FYS4411 - Project 1

Scrap I/O and friends

May 3, 2018

Abstract

This is where we put the abstract. It will be so good, like, the best. The greatest, probably.

1 Introduction

The aim of this project is to evaluate the case of a hard sphere trapped Bose gas using Variational Monte Carlo simulations. We want to find the ground state energy for systems containing different number of particles given a trial wave function. As the number of particles grow, so does the complexity of the wave function. What we are looking at is a quantum many-body-problem

$$\imath\hbar\frac{\partial}{\partial x}|\Psi(t)\rangle = \hat{H}|\Psi(t)\rangle$$

and with a growing exponential complexity in the wave function the more impossible the calculation of the Hamiltonian and the energy becomes. The Variatonal Monte Carlo method allows us to look for the low ground state energy, but we will be using Metropolis-Hastings algorithm and Importance Sampling to make the selection of movements of particles in the system.

Our trail wave function consist of one variational parameter, $\Psi_T(\mathbf{R};\alpha)$. We want to find the optimal value for α that minimizes the energy. For that we use the method of Steepest Descent. For the statistical analysis and error estimates of the numerical data we have used blocking techniques.

2 Theory

2.1 Variational Monte Carlo

Monte Carlo simulations are widely used methods in numerical science that employs random walkers. In this project we are taking a closer look at trapped bosons. We are given a trail wave function we assume is as close to the real case as possible, $\Psi_T(\mathbf{R};\alpha)$ where $\mathbf{R} = (\mathbf{R}_1, ..., \mathbf{R}_N)$ is the position of the different particles. From quantum mechanics we know the probability distribution is given by the wave function.

$$P(\mathbf{R};\alpha) = \frac{|\Psi_T(\mathbf{R};\alpha)|^2}{\int |\Psi_T(\mathbf{R};\alpha)|^2 d\mathbf{R}}$$

Monte Carlo integration allow us to evaluate the integral at hand. The expectation value of the Hamiltonian is given as follows.

$$\langle \widehat{\mathbf{H}} \rangle = \frac{\int d\mathbf{R} \Psi^*(\mathbf{R}) H(\mathbf{R}) \Psi(\mathbf{R})}{\int d\mathbf{R} \Psi^*(\mathbf{R}) \Psi(\mathbf{R})}$$

The variational principle states that the expectation value of the Hamiltonian is an upper-bound for the ground state energy of the Hamiltonian.

$$E_0 \leq \langle H \rangle$$

This is what the Variational Monte Carlo method bases itself on. Given a probability distribution we can evaluate the wave function and look for a local minimum. We define the local energy by

$$\widehat{\mathbf{E}}_L(\mathbf{R};\alpha) = \frac{1}{\Psi_T(\mathbf{R};\alpha)} \widehat{\mathbf{H}} \Psi_T(\mathbf{R};\alpha).$$

Then the expectation value of the local energy is given by

$$\langle \widehat{\mathbf{E}}_L \rangle = \int P(\mathbf{R}) \widehat{\mathbf{E}}_L d\mathbf{R} \approx \frac{1}{N} \sum_{i=1}^{N} \mathbf{E_L}(x_i)$$

where N is the number of Monte Carlo cycles. Now we can calculate the probability distribution and the local energy. And for each cycle we propose a new configuration $\mathbf{R}_p$ for the system and hopefully we come closer to the ground state.

$$\mathbf{R}_p = \mathbf{R} + r * \Delta \mathbf{R}$$

2.2 The Metropolis Algorithm

How we select the new configurations during the simulation is given by the Metropolis Algorithm. Ideally we would have a transition probability matrix which told us how likely it would be for each particle to move in the configuration space. Since we don't we try and

model it. We define the following entities:

$P_i^{(n)} \rightarrow$ probability of finding the system in state i at the n'th step.
$j \rightarrow$ possible new step
$A_{i\rightarrow j} \rightarrow$ probability of acceptance.
$T_{j\rightarrow i} \rightarrow$ probability of making the transition

Now we can say that a transition probability matrix can be constructed by $T_{j\rightarrow i}A_{j\rightarrow i}$. The probability of finding the system in state i at step n is

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

We want to push the system towards high density regional space of $P_i^{(n)}$. /We want to select states according to the probability distribution. In that way to converge to the desired stationary distribution p_i.

$$P_i^{(n\rightarrow\infty)} \rightarrow p_i$$

Using this statement and the fact that the sum over all possible transitional probabilities is one we can rewrite the above equation 1.

$$\sum_j [p_j T_{j\rightarrow i} A_{j\rightarrow i} - p_i T_{i\rightarrow j} A_{i\rightarrow j}] = 0$$

In order to stop at the configuration where we have reached equilibrium we us the condition of detailed balance, and it gives us

$$\frac{A_{j\rightarrow i}}{A_{i\rightarrow j}} = \frac{p_i T_{i\rightarrow j}}{p_j T_{j\rightarrow i}}.$$

We choose to accept when the acceptance is bigger than 1.

$$A_{j\rightarrow i} = \min\left(1, \frac{p_i T_{i\rightarrow j}}{p_j T_{j\rightarrow i}}\right)$$

What we must calculate is equation 2. And an example of the code is given further down, in Algorithm 1

$$\frac{p_i T_{i\rightarrow j}}{p_j T_{j\rightarrow i}} = \frac{|\Psi_T(\mathbf{R}_p)|^2}{|\Psi_T(\mathbf{R})|^2} \tag{2}$$

2.3 Importance Sampling

Now we move to an extension of the Metropolis algorithm, the Importance sampling. This method provides us with a better way of suggesting new moves. The expression for the new move, y, is given by the solution of the Langevin equation. It reads

Data: matrix $\mathbf{R}$
Result: float $\mathbf{E}_L$
number of MC-cycles N;
initialize $\mathbf{R}$;
calculate $|\Psi_T(\mathbf{R})|^2$;
for i *in* $[0, N]$ **do**
    $\mathbf{R}_p = \mathbf{R} + r * \Delta\mathbf{R}$;
    calculate $\omega = \frac{|\Psi_T(\mathbf{R}_p)|^2}{|\Psi_T(\mathbf{R})|^2}$;
    if $q \leq \omega$ **then**
        accept new move;
    else
        reject new move;
    end
    update energy, $\mathbf{E}_L$;
end

Algorithm 1: Monte Carlo with Metropolis-Hastings

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

where ξ is a gaussian random variable, D is the diffusion coefficient and is $\frac{1}{2}$ and Δt is the time step which take values between $[0.001, 0.01]$. $F(x)$ is the drift force and is given by the gradient of the wave function.

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

It is the drift force that ensures us we move particles towards regions of configuration space where the trail wave function is large. This method increases the efficiency of our program, since the standard Metropolis can suggest new moves in every direction with same probability. From the solution of the Fokker-Planck equation we get a new transition probability, the Green's function.

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

Now equation 2 from Metropolis algorithm changes to

$$\frac{p_i T_{i\to j}}{p_j T_{j\to i}} = \frac{G(x, y, \Delta t)|\Psi_T(y)|^2}{G(y, x, \Delta t)|\Psi_T(x)|^2},$$

and we must calculate the drift force in order to find new moves and the Green's function to accept/reject these moves.

2.4 Gradient Descent

Like most problems in physics the variational quantum problem is one of optimization. In this project we are optimizing the expectation value of the energy over the parameter α. In the naive implementation we simply span a reasonable range of α [1]. For a multi-parameter variational problem this approach is too computationally expensive to implement in most cases. To remedy this problem iterative gradient methods have been developed to make more intelligent choices for the variational parameters. In this project we have opted to implement a simple gradient descent method. The gradient descent is ordinarily presented as

$$\alpha_{i+1} = \alpha_i - \gamma \frac{\partial \langle E_L(\alpha_i) \rangle}{\partial \alpha_i} \tag{3}$$

Where we have opted to update the parameter γ iteratively by the Barzilai-Borwein formula which is a method that uses an iterative approach to the step size γ derived in much the same manner as for quasi-Newton methods[**OptTeory**]. The only constraint we have to enforce is that $0 < \alpha$. In the algorithm we have a simple rule that checks the sign of α and makes a correction if it is out-of-bounds. For $i > 0$ the step size γ_i is then given as:

$$\gamma_i = (\alpha_{i+1} - \alpha_i) \cdot \frac{1}{\langle E_L \rangle^{\alpha}_{i+1} - \langle E_L \rangle^{\alpha}_{i}} \tag{4}$$

