Ultra Cold = Ultra Cool
A Numerical Study of Trapped Bosons Devising a Variational Monte Carlo Solver
FYS4411 - Project 1

Brage Brevig and Karl Henrik Fredly
(Dated: April 6, 2021)

Employing a Variational Monte Carlo Method, we study quantum mechanical properties in systems of N trapped bosons. In light of the performance of our program, we find that the VMC solver produces results which are in excellent agreement with benchmark analytical results. Implementing importance sampling and parallelizing our code, we discover that there is a potential for doubling the number of accepted steps in the Metropolis - Hastings algorithm, at the cost of doubling the CPU run time - which is potentially remedied by distributing the work load across several threads when parallelizing, where we can report up towards a factor of 1.5 in terms of CPU time speed - up having four available core threads. In terms of results of interest pertaining to the physical system we are studying, we confirm that higher densities in the systems leads to greater energies and delocalization of the individual particles. The same effect is observed when increasing the correlation strength between the particles. The obtained energies for different systems are tabulated, and where possible, compared against analytical or referenced values with low relative errors. We report standard errors in the calculation of the energies on the order of 10^{-6} at its lowest and relative error to the referenced values on the order of 10^{-4} at the most.

I. INTRODUCTION

The scope of this numerical project is two-fold. For one, we wish to develop a professional - like Variational Monte Carlo (VMC) solver by implementing a handful of key numerical architectures such as importance sampling, optimization methods and resampling methods into an otherwise straight forward Markov Chain Monte Carlo algorithm. Our remaining business then will be to study N-body systems of trapped bosons, where the ultimate aim is to fully describe large bosonic systems whilst also realizing boson-boson interactions. Essentially, we are doing preliminary work for studying Bose - Einstein Condensates (BECs). Although BECs is a theoretical tool more than anything else, it has become increasingly important to understand these ultra cold, coherent systems as they lay the foundation for future research into areas such as quantum magnetism, low dimensional disorder and non-equilibrium dynamics in quantum systems [1].

Going forward, we shall establish the necessary physical and mathematical framework for working with trapped bosons, before moving on to exploring the various numerical methods employed in this study. We shall then present selected results from our studies of bosonic systems, and discuss these in light of both their physical interpretation and their validity in terms of how they were obtained. Lastly, we make our summary remarks on the project in its entirety.

II. THEORY & BACKGROUND

A. Trapped Bosonic Systems

It will be our business to study and evaluate the ground state properties of a hard sphere Bose gas devising a VMC solver with a specific trial wave function. We shall study systems of N bosons in which the particles are trapped in both spherical and elliptical harmonic traps. The traps may be one -, two - or three - dimensional. The harmonic potential for each case is given by

$$V_{HO}(x,y,z) = \begin{cases} \frac{1}{2}m\omega_0^2 r^2 & (S) \\ \frac{1}{2}m\left(\omega_0^2(x^2+y^2)+\omega_z^2 z^2\right) & (E) \end{cases} \tag{1}$$

where ω_0 denotes the potential strength of the harmonic trap. In the case of an elliptical trap, ω_0 is a measure of potential strength in the xy - plane, whilst ω_z measures the potential strength in the z - direction. The N-body Hamiltonian is given by

$$H = \sum_i^N \left(-\frac{\hbar}{2m}\nabla_i^2 + V_{HO}(x_i,y_i,z_i)\right) + \sum_{i<j} V_{bb}(\mathbf{r}_i,\mathbf{r}_j) \tag{2}$$

with an interaction potential V_{bb} governed by

$$V_{bb}(\mathbf{r}_i,\mathbf{r}_j) = \begin{cases} \infty & \text{if } |\mathbf{r}_i-\mathbf{r}_j| \leq a \\ 0 & \text{if } |\mathbf{r}_i-\mathbf{r}_j| > a \end{cases} \tag{3}$$

for a value a pertaining to the hard-core diameter of the spherical bosons. The repulsive interaction becomes infinitely large if the bosons touch or overlap, and is otherwise always zero.

In this project we employ a trial wave function on the form

$$\Psi(\mathbf{r}_1, \mathbf{r}_2, ..., \mathbf{r}_N, \alpha, \beta) = \prod_i g(\mathbf{r}_i, \alpha, \beta) \prod_{j<k} f(a, \mathbf{r}_j, \mathbf{r}_k) \tag{4}$$

making the ansatz that the wave function should be a product of single-particle wave functions g and correlation wave functions f. Here α and β are variational parameters in the single-particle wave function proportional to the ground state wave function for the classical harmonic oscillator

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

and the correlation wave functions f are defined as following

$$f(a, \mathbf{r}_j, \mathbf{r}_k) = \begin{cases} 0 & \text{if } |\mathbf{r}_j - \mathbf{r}_k| \leq a \\ 1 - \frac{a}{|\mathbf{r}_j - \mathbf{r}_k|} & \text{if } |\mathbf{r}_j - \mathbf{r}_k| > a \end{cases} \tag{6}$$

B. The Variational Method

The Variational Method is a quantum mechanical tool for studying the ground state, and some excited states, in systems where the exact wave function is unknown to us. Having knowledge about the Hamiltonian of the system, we make an educated guess towards the shape and behaviour of the ground state wave function, making a trial wave function $\Psi_T(\mathbf{r}\ ; \alpha, \beta, ...)$ depending on one or more *variational* parameters. It is our business then, using the Variational Method, to find values for these parameters such that the expectation value of the ground state energy is minimized. Fixing the parameters with the obtained values, we have made an estimate of the upper bound of the ground state energy, and also an approximation to the ground state wave function. Speaking the language of quantum mechanics, we know that the expectation value of the Hamiltonian $\langle H \rangle$ is computed through $\langle \varphi | H | \varphi \rangle$ using bra-ket notation. Any valid wave function, including our own trial wave function, must be assumed to be a linear combination of the Hamiltonians eigenfunctions

$$\varphi = \sum_n c_n \psi_n$$

which are normalized and orthogonal. The computation of the expectation value then becomes an inner product

$$\langle H \rangle = \langle \sum_m c_m \psi_m | H \sum_n c_n \psi_n \rangle = \sum_n \sum_m c_m^\star c_n E_n \langle \psi_m | \psi_n \rangle$$

or

$$\langle H \rangle = \sum_n c_n^2 E_n \geq E_0$$

implying that when we compute the expected energy for a given system using a trial wave function, it must be greater than or equal to the ground state energy. Since our trial wave function depends upon a variational parameter, so does the expected energy. Minimizing the expected energy then, ensures that we are estimating the lowest possible energy of the system - the ground state energy.

C. One Body Density

In order to better understand the ground state we are studying, a helpful quantity to explore is the one-body density. The one-body density is the probability density for a single particle and is given by

$$\rho(\mathbf{r}_1) = \int |\Psi(\mathbf{r}_1, \mathbf{r}_2, ..., \mathbf{r}_N)|^2 d\mathbf{r}_2 \mathbf{r}_3 ... \mathbf{r}_N \tag{7}$$

To estimate this integral, we could sample the position of a chosen particle at each step in the calculation. However, since the system we are analyzing has spherical symmetry in the case $\beta = 1$, and all particles share the same wavefunction, we will instead sample the distance from origo for all particles at each step. Storing each distance is expensive, so we will instead count how often a particle is in each distance interval from origo, which will give us a histogram showing the one-body density.

III. NUMERICAL METHODS AND IMPLEMENTATION

A. Brute Force Metropolis Algorithm

Going forward we shall describe how we have implemented the Metropolis Algorithm which serves as the backbone to our end result - a professional - like VMC code. The Metropolis Algorithm is a Brute Force Method for studying systems in which stochastic processes occur, and works on the principle of sampling many different configurations of the system at hand and using the obtained set of samples to draw statistical conclusions about quantities like mean energy and energy variance. In our case, we wish to estimate the ground state energy of systems of trapped bosons by iteratively changing the configuration of the system and sampling the local energies E_L with

$$\langle E_L \rangle = \int d\mathbf{r} P(\mathbf{r}) E_L(\mathbf{r})$$

for which $P(\mathbf{r})$ describes the probability distribution

$$P(\mathbf{r}) = \frac{|\Psi(\mathbf{r}\ ; \alpha)|^2}{\int |\Psi(\mathbf{r}\ ; \alpha)|^2 d\mathbf{r}} \tag{8}$$

and

$$E_L(\mathbf{r}\ ; \alpha) = \frac{1}{\Psi(\mathbf{r}\ ; \alpha)} H \Psi(\mathbf{r}\ ; \alpha)$$

describes the local energy. This way, we avoid evaluating the multidimensional integral pertaining to the normalization factor in the probability distribution, by rather employing Monte Carlo integration such that

$$\langle E_L \rangle = \int d\mathbf{r} P(\mathbf{r}) E_L(\mathbf{r}) \sim \frac{1}{N} \sum_{i=1}^{N} E_L(\mathbf{r}_i \, ; \alpha) \tag{9}$$

