Monte Carlo Estimation of Ground State Energy for a Bose Gas

Dalen, Bendik Steinsvåg and Eriksen, Annika Gjolme

(Dated: Wednesday 7th April, 2021)

Presented is a paper on using Monte Carlo calculations to find the ground state energy of a trapped, hard sphere Bose gas. This was done using both a regular, brute force Metropolis algorithm and importance sampling based on the Fokker-Planck and the Langevin equations.

We found that the naive solution produced the expected energy and that the non-interacting case performed as expected. Adding in more functionality did introduce some uncertainty in the accuracy of the simulation.

While there were some statistical methods that didn't give the results we hoped for in terms of efficiency, we also noticed that the energies in the more complex system with a repulsive potential between the particles resulted in little to no effect on the energy. This would benefit from a second look.

I. INTRODUCTION

The physical problem we will be studying in this project is that of a trapped, hard sphere Bose gas. Specifically we want to estimate an upper bound for the ground state energy for a system with differing numbers of particles with a specific trial wave function. Finding this value analytically for a large amount of particles and for high values of the average gas parameter is virtually impossible. Thus many body methods like Monte Carlo calculations may be more appropriate.

In this project we use Monte Carlo simulation to study this problem, both using a brute force Metropolis algorithm and importance sampling. We will also study the differences of including and not including repulsive interaction between particles.

Monte Carlo methods are a widely used family of what can loosely be described as statistical simulation methods. The methods allow for solving ab initio problems in chemistry, physics and more as well as multidimensional integrals.

The Metropolis algorithm is a selection rule for the Monte Carlo simulation. It uses ratios of probabilities to find the likelihood of change rather than the often unknown direct transition probabilities.

Importance sampling is a variation on the Metropolis method, that makes two improvements. It uses the Langevin equation to get a better suggestion about new positions for the particles. It also uses the Fokker-Planck equation to solve Green's function to get a better transition probability.

All code used in solving the project can be found at the following Github repository: **https://github.com/ageriksen/variational-monte-carlo-fys4411**

II. THEORY AND ALGORITHMS

A. Physical Problem

1. Hamiltonian

The general two-body Hamiltonian of the system with N particles has the form

$$H = \sum_i^N \left(\frac{-\hbar^2}{2m}\nabla_i^2 + V_{ext}\left(\mathbf{r}_i\right) \right) + \sum_{i<j}^N V_{int}\left(\left|\mathbf{r}_i - \mathbf{r}_j\right|\right). \quad (1)$$

This can be separated into two parts. The first term is the onebody potential and the second is the repulsive interaction. This distinction separates the Hamiltonian into the one part working on each individual particle, and one part that details the interaction between the particles.

In the onebody potential we find V_{ext}, which is the harmonic trap we'll be using. It has the function

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

Here (S) means a spherical trap and (E) means an elliptical trap. ω_{ho}^2 is the trap frequency in the x, y plane, and ω_z^2 is the frequency in the z direction. For a spherical trap we have $\omega_z^2 = \omega_{ho}^2$. Inserting this we get

$$H_{ob} = \frac{1}{2}m\sum_i^N \left(-\hbar^2\nabla_i^2 + \omega_{ho}^2\left(x^2+y^2\right) + \omega_z^2 z^2\right) \quad (3)$$

The repulsive interaction has the formula

$$V_{int}\left(\left|\mathbf{r}_i - \mathbf{r}_j\right|\right) = \begin{cases} \infty & \left|\mathbf{r}_i - \mathbf{r}_j\right| \leq a \\ 0 & \left|\mathbf{r}_i - \mathbf{r}_j\right| > a \end{cases} \quad (4)$$

This essentially means that two particles can't get closer than a certain distance a. We also employ the shorthand

$$\sum_{i<j}^N V_{ij} \equiv \sum_{i=1}^N \sum_{j=i+1}^N V_{ij} \quad (5)$$

to make the text more readable.

For parts of the project we will be setting $a = 0$. This means that the repulsive interaction will always be 0, and thus be irrelevant. In other words we'll have $H = H_{ob}$.