We also introduce the notation $\frac{\partial \langle E^i_L \rangle}{\partial \alpha} = \langle E_L \rangle^{\alpha}_i$ for brevity, and the same notation will be used for the tiral wavefunction ψ_T. The partial derivative itself can be found by the equation[2]

$$\langle E_l \rangle^{\alpha}_i = 2 \left(\langle \frac{\psi^{\alpha}}{\psi} E_L \rangle^i - \langle \frac{\psi^{\alpha}}{\psi} \rangle \langle E_L \rangle^i \right) \tag{5}$$

Wherein the fraction of the partial derivative of the wavefunction by the wavefunction must be derived, which is trivial with the ansatz of the wavefunction.

$$\frac{\psi^{\alpha}}{\psi} = \frac{\psi \prod_i^N (x_i^2 + y_i^2 + \beta z_i^2)}{\psi} \tag{6}$$

$$= \prod_i^N (x_i^2 + y_i^2 + \beta z_i^2) \tag{7}$$

This holds for both the interactive and non-interactive case since the correlation wavefunction, $f(r_i, r_j)$, does not have any dependence on α
In the program the only addition to make is the computation of two additional expecation values[3]. Empirically we've also seen that we can reduce the number of MC-cycles quite

[1] an implementation of which can be seen in the file `main_b.cpp`

[2] The equation was retrieved from the lecture notes

[3] implemented in `vmc.cpp`

drastically and still get a good convergence to the optimal α. In summary gradient descent and other associated methods allows for a radical speedup in estimation of the optimal values of the variational parameters. The implementation of this method can be found in the file `main_f.cpp`.

2.5 Blocking Method

For the statistical analysis and error estimates we have used the technique of blocking. We are looking for the expectation value of the ground state energy of our system. In order to say something about how accurate these results are we want to look at the standard deviation.

If our samples where uncorrelated we could calculate the standard deviation through the following equation

$$\sigma = \sqrt{\frac{1}{n-1}\left(\langle E_L^2\rangle - \langle E_L\rangle^2\right)}.$$

It can be showed that for correlated samples the equation for the standard deviation is

$$\sigma = \sqrt{\frac{1+2\tau/\Delta t}{n-1}\left(\langle E_L^2\rangle - \langle E_L\rangle^2\right)}$$

where τ is the time between one sample and the next uncorrelated sample and Δt is the time between each sample. If we knew what τ was we could simply find our σ, but since we don't we use blocking to find it. The method is simple enough, we divide our samples of data into blocks and calculating the mean of each block. So if we have an amount of samples of $\langle E_L\rangle$, we divide these into M blocks and calculate the mean. By plotting the standard deviation as function of the block size we can keep blocking and blocking until we see that the std. dev. stops increasing, and this is where the blocks are uncorrelated. Now we have an estimate for τ and thus the standard deviation.

2.6 Numerical differentiation

To evaluate the local energy of the system as defined in the project it necessary to compute the second derivative of the trial wavefunction ψ_T. We've chosen to implement the numerical differentiation as a finite difference approximation. Let $\mathbf{R}$ be the row major $N \times D$ matrix where N is the number of particles and D is their dimension. Then the second derivative can be found by the procedure listed as algorithm 2

Since the derivative involves three function calls for each particle the numerical derivative will obviously be quite computationally expensive. It is noted that the differentiation could be substantially optimized from the version included in the code, but is outside the scope of this project.

Data: matrix $\mathbf{R}$
Result: float $\nabla^2\psi_T(R)$

$\Delta = 0$
$\mathbf{R_p} = \mathbf{R}$
$\mathbf{R_m} = \mathbf{R}$
for i *in* $[0, N-1]$ **do**
    for j *in* $[0, D-1]$ **do**
        $\mathbf{R_p}(i,j) += h$
        $\mathbf{R_m}(i,j) -= h$
        $\Delta = \psi_T(\mathbf{R_p}) + \psi_T(\mathbf{R_m}) - 2\psi_T(\mathbf{R})$
        $\mathbf{R_p}(i,j) -= h$
        $\mathbf{R_m}(i,j) += h$
    end
end
return $\frac{\Delta}{h^2}$

Algorithm 2: Numerical differentiation of the second order of the trial wavefunction on a system $\mathbf{R}$

2.7 One-body densities

The one body density can be defined in terms of an arbitrary position vector $\mathbf{r}_1$ as

$$\rho(\mathbf{r}_1) = \int d\mathbf{r}_2 \dots d\mathbf{r}_N |\Psi(\mathbf{r}_1, \mathbf{r}_2, \dots, \mathbf{r}_N)|^2 \tag{8}$$

It's a useful size to compute since it relates the number of particles in the simulation to the density by way of equation 9

$$N = \int d\mathbf{r} \rho(\mathbf{r}) \tag{9}$$

It also provides a complete picture of the relative distribution of particles. Which can be used to visually identify the ground state[4].

[4]In the odd case where the computation of $\frac{\partial \langle E_L(\alpha_i) \rangle}{\partial \alpha_i}$ for variational optimization is unavailable or too cumbersome one could ostensibly use very simple ML algorithms to optimize the variational parameter.

3 Method

Keeping the stuff here for examples. Might even be relevant.

3.1 Randomization

Randomizing the transaction factor ϵ, and the picking of financial agents, `agent_one` and `agent_two`, was done by initializing the following random number generators (RNGs):

```
std::random_device rd;
std::mt19937_64 gen(rd());
std::uniform_int_distribution<int> AgentPicker(0,NAgents-1);
std::uniform_real_distribution<double> TransactionFactorGenerator
   (0.0,1.0);
// Calling RNGs to initialize agents and transaction factor:
agent_one = AgentPicker(gen);
agent_two = AgentPicker(gen);
TransactionFactor = TransactionFactorGenerator(gen);
```

3.2 Conservation of money

A potential source of money "leaks" in the simulations is if `agent_one` = `agent_two`. In this case the system would "leak" an amount of money equal to $\epsilon(m_1 + m_2)$, propagating for each transaction where that agent is involved, and for each subsequent instance of the error. This was handled by a simple test

```
if (agent_one == agent_two){
    continue;
}
```

which throws away the transactions where this would happen.

4 Results

1b): Developing your code and 1d): A better statistical analysis

The results of the simulations are included in the appendix as figures 4 5, 6 and 7. The results were produced with the code in the repository[5]. Errors in the expectation value for the energy in the plots are computed with the blocking method[6]. The exact answer for the minimum of the expecation value of the energy can be derived quote easily from 26. The minimum of the local energy has to be where the kinetic and potential energies cancel exactly. In the case of the 3d particle we see the following after imposing natural units.

$$\min(E_L) \implies \left(\frac{1}{2}\right)[4\alpha^2(x^2+y^2+\beta^2 z^2)] = \frac{1}{2}(x^2+y^2+\beta z^2) \tag{10}$$

$$2\alpha^2 = \frac{1}{2} \tag{11}$$

$$\alpha = 0.5 \tag{12}$$

With α at the minimum we then expect the following values for $\langle E \rangle$

$$\begin{aligned} &\text{1D: } E_L|_{\alpha=0.5} = \sum_i^N \alpha = N_\alpha, \\ &\text{2D: } E_L|_{\alpha=0.5} = \sum_i^N 2\alpha = 2N\alpha \\ &\text{3D: } E_L|_{\alpha=0.5} = \sum_i^N 2\alpha + \alpha\beta = 3N\alpha \end{aligned} \tag{13}$$

In general the energy in the non-interactive case at the variational minimum for our trial wavefunction can then be expressed as

$$\min(\langle E[\alpha] \rangle) = 0.5 \cdot N \cdot D \tag{14}$$

Where N is the number of particles and D is their dimension. From the figures referenced in the beginning of this section it is clear that both the numerical and analytic solutions converge to the correct solution.

1e: The repulsive interaction

Comment to be added to discussion: we see that the expectation value of the local energy increases by a small amount. This is consistent with what we see in Fig. 2 in (ref. to paper in Physics by J.L. DuBois).

[5] at commit `b0ff612bc6666335b106af5e22a7a13a13c7cff7`

[6] the blocking implementation is a copy of the code in Ref. [**ComphysGit**]

N	α	σ^2	$\langle E_L \rangle$	$\frac{\langle E_L \rangle}{N}$	CPU-time [sek]
10	0.480013	1.73405e-06	21.643	2.1643	12.31
50	0.480127	1.75833e-06	111.066	2.2213	1253.91
100	0.480279	1.75902e-06	228.682	2.2868	9854.01

Table 1: VMC with Metropolis sampling. Calculations done in 3 dimensions with number of cycles $N_{MC} = 10^6$ for an elliptical trap, $\beta = 2.82843$ with interaction, $a = 0.0043$. Step length $t = 0.5$.

