Variational Monte Carlo studies of quantum mechanical systems (WIP title)

Anna Stray Rongve Knut Magnus Aasrud
Amund Midtgard Raniseth

March 18, 2021

Abstract

1 Introduction

Finding the ground state of a confined Bose-gas can be a difficult task to do analytically. As such, we shall implement a variational Monte Carlo solver specialized for the problem at hand, and solve it numerically. This is done in Rust, a fast, safe and modern language.

We will first introduce the theory behind our approach and find an analytical solution for a simplified system. This is done to have a benchmark from which we can compare the performance of our numerical solver. Thereafter we present the methodology behind our codebase and how its implemented. The sections thereafter show our findings and a discussion regarding them.

2 Theory

The system in question is a hard sphere Bose gas located in a potential well. The potential is an *elliptical harmonic trap*, described for each particle by

$$V_{\text{ext}}(\mathbf{r}) = \frac{1}{2}m\left(\omega_{\text{ho}}^2(r_x^2 + r_y^2) + \omega_z^2 r_z^2\right). \tag{1}$$

Here, $\mathbf{r}$ is the position of the particle. Note that setting $\omega_{\text{ho}} = \omega_z$ results in eq. (1) evaluating to $V_{\text{ext}}(\mathbf{r}) = \frac{1}{2}m\omega_{\text{ho}}^2 r^2$, which represents the *spherical* case of the elliptical harmonic trap. As a simplification, we hereby denote the spherical case as (S) and the general elliptical case as (E).

In addition to this external potential, we represent the inter-boson interactions with the following pairwise, repulsive potential[1]:

$$V_{\text{int}}(|\mathbf{r}_i - \mathbf{r}_j|) = \begin{cases} \infty & |\mathbf{r}_i - \mathbf{r}_j| \leq a \\ 0 & |\mathbf{r}_i - \mathbf{r}_j| > a \end{cases}. \tag{2}$$

Eq. (1) and eq. (2) evaluate to the following two-body Hamiltonian:

$$H = \sum_i^N \left(-\frac{\hbar^2}{2m} \nabla_i^2 + V_{\text{ext}}(\mathbf{r}_i) \right) + \sum_{i<j}^N V_{\text{int}}(|\mathbf{r}_i - \mathbf{r}_j|). \tag{3}$$

The term $-\frac{\hbar^2}{2m}\nabla_i^2$ stems from the kinetic energy of the system and the index notation used is described in A.2.1. By scaling into length units of a_{ho} and energy units of $\hbar\omega_{\text{ho}}$, this equation is further simplified into:

$$H = \frac{1}{2} \sum_i^N \left(-\nabla_i^2 + r_{x,i}^2 + r_{y,i}^2 + \gamma^2 r_{z,i}^2 \right) + \sum_{i<j}^N V_{\text{int}}(|\mathbf{r}_i - \mathbf{r}_j|). \tag{4}$$

The derivation of (4) is explained in A.3.4.

2.1 The variational principle

Given the above Hamiltonian, we can introduce the concept of a *trial wave function* $\Psi_T(\alpha)$. This is a normalized ansatz to the ground state wave function, parametrized by the parameter(s) α. This gives us a way of deploying the *variational principle* by varying said parameter α to our needs:

We know that for any normalized function Ψ_T, the expected energy is higher than the ground state energy (as proved in [2] on p. 293-294), viz.

$$\langle E(\alpha) \rangle = \langle \Psi_T(\alpha) | H | \Psi_T(\alpha) \rangle \geq E_0 = \langle \Psi_0 | H | \Psi_0 \rangle. \tag{5}$$

Thus, minimizing over α will give an approximation of the true ground state (perhaps even an accurate answer).

Evaluating this integral is computationally demanding. Hence, we utilize Monte Carlo integration to allow scalability. This is done by changing the particles positions where the shifting follows some rules. For each change, the local energy is sampled resulting in an expectation value of the energy $\langle E \rangle$ for the Hamiltonian. Repeating this for different α's, we seek the value of α that results in the lowest energy and which hopefully approximates the ground state.