We also have the characteristic length of the trap, which is $a_{ho} \equiv (\hbar/m\omega_{ho})^{\frac{1}{2}}$. At one point we will introduce length unis in units of a_{ho} and energy in units of $\hbar\omega_{ho}$. This mean the Hamiltonian can be recast into

$$H = \sum_{i=1}^{N} \frac{1}{2}\left(-\nabla_i^2 + x_i^2 + y_i^2 + \gamma^2 z_i^2\right) + \sum_{i<j} V_{int}\left(|\mathbf{r}_i - \mathbf{r}_j|\right), \tag{6}$$

where $\gamma = \frac{\omega_z}{\omega_{ho}}$. See section A 7 for further details.

2. *Wave function*

Our trial wave function for the ground state with N atoms is given by

$$\begin{aligned}\Psi_T(\mathbf{r}) &= \Psi_T\left(\mathbf{r}_1, \mathbf{r}_2, \dots \mathbf{r}_N, \alpha, \beta\right) &(7)\\ &= \left[\prod_i g\left(\alpha, \beta, \mathbf{r}_i\right)\right]\left[\prod_{j<k} f\left(a, |\mathbf{r}_j - \mathbf{r}_k|\right)\right] &(8)\end{aligned}$$

This too can be separated into a onebody part and a correlation part.

g represents the onebody wave function. It has the formula

$$g\left(\alpha, \beta, \mathbf{r}_i\right) = \exp\left[-\alpha\left(x_i^2 + y_i^2 + \beta z_i^2\right)\right], \tag{9}$$

where α and β are variational parameters. For spherical traps we also have $\beta = 1$.

The correlation wave function has the form

$$f\left(a, |\mathbf{r}_i - \mathbf{r}_j|\right) = \begin{cases} 0 & |\mathbf{r}_i - \mathbf{r}_j| \le a \\ \left(1 - \frac{a}{|\mathbf{r}_i - \mathbf{r}_j|}\right) & |\mathbf{r}_i - \mathbf{r}_j| > a \end{cases} \tag{10}$$

As you can see when $a = 0$, f will always be 1 and will thus be irrelevant.

B. Monte Carlo Algorithms

Monte Carlo methods simulate a system by sampling a Probability Distribution Function (PDF) and averaging the result of each sample, or trial. The requirement for a Monte Carlo simulation is that the system in question can be described by a PDF. The PDF is then sampled and each such trial is tallied and averaged.

The law of large numbers shows how the true mean of the system can be approached to an arbitrary precision depending on the amount of samples. Often a variance can also be found which allows for an estimate of the number of trials needed for a given precision.

The trials need to be tallied appropriately to produce the desired result, but the core of the method is in stochastic sampling techniques, perhaps with some algebra, to arrive at a solution to the problem at hand. A natural class of relevant systems are random processes, e.g. diffusion, as they can be described via PDFs. The definition is valid in a broader context. So long as the solution to the problem can be expressed in terms of a PDF, a Monte Carlo method is viable.

The basic Monte Carlo strategy can be boiled down into 4 basic points. 1) The random variables, 2) the PDFs, 3) the moments of the PDFs and 4) the pertinent variance σ^2.

The moments are the "expectation values" of the n-th power of the value at hand,

$$\langle x^n \rangle \equiv \int x^n p(x) dx.$$

the 0th moment gives the normalization requirement for the PDF, the 1st moment is the expectation value μ. The $p(x)$ here is the PDF.

There are also central moments,

$$\langle (x - \langle x \rangle)^n \rangle \equiv \int (x - \langle x \rangle)^n p(x) dx.$$

The 0th and 1st moments are trivial, while the 2nd central moment gives the variance σ^2. Expanding the left hand side of the equation above gives the simplified $\langle x^2 \rangle - \langle x \rangle^2$.