1f: Finding the best parameter

Applying the gradient descent method to tune the variational parameter resulted in satisfying convergence to the optimal value for the variational parameter α. The simulation results are included in figure 1[7]. After the convergence was shown to be satisfactory for the non-interactive case simulations were run for the hamiltonian with interaction [8]. The result shows for the variational parameter as $\alpha_{min} \approx 0.4988\ldots$. These findings are illustrated in figure 2.

1g: Onebody densities

The onebody density was computed the counting of particles in areas around the origo. The origo was chosen as it is the natural focal point of particles in the harmonic oscillator as formulated in the project. The bins were equipartitioned by a set step length trivially determined as $\Delta L = L/n_{bins}$ where L denotes the endpoint of the interval and n_{bins} the total number of bins. The implementation of the algorithm can be found in `vmc.cpp`. A simulation was run to compute the onebody densities for both the interactive and non-interacitve case for the respective minimas of the variational parameter α.

[7]The simulations were produced with the file `mainf_f.cpp` at commit `2d03030ec5a711383d582dde23869483d122c6b6`

[8]commit `f285327a7d95349a874aafe16d8ecd5f48c5cef5`

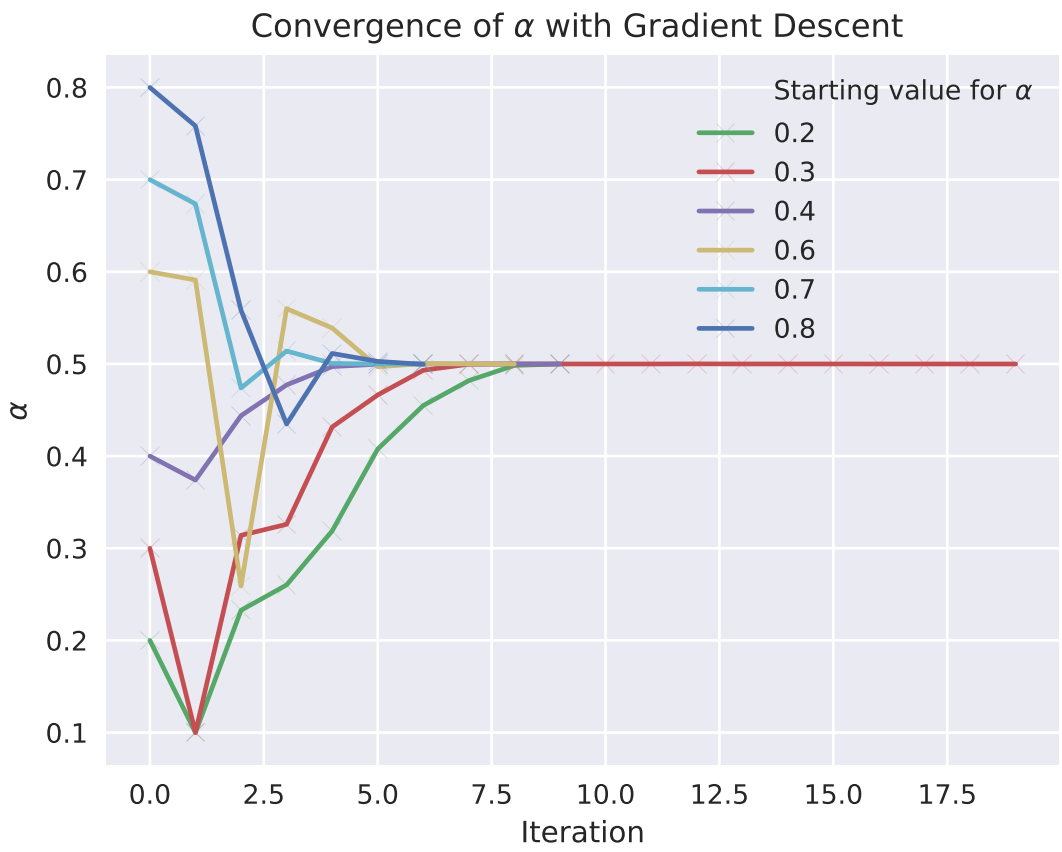


Figure 1: Finding the optimal α with GD. This plot show the number of iterations it took to reach the minimum for the non-interactive spherical trap.

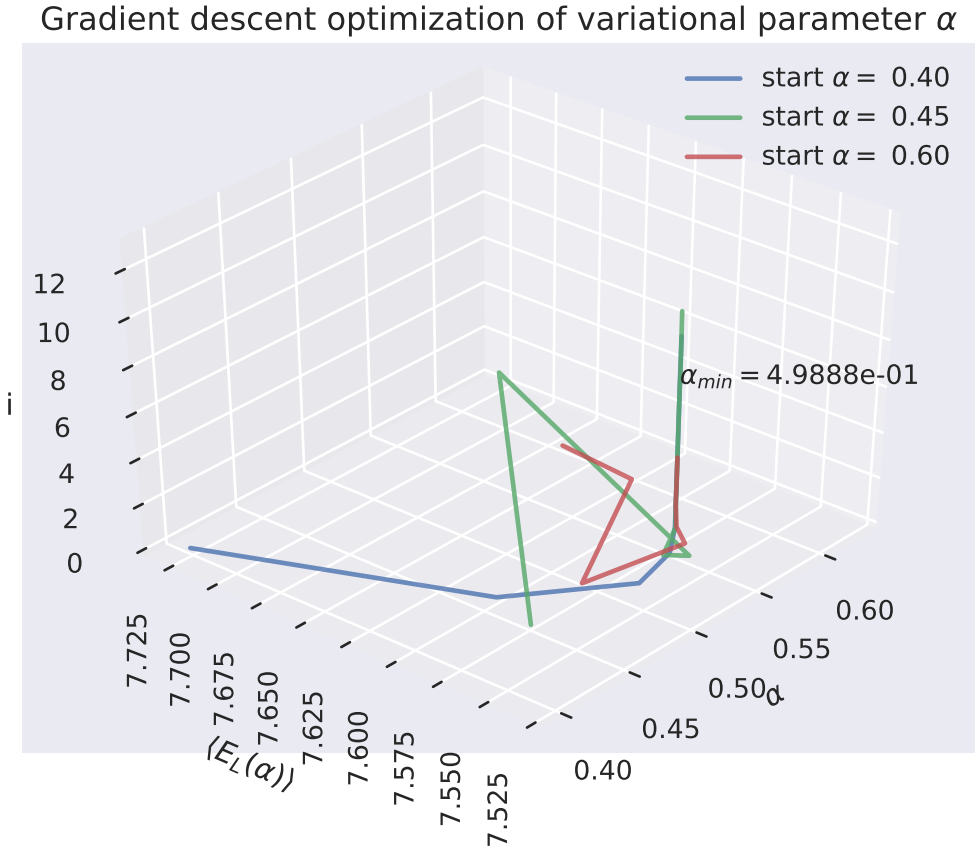


Figure 2: Finding the optimal α with GD. This plot shows the first central moment for the local energy on the y-axis, the variational parameter on the x axis and the GD iteration number on the z-axis for the interactive spherical trap. We observe that the GD algorithm finds a satisfyingly stable minimum with few iterations