2.2 Wave function

For N particles, we use the following trial wave function:

$$\Psi_T(\mathbf{r}_1, ..., \mathbf{r}_N, \alpha, \beta) = \prod_i g(\alpha, \beta, \mathbf{r}_i) \prod_{j<k} f(a, |\mathbf{r}_j - \mathbf{r}_k|) \tag{6}$$

Once again, the index notation is described in A.2.1. Here we've used that

$$g(\alpha, \beta, \mathbf{r}_i) = e^{-\alpha(x_i^2 + y_i^2 + \beta z_i^2)},$$
$$\text{and } f(a, |\mathbf{r}_i - \mathbf{r}_j|) = \begin{cases} 0 & |\mathbf{r}_i - \mathbf{r}_j| \leq a \\ 1 - \frac{a}{|\mathbf{r}_i - \mathbf{r}_j|} & |\mathbf{r}_i - \mathbf{r}_j| > a \end{cases}.$$

Simplifying the trial wave function can prove useful, in order to reduce the number of floating point operations. An analytical expression is also convenient for comparison with the numerical calculations.

2.3 Importance sampling

Importance sampling, compared to the brute force Metropolis sampling, sets a bias on the sampling, leading it on a better path. This means that the desired standard deviation is acquired after fewer Monte Carlo cycles.

For our quantum mechanical scenario with boson particles in a magnetic trap, the bias has its root in the so-called quantum force. This quantum force pushes the walker (the boson particle) to the regions where the trail wave function is large. It is clear that this yields a faster convergence compared to the Metropolis algorithm, where the walker has the same probability of moving in all directions.

The quantum force $\mathbf{F}$ is given by the formula

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

which is derived from the Fokker-Planck equation, using the Langevin equation to generate the next step with Euler's method, and by making the probability density converge to a stationary state.

2.3.1 Fokker-Planck

For one particle (or walker), the one-dimensional Fokker-Planck equation for a diffusion process is:

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

Where $P(x,t)$ is a time-dependent probability density, D is the diffusion coefficient and F is a drift term which in our case is driven by the quantum force.

2.3.2 Langevin equation

The Langevin equation solution gives the position of the walker in the next timestep. The Langevin equation is:

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

Converting this to a function yielding the new position y in a computational manner, we use Euler's method.

$$y = x + DF(x)\Delta t + \xi\sqrt{\Delta t} \tag{7}$$

Where the symbols represent:

Table 1: Explanation of variables used in (7).

Variable	**Description**
y	New position
x	Current position
DF(x)	Diffusion and Drift at the old position
D	In atomic units: 1/2, from the kinetic energy operator
Δt	Chosen time-step
ξ	Gaussian random variable

Examples of timesteps giving stable values of the ground state energy is $\Delta t \in [0.001, 0.01]$

2.3.3 Fokker-Planck and Langevin equation in importance sampling

In order to use these equations for our importance sampling, we start with the original Fokker-Planck equation.

After inserting D as the diffusion coefficient and $\mathbf{F_i}$ as component i of the drift velocity, we can make the probability density converge to a stationary state by setting its partial derivative over time to zero.

$$\frac{\partial P}{\partial t} = \sum_i D\frac{\partial}{\partial \mathbf{x_i}}\left(\frac{\partial}{\partial \mathbf{x_i}} - \mathbf{F_i}\right)P(\mathbf{x},t)$$

Where then $\frac{\partial P}{\partial t} = 0$, and by expanding the parenthesis and moving the double partial derivative over to the other side, we obtain:

$$\frac{\partial^2 P}{\partial \mathbf{x_i}^2} = P\frac{\partial}{\partial \mathbf{x_i}}\mathbf{F_i} + \mathbf{F_i}\frac{\partial}{\partial \mathbf{x_i}}P$$

By inserting $g(\mathbf{x})\frac{\partial P}{\partial x}$ for the drift term, $\mathbf{F}$, we get

$$\frac{\partial^2 P}{\partial \mathbf{x_i}^2} = P\frac{\partial g}{\partial P}\left(\frac{\partial P}{\partial \mathbf{x}_i}\right)^2 + Pg\frac{\partial^2 P}{\partial \mathbf{x}_i^2} + g\left(\frac{\partial P}{\partial \mathbf{x}_i}\right)^2$$

Where again the left hand side can be set to zero to comply with the fact that at a stationary state, the probability density is the same for all walkers. [THIS MUST BE FALSE??? WHY can we really set this term to zero??]