Discretizing the momenta we can find that the variance can, through the central limit theorem, be written as

$$\sigma_N^2 \sim \frac{1}{N}(\langle f^2 \rangle - \langle f \rangle^2) = \frac{\sigma_f^2}{N}.$$

And we can see that the standard deviation $\sigma \sim \frac{1}{\sqrt{N}}$. The aim of Monte Carlo methods is to get the standard deviation as small as possible for a given number of measurements N. Each sample then represents one measurement. The accuracy gain here is less than more traditional methods of integrations, e.g. Simpsons, but it gets more efficient at higher dimensions. So long as the number of dimensions of the integral are greater than twice the degree of power of the truncation error for the method in question, since Monte Carlo always scales according to $\sqrt{N}$ regardless of the number of dimensions.

Essentially, a Monte Carlo method contains

- PDF characterizing the system
- RNG $\in [0, 1]$
- sampling rule
- error estimation
- technique to improve the error

Some improvement to the brute force method comes in the mapping of the RNG as well as the choice in PDF $p(x)$ so that the cumulative probability mainly samples the relevant areas of the system so that we waste fewer cycles sampling. [3]

The sampling rule is an important part of the Monte Carlo simulation and amongst the popular rules is the Metropolis algorithm.

C. Metropolis Algorithm

All Monte Carlo schema are based on Markov chains. Random walks with a select probability for each move. Each move is independent of the history of the system. The reason behind this is that given enough time, the Markov chain leads to the most likely state for the system starting from a random point. It moves towards equilibrium. In essence, the Markov process is a discretized diffusion equation,

$$\frac{\partial p(x,t)}{\partial t} = D\frac{\partial^2 p(x,t)}{\partial x^2}.$$

D here being the Diffusion constant. The diffusion function has, given the constraints of a PDF a stable mean in time, but a varying variance.

$$\frac{\partial \langle x \rangle}{\partial t} = 0,$$
$$\frac{\partial \langle x^2 \rangle}{\partial t} = ... = 2D \rightarrow \langle x^2 \rangle = 2Dt,$$
$$\sigma^2 = \langle x^2 \rangle - \langle x \rangle^2 = 2Dt.$$

Which leads to an Root Mean Square(RMS) displacement after a time t,

$$\sqrt{2Dt} \propto \sqrt{t}.$$

In essence, the probability of being in a position i after one timestep from $t = 0$

$$p_i(t = \epsilon) = P(j \rightarrow i)p_j(t = 0).$$

$P(j \rightarrow i) = P_{ij}$ is the transition probability from state j to state i.

Typically, the transition probability is not known. The Metropolis algorithm solves this by modeling it as a product of 2 possibilities, $A(j \rightarrow i)$ and $T(j \rightarrow i)$. These are the probability for accepting a move from j to i and the probability for making the transition to i, being in position j. A for acceptance and T for transition. The total transition probability is then

$$P(j \rightarrow i) = A(j \rightarrow i)T(j \rightarrow i).$$

We can express the Metropolis algorithm as

- Suggest a move to new state, i, with some moving probability $T_{j\rightarrow i}$
- Probability $A_{j\rightarrow i}$ of accepting the move or rejecting with $(1 - A_{j\rightarrow i})$ probability, sampling instead with the current position j.

Using the base shape of the algorithm, we can express the probability of being in a position i at the next instance could be expressed,

$$p_i(t+1) = \sum_j [p_j(t)T_{j\rightarrow i}A_{j\rightarrow i} + p_i(t)T_{i\rightarrow j}(1 - A_{i\rightarrow j})]$$
$$p_i(t+1) - p_i(t) = \sum_j [p_j(t)T_{j\rightarrow i}A_{j\rightarrow i} - p_i(t)T_{i\rightarrow j}A_{i\rightarrow j}].$$

The last change is based on the normalization requirement for probabilities, $\sum_j T_{j\rightarrow j}$, and is similar to the Master equation relating temporal dependence of a PDF with transition rates. Calling back to the diffusion function, when $t \rightarrow \inf$, the change in state over time should go to 0. To ensure that we reach equilibrium over time and not a cyclical change in state, we invoke detailed balance