However, before we may begin sampling the local energies, we must allow the system to reach equilibrium as we are dealing with a Markov Chain process. The process of reaching equlibrium, or *thermalizing*, is very slow if the spawned random walkers are allowed to span the entire configuration space. We therefore impose a set of rules that in theory guarantees steady thermalization - namely, detailed balance and ergodicity. Detalied balance may be achieved by demanding that

$$P_i W(i \to j) = P_j W(j \to i)$$

where $W(i \to j)$ describes the transition probability between two states i and j, and P_i describes the probability density at state i. If we also make sure that the random walkers may reach any state also being a part of the distribution function we have ensured that our sampling is ergodic. The Metropolis algorithm bases itself on these two principles, and by introducing a selection probability $T(i \to j)$ and acceptance probability $A(i \to j)$ we model the transition probability as the product of these two. Inserting this model into our required detailed balance definition, we see that

$$P_i T(i \to j) A(i \to j) = P_j T(j \to i) A(j \to i)$$

$$\implies \frac{A(j \to i)}{A(i \to j)} = \frac{P_j T(i \to j)}{P_i T(j \to i)}$$

We are sampling from a distribution $P = |\Psi|^2$, and due to the requirement of detailed balance the probability of selecting state j given state i is equal to the probability of selecting state i given state j, so we may simplify further

$$\frac{A(j \to i)}{A(i \to j)} = \frac{|\Psi_j|^2}{|\Psi_i|^2}$$

A simple recipe for implementing the Metropolis algorithm then becomes to

- Initialize N particles in a random configuration
- Select a random walker and suggest the transition into a new state by moving the walker randomly
- Compute the acceptance ratio as in REF and compare it against a random number drawn between 0 and 1 - if the random number is smaller than the acceptance ratio, we accept the proposed transition and spawn a new random walker, repeating the process. If not, the random walker is reset.
- When, and only when we have moved all particles in the system are we done.

B. Importance Sampling and the Metropolis - Hastings Algorithm

The Brute Force Method we have described hitherto acts a backbone for our VMC code, yet it lacks a handful of easily implemented auxiliary elements which will help us avoid wasting CPU cycles and improve the precision of our results. One such element is importance sampling. As the Brute Force Method is based on sampling configuration space at entirely random coordinates, we are always at risk of suggesting new configurations which are similar to the previous one, effectively locking our particle in place for many cycles. In order to remedy this issue, we model our system as if moving dynamically in time due to a drift force $F(\mathbf{r})$ by the Langevin equation

$$\frac{\partial \mathbf{r}(t)}{\partial t} = DF(\mathbf{r}(t)) + \eta(t) \tag{10}$$

Taken from the field of molecular dynamics, this equation originally describes the Brownian motion of particles diffusing in a liquid medium, where D denotes the diffusion constant and $\eta(t)$ accounts for the seemingly random collisions between the dissolved particles and the solvent molecules. In our case, we devise this equation and its solution

$$\mathbf{r}_j = \mathbf{r}_i + DF(\mathbf{r}_i)\Delta t + \xi\sqrt{\Delta t} \tag{11}$$

to suggest transitions in configuration space which are biased by the wave function itself as

$$F(\mathbf{r}_i) = \frac{2}{\Psi(\mathbf{r}_i)} \nabla \Psi(\mathbf{r}_i)$$

For an excellent derivation of this expression, please refer to [2]. As we may see in equation (11), we suggest new positions by "pushing" the particles from position i to a position j by the drift force term, as well as accounting for any random motion occurring with the last term. Here, Δt denotes a time step which in our case lies within the range of $[10^{-3}, 10^{-1}]$. This range was determined by trial and error, and the results pertaining to this time dependency may be found in the Results section.

The formulae and analytical expressions for the drift force, which may be found in Appendix B, suggest that particles located around the centre of the wave function experience very little drift force, whilst those far away from the centre of the wave function where the probability density is low experience a great drift force 'pushing' the particles in the direction of the higher probability density centre. Consequently, we increase the likelihood of sampling configurations with much higher transition probabilities in any direction such that we avoid wasting our CPU cycles proposing new configurations that are

rejected over and over again. As we now have changed the method for proposing new configurations, we consequentially need a new method for accepting them. The Fokker - Planck equation

$$\frac{\partial P(\mathbf{r}_i,t)}{\partial t} = D\frac{\partial}{\partial \mathbf{r}_i}\left(\frac{\partial}{\partial \mathbf{r}_i} - F\right)P(\mathbf{r}_i,t) \tag{12}$$

describes the time - dependent evolution of a probability distribution $P(\mathbf{r}_i,t)$ and is a natural choice when devised alongside the Langevin equation. It's solution is given by the Green's function

$$G(\mathbf{r}_j,\mathbf{r}_i,\Delta t) = \frac{1}{(4\pi D\Delta t)^{3N/2}} e^{-(\mathbf{r}_j-\mathbf{r}_i-D\Delta t F(\mathbf{r}_i))^2/4D\Delta t} \tag{13}$$

describing the probability for selecting a transition y given that we previously were in a state x. In turn, the acceptance ratio is replaced with $A(\mathbf{r}_i \rightarrow \mathbf{r}_j) = \min(1, q(\mathbf{r}_i,\mathbf{r}_j))$ with

$$q(\mathbf{r}_i,\mathbf{r}_j) = \frac{G(\mathbf{r}_i,\mathbf{r}_j,\Delta t)|\Psi(\mathbf{r}_j)|^2}{G(\mathbf{r}_j,\mathbf{r}_i,\Delta t)|\Psi(\mathbf{r}_i)|^2}$$

Again, for an in detail derivation of the Green's function ratio as a simplified analytical expression, refer to [2]. The recipe for implementing the Metropolis - Hastings algorithm is not much different from what we saw for the Metropolis algorithm. The only difference is that we now must propose new states by calculating the drift force acting on each of the random walkers and use the Langevin equation to update their positions.

It should be noted here that even though we are able to achieve a much higher acceptance rates with importance sampling, the method comes with a relatively large computational price tag. Compared to the Metropolis algorithm, the Metropolis - Hastings algorithm is much more involved and requires iterative computations of the drift force, the new positions we wish to suggest, as well as re-evaluation of the Green's function ratio in each sampling step. As we will present in the Results section, this typically doubles the run time compared to devising the Brute Force method. This also gives rise to the motivation for making a parallelized, effective code for launching large MC simulations of many body systems.

C. Optimization Methods and Their Implementation

To find the optimal variational parameters that minimize the energy we use gradient descent. We start with an initial guess of the variational parameter, calculate the gradient of the energy with respect to the variatonal parameter, and update the parameter in the direction of the negative gradient to approach the minimum. This procedure is repeated until the variational parameter converges, or until we cross a tolerance for how small the gradient should be. In the case where the bosons interact, we have found that the former typically will not happen for reasonable amounts of iterative computations of the gradient. We therefore focus on the latter, and perform a series of simulations for the same system in order to determine an average optimal parameter used to launch the final simulations in which we extract the estimated ground state energy.

The derivative of the energy with respect to the parameter α in our case is given by:

$$\begin{aligned}
\bar{E_a} &= \frac{d}{d\alpha}\int dr \frac{1}{\psi}H\psi = \int dr\Big(-\frac{1}{\psi^2}\bar{\psi_\alpha}H\psi + \frac{1}{\psi}\frac{d}{d\alpha}H\psi\Big) \\
&= \int dr\Big(-\frac{\bar{\psi_\alpha}}{\psi}\frac{1}{\psi}H\psi + \frac{1}{\psi}H\frac{\psi}{\psi}\bar{\psi_\alpha}\Big) \\
&= \int dr\Big(-\frac{\bar{\psi_\alpha}}{\psi}\frac{1}{\psi}H\psi + \frac{1}{\psi}H\psi\frac{\bar{\psi_\alpha}}{\psi}\Big) \\
&= 2\Big(\langle\frac{\bar{\psi_\alpha}}{\psi}E_L\rangle - \langle\frac{\bar{\psi_\alpha}}{\psi}\rangle\langle E_L\rangle\Big)
\end{aligned} \tag{14}$$

The expected values which go into this expression are calculated using the Metropolis algorithm.