For this to be solvable, the remaning terms have to cancel each other. This is only possible when $g = P^{-1}$, which gives the aformentioned quantum force, $\mathbf{F}$,

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

From here, The Green's function is deployed as

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

Which will be part of the proposal distribution, $q(y, x)$ as

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

2.4 Analytical derivations

2.4.1 Local energy simple Gaussian wave function

As a test case to be compared against our numerical implementation, we want to find an analytical expression for the energy of the trial wave function(Ref)(local energy). We only study the harmonic oscillator potential and disregard the two-body potential. This is simply done by setting the parameter $a = 0$ which by (2) gives $V_{\text{int}} = 0$. First β is set to 1 to find the relevant local energies for one to three dimensions for both one and N particles. The simplest Gaussian wavefunction then becomes:

$$\Psi_T(\mathbf{r_1}, \mathbf{r_2}, \ldots, \mathbf{r_N}, \ , \ ) = \prod_i \exp(-\alpha r_i^2).$$

The energy is here given by

$$E_L(\mathbf{r}) = \frac{1}{\Psi_T(\mathbf{r})} H\Psi_T(\mathbf{r}) = \frac{1}{\Psi_T(\mathbf{r})} \left[\sum_i^N \left(\frac{-\hbar^2}{2m} \nabla_i^2 + V_{\text{ext}}(\mathbf{r}_i) \right) \right] \Psi_T(\mathbf{r})$$
$$= \frac{1}{\Psi_T(\mathbf{r})} \left[\sum_i^N \left(\frac{-\hbar^2}{2m} \nabla_i^2 \Psi_T(\mathbf{r}) + V_{\text{ext}}(\mathbf{r}_i) \Psi_T(\mathbf{r}) \right) \right].$$

We simplify $\nabla_i^2 \Psi_T$ as shown in A.3.1 to get

$$\nabla^2 \Psi_t(\mathbf{r}) = -2\alpha \Psi_T \left(\dim -2\alpha \mathbf{r}_i^2 \right), \tag{8}$$

where dim is the dimension of the system (1, 2 or 3). Given eq. (8), we find that the local energy for N particles in the case of the simple Gaussian wavefunction is

$$E_L(\mathbf{r}) = \frac{\hbar^2}{m} \alpha N \dim + \left(\frac{1}{2} m \omega_{\text{ho}}^2 - 2\alpha^2 \right) \sum_i^N \mathbf{r}_i^2, \tag{9}$$

as shown in A.3.2. We can simplify this even further by scaling, namely setting $\hbar = m = 1$, which gives us the equation

$$E_L(\mathbf{r}) = N\alpha \dim + \left(\frac{1}{2} m \omega_{\text{ho}}^2 - 2\alpha^2 \right) \sum_i^N \mathbf{r}_i^2 \tag{10}$$

An even simpler analytic expression is obtained by setting $\omega_{\text{ho}} = 1$ and taking the derivate of the local energy with respect to r_i, giving $\alpha = 0.5$.

$$E_L = \frac{N \dim}{2}$$

2.4.2 Drift force

The following expression for the drift force will be used to **explanation**

$$F = \frac{2\nabla_k \Psi_T(\mathbf{r})}{\Psi_T(\mathbf{r})} = -4\alpha \mathbf{r}_k$$

applying the gradient operator to the trail wavefunction is already shown (appendix: Second derivative of trial wave function).

2.4.3 Local energy for full wave function

With $\beta \neq 0$ and a > 0 the wave function becomes a bit more complicated as the potential/Gaussian can be can now be elliptical and the wave function contains the Jastrow factor. Difficult to find an analytical expression for the derivate of the trail wave function.

$$E_L(\mathbf{r}) \frac{1}{\Psi_T(\mathbf{r})} \sum_i^N \nabla_i^2 \Psi_T(\mathbf{r}),$$

Rewriting the full wave function

$$\Psi_T(\mathbf{r}) = \Psi_T(\mathbf{r}_1, \mathbf{r}_2, ... \mathbf{r}_N, \alpha, \beta) = \left[\prod_i g(\alpha, \beta, \mathbf{r}_i) \right] \left[\prod_{j<k} f(a, |\mathbf{r}_j - \mathbf{r}_k|) \right],$$

to the following

$$\Psi_T(\mathbf{r}) = \left[\prod_i^N \phi(\mathbf{r}_i) \right] \exp\left(\sum_{i<j} u(r_{ij}) \right)$$

where

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

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

$$r_{ij} = |r_i - r_j|$$

The first derivative for the k'th particle then is a bit tricky to calculate, so the result will be presented her while the full calculation is in **REF APPENDIX**. The analytical expression becomes

$$\nabla_k \Psi_T(\mathbf{r}) = \nabla_k \phi(\mathbf{r}_4 k) \left[\prod_{i \neq k}^N \phi(\mathbf{r}_k) \right] \exp\left(\sum_{j<m}^N u(r_{jm}) \right) \left[\prod_i^N \phi(\mathbf{r}_i) \right] \exp\left(\sum_{j<m}^N u(r_{jm}) \right) \sum_{l \neq k}^N \nabla_k(r_{kl}),$$