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

This helps ensure correct distribution appears at equilibrium, e.g. the Boltzmann distribution.

$$p_i = \frac{\exp(-\beta E_i)}{Z}$$
$$Z = \sum_j \exp(-\beta E_j)$$
$$\beta = \frac{1}{k_B T}$$

β being the inverse temperature of the system, the Z is a normalizing constant to ensure the overall probability is 1. Z is difficult to impossible to know in most systems. Using the condition for detailed balance we can remove this constant.

$$\frac{p_i}{p_j} = \exp(\beta(E_i - E_j))$$

With this, we can begin to see a basic version of the Metropolis algorithm. Due to the difficulty in knowing exactly both the normalization factor and the transition probability, we can write the detailed balance in as

$$\frac{A_{j\rightarrow i}}{A_{i\rightarrow j}} = \frac{p_i T_{i\rightarrow j}}{p_j T_{j\rightarrow i}} = \exp(-\beta(E_i - E_j)).$$

This is under the assumption, for the brute force Metropolis, that the transition probability is symmetric $T_{j\rightarrow i} = T_{i\rightarrow j}$.

We are here principally interested in a new state $E_i < E_j$. While a move to a lower energy should be accepted, only accepting moves that lower energy would severely bias our averages. There are changes that would occur in a system that would not directly lead to an equilibrium which would, nonetheless have a chance to occur. Therefore, we can set $A_{j\rightarrow i} = 1$. This gives the other acceptance probability as

$$A_{i\rightarrow j} = \exp(-\beta(E_j - E_i)).$$

Note the change in subscript j and i.

A way to encode this, is as follows

$$A(j \rightarrow i) = \begin{cases} \exp(-\beta(E_i - E_j)) & E_i - E_j > 0 \\ 1 & else \end{cases}.$$

This leads us to the basic, brute force Metropolis algorithm, assuming an initial position.

- have a initial value $x^{(i)}$
- generate a trial value y with some probability $T(x^{(i)} \rightarrow y)$
- check the new value

$$x^{(i+1)} = \begin{cases} y & A(x^{(i)} \rightarrow y) \\ x^{(i)} & (1 - A(x^{(i)} \rightarrow y)) \end{cases}$$

- given symmetric $T(x^{(i)} \rightarrow y)$, defined

$$A(x \rightarrow y) = \min(\frac{p(y)}{p(x)}, 1)$$

For this project, we assume the symmetry of transition probability.

D. Ground State and Local Energy

With all those tools in place we can finally find the ground state energy. The local energy in a MC cycle is defined as

$$E_L(\mathbf{r}) = \frac{1}{\Psi_T(\mathbf{r})} H\Psi_T(\mathbf{r}). \tag{11}$$

As you do more cycles the average of the local energy approaches the ground state energy,

$$E = \sum_i E_{L,i}(\mathbf{r}). \tag{12}$$

What we have so far will be pretty descent estimate of the the ground state energy, but it can be better. The simplest method is known as equilibration, and means we let a portion of the MC cycles run before we start calculating the average. Because the statistical models we have generally rely on a system being in or close to equilibrium, letting the system equilibrate for some time before making any measurements would improve the accuracy of the results. In addition, we will use two more methods that improves our estimation, as well as discuss how we are going calculate the error.

E. Importance Sampling

Importance sampling involves using the Fokker-Planck and the Langevin equations to improve the Metropolis algorithm. First lets look at the Langevin equation.

1. Langevin Equation

The Langevin equation is used to calculate the new coordinates of the particles. It reads

$$\frac{\partial x(t)}{\partial t} = DF(x(t)) + \eta. \tag{13}$$

Here η is a random variable, F is a drift term and D is the diffusion coefficient. When using atomic units we have $D = 1/2$. Using Euler's method, this can be recast into

$$y = x + DF(x)\Delta t + \xi\sqrt{\Delta t}. \tag{14}$$

Here x is the old position, y is the new position, ξ is a Gaussian random variable and Δt is a chosen time step. Usually we set $\Delta t \in [0.001, 0.01]$ as it generally yield stable values of the ground state energy.