The derivative of the trial wavefunction with respect to α divided by the wavefunction is given by:

$$\begin{aligned}
\frac{\bar{\psi_\alpha}}{\psi} &= \frac{1}{\psi}\frac{\partial}{\partial\alpha}\prod_i g(\mathbf{r}_i,\alpha,\beta)\prod_{j<k} f(a,\mathbf{r}_j,\mathbf{r}_k) \\
&= \frac{1}{\prod_i g(\mathbf{r}_i,\alpha,\beta)}\frac{\partial}{\partial\alpha}\prod_i \exp\big(-\alpha(x_i^2+y_i^2+\beta z_i^2)\big) \\
&= -\sum_i x_i^2 + y_i^2 + \beta z_i^2
\end{aligned} \tag{15}$$

Note how the derivative scales with the number of particles. This is because the total energy scales with the number of particles. We found that scaling the derivatives by the number of particles lead to much more consistent results across simulations with few and many particles. This is consistent with how the energy per particle changes very little with the number of particles.

With the derivatives found, we update the variational parameter by the negative gradient times a learning rate.

$$\alpha \leftarrow \alpha - \gamma\bar{E_\alpha} \tag{16}$$

If the learning rate is too small, the variatonal parameter converges too slowly, and if the learning rate is too large, it diverges. We find the best learning rate by trial and error. The performance of different learning rates is shown in the results.

D. Resampling Methods - Blocking

We estimate the expected value of the energy with an average over energy samples 9. This estimate will include errors due to the stochastic nature of the VMC method, correlation between samples where not much has changed in the system and correlation due to the use of psuedo-random numbers. The sources of correlation means that the standard deviation of the energy samples will be an optimistic estimate of the error in the system. We will therefore employ the blocking method to compute an error which accounts for correlation in the data. We will use the blocking method as implemented and explained by Marius Jonsson [3].

E. Parallelization

VMC calculations lend themselves perfectly to parallelization, since they consist of many small calculations which are averaged over in the end. We have implemented parallelization using OpenMP, where each thread has its own system of particles. Each thread must complete the full number of calibration steps, and then the full number of steps divided by the number of threads. At the end of the calculation, the expected values calculated in each thread are averaged. The speedup from parallelization is shown in the results.

IV. RESULTS

A. Benchmarks

All code used to set up the calculations which produced these results is in the "main.cpp" file in the codebase [4]. Most results have their own function which must be called to produce the results. We will also give the most important parameters for the simulations in figure and table texts.

Some of the numerical results shown have an uncertainty of around 10^{-6} due to the standard c++ formatting of numbers when printed or written to file. All presented energies are given per particle in the system and in natural units.

We ran small VMC calculations for the system without any interaction between the particles to compare our results with analytical results. The results for different numbers of particle is shown in table I.

N	α	$\langle E_L \rangle$	$\langle E_L \rangle_{exact}$	σ_E	Accept%	GD Time(s)	E Time(s)
1	0.5	1.5	1.5	0	75.079	9.72	9.69
10	0.5	1.5	1.5	0	75.078	11.17	16.12
100	0.5	1.5	1.5	0	75.093	21.83	69.07
500	0.5	1.5	1.5	0	75.076	89.61	302.40

Table I. Benchmarks for the calculation with N particles in 3 dimensions, no interaction, and importance sampling. The parameters used were $dt = 0.1$, $\alpha_0 = 0.1$, 20000 steps, 10000 calibration, learning rate of 0.02 for the GD. 10^7 steps, 10^5 calibration for the energy calculation. The values shown are in order: Number of particles, optimal alpha found, energy estimate, exact energy, standard error, % accepted steps, time the gradient descent took, time the final energy calculation took.

We also did a calculation for 1D, 2D and 3D shown in table II.

	α	$\langle E_L \rangle$	$\langle E_L \rangle_{exact}$	σ_E	Accept%	GD Time(s)	E Time(s)
1D	0.5	0.5	0.5	0	90.881	6.01	10.68
2D	0.5	1.0	1.0	0	82.606	3.93	13.86
3D	0.5	1.5	1.5	0	75.080	2.82	17.07

Table II. Benchmarks for the calculation with 10 particles in 1, 2 and 3 dimensions, no interaction, and importance sampling. The parameters used were $dt = 0.1$, $\alpha_0 = 0.1$, 20000 steps, 10000 calibration, learning rate of 0.1 for the GD. 10^7 steps, 10^5 calibration for the energy calculation. The values shown are in order: Number of particles, optimal alpha found, energy estimate, exact energy, standard error, % accepted steps, time the gradient descent took, time the final energy calculation took.

Finally, we calculated the energy (per particle) of the system for many different values of α and plotted it in figure 1 next to the analytical formula for the energy 41 (per particle).

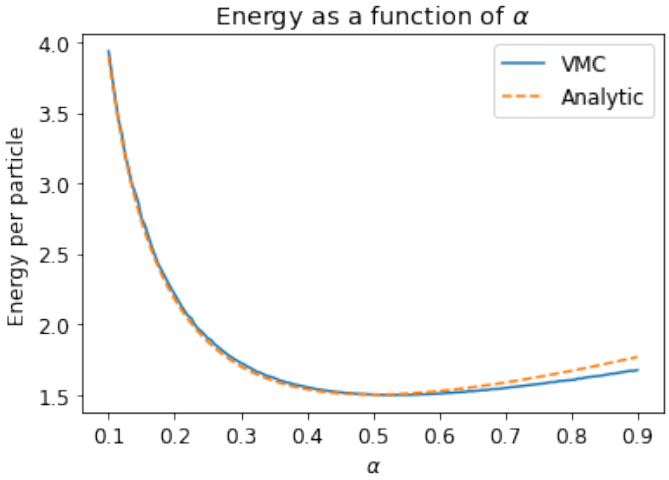

Figure 1. The energy per particle calculated with VMC, and analytically side by side. Calculated with $2{\cdot}10^6$ cycles at each point with importance sampling, dt=0.1, 3D, 10 particles.

B. Evaluation of the Methods

The performance of importance sampling, parallelization and gradient descent is shown in this section. These simulations are meant as an illustration of performance in general. Each set of parameters leads to different run speeds and gradient behavior.

The effect of importance sampling on the acceptance rate and speed is shown in table III, together with the effect of running on 4 threads instead of 1.

	Time [s]	Acceptance %
Importance Sampling	60.69	94.36
No Importance Sampling	32.23	48.07
1 Thread	135.99	
4 Threads	52.97	

Table III. A tabular performance comparison when employing auxillary numerical methods like importance sampling and code parallelization. The sampling methods were run with dt=0.1 / steplength=1, 10^4 equilibration steps, 10^5 sampled steps with $\alpha = 0.498944$, 3D, 5 particles. Parallelization was tested with 10^5 equilibration steps and 10^7 sampled steps, 3D, 5 particles.

Four gradient descent calculations with different learning rates are shown in Figure 2.

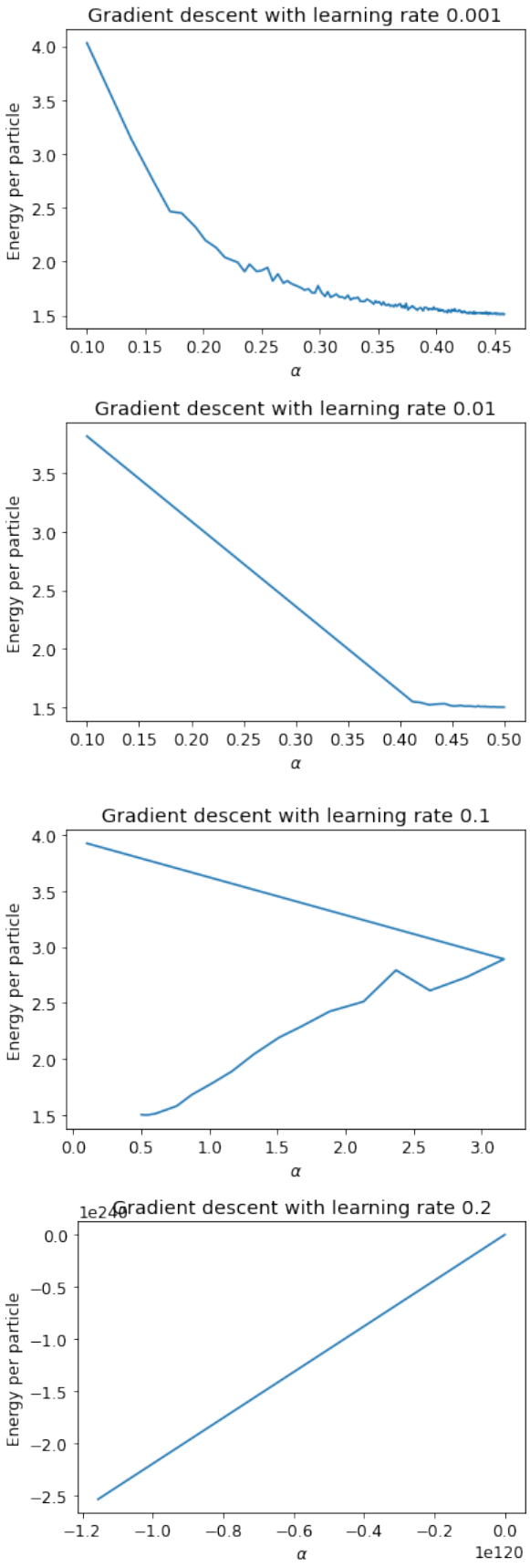

Figure 2. The energy and α value at each step of a gradient descent calculation. Each gradient step had 10^4 equilibration steps and $2 \cdot 10^4$ sampled steps. A max of 200 iterations of gradient descent were run. 3D, 10 particles, importance sampling with dt=0.1.