The Laplacian is derived in **REF APPENDIX** resulting in the following analytical expression

$$\begin{aligned}\frac{1}{\Psi_T(\mathbf{r})}\nabla_k^2\Psi_T(\mathbf{r}) = \frac{\nabla_k\phi(\mathbf{r}_k)}{\phi(\mathbf{r}_k)} &+ 2\frac{\nabla_k\phi(\mathbf{r}_k)}{\phi(\mathbf{r}_k)}\sum_{j\neq k}\frac{\mathbf{r}_j-\mathbf{r}_k}{\mathbf{r}_{jk}}u'(r_{lk}) \\ &+ \sum_{j\neq k}\sum_{l\neq k}\frac{\mathbf{r}_j-\mathbf{r}_k}{\mathbf{r}_{jk}}u'(r_{lk}) \\ &+ \sum_{j\neq k}\sum_{l\neq k}\frac{\mathbf{r}_j-\mathbf{r}_k}{\mathbf{r}_{jk}}\frac{\mathbf{r}_l-\mathbf{r}_k}{\mathbf{r}_{lk}}u'(r_{jk})u'(r_{lk}) \\ &+ \sum_{l\neq k}\frac{2}{r_{lk}}u'(r_{lk}) + u''(r_{lk})\end{aligned}$$

Where

$$\frac{\nabla_k\phi(\mathbf{r}_k)}{\phi(\mathbf{r}_k)} = -2\alpha\begin{bmatrix}x_k^2\\ y_k^2\\ \beta z_k^2\end{bmatrix},$$

$$\frac{\nabla_k^2\phi(\mathbf{r}_k)}{\phi(\mathbf{r}_k)} = 2\alpha(2\alpha)[x_k^2+y_k^2+\beta^2 z_k^2] - 2 - \beta),$$

$$u'(r_{ij}) = \frac{r_{ij}}{r_{ij}-a}, \quad \text{for} \quad r_{ij} > a,$$

$$u''(r_{ij}) = \frac{a(a-2r_{ij})}{r_{ij}^2(a-r_{ij})^2}, \quad \text{for} \quad r_{ij} > a,$$

3 Method

Rust is used to preform the numerical calculations. ## Variational Monte Carlo

Firstly the ground state energy of spherical harmonic oscillator ($\beta = 1$) was calculated using Variational Monte Carlo with standard Metropolis sampling. To get the code up and running we started with the simplest scenario in one dimension without any interaction. Neutral units were implemented. Comparing the calculations from the analytic expression and the ones from the numerical calculated kinetic energy in order to compare CPU time for the two. We also compared CPU time for different number of atoms (N = 1 , 10 , 100 and 500). Thereafter the same was done in two and three dimensions.

3.1 Natrual length scale

1) Avoid underflow
2) Most quantum mechanical sytems have energy in units of $\hbar\omega$

Factor out $\hbar\omega_{ho}$ from the Hamiltonian

$$H = \frac{1}{2}\sum_{i}^{N}\left(-\frac{\hbar^2}{m}\ \nabla_i^2 + m\left[\omega_{ho}(x_i^2+y_i^2)+\omega_z^2 z_i^2\right]\right)$$

$$= \frac{\hbar\omega_{ho}}{2}\sum_{i}^{N}\left(-\frac{\hbar}{m\omega_{ho}}\nabla_i^2 + \frac{m\omega_{ho}}{\hbar}\left[x_i^2+y_i^2+\frac{\omega_z^2}{\omega_{ho}^2}z_i^2\right]\right)$$