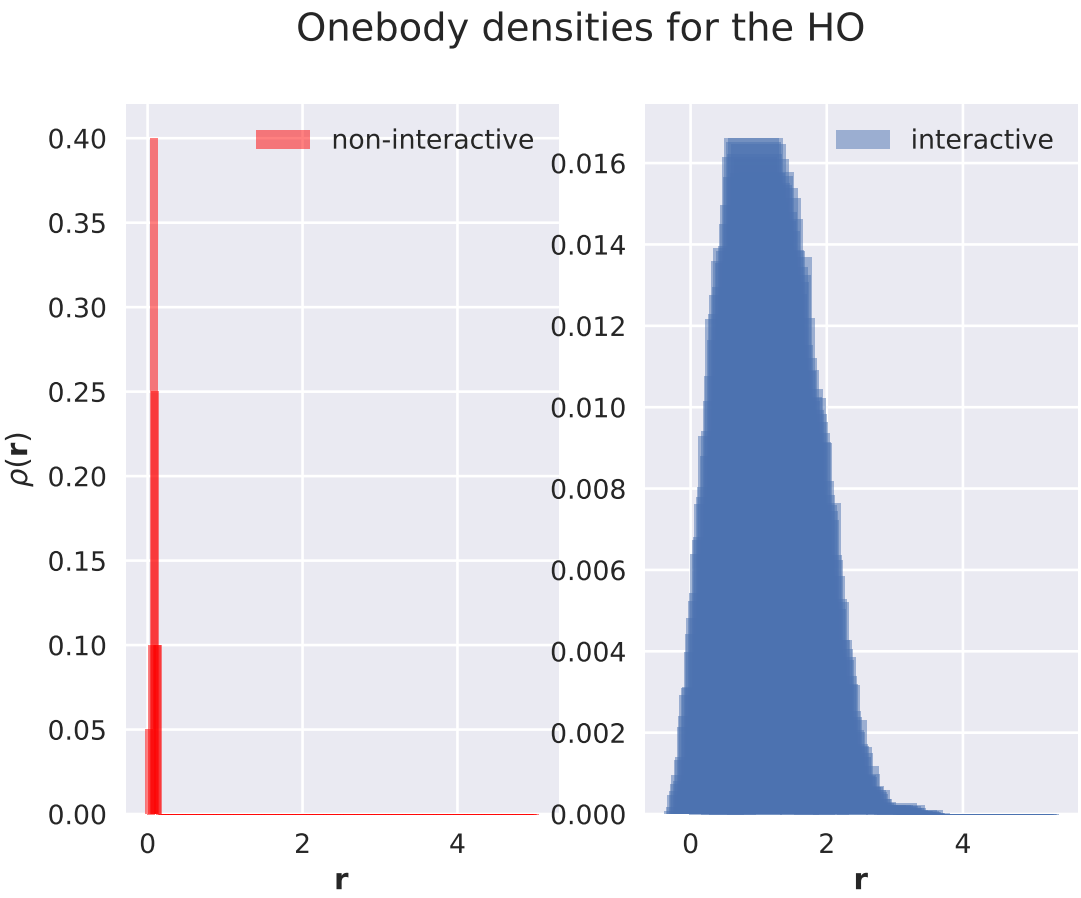


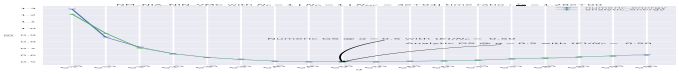
Figure 3: With the optimal parameters for alpha the onebody densities are computed and shown above. We observe that the jastrow factor has a large and pronounced effect on the relative distribution of particles. Additionally we note that for the non-interactive case we expect a density that approximates a delta function as figuratively all bosons will be in the HO ground state.

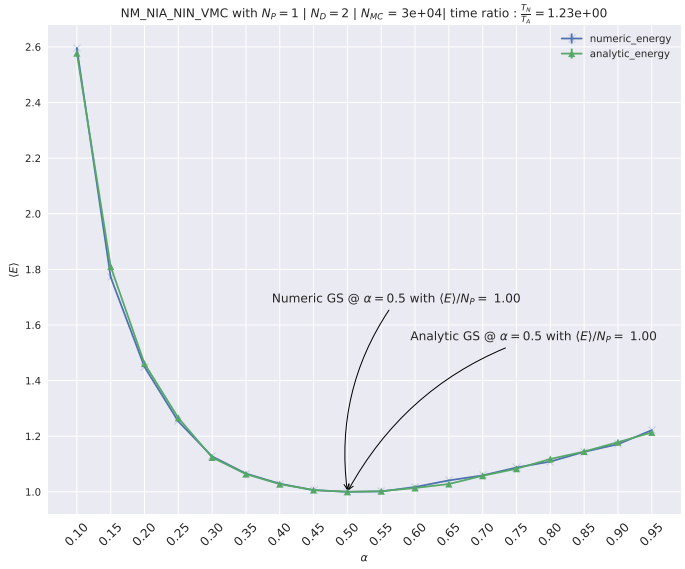
5 Discussion

Tekst

6 Conclusion

Herein lies the conclusions of yonder project, verily I say!





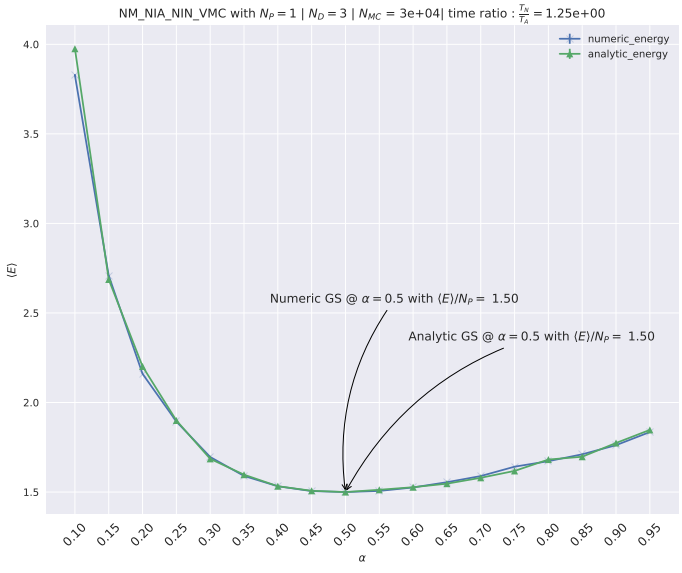


Figure 4: Plots for simulations with 1 particle and 1-3 dimensions. The CPU time difference is noted in the figure captions

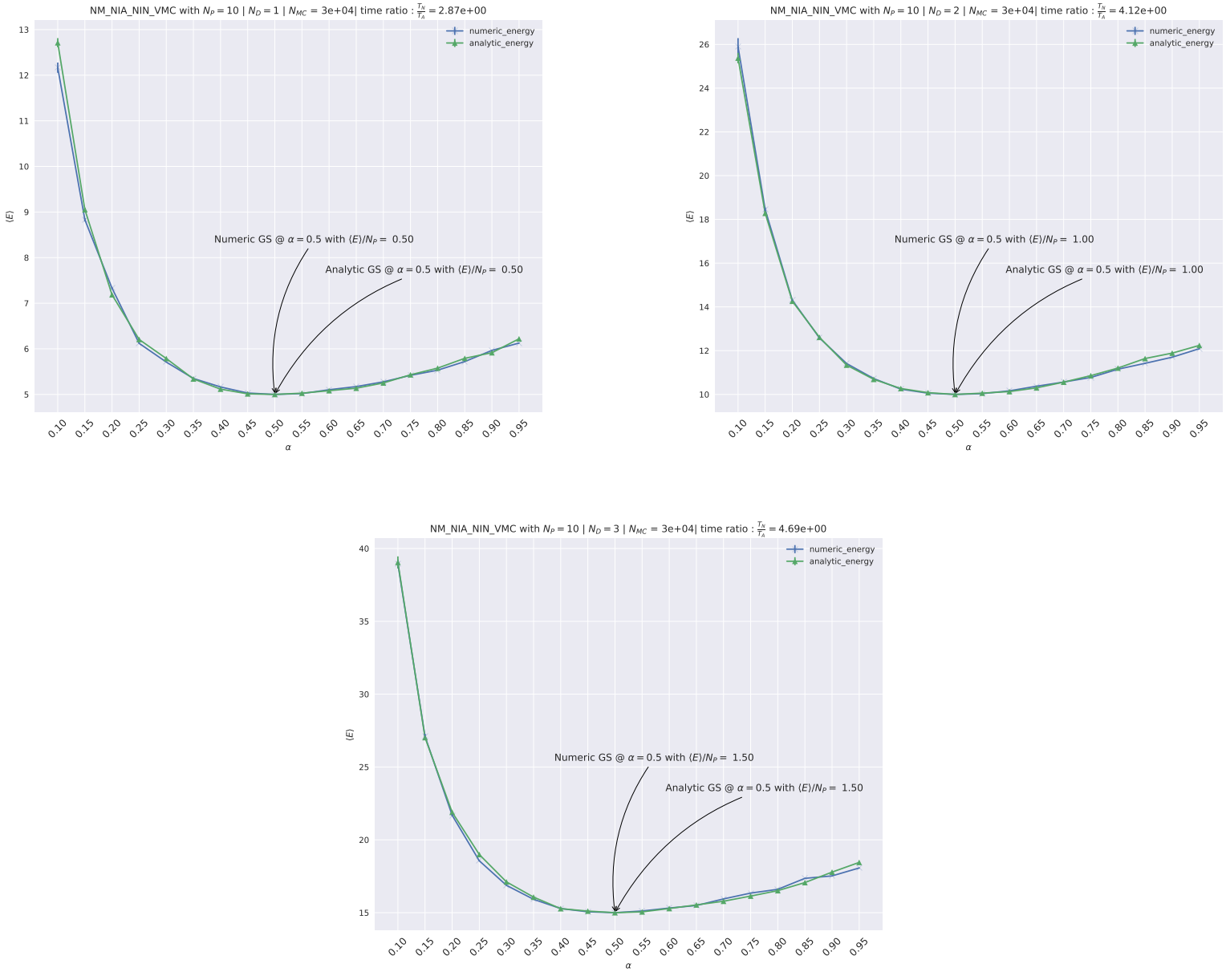


Figure 5: Plots for simulations with 10 particles and 1-3 dimensions. The CPU time difference is noted in the figure captions

