(z_k - z_j)^2}{r_{kj}^3} \right) u'(r_{kj}) \\ &\quad + \left(\frac{(x_k - x_j)^2 + (y_k - y_j)^2 + (z_k - z_j)^2}{r_{kj}^2} \right) u''(r_{kj}) \\ &= \sum_{j\neq k} \left[\left(\frac{3}{r_{kj}} - \frac{r_{kj}^2}{r_{kj}^3} \right) u'(r_{kj}) + \frac{r_{kj}^2}{r_{kj}^2} u''(r_{kj}) \right] \\ &= \sum_{j\neq k} \left[\frac{2}{r_{kj}} u'(r_{kj}) + u''(r_{kj}) \right] \end{aligned} \tag{68}$$

Finally adding equations 60, 61, 62, 68 and the first term of equation 59, we have

$$\begin{aligned} \frac{1}{\Psi_T(\mathbf{R})} \nabla^2 \Psi_T(\mathbf{R}) &= \frac{\nabla_k^2 \Phi(\mathbf{r}_k)}{\Phi(\mathbf{r}_k)} + 2 \frac{\nabla_k \Phi(\mathbf{r}_k)}{\Phi(\mathbf{r}_k)} \cdot \sum_{j\neq k} \frac{\mathbf{r}_k - \mathbf{r}_j}{r_{kj}} u'(r_{kj}) \\ &\quad + \sum_{i,j\neq k} \frac{(\mathbf{r}_k - \mathbf{r}_i)(\mathbf{r}_k - \mathbf{r}_j)}{r_{ki} r_{kj}} u'(r_{kj}) + \sum_{j\neq k} \left(\frac{2}{r_{kj}} u'(r_{kj}) + u''(r_{kj}) \right) \end{aligned} \tag{69}$$