$\frac{\hbar\omega_{ho}}{2}$ has the unit of length, hence $\frac{\hbar\omega_{ho}}{2} = \sqrt{\text{length}}$, which is a natrual length scale. Defining

$$a_{ho} = \sqrt{\frac{\hbar}{m\omega_{ho}}},$$

and

$$\gamma = \frac{\omega_z}{\omega_{ho}}$$

Divide the Hamiltoninan with the factor of a to be able to express the energy in units of $\hbar\omega$

$$H = \frac{1}{2}\sum_{i}^{N}\left(-a_{ho}^2\ \nabla_i^2 + a_{ho}^{-2}\left(x_i^2+y_i^2+\gamma^2 z_i^2\right)\right)$$

As for coordinates and particle(boson) diameter (scaling x_i, y_i, z_i and the Laplacian) by doing as follows

$$a_0 = \frac{a}{a_{ho}}, \quad \mathbf{r_0} = \frac{\mathbf{r}}{a_{ho}}$$

Scaled Hamiltonian

$$H = \frac{1}{2}\sum_{i}^{N}\left(-\nabla_i^2 + x_i^2+y_i^2+\gamma^2 z_i^2\right)$$

Fixed parameters (to only have one variational paramter): Boson diamter: $\frac{a}{a_{ho}} = ??$

$\gamma = \beta = ...$

3.2 Auto-vectorization

Auto-vectorization in Rust is almost as easy as in C++, and can be applied by setting `RUSTFLAGS = "-C opt-level=3 -C target-cpu=native"` in the *Cargo.toml* file, which basically inputs the parameters to the compiler at compiletime. The first flag tells the compiler to run all possible optimizations. Setting *opt-level=2* is the same as running the alias *-O* which only runs some optimizations cite: [[https://doc.rust-lang.org/rustc/codegen-options/index.html#opt-level]]. *target-cpu* tells the compiler which cpu to compile specific code for. By inserting *native*, the compiler will compile for the cpu the compiler is run at. cite:[[https://doc.rust-lang.org/rustc/codegen-options/index.html#target-cpu]]

However, simple loops like `for i in 0..n` will not be properly vectorized due to the fact that the compiler cannot guarantee that the length of the loop is within bounds of the slice iterated over. The easiest way to ensure that this does not happen is to use an interator. If this cannot be done, hinting to LLVM the length of the slice would also eliminate the bound checks. An example is to define the slice as `let x = &x[0..n];` # Results

4 Discussion

5 Conclusion

A Appendix

A.1 Source code

All source code for both the Rust VMC implementation and this document is found on the following GitHub Repository

https://github.com/kmaasrud/vmc-fys4411

A.2 Notation and other explanations

A.2.1 Index notation for sums and products

For products and sums, the following convention is used:

$$\sum_{i<j}^{N} = \sum_{i=1}^{N}\sum_{j=i+1}^{N}, \quad \text{or} \quad \prod_{i<j}^{N} = \prod_{i=1}^{N}\prod_{j=i+1}^{N}$$

A.3 Calculations

A.3.1 Second derivative of trial wave function

$$\begin{aligned}
\nabla_i^2\Psi_T(\mathbf{r}) &= \nabla_i \cdot \left[\frac{\partial}{\partial x_i}, \frac{\partial}{\partial y_i}, \frac{\partial}{\partial z_i}\right]\Psi_T(\mathbf{r}) \\
&= \nabla_i \cdot \left[\frac{\partial}{\partial x_i}\exp\left(-\alpha\mathbf{r}_i^2\right), \frac{\partial}{\partial y_i}\exp\left(-\alpha\mathbf{r}_i^2\right), \frac{\partial}{\partial z_i}\exp\left(-\alpha\mathbf{r}_i^2\right)\right] \\
&= \nabla_i \cdot \left[-2\alpha x_i\exp\left(-\alpha\mathbf{r}_i^2\right), -2\alpha y_i\exp\left(-\alpha\mathbf{r}_i^2\right), -2\alpha z_i\exp\left(-\alpha\mathbf{r}_i^2\right)\right] \\
&= -2\alpha\left[\exp\left(-\alpha\mathbf{r}_i^2\right)(1-2\alpha x_i^2), \exp\left(-\alpha\mathbf{r}_i^2\right)(1-2\alpha y_i^2), \exp\left(-\alpha\mathbf{r}_i^2\right)(1-2\alpha z_i^2)\right] \\
&= -2\alpha\Psi_T\left[(1-2\alpha x_i^2), (1-2\alpha y_i^2), (1-2\alpha z_i^2)\right] \\
&= -2\alpha\Psi_T\sum_{d=x,y,z} 1-2\alpha d_i^2 \\
&= -2\alpha\Psi_T(\dim -2\alpha\mathbf{r}_i^2)
\end{aligned}$$