C. Physical Results in the Interacting Systems

We performed large scale calculations consisting of several gradient descent calculations on each system, and then a large calculation using the average α from the gradient descents to get a good estimate of the energy and standard error. The physical results from these calculations are shown in table IV, and the metrics of the calculations are shown in table V.

N	a	α	$\langle E_L \rangle$	$\langle E_L \rangle_{ref}$	σ_E
2	0	0.500000	1.5	1.5	$3.48{\cdot}10^{-10}$
3	0	0.500000	1.5	1.5	$1.25{\cdot}10^{-9}$
5	0	0.500000	1.5	1.5	$3.56{\cdot}10^{-10}$
10	0	0.500000	1.5	1.5	0
20	0	0.500000	1.5	1.5	0
2	0.00433	0.499743	1.501713	-	$6.80{\cdot}10^{-6}$
3	0.00433	0.499485	1.50343	1.50345	$1.48{\cdot}10^{-5}$
5	0.00433	0.498944	1.50684	1.50689	$3.33\ {\cdot}10^{-5}$
10	0.00433	0.497685	1.5153	1.51537	$1.04{\cdot}10^{-4}$
20	0.00433	0.495496	1.5319	1.5320	$3.18{\cdot}10^{-4}$
2	0.0433	0.497426	1.51711	-	$3.23{\cdot}10^{-5}$
3	0.0433	0.494881	1.53386	-	$4.42{\cdot}10^{-5}$
5	0.0433	0.484810	1.56743	-	$8.19{\cdot}10^{-5}$
10	0.0433	0.480227	1.72	-	$5.61{\cdot}10^{-2}$
20	0.0433	0.474472	1.7746	-	$8.09{\cdot}10^{-4}$

Table IV. Tabular overview of expected energies alongside the corresponding values for α and the standard error σ_E for varying values of the number of bosons and the hard shell diameter a. The reference values are taken from an article on BECs studied under a Diffusion Monte Carlo (DMC) method[5], and the trivial energies from equation 41. Each of the maximum 200 gradient descent iterations consisted of 10^4 equilibration steps and 10^5 sampling steps. The final energy calculation consisted of 10^5 equilibration steps and 10^7 sampling steps. The calculation also used 3D, dt=0.01, learning rate = 0.01.

N	a	α	Accept %	GD Time(s)	E Time(s)
2	0	0.500000	97.06	9.21	11.96
3	0	0.500000	97.06	9.64	14.47
5	0	0.500000	97.06	11.22	17.69
10	0	0.500000	97.06	17.81	27.66
20	0	0.500000	97.06	26.71	47.75
2	0.00433	0.499743	97.07	262.90	59.72
3	0.00433	0.499485	97.07	372.83	80.74
5	0.00433	0.498944	97.07	746.79	140.79
10	0.00433	0.497685	97.12	2301.44	314.23
20	0.00433	0.495496	97.15	5710.55	661.40
2	0.0433	0.497426	96.97	52.97	17.56
3	0.0433	0.494881	96.84	111.37	22.56
5	0.0433	0.484810	96.61	177.73	33.15
10	0.0433	0.480227	95.52	433.60	67.35
20	0.0433	0.474472	94.20	606.04	146.51

Table V. Tabular overview of performance metrics of the same VMC calculations as in table IV. The gradient descent calculationts and the final energy calculation were timed separately. Each of the maximum 200 gradient descent iterations consisted of 10^4 equilibration steps and 10^5 sampling steps. The final energy calculation consisted of 10^5 equilibration steps and 10^7 sampling steps. The calculation also used 3D, dt=0.01, learning rate = 0.01.

We repeated a similar setup for systems of particles in an elliptically deformed trap. The results for interacting particles is shown in table VI.

N	α	$\langle E_L \rangle$	σ_E	Accept%	GD Time(s)	E Time(s)
10	0.391	1.923	0.001	96.39	813.11	64.07
50	0.379	2.008	0.001	96.43	6125.07	525.96
100	0.385	2.089	0.001	95.92	2550.49*	1705.40

Table VI. A tabular overview of the obtained optimal parameters and expected energies for different numbers of interacting bosons N in an elliptically deformed trap with $\lambda = \beta = \sqrt{8}$. These results were obtained using 10^5 equilibration steps and 10^7 sampling steps. In the gradient descent calculations, 10^4 equilibration steps and 10^5 sampling steps were used. For 100 particles, the simulation was not run in series as with the others to find an average optimal parameter due to time considerations.

The results for non-interacting particles in an elliptically deformed trap is shown in table VII.

N	α	$\langle E_L \rangle$	σ_E	Accept%	GD Time(s)	E Time(s)
10	0.390	1.902	0.001	96.34	163.04	30.25
50	0.392	1.902	0.001	96.32	575.35	101.55
100	0.390	1.902	0.001	96.35	1059.04	194.03

Table VII. A tabular overview of the obtained optimal parameters and expected energies for different numbers of non-interacting bosons N in an elliptically deformed trap with $\lambda = \beta = \sqrt{8}$. These results were obtained using 10^5 equilibration steps and 10^7 sampling steps. Here, the gradient descent method was run with 10^3 equilibration steps and 10^4 sampling steps due to the very fast convergence towards a steady state in the non-interacting case.

Finally we computed the one-body density for systems with different numbers of particles, as shown in figure 3, and for systems with different hard shell radii as shown in figure 4.

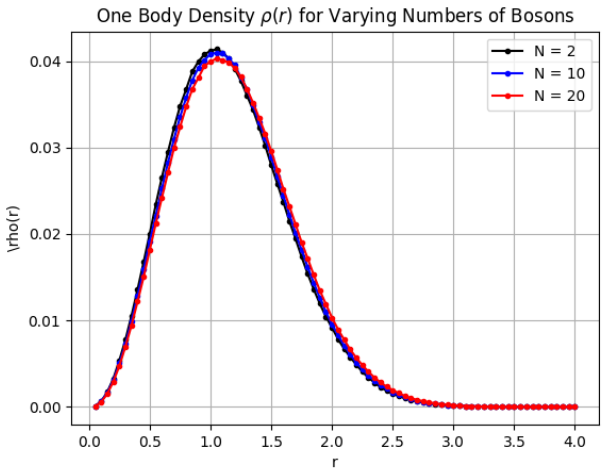

Figure 3. A plot of the one body density distribution for different numbers of bosons N. These results were obtained using 10^5 equilibration steps and 10^6 sampling steps.

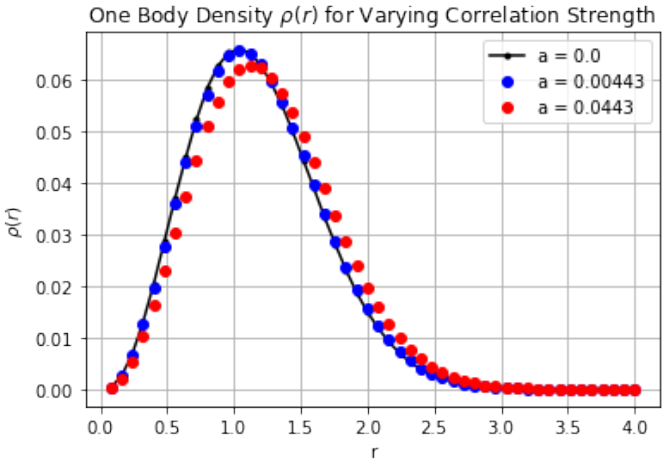

Figure 4. A plot of the one body density distribution for different values of the hard shell radius a. These results were obtained with 10^6 equilibration steps, 10^7 sampling steps for 5 particles.

V. DISCUSSION

A. Benchmarks

Tables I and II are demonstrations of the performance of the VMC method for known reference values of the ground state energy of systems with different numbers of bosons in different dimensions. These are presented here to show that for the non-interacting case, the numerical results are in exceptional agreement with the analytical values in one, two and three dimensions. What more is interesting here is that the program performed very well with a rather large timestep $\Delta t = 0.1$ giving rise to a relatively low acceptance ratio compared to what was necessary in later simulations for us to reach a steady state where could begin sampling energies. Notice also how for lower dimensions the acceptance ratio becomes larger. We believe that for these lower dimensional systems, there are fewer possible unfavourable transitions to select[6] - and so naturally, the total number of accepted transitions increases. The acceptance ratio bump comes at the cost of CPU cycles, as is obvious from Table II. The constant motion in one dimension causes the thermalization time to increase, which in turn affects the convergence rate of the gradient descent method. These remarks were carried on into the simulation of systems in which the bosons are allowed to interact.