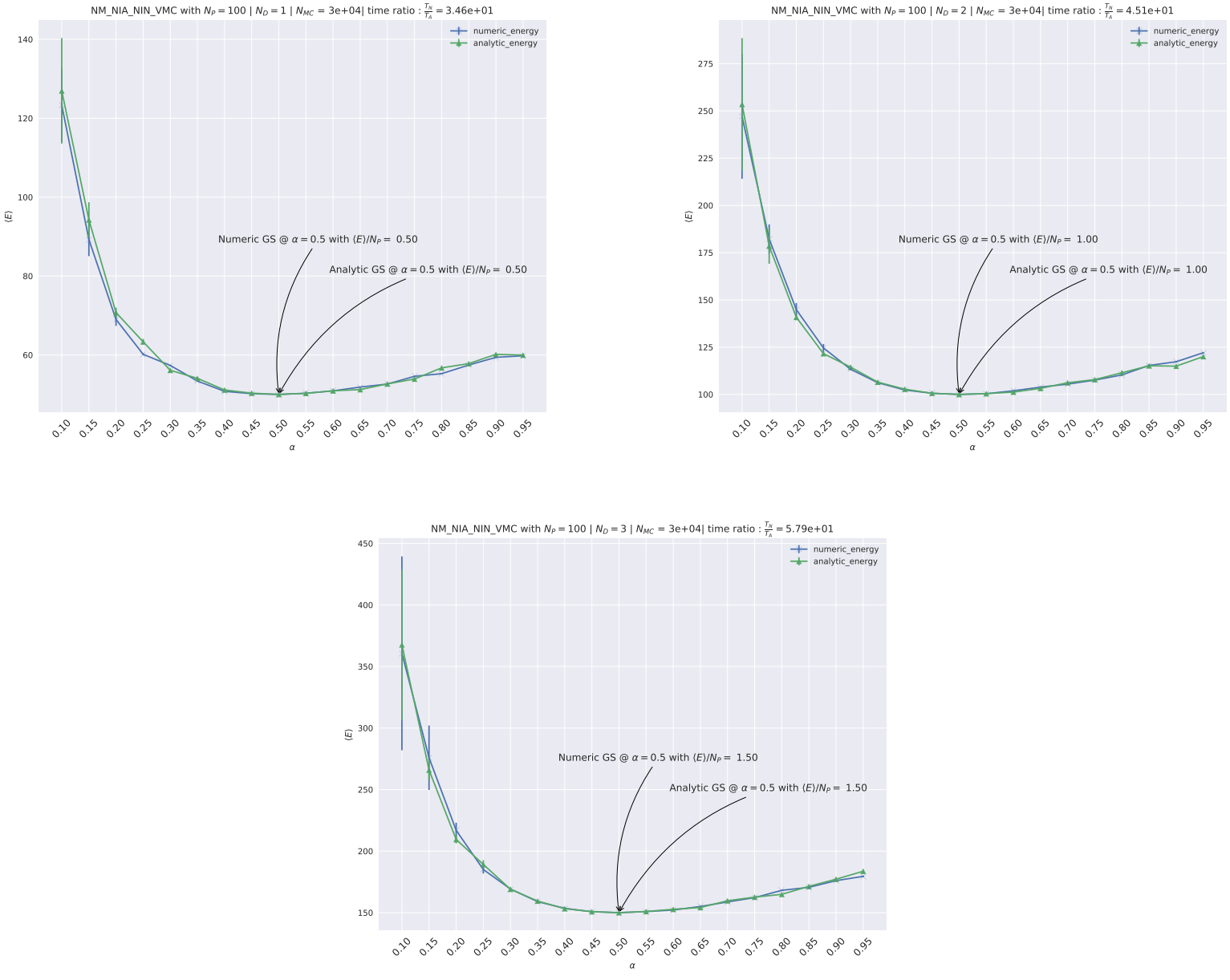
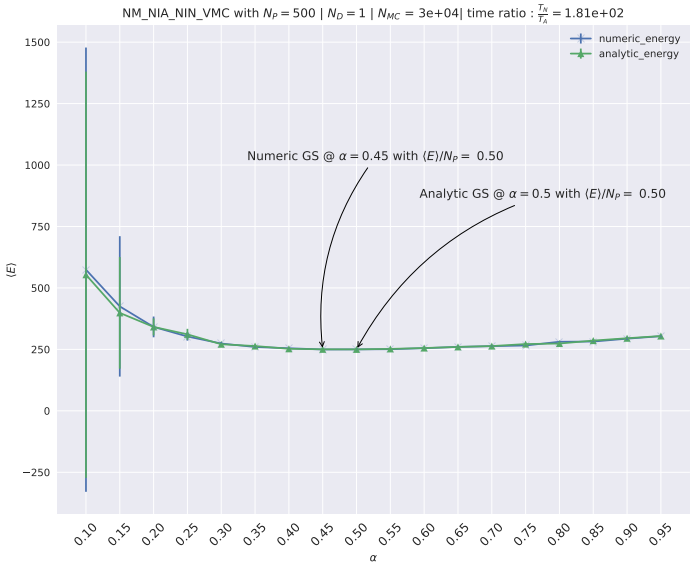
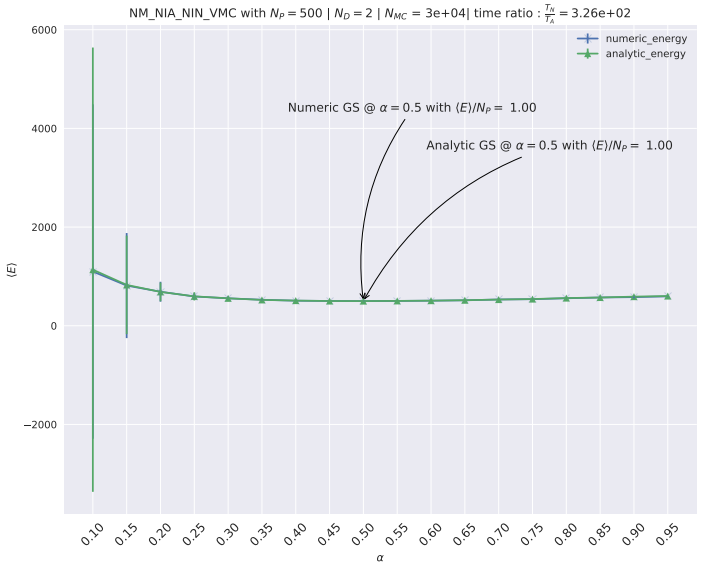


Figure 6: Plots for simulations with 100 particles and 1-3 dimensions. The CPU time difference is noted in the figure captions





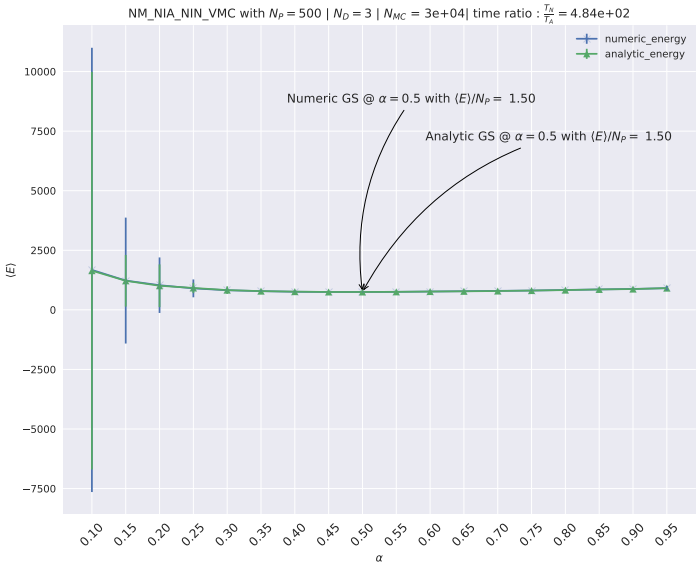


Figure 7: Plots for simulations with 500 particles and 1-3 dimensions. The CPU time difference is noted in the figure captions

Appendix A - Problem 1

The local energy is given by

$$E_L(\mathbf{r}) = \frac{1}{\Psi_T(\mathbf{r})}\hat{H}\Psi_T(\mathbf{r})$$

with trial wave equation

$$\Psi_T(\mathbf{r}) = \prod_i g(\alpha, \beta, \mathbf{r_i}) \prod_{i<j} f(a, |\mathbf{r_i} - \mathbf{r_j}|)$$