A.3.2 Local energy for Gaussian wave function

Starting with

$$E_L(\mathbf{r}) = \frac{1}{\Psi_T(\mathbf{r})}\left[\sum_i^N\left(\frac{-\hbar^2}{2m}\nabla_i^2\Psi_T(\mathbf{r}) + V_{\text{ext}}(\mathbf{r}_i)\Psi_T(\mathbf{r})\right)\right],$$

and using the result from A.3.1, this results in:

$$E_L(\mathbf{r}) = \frac{1}{\Psi_T(\mathbf{r})}\left[\sum_i^N \left(\frac{\hbar^2\alpha}{m}(\dim -2\alpha\mathbf{r}_i^2) + \frac{1}{2}m\omega_{\text{ho}}^2\mathbf{r}_i^2\right)\Psi_T(\mathbf{r})\right]$$
$$= \frac{\hbar^2}{m}\alpha N\dim + \left(\frac{1}{2}m\omega_{\text{ho}}^2 - 2\alpha^2\right)\sum_i^N \mathbf{r}_i^2$$

A.3.3 Gradient and Laplacian for trial wave function general case

Gradient Rewriting the wave function to

$$\Psi_T(\mathbf{r}) = \left[\prod_i^N \phi(\mathbf{r}_i)\right]\exp\left(\sum_{i<j} u(r_{ij})\right)$$

where $r_{ij} = |r_i - r_j|$ and we set $u(r_{ij}) = \ln f(r_{ij})$. Lastly $g(\alpha, \beta, \mathbf{r}_i)$ is redefined to the following function

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

For convenience

$$\Psi_1(\mathbf{r}_i) = \prod_i^N \phi(\mathbf{r}_i)$$

and

$$\Psi_2(\mathbf{r}_{ij}) = \exp\left(\sum_{i<j} u(r_{ij})\right)$$

where Ψ_1 and Ψ_2 is the one-body and correlated part of the wave function, respectively. Both parts have simple dependency of the k'th particle. Ψ_1 is a product of one-body wave functions with only one factor dependent of $\mathbf{r}_k$ and Ψ_2 is $\mathbf{r}_k$ - dependent for the pairs $\sum_{i\neq k} u(\mathbf{r}_{ik})$. Hence the first derivatives becomes

$$\nabla_k\Psi_1(\mathbf{r}) = \left[\prod_{i\neq k}^N \phi(\mathbf{r}_i)\right]\nabla_k\phi(\mathbf{r}_k)$$

$$\nabla_k\Psi_2(\mathbf{r}_{ij}) = \exp\left(\sum_{i<j} u(r_{ij})\right)\sum_{i\neq k}\nabla_k u(\mathbf{r}_{ik})$$

Giving the first derivate of the trail wave function

$$\nabla_k \Psi_T(\mathbf{r}) = \nabla_k \phi(\mathbf{r}_k) \left[\prod_{i \neq k}^{N} \phi(\mathbf{r}_i) \right] \exp\left(\sum_{i<j} u(r_{ij}) \right) + \prod_{i}^{N} \phi(\mathbf{r}_i) \exp\left(\sum_{i<j} u(r_{ij}) \right) \sum_{i \neq k} \nabla_k u(\mathbf{r}_{ik})$$

Laplacian The Laplacian of the wacefunction needs to be evaluated in order to calculate

$$\frac{1}{\Psi_T(\mathbf{r})} \nabla_k \nabla_k \Psi_T(\mathbf{r})$$

The last part, $\nabla_k \Psi_T(\mathbf{r})$ is calculated in the section above / equation (**Reference here**). Next step is then to calculate