Figure 1 is more than anything a visual confirmation that the program works for the non-interacting case. Here we see that the VMC solver follows the analytical α dependency, although with some discrepancy. This result was produced using two million Monte Carlo cycles, but should have been produced with many more in order to avoid the error here. This was taken into consideration later on when simulating larger systems with the possibility of interaction.

B. Evaluation of the Methods

The true challenge of the VMC architecture is to constantly fine tune the (many) parameters determining the outcome of the simulations. Amongst the most important ones are the timestep Δt and learning rate γ used to correctly implement importance sampling and the gradient descent method, respectively. Table III is a very simple, yet effective comparison of the performance based effects the implementation of importance sampling has on the final outcome in our larger simulations. These results were taken from a simulation of five non-interacting bosons in three dimensions. As we may see, importance sampling does improve acceptance rate, but at the cost of CPU cycles as discussed in Section III. It is also a rather dramatic run time increase, close to 100% which again leads to the same increase in accepted steps which improves the effectiveness of the sampling. We would like here to also demonstrate briefly how important a parallelized code is in light of performing very large simulations devising VMC. Having not parallelized the code, the lone thread working away at sampling and computing averages spends approximately 150% more time to finish the task, compared to the case where the work load is distributed across four threads. Having an effectively parallelized code, whilst also focusing

on reducing the number of Floating Point Operations (FLOPs) where possible has been essential to the final architecture of this numerical project. [7]

Figure 2 is an excellent presentation of why fine tuning the learning rate used in the gradient descent (GD) method is crucial to achieving high precision in the search on an optimal parameter α. For a learning rate that is too small, GD calculations are too slow for a given number of maximum iterations and do not reach convergence before the iterations break. If the learning rate is too large, there is dramatic divergence. The third plot from top to bottom shows the behaviour of a learning rate that is very close, but not close enough, to a perfect learning rate. It is extremely close to overshooting in its first iteration and diverging thereafter, but is able to regress and find the correct parameter. This may not be the case in the same simulation is run in series. The learning rate that was well suited for this project turned out to be $\gamma = 0.01$, for which we see very fast convergence towards the correct parameter.

The results pertaining to the total time spent on GD calculations and consequent large simulations using the optimal parameters obtained devising GD are found in Table V. These are first and foremost demonstrative in the sense that they represent the large number of calculations needed in order to obtain decent results using VMC. What is more interesting is that these times were obtained by performing GD on the same system with different seeds sequentially, and averaging the obtained optimal parameters. This proved to be faster and more accurate than spending enormous amounts of CPU cycles on one GD calculation per system. For systems with boson-boson interaction, the GD method never converged towards a fixed optimal parameter, but instead to a common area in which the final parameter would be obtained after the GD method was forced to break. Instead of spending hours on end waiting for a single GD calculation to break on some random parameter within that area and use only that one to launch our larger simulations, we visited the same area of parameters multiple times and averaged over the obtained parameters. In our final simulations we made ten simulations in a series for each system, but we achieved decent results for only five serial simulations, too.

C. Physical Results in the Interacting System

Tables IV and V are a display of both the obtained results for specific simulations of both interacting and non-interacting bosons, as well as the performance metrics of our program. Unfortunately, we have been unable to find reference values for systems where the bosons have a hard shell diameter $a = 10a_{Rb}/a_{HO}$. Yet, we may remark that under the condition of stronger correlation, the average energy per particle increases, which agrees well with what one would expect when the correlation term is repulsive. It is also clear from the different simulations that changing the hard shell diameter from 0 to $4.33{\cdot}10^{-3}$ does not dramatically change the average energy of the particles, which is also true for the ten-fold increase from $4.33{\cdot}10^{-3}$ to $4.33{\cdot}10^{-2}$. In the lack of reference values, we may at least pay some mind to the fact that the standard errors obtained from blocking our obtained data are small. This implies a certain stability in the simulations after thermalization, increasing our confidence in the validity of the results, although we have no way of confirming this as of now. More importantly, there is excellent agreement between the results for the case in which $a = 4.33 \cdot 10^{-3}$ and our reference values taken obtained in DMC simulations of the same systems.

Figures 3 and 4 are representative of how the particles in dilute BECs are distributed when the system has reached a steady, coherent state. In the first case, we chose to illustrate how denser BECs display greater delocalization than their more diluted counterparts. Although the density of the BECs is not dramatically increased here, it is apparent that there is a slight shift in the distribution going from two to twenty confined bosons. The shift in the distribution occurring when going from two to ten particles is close to unobservable with the resolution in the given figure, yet a keen eye will catch it. It is nevertheless safe to say that denser BECs lends themselves to the semantics of being more 'spread out'. Focusing on Figure 4 now, we are here demonstrating the changes in one body densities due to variations in the correlation strength between the bosons. As was explained in Section II, the correlation term devised in this project is a repulsive potential, which is in complete agreement with the results - the greater the correlation strength, the more the bosons repel each other and so the more strongly correlated BECs display much greater delocalization.

Tables VI and VII are presented as our final results, in which we study systems of both interacting and non-interacting bosons confined in an elliptically deformed trap with $\lambda = \beta = \sqrt{8}$. These results agree well with the fact that (1) the average energy of the non-interacting particles increases as they are confined in a greater potential than is the case for a spherical trap and (2) the interacting particles experience an even greater increase in average energy as they are not only confined in a stronger potential, but they are also confined closer to each other.

VI. CONCLUSION

The Variational Monte Carlo solver devised in this numerical project has proved to be a powerful tool for studying both interacting and non-interacting bosons confined in two different oscillator traps - spherical and

elliptical. We have found that our numerical results are in agreement with benchmark analytical values, as well as reference values taken from a study of Diffusion Monte Carlo methods for the same systems dealt with here. In the cases where we lack analytical or reference values, we are confident in the validity of our results where they show small standard errors after having been put through the wringer (a Blocking method code provided by M. Jonsson) and a natural behaviour well in agreement with the behaviour of the values we know to be valid. Extrapolating the distribution of particle densities (one body densities) for different numbers of bosons and different hard shell radii a we find that more densely populated systems have higher average energies per particle, as well as larger delocalization. The same trends are observed for increasing hard shell radii.

We have also been meticulous in presenting sufficient amounts of performance based results. We have found that importance sampling, along with effective parallelization, ensures a (relatively) fast and precise VMC solver. Including importance sampling leads to a near doubling in the percentage of accepted steps, whilst also doubling the execution time of the sampling based calculations. While the effective sampling is important, the slow down of the program is important to remedy as the VMC method in it self will be slow for very large systems. By parallelizing the code, we experience run time speed - ups up to a factor of 1.5 using only four threads to distribute the work load. Having more will only increase the speed - up. Last, but not least, we have discovered that for the interacting system, the gradient descent method does not seem to converge within a reasonable amount of iterative calculations, nor for a reasonable amount of Monte Carlo cycles. We found that running simulations of the same system with different random seeds in series for fewer GD iterations and fewer Monte Carlo cycles gave us optimal parameters leading to energies in excellent agreement with reference values.

A future prospect will be for us to determine a surefire way to achieve fast convergence with the GD method - be that by continuing to simulate systems in series or finding a less computationally expensive way of performing one GD calculating per system. Performing calculations with a dynamically changing learning rate, or using gradient descent with momentum should also be explored. Moreover, there exists a myriad of more elegant gradient methods which may be implemented to improve the optimization of α - the stochastic gradient method, conjugate gradient method to mention a few. To further validate the functionality of our code architecture, we wish to implement new object Hamiltonians we may study, hopefully with more available reference values. Possible extensions besides this reach far - modifying the architecture to include the possibility of studying fermions is one of the more relevant ones.

[1] I. Bloch, J. Dalibard, and W. Zwerger, Article Review (2007).
[2] J. Høgberget, Master thesis (2013).
[3] M. Jonsson, Physical Review E **98** (2018), 10.1103/PhysRevE.98.043304.
[4] "Github repository with code and results: https://github.com/KarlHenrik/FYS4411-Gruppe/tree/main/Project1," (5.4.2021).
[5] D. Blume and C. H. Greene, Physical Review A **63** (2001), 10.1103/PhysRevA.63.063601.
[6] It's difficult to get stuck in one dimension - at some point you must move forwards or backwards.
[7] Which was the total number of available threads for the author performing this test. The second author with his whopping twelve threads should probably have done the test to accentuate the difference further.