2. Fokker-Planck

The Fokker-Planck equation gives us a formula for the drift term which we can then use to solve Green's function, giving us a new transition probability.

Fokker-Planck for a time-dependent probability density $P(\mathbf{x}, t)$ reads:

$$\frac{\partial P}{\partial t} = \sum_i D\frac{\partial}{\partial \mathbf{x_i}}\left(\frac{\partial}{\partial \mathbf{x_i}} - \mathbf{F_i}\right)P(\mathbf{x}, t). \tag{15}$$

Since the probability density converges we can set the left side to zero. From this the drift term $\mathbf{F}$ can be derived,

$$\mathbf{F} = 2\frac{1}{\Psi_T}\nabla\Psi_T, \tag{16}$$

where Ψ_T is the wave function. This formula is known as the quantum force.

This drift term can then be used to solve Green's function:

$$G(y, x, \Delta t) = \tag{17}$$

$$\frac{1}{(4\pi D\Delta t)^{3N/2}} \exp\left(-(y - x - D\Delta t F(x))^2/4D\Delta t\right), \tag{18}$$

where N is the number of particles, and the other variables are the same as for the Langevin equation. We can then replace our transition probability with

$$A(x \to y) = \min\left(1, q(y, x)\right) \tag{19}$$

$$= \min\left(1, \frac{G(x, y, \Delta t)\,|\Psi_T(y)|^2}{G(y, x, \Delta t)\,|\Psi_T(x)|^2}\right), \tag{20}$$

which will hopefully mean we approach the ground state energy faster.

F. Steepest Descent

The Steepest Descent algorithm is used to optimize a variable with a few Monte Carlo cycles. This is done by guessing at the variable, doing some cycles with it, then updating the variable. Then repeat for a certain number of iterations. This relies on the idea that a function $F(\mathbf{x})$ will decrease fastest if $\mathbf{x}$ goes in the direction of the negative gradient. This can be represented mathematically as

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \eta_k \nabla F\left(\mathbf{x}_k\right) \tag{21}$$

where k represents each iteration.

η_k is called the learning rate. As long as it is small enough we will have $F\left(\mathbf{x}_{k+1}\right) \leq F\left(\mathbf{x}_k\right)$, meaning the scheme always moves towards smaller values, or a minimum. We thus need to choose a learning rate that is large enough for the system to converge in a reasonable time frame, but not so large that it produces erratic behaviour.

Another potential problem with steepest descent is that we can't be sure whether the value we are approaching is a local or a global minimum. There are other gradient descent methods that try to alleviate both these problems, however for the problem we are studying here steepest descent will work well enough.

The variable we want to optimize is α, and the function we want to optimize α for is the ground state energy. This means we can write

$$\alpha_{k+1} = \alpha_k - \eta\frac{\partial}{\partial\alpha}E\left(\alpha_k; \mathbf{r}\right). \tag{22}$$

$\frac{\partial E}{\partial \alpha}$ can be written as

$$\frac{\partial E}{\partial \alpha} = \frac{2}{n}\left(\sum_i^n E_{L,i}\nabla\Psi_i - \sum_i^n E_{L,i}\sum_i^n \nabla\Psi_i\right) \tag{23}$$

were n is the number of MC cycles $E_{L,i}$ is the local energy at cycle i, and $\nabla\Psi_i$ is the derivative of the wave function at cycle i.

The algorithm thus looks like:

- Initialise α_0 as a guess for α
- Then at each iteration:
  - Run n MC cycles with α_i
  - At each cycle save $E_{L,i}$ and $\nabla\Psi_i$
  - Calculate the change in α, $d\alpha_i = \frac{2}{n}\left(\sum_i^n E_{L,i}\nabla\Psi_i - \sum_i^n E_{L,i}\sum_i^n \nabla\Psi_i\right)$
  - Update α, $\alpha_{i+1} = \alpha_i - \eta d\alpha_i$
- Repeat until either a set number of iterations have been done, or $d\alpha_i$ is sufficiently small

G. Resampling Methods

Resampling methods are a measure to increase confidence in the statistics of a set of measurements. The core concept is, as the name suggests, redrawing from the current data set. The base assumption here is that the data points are in essence independent and identically distributed (iid). If this is the case, several sets drawn from the original data set will have low to no correlation and can be used to gain a greater confidence in sample means and to reduce the statistical standard error.