$$\begin{aligned}
\nabla_k \nabla_k \Psi_T(\mathbf{r}) =& \nabla_k \left(\nabla_k \phi(\mathbf{r}_k) \left[\prod_{i \neq k} \phi(\mathbf{r}_i) \right] \exp\left(\sum_{j<m} u(r_{jm}) \right) \right. \\
&\left. + \prod_{i} \phi(\mathbf{r}_i) \exp\left(\sum_{j<m} u(r_{jm}) \right) \sum_{l \neq k} \nabla_k u(\mathbf{r}_{kl}) \right) \\
\nabla_k \nabla_k \Psi_T(\mathbf{r}) =& \prod_{i \neq k} \left[\nabla_k^2 \phi(\mathbf{r}_k) \exp\left(\sum_{j<m} u(r_{jm}) \right) + \nabla_k \phi(\mathbf{r}_k) \cdot \nabla_k \exp\left(\sum_{j<m} u(r_{jm}) \right) \right] \\
&+ \nabla_k \prod_{i} \phi(\mathbf{r}_i) \exp\left(\sum_{j<m} u(r_{jm}) \right) \sum_{l \neq k} \nabla_k u(r_{kl}) \\
&+ \nabla_k \exp\left(\sum_{j<m} u(r_{jm}) \right) \prod_{i} \phi(\mathbf{r}_i) \sum_{l \neq k} \nabla_k u(r_{kl}) \\
&+ \nabla_k \sum_{l \neq k} \nabla_k u(r_{kl}) \prod_{i} \phi(\mathbf{r}_i) \exp \sum_{j<m} u(r_{jm})
\end{aligned}$$

In order to avoid writing long calculations, the three main gradients are calculated below. The last of the three following expressions/equations is a bit more of a hazard to calculate. First the product rule is used. Then a rule for the gradient is applied where the gradient of a unit vector is 2 divided by its magnitude. u' is parallel to the unit vector, hence their product becomes a scalar, the second derivate of u.

$$\nabla_k \exp\left(\sum_{j<m} u(r_{jm} \right) = \exp\left(\sum_{j<m} u(r_{jm} \right) \sum_{l \neq k} \nabla_k u(r_{kl})$$

$$\nabla_k \prod_{i} \phi(\mathbf{r}_i) = \prod_{i \neq k} \phi(\mathbf{r}_i) \nabla_k \phi(\mathbf{r}_k)$$

$$\begin{aligned}\nabla_k \sum_{l\neq k} \nabla_k u(r_{kl}) &= \sum_{l\neq k} \nabla_k \left(\frac{\mathbf{r}_l - \mathbf{r}_k}{\mathbf{r}_{lk}} u'(r_{lk}) \right) \\ &= \sum_{l\neq k} \left(\nabla_k \frac{\mathbf{r}_l - \mathbf{r}_k}{\mathbf{r}_{lk}} u'(r_{lk}) + \frac{\mathbf{r}_l - \mathbf{r}_k}{\mathbf{r}_{lk}} \nabla_k u'(r_{lk}) \right) \\ &= \sum_{l\neq k} \frac{2}{r_{lk}} + u''(r_{lk})\end{aligned}$$

Finally the Laplacian can be calculated, by reintroducing the fraction $\frac{1}{\Psi_T(\mathbf{r})}$

$$\begin{aligned}\frac{1}{\Psi_T(\mathbf{r})} \nabla_k^2 \Psi_T(\mathbf{r}) =& \frac{\prod_{i\neq k} \phi(\mathbf{r}_i)}{\prod_i \phi(\mathbf{r}_i)} \left(\nabla_k^2 \phi(\mathbf{r}_k) + \nabla_k \phi(\mathbf{r}_k \sum_{l\neq k} \nabla_k u(r_{kl}) \right) + \left(\frac{\nabla_k \phi(\mathbf{r}_i)}{\phi(\mathbf{r}_i)} \sum_{l\neq k} \nabla_k u(r_{kl}) \right) \\ &+ \sum_{l\neq k} \nabla_k u(r_{kl}) + \sum_{j\neq k} \nabla_k u(r_{kj}) + \nabla_k \sum_{l\neq k} \nabla_k u(r_{kl})\end{aligned}$$

The second and third terms are the same. Two of the terms are shown in the calculations above and $\nabla_k u(r_{kl})$ is the unit vector multiplied with the derivate of a scalar. Then we have the final expression