VII. APPENDIX

A. Appendix A

VIII. DEFINING THE SYSTEM

Harmonic potential (Spherical and Elliptical):

$$V_{ext}(\mathbf{r}) = \begin{cases} \frac{1}{2}m\omega_{ho}^2 r^2 & (S) \\ \frac{1}{2}m[\omega_{ho}^2(x^2+y^2)+\omega_z^2 z^2] & (E) \end{cases} \tag{17}$$

Inter-boson interaction:

$$V_{int}(|\mathbf{r}_i - \mathbf{r}_j|) = \begin{cases} \infty & |\mathbf{r}_i - \mathbf{r}_j| \leq a \\ 0 & |\mathbf{r}_i - \mathbf{r}_j| > a \end{cases} \tag{18}$$

Hamiltonian:

$$H = \sum_i^N \left(\frac{-\hbar^2}{2m} \bigtriangledown_i^2 + V_{ext}(\mathbf{r}_i) \right) + \sum_{i<j}^N V_{int}(\mathbf{r}_i, \mathbf{r}_j), \tag{19}$$

Trial wave function:

$$\begin{aligned} \Psi_T(\mathbf{r}) &= \Psi_T(\mathbf{r}_1, \mathbf{r}_2, \dots \mathbf{r}_N, \alpha, \beta) \\ &= \left[\prod_i g(\alpha, \beta, \mathbf{r}_i) \right] \left[\prod_{j<k} f(a, |\mathbf{r}_j - \mathbf{r}_k|) \right], \end{aligned} \tag{20}$$

Harmonic oscillator function for the ground state:

$$g(\alpha, \beta, \mathbf{r}_i) = \exp\{[-\alpha(x_i^2 + y_i^2 + \beta z_i^2)]\}. \tag{21}$$

The correlation wave function:

$$f(a, |\mathbf{r}_i - \mathbf{r}_j|) = \begin{cases} 0 & |\mathbf{r}_i - \mathbf{r}_j| \leq a \\ (1 - \frac{a}{|\mathbf{r}_i - \mathbf{r}_j|}) & |\mathbf{r}_i - \mathbf{r}_j| > a. \end{cases} \tag{22}$$

Local energy:

$$E_L(\mathbf{r}) = \frac{1}{\Psi_T(\mathbf{r})} H \Psi_T(\mathbf{r}), \tag{23}$$

IX. 1A) NO INTER-BOSON INTERACTION

We want to find an analytical expression for the local energy of the trial wave function 20.
We start simple by ignoring the inter-boson interaction and assuming a spherical potential($a = 0$, $\beta = 1$).

A. 1a) One particle, one dimension

First derivative in one dimension:

$$\begin{aligned} \frac{d}{dx}\Psi_T &= \frac{d}{dx} g(\alpha, \beta, x) \\ &= \frac{d}{dx} \exp\{-\alpha x^2\} \\ &= -2x\alpha \exp\{-\alpha x^2\} \end{aligned} \tag{24}$$

Second derivative in one dimension:

$$\begin{aligned}\frac{d^2}{dx^2}\Psi_T &= -2\alpha\frac{d}{dx}x\exp\{-\alpha x^2\}\\ &= -2\alpha\exp\{-\alpha x^2\} + 4x^2\alpha^2\exp\{-\alpha x^2\}\\ &= -2\alpha(1-2\alpha x^2)\exp\{-\alpha x^2\}\end{aligned} \tag{25}$$

Hamiltonian in one dimension:

$$\begin{aligned}H\Psi_T &= \left(\frac{-\hbar^2}{2m}\frac{d^2}{dx^2} + V_{ext}(x)\right)\Psi_T\\ &= \frac{\alpha\hbar^2}{m}(1-2\alpha x^2)\exp\{-\alpha x^2\} + \frac{1}{2}m\omega_{ho}^2x^2\exp\{-\alpha x^2\}\\ &= (\frac{\alpha\hbar^2}{m}(1-2\alpha x^2) + \frac{1}{2}m\omega_{ho}^2x^2)\exp\{-\alpha x^2\}\end{aligned} \tag{26}$$

Local energy in one dimension:

$$\frac{1}{\Psi_T}H\Psi_T = \frac{\alpha\hbar^2}{m}(1-2\alpha x^2) + \frac{1}{2}m\omega_{ho}^2x^2 \tag{27}$$

B. 1a) One particle, three dimensions

First derivative in three dimensions:

$$\begin{aligned}\nabla\Psi_T &= \Big[\frac{d}{dx}g(\alpha,\beta,\mathbf{r}), \frac{d}{dy}g(\alpha,\beta,\mathbf{r}), \frac{d}{dz}g(\alpha,\beta,\mathbf{r})\Big]\\ &= \Big[-2x\alpha\exp\{-\alpha(x^2+y^2+z^2)\},\\ &\quad -2y\alpha\exp\{-\alpha(x^2+y^2+z^2)\},\\ &\quad -2z\alpha\exp\{-\alpha(x^2+y^2+x^2)\}\Big]\end{aligned} \tag{28}$$

Second derivative in three dimensions:

$$\begin{aligned}\nabla^2\Psi_T =& \frac{d^2}{dx^2}g(\alpha,\beta,\mathbf{r}) + \frac{d^2}{dy^2}g(\alpha,\beta,\mathbf{r}) + \frac{d^2}{dz^2}g(\alpha,\beta,\mathbf{r})\\ =& -2\alpha(1-2\alpha x^2)\exp\{-\alpha(x^2+y^2+z^2)\}\\ &-2\alpha(1-2\alpha y^2)\exp\{-\alpha(x^2+y^2+z^2)\}\\ &-2\alpha(1-2\alpha z^2)\exp\{-\alpha(x^2+y^2+z^2)\}\\ =& -2\alpha(3-2\alpha(x^2+y^2+z^2))\exp\{-\alpha(x^2+y^2+z^2)\}\\ =& -2\alpha(3-2\alpha r^2)\Psi_T\end{aligned} \tag{29}$$

Hamiltonian in three dimensions:

$$\begin{aligned}H\Psi_T &= \left(\frac{-\hbar^2}{2m}\nabla^2 + V_{ext}(\mathbf{r})\right)\Psi_T\\ &= \left(\frac{\hbar^2}{m}\alpha(3-2\alpha r^2) + \frac{1}{2}m\omega_{ho}^2r^2\right)\Psi_T\end{aligned} \tag{30}$$

Local energy in three dimensions:

$$\frac{1}{\Psi_T}H\Psi_T = \left(\frac{\hbar^2}{m}\alpha(3-2\alpha r^2) + \frac{1}{2}m\omega_{ho}^2r^2\right) \tag{31}$$

C. 1a) N particles, one dimension

First derivative in one dimension:

$$\begin{aligned}\sum_i^N \frac{d}{dx_i}\Psi_T &= \sum_i^N \frac{d}{dx_i}\prod_i g(\alpha,\beta,x_i) \\ &= \sum_i^N -2x_i\alpha \prod_i g(\alpha,\beta,x_i)\end{aligned} \tag{32}$$

Second derivative in one dimension:

$$\begin{aligned}\sum_i^N \frac{d^2}{dx_i^2}\Psi_T &= \sum_i^N \frac{d^2}{dx_i^2}\prod_i g(\alpha,\beta,x_i) \\ &= \sum_i^N -2\alpha(1-2\alpha x_i^2)\prod_i g(\alpha,\beta,x_i) \\ &= \sum_i^N -2\alpha(1-2\alpha x_i^2)\Psi_T\end{aligned} \tag{33}$$

Hamiltonian in one dimension:

$$\begin{aligned}H\Psi_T &= \sum_i^N \left(\frac{-\hbar^2}{2m}\frac{d^2}{dx_i^2} + V_{ext}(x_i)\right)\Psi_T \\ &= \sum_i^N \left(\frac{\hbar^2}{m}\alpha(1-2\alpha x_i^2) + \frac{1}{2}m\omega_{ho}^2 x_i^2\right)\Psi_T\end{aligned} \tag{34}$$

Local energy in one dimension:

$$\frac{1}{\Psi_T}H\Psi_T = \sum_i^N \left(\frac{\hbar^2}{m}\alpha(1-2\alpha x_i^2) + \frac{1}{2}m\omega_{ho}^2 x_i^2\right) \tag{35}$$

D. 1a) N particles, three dimensions

First derivative in three dimensions:

$$\begin{aligned}\sum_i^N \nabla_i\Psi_T &= \sum_i^N \Big[\frac{d}{dx_i}g(\alpha,\beta,\mathbf{r}), \frac{d}{dy_i}g(\alpha,\beta,\mathbf{r}), \frac{d}{dz_i}g(\alpha,\beta,\mathbf{r})\Big] \\ &= \sum_i^N \Big[-2x_i\alpha\Psi_T, \\ &\qquad -2y_i\alpha\Psi_T, \\ &\qquad -2z_i\alpha\Psi_T\Big]\end{aligned} \tag{36}$$