The Hamiltonian becomes

$$\hat{H} = \sum_i^N \left(\frac{-\hbar^2}{2m}\nabla_i^2 + V_{ext}(\mathbf{r_i}) \right) + \sum_{i>j}^N V_{int}(\mathbf{r_i}, \mathbf{r_j})$$

where

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

and

$$V_{int}(|\mathbf{r_i} - \mathbf{r_j}|) = \begin{cases} \infty & |\mathbf{r_i} - \mathbf{r_j}| \leq a \\ 0 & |\mathbf{r_i} - \mathbf{r_j}| > a \end{cases}$$

First we solve only the harmonic oscillator (a=0) and we use $\beta = 1$ for one particle in 1D. And for one particle the trial wave equation, Hamiltonian and local energy becomes

$$\Psi_T(\mathbf{r}) = g(\alpha, x) = e^{-\alpha x_i^2}, \tag{15}$$

$$\hat{H} = \left(\frac{-\hbar^2}{2m}\nabla^2 + \frac{1}{2}m\omega_{ho}^2 x^2 \right) \tag{16}$$

$$E_L = \frac{1}{e^{-\alpha x^2}} \left(\frac{-\hbar^2}{2m}\nabla^2(e^{-\alpha x^2}) + \frac{1}{2}m\omega_{ho}^2 x^2 (e^{-\alpha x^2}) \right) \tag{17}$$

We then need to compute the laplacian of the trial wave-function.

$$\nabla^2(e^{-\alpha x^2}) = \frac{d^2 e^{-\alpha x^2}}{dx^2} \tag{18}$$

$$= \frac{d}{dx}\left(\frac{de^{-\alpha x^2}}{dx} \right) \tag{19}$$

$$= \frac{d}{dx}(-2\alpha x \cdot e^{-\alpha x^2}) \tag{20}$$

$$= (4\alpha^2 x^2 - 2\alpha)e^{-\alpha x^2} \tag{21}$$

Finally, the local energy becomes

$$E_L = \frac{e^{-\alpha x^2}}{e^{-\alpha x^2}} \left(\frac{-\hbar^2}{2m}(4\alpha^2 x^2 - 2\alpha) + \frac{1}{2} m\omega_{ho}^2 x^2 \right) \tag{22}$$

$$E_L = \frac{-\hbar^2}{2m}(4\alpha^2 x^2 - 2\alpha) + \frac{1}{2} m\omega_{ho}^2 x^2 \tag{23}$$

Solving the same problem for one particle in 2D and 3D we get something similar,

$$E_L = (\frac{-\hbar^2}{2m})4\alpha^2(x^2 + y^2) - 4\alpha + \frac{1}{2} m\omega_{ho}^2 (x^2 + y^2), \tag{24}$$

$$E_L = (\frac{-\hbar^2}{2m})4\alpha^2(x^2 + y^2 + z^2) - 6\alpha + \frac{1}{2} m\omega_{ho}^2 (x^2 + y^2 + z^2) \tag{25}$$

We can now easily see that the local energy energy for N number of particles in 1D, 2D and 3D becomes

$$\begin{aligned}
&\text{1D: } E_L = \sum_i^N (\frac{-\hbar^2}{2m})[4\alpha^2(x^2) - 2\alpha] + \frac{1}{2} m\omega_{ho}^2 (x^2), \\
&\text{2D: } E_L = \sum_i^N (\frac{-\hbar^2}{2m})[4\alpha^2(x^2 + y^2) - 4\alpha] + \frac{1}{2} m\omega_{ho}^2 (x^2 + y^2) \\
&\text{3D: } E_L = \sum_i^N (\frac{-\hbar^2}{2m})[4\alpha^2(x^2 + y^2 + \beta^2 z^2) - 4\alpha - 2\alpha\beta] + \frac{1}{2} m\omega_{ho}^2 (x^2 + y^2 + \beta z^2)
\end{aligned} \tag{26}$$

since the energy is simply the sum of the derivatives of each particle.
Now we can try to solve the complete problem. The trial wave function is now

$$\Psi_T(\mathbf{r}) = \prod_i g(\alpha, \beta, \mathbf{r_i}) \prod_{i<j} f(a, |\mathbf{r_i} - \mathbf{r_j}|)$$

and first we rewrite it using

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

and

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

getting

$$\Psi_T(\mathbf{r}) = \prod_i \phi(r_i) \exp\left(\sum_{i<j} u(r_{ij})\right).$$

The local energy for this problem becomes:

$$E_L = \frac{1}{\Psi_T(\boldsymbol{r})}\left(\sum_i^N \left(\frac{\hbar^2}{2m}\nabla_i^2\Psi_T(\mathbf{r}) + \frac{1}{2}m\omega_{ho}^2 r^2\Psi_T(\mathbf{r})\right) + \sum_{i<j}^N V_{int}(\mathbf{r_i},\mathbf{r_j})\Psi_T(\mathbf{r})\right). \quad (27)$$

The difficulty in (27) is solving the derivatives of the wave equation given the complexity of the exponential. We begin with the first derivative.

$$\nabla_i^2 \prod_i \phi(\mathbf{r_i}) \exp\left(\sum_{i<j} u(r_{ij})\right) = \nabla_i \cdot \nabla_i \prod_i \phi(\mathbf{r_i}) \exp\left(\sum_{i<j} u(r_{ij})\right)$$

The first derivative of particle k:

$$\nabla_k \prod_i \phi(\mathbf{r_i}) \exp\left(\sum_{i<j} u(r_{ij})\right) = \nabla_k \left(\prod_i \phi(\mathbf{r_i})\right) \exp\left(\sum_{i<j} u(r_{ij})\right) + \nabla_k \left(\exp\left(\sum_{i<j} u(r_{ij})\right)\right) \prod_i \phi(r_i)$$

$$\begin{aligned}
\nabla_k \left(\prod_i \phi(\mathbf{r_i})\right) &= \nabla_k(\phi(r_1)(\phi(r_2)...(\phi(r_k)..(\phi(r_N)) \\
&= \nabla_k \left(e^{-\alpha(x_1^2+y_1^2+z_1^2)} e^{-\alpha(x_2^2+y_2^2+z_2^2)} ... e^{-\alpha(x_k^2+y_k^2+z_k^2)} ... e^{-\alpha(x_N^2+y_N^2+z_N^2)}\right) \\
&= \nabla_k \phi(r_k) \left[\prod_{i\neq k} \phi(\mathbf{r_i})\right]
\end{aligned}$$

$$\begin{aligned}
\nabla_k \exp\left(\sum_{i<j} u(r_{ij})\right) &= \nabla_k \exp(u(r_{12}) + u(r_{13}) + ... + u(r_{23}) + ... + u(r_{kj}) + .. + u(r_{N-1,N})) \\
&= \exp\left(\sum_{i<j} u(r_{ij})\right) \sum_{i\neq k} \nabla_k u(r_{kj})
\end{aligned}$$

And the first derivative of the trial wave equation is

$$\nabla_k \Psi_T(\mathbf{r}) = \nabla_k \phi(r_k) \left[\prod_{i\neq k} \phi(\mathbf{r_i})\right] \exp\left(\sum_{i<j} u(r_{ij})\right) + \prod_i \phi(\mathbf{r_i}) \exp\left(\sum_{i<j} u(r_{ij})\right) \sum_{j\neq k} \nabla_k u(r_{kj}).$$

Now we find the second derivative of the wave function.

$$\frac{1}{\Psi_T(\mathbf{r})}\nabla_k^2\Psi_T(\mathbf{r}) = \frac{1}{\nabla_k \prod_i \phi(\mathbf{r_i}) \exp\left(\sum_{i<j} u(r_{ij})\right)} \left(\nabla_k \left(\nabla_k \phi(r_k) \left[\prod_{i\neq k} \phi(\mathbf{r_i}) \right] \right) \cdot \exp\left(\sum_{i<j} u(r_{ij}) \right) \right.$$

$$+ \nabla_k \phi(r_k) \left[\prod_{i\neq k} \phi(\mathbf{r_i}) \right] \cdot \nabla_k \left(\exp\left(\sum_{i<j} u(r_{ij}) \right) \right)$$

$$+ \nabla_k \left(\prod_i \phi(\mathbf{r_i}) \right) \cdot \exp\left(\sum_{i<j} u(r_{ij}) \right) \sum_{j\neq k} \nabla_k u(r_{kj})$$

$$+ \prod_i \phi(\mathbf{r_i}) \cdot \nabla_k \left(\exp\left(\sum_{i<j} u(r_{ij}) \right) \right) \sum_{j\neq k} \nabla_k u(r_{kj})$$

$$\left. + \prod_i \phi(\mathbf{r_i}) \exp\left(\sum_{i<j} u(r_{ij}) \right) \cdot \nabla_k \left(\sum_{j\neq k} \nabla_k u(r_{kj}) \right) \right)$$