$$\begin{aligned}\frac{1}{\Psi_T(\mathbf{r})} \nabla_k^2 \Psi_T(\mathbf{r}) =& \frac{\nabla_k \phi(\mathbf{r}_k)}{\phi(\mathbf{r}_k)} + 2 \frac{\nabla_k \phi(\mathbf{r}_k)}{\phi(\mathbf{r}_k)} \sum_{j\neq k} \frac{\mathbf{r}_j - \mathbf{r}_k}{\mathbf{r}_{jk}} u'(r_{lk}) \\ &+ \sum_{j\neq k} \sum_{l\neq k} \frac{\mathbf{r}_j - \mathbf{r}_k}{\mathbf{r}_{jk}} u'(r_{lk}) + \sum_{j\neq k} \sum_{l\neq k} \frac{\mathbf{r}_j - \mathbf{r}_k}{\mathbf{r}_{jk}} \frac{\mathbf{r}_l - \mathbf{r}_k}{\mathbf{r}_{lk}} u'(r_{jk}) u'(r_{lk}) \\ &+ \sum_{l\neq k} \frac{2}{r_{lk}} u'(r_{lk}) + u''(r_{lk})\end{aligned}$$

A.3.4 Scaling of repulsion Hamiltonian

We have the initial expression for the Hamiltonian, (3). Inserting (1), we get:

$$H = \frac{1}{2} \sum_i^N \left(-\frac{\hbar^2}{m} \nabla_i^2 + m \left(\omega_{\text{ho}}^2 (r_{x,i}^2 + r_{y,i}^2) + \omega_z^2 r_{z,i}^2 \right) \right) + \sum_{i<j}^N V_{\text{int}}(|\mathbf{r}_i - \mathbf{r}_j|).$$

We now introduce the scaled length unit $r' = \frac{r}{a_{\text{ho}}}$ which in turn leads to $\nabla_i'^2 = a_{\text{ho}}^2 \nabla_i^2$.

$$H = \frac{1}{2}\sum_{i}^{N}\left(-\frac{\hbar^2}{ma_{\rm ho}^2}\nabla_i'^2 + ma_{\rm ho}^2\left(\omega_{\rm ho}^2(r_{x,i}'^2 + r_{y,i}'^2) + \omega_z^2 r_{z,i}'^2\right)\right) + \sum_{i<j}^{N} V_{\rm int}(|\mathbf{r}_i - \mathbf{r}_j|)$$

Inserting the definition of $a_{\rm ho} = \frac{\hbar}{m\omega_{\rm ho}}$, we get

$$H = \frac{1}{2}\sum_{i}^{N}\left(-\hbar\omega_{\rm ho}\nabla_i'^2 + \hbar\omega_{\rm ho}\left((r_{x,i}'^2 + r_{y,i}'^2) + \frac{\omega_z^2}{\omega_{\rm ho}^2} r_{z,i}'^2\right)\right) + \sum_{i<j}^{N} V_{\rm int}(|\mathbf{r}_i - \mathbf{r}_j|),$$

$$H = \frac{\hbar\omega_{\rm ho}}{2}\sum_{i}^{N}\left(-\nabla_i'^2 + (r_{x,i}'^2 + r_{y,i}'^2) + \gamma^2 r_{z,i}'^2)\right) + \sum_{i<j}^{N} V_{\rm int}(|\mathbf{r}_i - \mathbf{r}_j|),$$

where $\gamma = \frac{\omega_z}{\omega_{\rm ho}}$. We lastly reorganize the above to obtain a scaled Hamiltonian $H' = \frac{H}{\hbar\omega_{\rm ho}}$ and also make sure to scale the function $V_{\rm int} \to V_{\rm int}'$ by transitioning from $a \to a' = \frac{a}{a_{\rm ho}}$.

$$H' = \frac{1}{2}\sum_{i}^{N}\left(-\nabla_i'^2 + r_{x,i}'^2 + r_{y,i}'^2 + \gamma^2 r_{z,i}'^2\right) + \sum_{i<j}^{N} V_{\rm int}'(|\mathbf{r}_i' - \mathbf{r}_j'|). \tag{11}$$

By ensuring that we used scaled length units of $r' = \frac{r}{a_{\rm ho}}$ and scaled energy units of $E' = \frac{E}{\hbar\omega_{\rm ho}}$, equation (11) holds. For simplification, we will not use the primed notation outside this derivation.

References

[1] M. Hjorth-Jensen, "Project 1." Jan. 2021.

[2] David. J. Griffiths, *Introduction to Quantum Mechanics*. 2005.

[3] K. M. Aasrud, A. S. Rongve, and A. M. Raniseth, "Project 3." Oct. 2019, [Online]. Available: https://github.com/kmaasrud/gq-mcm-fys3150/blob/master/doc/Project-3_Aasrud-Raniseth-Rongve.pdf.