Second derivative in three dimensions:

$$\begin{aligned}\sum_i^N \nabla_i^2 \Psi_T &= \sum_i^N \Big(\frac{d^2}{dx_i^2} g(\alpha,\beta,\mathbf{r}) + \frac{d^2}{dy_i^2} g(\alpha,\beta,\mathbf{r}) + \frac{d^2}{dz_i^2} g(\alpha,\beta,\mathbf{r}) \Big) \\ &= \sum_i^N \Big(-2\alpha(1-2\alpha x_i^2)\Psi_T \\ &\qquad -2\alpha(1-2\alpha y_i^2)\Psi_T \\ &\qquad -2\alpha(1-2\alpha z_i^2)\Psi_T \Big) \\ &= \sum_i^N \Big(-2\alpha(3-2\alpha(x_i^2+y_i^2+z_i^2)) \Big)\Psi_T \\ &= \sum_i^N \Big(-2\alpha(3-2\alpha r_i^2) \Big)\Psi_T \end{aligned} \tag{37}$$

Hamiltonian in three dimensions:

$$\begin{aligned} H\Psi_T &= \sum_i^N \left(\frac{-\hbar^2}{2m} \bigtriangledown_i^2 + V_{ext}(\mathbf{r}_i) \right) \\ &= \sum_i^N \left(\frac{\hbar^2}{m}\alpha(3-2\alpha r_i^2) + \frac{1}{2} m\omega_{ho}^2 r_i^2 \right) \Psi_T \end{aligned} \tag{38}$$

Local energy in three dimensions:

$$\frac{1}{\Psi_T} H\Psi_T = \sum_i^N \left(\frac{\hbar^2}{m}\alpha(3-2\alpha r_i^2) + \frac{1}{2} m\omega_{ho}^2 r_i^2 \right) \tag{39}$$

To find the expected value of energy takes a couple of steps. First we change the wavefunction to be radial and normalize it so that:

$$\Psi_T(r;\alpha)^2 = \sqrt{\frac{\alpha^3 2^7}{\pi}} r^2 e^{-2\alpha r^2} \tag{40}$$

Then we find the expectation value of the energy (using an integral calculator):

$$\begin{aligned} \int_{-\infty}^{\infty} \Psi_T^* H \Psi_T dr &= \sum_i^N \left(\int_{-\infty}^{\infty} (\alpha(3-2\alpha r_i^2) + \frac{1}{2} r_i^2) \Psi_T(r;\alpha)^2 dr_i \right) \\ &= \sum_i^N \frac{3}{2}\alpha + \frac{3}{8\alpha} \end{aligned} \tag{41}$$

For one dimension, you only need to divide the energy by 3.

E. Appendix B

1.5

X. DERIVATION OF AN ANLYTICAL EXPRESSION FOR THE DOUBLE DERIVATIVE OF THE FULL WAVE FUNCTION

We shall now explain in detail how to obtain the local energy of a particle k for an N - dimensional bosonic system. Consider first the trial wave function given by

$$\Psi_T(\mathbf{r}) = \prod_i g(\mathbf{r}\ ;\alpha,\beta) \prod_i f(r_{i,j}\ ;a)$$

Let $r_{i,j} = |\mathbf{r}_i - \mathbf{r}_j|$, $\ln f(a, r_{i,j}) = u(r_{i,j})$ and $g(\alpha, \beta, \mathbf{r}_i) = \varphi(\mathbf{r}_i)$ such that we may rewrite the trial function by

$$\Psi_T(\mathbf{r}) = \left(\prod_i \varphi(\mathbf{r}_i)\right) \exp\left(\sum_{i<j} u(r_{i,j})\right) \tag{42}$$

The first derivative of this function with respect to particle k becomes, by the product rule

$$\nabla_k \Psi_T(\mathbf{r}) = \left(\nabla_k \prod_i \varphi(\mathbf{r}_i)\right) \exp\left(\sum_{i<j} u(r_{i,j})\right) + \left(\prod_i \varphi(\mathbf{r}_i)\right) \nabla_k \exp\left(\sum_{i<j} u(r_{i,j})\right) \tag{43}$$

As

$$\prod_i \varphi(\mathbf{r}_i) = \varphi(\mathbf{r}_1)\varphi(\mathbf{r}_2)...\varphi(\mathbf{r}_k)...\varphi(\mathbf{r}_N),$$

we must have

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

and similarly, as

$$\exp\left(\sum_{i<j} u(r_{i,j})\right) = \exp\left(u(r_{1,2}) + u(r_{1,3}) + ... + u(r_{k,i}) + ... + u(r_{N-1,N})\right)$$

the gradient occurs for each term involving the interaction between particle k and the $N-1$ other particles such that

$$\nabla_k \exp\left(\sum_{i<j} u(r_{i,j})\right) = \sum_{\ell \neq k} \nabla_k u(r_{\ell,k})$$

The full expression for the first derivative of the trial wave function then becomes

$$\begin{aligned} \nabla_k \Psi(\mathbf{r}) =& \nabla_k \varphi(\mathbf{r}_k) \prod_{i \neq k} \varphi(\mathbf{r}_i) \exp\left(\sum_{i<j} u(r_{i,j})\right) \\ &+ \left(\prod_i \varphi(\mathbf{r}_i)\right) \exp\left(\sum_{i<j} u(r_{i,j})\right) \sum_{\ell \neq k} \nabla_k u(r_{\ell,k}) \end{aligned} \tag{44}$$

Now, for the second derivative $\nabla_k^2 \Psi(\mathbf{r})$ we employ the product rule to first see that

$$
\begin{aligned}
\nabla_k^2\Psi(\mathbf{r}) = (\nabla_k \cdot \nabla_k\varphi(\mathbf{r}_k)) \prod_{i\neq k}\varphi(\mathbf{r}_i)\exp\left(\sum_{i<j}u(r_{i,j})\right) \\
+ \nabla_k\varphi(\mathbf{r}_k)\left(\nabla_k\prod_{i\neq k}\varphi(\mathbf{r}_i)\right)\exp\left(\sum_{i<j}u(r_{i,j})\right) \\
+ \left(\nabla_k\varphi(\mathbf{r}_k)\prod_{i\neq k}\varphi(\mathbf{r}_i)\right)\nabla_k\exp\left(\sum_{i<j}u(r_{i,j})\right) \\
+ \left(\nabla_k\prod_{i}\varphi(\mathbf{r}_i)\right)\exp\left(\sum_{i<j}u(r_{i,j})\right)\sum_{\ell\neq k}\nabla_k u(r_{\ell,k}) \\
+ \left(\prod_{i}\varphi(\mathbf{r}_i)\right)\nabla_k\exp\left(\sum_{i<j}u(r_{i,j})\right)\sum_{\ell\neq k}\nabla_k u(r_{\ell,k}) \\
+ \left(\prod_{i}\varphi(\mathbf{r}_i)\right)\exp\left(\sum_{i<j}u(r_{i,j})\right)\left(\nabla_k\sum_{\ell\neq k}\nabla_k u(r_{\ell,k})\right)
\end{aligned}
\tag{45}
$$

Luckily, we have done the necessary computations which we may here employ in order to arrive at

$$
\begin{aligned}
\nabla_k^2\Psi(\mathbf{r}) = \nabla_k^2\varphi(\mathbf{r}_k)\prod_{i\neq k}\varphi(\mathbf{r}_i)\exp\left(\sum_{i<j}u(r_{i,j})\right) \\
+ 2\left(\nabla_k\varphi(\mathbf{r}_k)\prod_{i\neq k}\varphi(\mathbf{r}_i)\right)\exp\left(\sum_{i<j}u(r_{i,j})\right)\sum_{\ell\neq k}\nabla_k u(r_{\ell,k}) \\
+ \left(\prod_{i}\varphi(\mathbf{r}_i)\right)\exp\left(\sum_{i<j}u(r_{i,j})\right)\sum_{n\neq k}\nabla_k u(r_{n,k})\sum_{\ell\neq k}\nabla_k u(r_{\ell,k}) \\
+ \left(\prod_{i}\varphi(\mathbf{r}_i)\right)\exp\left(\sum_{i<j}u(r_{i,j})\right)\left(\sum_{\ell\neq k}\nabla_k^2 u(r_{\ell,k})\right)
\end{aligned}
\tag{46}
$$

As we are interested in $\nabla^2\Psi(\mathbf{r})/\Psi(\mathbf{r})$ in order to determine the local energy, we may divide equation (46) by $\Psi(\mathbf{r})$ throughout to arrive at

$$
\begin{aligned}
\frac{1}{\Psi(\mathbf{r})}\nabla_k^2\Psi(\mathbf{r}) = \frac{\nabla_k^2\varphi(\mathbf{r}_k)}{\varphi(\mathbf{r}_k)} + 2\frac{\nabla_k\varphi(\mathbf{r}_k)}{\varphi(\mathbf{r}_k)}\sum_{\ell\neq k}\nabla_k u(r_{\ell,k}) \\
+ \sum_{n\neq k}\nabla_k u(r_{n,k})\sum_{\ell\neq k}\nabla_k u(r_{\ell,k}) \\
+ \sum_{\ell\neq k}\nabla_k^2 u(r_{\ell,k})
\end{aligned}
\tag{47}
$$