Solving these equations separately makes it easier.

$$\frac{\nabla_k^2 \phi(r_k) \left[\prod_{i\neq k} \phi(\mathbf{r_i}) \right] \cdot \exp\left(\sum_{i<j} u(r_{ij}) \right)}{\nabla_k \prod_i \phi(\mathbf{r_i}) \exp\left(\sum_{i<j} u(r_{ij}) \right)} = \frac{\nabla_k^2 \phi(r_k)}{\phi(r_k)}$$

$$\frac{\nabla_k \phi(r_k) \left[\prod_{i\neq k} \phi(\mathbf{r_i}) \right] \cdot \nabla_k \left(\exp\left(\sum_{i<j} u(r_{ij}) \right) \right)}{\nabla_k \prod_i \phi(\mathbf{r_i}) \exp\left(\sum_{i<j} u(r_{ij}) \right)} = \frac{\nabla_k \phi(r_k)}{\phi(r_k)} \sum_{j\neq k} \nabla_k u(r_{kj})$$

$$\frac{\nabla_k \left(\prod_i \phi(\mathbf{r_i}) \right) \cdot \exp\left(\sum_{i<j} u(r_{ij}) \right) \sum_{j\neq k} \nabla_k u(r_{kj})}{\nabla_k \prod_i \phi(\mathbf{r_i}) \exp\left(\sum_{i<j} u(r_{ij}) \right)} = \frac{\nabla_k \phi(r_k)}{\phi(r_k)} \sum_{j\neq k} \nabla_k u(r_{kj})$$

$$\frac{\prod_i \phi(\mathbf{r_i}) \cdot \nabla_k \left(\exp\left(\sum_{i<j} u(r_{ij}) \right) \right) \sum_{j \neq k} \nabla_k u(r_{kj})}{\nabla_k \prod_i \phi(\mathbf{r_i}) \exp\left(\sum_{i<j} u(r_{ij}) \right)} = \sum_{i \neq k} \nabla_k u(r_{ki}) \sum_{j \neq k} \nabla_k u(r_{kj})$$

$$\frac{\prod_i \phi(\mathbf{r_i}) \exp\left(\sum_{i<j} u(r_{ij}) \right) \cdot \nabla_k \left(\sum_{j \neq k} \nabla_k u(r_{kj}) \right)}{\nabla_k \prod_i \phi(\mathbf{r_i}) \exp\left(\sum_{i<j} u(r_{ij}) \right)} = \sum_{j \neq k} \nabla_k^2 u(r_{kj})$$

Putting them together again we get the following

$$\frac{1}{\Psi_T(\mathbf{r})} \nabla_k^2 \Psi_T(\mathbf{r}) = \frac{\nabla_k^2 \phi(r_k)}{\phi(r_k)} + 2 \frac{\nabla_k \phi(r_k)}{\phi(r_k)} \sum_{j \neq k} \nabla_k u(r_{kj}) + \sum_{i \neq k} \nabla_k u(r_{ki}) \sum_{j \neq k} \nabla_k u(r_{kj}) + \sum_{j \neq k} \nabla_k^2 u(r_{kj}) \tag{28}$$

We solve the first and second derivatives of $u(r_{kj})$.

$$\nabla_k u(r_{kj}) = \left(\vec{i} \frac{\partial}{\partial x_k} + \vec{j} \frac{\partial}{\partial y_k} + \vec{k} \frac{\partial}{\partial z_k} \right) u(r_{kj}) \tag{29}$$

From Rottmann p. 128 we have that

$$\begin{aligned}
\frac{\partial u(r_{kj})}{\partial x_k} \vec{i} &= \frac{\partial u(r_{kj})}{\partial r_{kj}} \frac{\partial r_{kj}}{\partial x_k} \vec{i} \\
&= u'(r_{kj}) \frac{\partial \sqrt{|x_k - x_j|^2 + |y_k - y_j|^2 + |z_k - z_j|^2}}{\partial x_k} \vec{i} \\
&= u'(r_{kj}) \frac{1}{2} 2 |x_k - x_j| \frac{1}{r_{kj}} \vec{i} \\
&= \frac{u'(r_{kj}) |x_k - x_j|}{r_{kj}} \vec{i} \\
\frac{\partial u(r_{kj})}{\partial y_k} \vec{j} &= \frac{u'(r_{kj}) |y_k - y_j|}{r_{kj}} \vec{j} \\
\frac{\partial u(r_{kj})}{\partial z_k} \vec{k} &= \frac{u'(r_{kj}) |z_k - z_j|}{r_{kj}} \vec{k}
\end{aligned}$$

where we have used the fact that $r_{kj} = \sqrt{|x_k - x_j|^2 + |y_k - y_j|^2 + |z_k - z_j|^2}$. Now equation (29) becomes

$$\begin{aligned}
\nabla_k u(r_{kj}) &= u'(r_{kj}) \frac{(|x_k - x_j| \vec{i} + |y_k - y_j| \vec{j} + |z_k - z_j| \vec{k})}{r_{kj}} \\
&= u'(r_{kj}) \frac{(\boldsymbol{r_k} - \boldsymbol{r_j})}{r_{kj}}.
\end{aligned}$$

And for the second derivative have that

$$\nabla_k^2 u(r_{kj}) = \left(\frac{\partial^2}{\partial x_k^2} + \frac{\partial^2}{\partial y_k^2} + \frac{\partial^2}{\partial z_k^2} \right) u(r_{kj}) \tag{30}$$

and use Rottmann p. 128 again and see that

$$\begin{aligned}
\frac{\partial^2 u(r_{kj})}{\partial x_{kj}^2} &= \frac{\partial^2 u(r_{kj})}{\partial r_{kj}^2} \left(\frac{\partial r_{kj}}{\partial x_{kj}} \right)^2 + \frac{\partial u(r_{kj})}{\partial r_{kj}} \frac{\partial^2 r_{kj}}{\partial x_{kj}^2} \\
&= u''(r_{kj}) \frac{(x_k - x_j)^2}{r_{kj}^2} + u'(r_{kj}) \frac{\partial^2 r_{kj}}{\partial x_{kj}^2} \\
\frac{\partial^2 r_{kj}}{\partial x_{kj}^2} &= \frac{\partial^2 \sqrt{|x_k - x_j|^2 + |y_k - y_j|^2 + |z_k - z_j|^2}}{\partial x_{kj}^2} \\
&= \frac{\partial \frac{x_k - x_j}{\sqrt{|x_k - x_j|^2 + |y_k - y_j|^2 + |z_k - z_j|^2}}}{\partial x_{kj}} \\
&= \frac{r_{kj} - \frac{1}{2}(x_k - x_j) \cdot \frac{2(x_k - x_j)}{r_{kj}}}{r_{kj}^2} \\
&= \frac{1}{r_{kj}} - \frac{(x_k - x_j)^2}{r_{kj}^3}.
\end{aligned}$$

These equations put together and solving with respect to y and z we get

$$\begin{aligned}
\frac{\partial^2 u(r_{kj})}{\partial x_{kj}^2} &= u''(r_{kj}) \frac{(x_k - x_j)^2}{r_{kj}^2} + u'(r_{kj}) \left(\frac{1}{r_{kj}} - \frac{(x_k - x_j)^2}{r_{kj}^3} \right), \\
\frac{\partial^2 u(r_{kj})}{\partial y_{kj}^2} &= u''(r_{kj}) \frac{(y_k - y_j)^2}{r_{kj}^2} + u'(r_{kj}) \left(\frac{1}{r_{kj}} - \frac{(y_k - y_j)^2}{r_{kj}^3} \right), \\
\frac{\partial^2 u(r_{kj})}{\partial z_{kj}^2} &= u''(r_{kj}) \frac{(z_k - z_j)^2}{r_{kj}^2} + u'(r_{kj}) \left(\frac{1}{r_{kj}} - \frac{(z_k - z_j)^2}{r_{kj}^3} \right).
\end{aligned}$$

Now we can add them all together and equation (30) becomes

$$
\begin{aligned}
\nabla_k^2 u(r_{kj}) &= \frac{u''(r_{kj})}{r_{kj}^2}((x_k - x_j)^2 + (y_k - y_j)^2 + (z_k - z_j)^2) \\
&+ u'(r_{kj}) \left(\frac{3}{r_{kj}} - \frac{(x_k - x_j)^2 + (y_k - y_j)^2 + (z_k - z_j)^2}{r_{kj}^3} \right) \\
&= \frac{u''(r_{kj}) r_{kj}^2}{r_{kj}^2} + u'(r_{kj}) \left(\frac{3}{r_{kj}} - \frac{r_{kj}^2}{r_{kj}^3} \right) \\
&= u''(r_{kj}) + u'(r_{kj}) \frac{2}{r_{kj}}
\end{aligned}
$$