That the sample is iid is important in that we want to minimize the correlation between samples. The correlation can be modeled as partial sums,

$$f_d = \frac{1}{n-d}\sum_{k=1}^{n-d}(x_k - \bar{x}_n)(x_{k+d} - \bar{x}_n).$$

Which in turn lets us define the correlation term of the error function as

$$\frac{2}{n}\sum_{k<l}(x_k - \bar{x}_n)(x_l - \bar{x}_n) = 2\sum_{d=1}^{n-1} f_d$$

where f_d reflects the correlation between measurements separated by the distance d between them. When $d = 0$ f gives the sample variance. From here,

$$\kappa_d = \frac{f_d}{var(x)}$$

is the autocorrelation function. This is a usefull measure of pairwise correlation starting at 1 for sampling 2 of the same variable.

With this, the sample error can be expressed in terms of the autocorrelation function,

$$\begin{aligned} err_X^2 &= \frac{1}{n}var(x) + \frac{2}{n}var(x)\sum_{d=1}^{n-1}\kappa_d \\ &= \left(1 + 2\sum_{d=1}^{n-1}\kappa_d\right)\frac{1}{n}var(x) \\ &= \frac{\tau}{n}var(x). \end{aligned}$$

With the *autocorrelation time* τ:

$$\tau = 1 + \sum_{d=1}^{n-1}\kappa_d.$$

A perfectly uncorrelated set of measurements has a $\tau = 1$. For more correlated measurements, the autocorrelation time reduces the effective sample size,

$$n_{eff} = \frac{n}{\tau}.$$

Not accounting for the effective reduction will cause the estimated error to also be less than the approximation

$$err_X^2 \approx \frac{var(x)}{n}.$$

A small enough correlation time corresponds to a good approximation of the error.

Resampling methods require fitting the same statistical methods repeatedly on different subsets of the original data and are therefore computationally taxing. Current computational power mitigate this, so as to not make it prohibitive.

The simulations performed are treated as computer experiments. This is especially so for Monte Carlo methods. This allows us to use the same methods that would otherwise be used for analysing other experimental data. While systematical errors need to be confronted on a case by case basis, the statistical errors of the simulations can be further analyzed and reduced with resampling.

There are several resampling methods available. Examples include Jackknife, cross-validation, bootstrap and blocking. For this project we use the bootstrap and blocking methods.

1. *Bootstrap*

The Bootstrap method is non-parametric. It is general, though there are edge cases where it fails. An advantage is that the method does not require distributional assumptions, such as normally distributed errors. For smaller datasets and poorly behaved data, the Bootstrap can provide good results. The Bootstrap method can work on statistics with hard to derive sampling distributions. Bootstrap is relatively simple to apply for difficult data-collection plans.

For a given data set $\mathbf{x} = (x_1, x_2, ..., x_m)$ the bootstrap in essence,

- iterate k times:

– Draw, with replacement, n samples $\mathbf{x}^* = (x_1^*, x_2^*, ..., x_n^*)$ from the original set $\mathbf{x}$.

– compute the estimator $\hat{\theta}^*$ with $\mathbf{x}^*$.

With the $\hat{\theta}^*$ we can now plot a histogram of the estimator and statistical values such as the variance $\hat{\sigma}^2$

2. Blocking

The Blocking method was popularized by Flyvberg and Pedersen [1] in 1989, with some further work by Marius Jonsson [4] in 2018. The method is a staple for measuring the sample mean $\bar{X}$.