Our remaining business now is to determine the analytical expressions for $\nabla_k^2\varphi(\mathbf{r}_k)$, $\nabla_k\varphi(\mathbf{r}_k), \nabla_k^2 u(r_{k,i})$ and $\nabla_k u(r_{k,i})$ Beginning with the latter, we may first observe that by the chain rule

$$
\nabla_k u(r_{k,i}) = \frac{\partial u(r_{k,i})}{\partial r_{k,i}}\frac{\partial r_{k,i}}{\partial \mathbf{r}_k} = u'(r_{k,i})\frac{\partial}{\partial \mathbf{r}_k}|\mathbf{r_k}-\mathbf{r_i}| = u'(r_{k,i})\frac{(\mathbf{r}_k-\mathbf{r}_i)}{|\mathbf{r}_k-\mathbf{r}_i|}
$$

Moving on to the second derivative, we see that

$$\begin{aligned}\nabla_k^2 u(r_{k,i}) &= \frac{\partial}{\partial \mathbf{r}_k}\frac{\partial u(r_{k,i})}{\partial r_{k,i}}\frac{\partial r_{k,i}}{\partial \mathbf{r}_k} + \frac{\partial u(r_{k,i})}{\partial r_{k,i}}\frac{\partial^2 r_{k,i}}{\partial \mathbf{r}_k^2} \\ &= \frac{\partial^2 u(r_{k,i})}{\partial r_{k,i}^2}\left(\frac{\partial r_{k,i}}{\partial \mathbf{r}_k}\right)^2 + \frac{\partial u(r_{k,i})}{\partial r_{k,i}}\frac{\partial^2 r_{k,i}}{\partial \mathbf{r}_k^2}\end{aligned} \tag{48}$$

This time around, we must be a little more careful with the treatment of $\mathbf{r}_k$ as a vector when evaluating the partial derivative

$$\frac{\partial}{\partial \mathbf{r}_k}\frac{\partial r_{k,i}}{\partial \mathbf{r}_k} = \left(\mathbf{i}\frac{\partial}{\partial x_k} + \mathbf{j}\frac{\partial}{\partial y_k} + \mathbf{k}\frac{\partial}{\partial z_k}\right)\frac{(\mathbf{r}_k - \mathbf{r}_i)}{|\mathbf{r}_k - \mathbf{r}_i|}$$

Differentiating then, with only respect to the x-component as an example, leads to

$$\frac{\partial}{\partial x_k}\frac{x_k - x_i}{|\mathbf{r}_k - \mathbf{r}_i|} = \frac{|\mathbf{r}_k - \mathbf{r}_i| - \frac{(x_k - x_i)^2}{|\mathbf{r}_k - \mathbf{r}_i|}}{|\mathbf{r}_k - \mathbf{r}_i|^2} = \frac{1}{r_{k,i}} - \frac{(x_k - x_i)^2}{r_{k,i}^3}$$

Applying the similar process to the two other Cartesian coordinates leads to a full expression

$$\frac{\partial}{\partial \mathbf{r}_k}\frac{(\mathbf{r}_k - \mathbf{r}_i)}{|\mathbf{r}_k - \mathbf{r}_i|} = \frac{3}{r_{k,i}} - \frac{(\mathbf{r}_k - \mathbf{r}_i)^2}{|\mathbf{r}_k - \mathbf{r}_i|^3} = \frac{2}{r_{k,i}}$$

The squared term in equation (48) equates to 1 as the term is a unit vector

$$\left(\frac{\partial r_{k,i}}{\partial \mathbf{r}_k}\right)^2 = \left(\frac{(\mathbf{r}_k - \mathbf{r}_i)}{|\mathbf{r}_k - \mathbf{r}_i|}\right)^2 = 1$$

Using our results, we may now see that

$$\nabla_k^2 u(r_{k,i}) = u''(r_{k,i}) + \frac{2}{r_{k,i}}u'(r_{k,i})$$

In order to determine the analytical expression for the derivatives of the correlation function $u(r_{k,i})$ we turn to its definition $u(r_{k,i}) = \ln f(r_{k,i}\ ;\ a)$ such that

$$u'(r_{k,i}) = \frac{d}{dr_{k,i}}\ln\left(1 - \frac{a}{r_{k,i}}\right) = \frac{a}{r_{k,i}^2 - a r_{k,i}} \tag{49}$$

Differentiating the result in equation (49) with respect to $r_{k,i}$ we arrive at

$$u''(r_{k,i}) = \frac{a^2 - 2a r_{k,i}}{(r_{k,i}^2 - a r_{k,i})^2} \tag{50}$$

Moving on to the gradients of $\varphi(\mathbf{r}_k)$ we see that

$$\nabla_k \varphi(\mathbf{r}_k) = \nabla_k e^{-\alpha(x_k^2 + y_k^2 + \beta z_k^2)} = -2\alpha\left(\mathbf{i}x_k + \mathbf{j}y_k + \mathbf{k}\beta z_k\right)\varphi(\mathbf{r}_k) \tag{51}$$

whilst differentiating a second time to obtain the second gradient of $\varphi(\mathbf{r}_k)$ leads to, by the product rule,

$$\nabla_k^2 \varphi(\mathbf{r}_k) = (-4\alpha - 2\alpha\beta)\varphi(\mathbf{r}_k) + 4\alpha^2(x_k^2 + y_k^2 + \beta^2 z_k^2)\varphi(\mathbf{r}_k) \tag{52}$$

We may now insert all of our results back into equation (47) such that

$$\begin{aligned}\frac{1}{\Psi(\mathbf{r}_k)}\nabla_k^2\Psi(\mathbf{r}_k) =& -4\alpha - 2\alpha\beta + 4\alpha^2(x_k^2 + y_k^2 + \beta^2 z_k^2) \\ &+ 4\alpha(\mathbf{i}x_k + \mathbf{j}y_k + \mathbf{k}\beta z)\sum_{i\neq k}\frac{a}{r_{k,i}^2 - a r_{k,i}}\hat{\mathbf{r}}_{k\to i} \\ &+ \left(\sum_{j\neq k}\frac{a}{r_{k,j}^2 - a r_{k,j}}\hat{\mathbf{r}}_{k\to j}\right)\left(\sum_{i\neq k}\frac{a}{r_{k,i}^2 - a r_{k,i}}\hat{\mathbf{r}}_{k\to i}\right) \\ &+ \sum_{i\neq k}\left(\frac{a^2 - 2a r_{k,i}}{(r_{k,i}^2 - a r_{k,i})^2} + \frac{a}{r_{k,i}^3 - a r_{k,i}^2}\right)\end{aligned} \tag{53}$$

where $\hat{\mathbf{r}}_{k\to i,j}$ should be read as the radial unit vector pointing from particle k to particle i or j.

XI. DRIFT FORCE

$$F_{1D} = \frac{2}{\Psi_T}\frac{\partial \Psi_T}{\partial x} = \frac{2}{\exp(-\alpha x^2)}\frac{\partial}{\partial x}\exp\left(-\alpha x^2\right) = -4\alpha x$$

$$F_{2D} = \frac{2}{\Psi_T}\left(\mathbf{i}\frac{\partial}{\partial x} + \mathbf{j}\frac{\partial}{\partial y}\right)\Psi_T = -4\alpha(x\mathbf{i} + y\mathbf{j})$$

$$F_{3D} = -4\alpha(x\mathbf{i} + y\mathbf{j} + \beta z\mathbf{k})$$

For N particles, omitting repulsion, the drift force take on the same analytical expressions as we are only moving a single particle at a time.
For the full wave function, including the correlation wave function $f(a, r_{k,i})$, we can use the first derivative in (44) to see that

$$F_k = \frac{2}{\Psi(\mathbf{r}_k)}\nabla_k \Psi(\mathbf{r}_k) = 2\frac{\nabla_k \phi(\mathbf{r}_k)}{\phi(\mathbf{r}_k)} + 2\sum_{i\neq k}\nabla_k u(r_{k,i}) \tag{54}$$

Both terms have analytical solutions found in (49) and (51), and give the analytical solution for the drift force

$$F_k = 2\left(-2\alpha\left(\mathbf{i}x_k + \mathbf{j}y_k + \mathbf{k}\beta z_k\right) + \sum_{i\neq k}\frac{a}{r_{k,i}^2 - a r_{k,i}}\hat{\mathbf{r}}_{k\to i}\right) \tag{55}$$