Now we can write out the complete second derivative, equation (28)

$$
\begin{aligned}
\frac{1}{\Psi_T(\mathbf{r})} \nabla_k^2 \Psi_T(\mathbf{r}) &= \frac{\nabla_k^2 \phi(r_k)}{\phi(r_k)} + 2 \frac{\nabla_k \phi(r_k)}{\phi(r_k)} \sum_{j \neq k} \frac{(\boldsymbol{r_k} - \boldsymbol{r_j})}{r_{kj}} u'(r_{kj}) \\
&+ \sum_{ij \neq k} \frac{(\boldsymbol{r_k} - \boldsymbol{r_i})}{r_{ki}} \frac{(\boldsymbol{r_k} - \boldsymbol{r_j})}{r_{kj}} u'(r_{ki}) u'(r_{kj}) + \sum_{j \neq k} \left(u''(r_{kj}) + \frac{2}{r_{kj}} u'(r_{kj}) \right).
\end{aligned}
$$

For the full analytical solution of the interacting problem we solve each term by it self.

$$
\begin{aligned}
\frac{1}{\phi(r_k)} \nabla_k^2 \phi(r_k) &= \frac{\nabla_k^2 \; exp(-\alpha(x_k^2 + y_k^2 + \beta z_k^2))}{exp(-\alpha(x_k^2 + y_k^2 + \beta z_k^2))} \\
&= ((2\alpha(2\alpha x_k^2 - 1)) + (2\alpha(2\alpha y_k^2 - 1)) + (2\alpha\beta(2\alpha\beta z_k^2 - 1))) \cdot \frac{\phi(r_k)}{\phi(r_k)} \\
&= -4\alpha^2 - 2\alpha\beta + 4\alpha^2(x_k^2 + y_k^2 + \beta z_k^2)
\end{aligned}
$$

$$
\begin{aligned}
\frac{1}{\phi(r_k)} \nabla_k \phi(r_k) &= \frac{\nabla_k \; exp(-\alpha(x_k^2 + y_k^2 + \beta z_k^2))}{exp(-\alpha(x_k^2 + y_k^2 + \beta z_k^2))} \\
&= (2\alpha x_k \vec{i} + 2\alpha y_k \vec{j} + 2\alpha\beta z_k \vec{k}) \cdot \frac{\phi(r_k)}{\phi(r_k)} \\
&= (2\alpha x_k \vec{i} + 2\alpha y_k \vec{j} + 2\alpha\beta z_k \vec{k})
\end{aligned}
$$

Wolfram alpha gives to solution the first and second derivatives of function $u(r_{kj})$.

$$
\begin{aligned}
u'(r_{kj}) &= \frac{d(\ln f(r_{kj}))}{dr_{kj}} \\
&= \frac{d\ \ln\left(1-\frac{a}{(r_k-r_j)}\right)}{dr_{kj}} \\
&= -\frac{a}{ar_{kj}-r_{kj}^2}
\end{aligned}
$$

$$
\begin{aligned}
u''(r_{kj}) &= \frac{d^2(\ln f(r_{kj}))}{dr_{kj}^2} \\
&= \frac{a(a-2r_{kj})}{r_{kj}^2(a-r_{kj})^2}
\end{aligned}
$$

Putting all of this together we end up with the following expression for the second derivative of the wave equation divided by it self:

$$
\begin{aligned}
\frac{1}{\Psi_T(\mathbf{r})}\nabla_k^2\Psi_T(\mathbf{r}) &= -4\alpha^2 - 2\alpha\beta + 4\alpha^2(x_k^2+y_k^2+\beta z_k^2) \\
&+ 2((2\alpha x_k\vec{i} + 2\alpha y_k\vec{j} + 2\alpha\beta z_k\vec{k}))\sum_{j\neq k}\left(\frac{(x_k-x_j)\vec{i}+(y_k-y_j)\vec{j}+(z_k-z_j)\vec{k}}{r_{kj}}\left(\frac{-a}{ar_{kj}-r_{kj}^2}\right)\right) \\
&+ \sum_{j\neq k}\left(\frac{(x_k-x_i)\vec{i}+(y_k-y_i)\vec{j}+(z_k-z_i)\vec{k}}{r_{ki}}\right)\left(\frac{(x_k-x_j)\vec{i}+(y_k-y_j)\vec{j}+(z_k-z_j)\vec{k}}{r_{kj}}\right) \\
&* \left(\frac{-a}{ar_{ki}-r_{ki}^2}\right)\left(\frac{-a}{ar_{kj}-r_{kj}^2}\right) \\
&+ \sum_{j\neq k}\left(\frac{a(a-2r_{kj})}{r_{kj}^2(a-r_{kj})^2} + \frac{2}{r_{kj}} - \frac{a}{ar_{kj}-r_{kj}^2}\right)
\end{aligned}
$$

As for the drift force we simply write down the solutions for one particle in 1D, 2D and 3D as they are just the first derivative of the wave function,

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

And are easily solved;

$$F = -4\alpha x,$$

$$F = -4\alpha(x+y),$$

$$F = -4\alpha(x + y + \beta z).$$

For N-particles we must solve $\nabla_k \Psi_T$. In 1D this is

$$\nabla_k \Psi_T = -2\alpha x_k \prod_{k \neq i}^{N} e^{-\alpha x_i^2},$$

$$F_k = -4\alpha x_k \frac{\prod_{k \neq i}^{N} e^{-\alpha x_i^2}}{\prod_i^N e^{-\alpha x_i^2}} = -4\alpha x_k \frac{1}{e^{-\alpha x_k^2}},$$

$$F = \sum_i^N -4\alpha x_i \frac{1}{e^{-\alpha x_i^2}}.$$

And for 2D and 3D;

$$F = \sum_i^N -4\alpha(x_i + y_i) \frac{1}{e^{-\alpha(x_i^2 + y_i^2)}},$$

$$F = \sum_i^N -4\alpha(x_i + y_i + \beta z_i) \frac{1}{e^{-\alpha(x_i^2 + y_i^2 + \beta z_i^2)}}.$$

Finally we can solve for N-particles with interaction.
We have already solved the first derivative of the wave equation when we dealt with the local energy problem.

$$\nabla_k \Psi_T(\mathbf{r}) = \nabla_k \phi(r_k) \left[\prod_{i \neq k} \phi(\mathbf{r_i}) \right] \exp\left(\sum_{i<j} u(r_{ij}) \right) + \prod_i \phi(\mathbf{r_i}) \exp\left(\sum_{i<j} u(r_{ij}) \right) \sum_{j \neq k} \nabla_k u(r_{kj})$$

$$\begin{aligned}
\nabla_k \phi(r_k) &= \frac{\partial}{\partial x_k} e^{(-\alpha(x_k^2 + y_k^2 + \beta z_k^2))} \vec{i} \\
&+ \frac{\partial}{\partial y_k} e^{(-\alpha(x_k^2 + y_k^2 + \beta z_k^2))} \vec{j} \\
&+ \frac{\partial}{\partial z_k} e^{(-\alpha(x_k^2 + y_k^2 + \beta z_k^2))} \vec{k} \\
&= (-2\alpha x_k \vec{i} - 2\alpha y_k \vec{j} - 2\alpha\beta z_k \vec{k}) e^{(-\alpha(x_k^2 + y_k^2 + \beta z_k^2))} \\
\nabla_k u(r_{kj}) &= u'(r_{kj}) \frac{(\mathbf{r_k} - \mathbf{r_j})}{r_{kj}} \\
&= -\frac{a(\vec{r_k} - \vec{r_j})}{a r_{kj}^2 - r_{kj})^3}
\end{aligned}$$

$$\nabla_k \Psi_T(\mathbf{r}) = (-2\alpha x_k \vec{i} - 2\alpha y_k \vec{j} - 2\alpha\beta z_k \vec{k})\Psi_T(\mathbf{r}) + \sum_{j \neq k} \left(-\frac{a(\vec{r_k} - \vec{r_j})}{a r_{kj}^2 - r_{kj})^3} \right) \Psi_T(\mathbf{r})$$

$$F = (-4\alpha x_k \vec{i} - 4\alpha y_k \vec{j} - 4\alpha\beta z_k \vec{k}) + \sum_{j \neq k} \left(-\frac{a(\vec{r_k} - \vec{r_j})}{a r_{kj}^2 - r_{kj})^3} \right)$$