The initial assumption is that $n = 2^d$. for some integer $d > 1$. Further, we assume $X_1, X_2, ..., X_n$ is a stationary time series. The set is assumed to be asymptotically uncorrelated. In vector notation, we get

$$\vec{X} = (X_1, X_2, ..., X_n).$$

The main strength of the blocking method is that it does not scale terribly with growing n, as well as growing more accurate with greater set sizes.

The blocking transformations are done by taking the pairwise mean of neighbouring elements in $\vec{X}$ and forming a new vector $\vec{X_1}$. Then find the pairwise mean of $\vec{X_1}$ to form $\vec{X_2}$ and so on and so forth. Recursively define $\vec{X_i}$ so,

$$(\vec{X_0})_k \equiv (\vec{X})_k.$$
$$(\vec{X}_{i+1})_k \equiv \frac{1}{2}((\vec{X_i})_{2k-1} + (\vec{X_i})_{2k}) \forall i \in [1, d-1].$$

$\vec{X}_k$ is subject to k blocking transformations. We then have d vectors with subsequent averages of measurements. If $\vec{X}$ is a stationary time series, then $\vec{X_i}$ is as well, $\forall i \in [0, d-1]$.

For each i we can compute autocovarriance, variance, sample mean $(\gamma_i, \sigma_i^2, \hat{X}_i)$ and number of observations, n_i. From the definition of $\vec{X_i}$, we can see that

$$n_i = \frac{n}{2^i}.$$

[2]

With $h = |i - j|$ we can define

$$\begin{aligned}\gamma_{k+1}(h) &= cov((X_{k+1})_i, (X_{k+1})_j) \\ &= \frac{1}{4} cov((X_k)_{2i-1} + (X_k)_{2i}, (X_k)_{2j-1} - (X_k)_{2j} \\ &= \begin{cases} \frac{1}{2}(\gamma_k(2h) + \gamma_k(2h+1)) & if h = 0 \\ \frac{1}{4}(\gamma_k(2h-1) + \gamma_k(2h+1)) + \frac{1}{2}\gamma_k(2h) & else \end{cases} [4]\end{aligned}$$

We can also derive the variance of the sample mean from the autocovariance function,

$$\begin{aligned}var(\bar{X}) &= \frac{1}{n^2} cov\left[\sum_{i=1}^{n} X_i, \sum_{j=1}^{n} X_j\right] \\ &= \frac{1}{n^2}[n\gamma(0) + (n-1)\gamma(1) + ... + \gamma(n-1)] \\ &+ (n-1)\gamma(-1) + (n-2)\gamma(-2) + ... + \gamma(1-n)] \\ &= \frac{\sigma^2}{n} + \frac{2}{n}\sum_{h=1}^{n-1}\left(1 - \frac{h}{n}\right)\gamma(h) \;\; if \gamma(0) = \sigma^2. [4]\end{aligned}$$

We can add the truncation error e_k,

$$e_k \equiv \frac{2}{n_k}\sum_{h=1}^{n_k-1}\left(1 - \frac{h}{n_k}\right)\gamma_k(h).[2]$$

for $i, j \in [0, d-1]$, $var(\bar{X}_i) = var(\bar{X}_j)$:

$$\begin{aligned}n_{j+1}\bar{X}_{n+1} &= \sum_{i=1}^{n_{j+1}} (\hat{X}_{j+1})_i = \frac{1}{2}\sum_{i=1}^{n_j/2} (\hat{X}_j)_{2i-1} + (\hat{X}_j)_{2i} \\ &= \frac{1}{2}\Big[(\hat{X}_j)_i + (\hat{X}_j)_2 + \cdots + (\hat{X}_j)_{n_j}\Big] \\ &= \frac{n_j}{2}\bar{X}_j = n_{j+1}\bar{X}_j.\end{aligned}$$

We can repeat this until we get

$$var(\bar{X}_i) = var(\bar{X}_0) = var(\bar{X}) \;\; \forall i \in [0, d-1].$$

Since $var(\bar{X}) = \frac{\sigma_k^2}{n_k} + e_k$ for all $k \in [0, d-1]$, and as the autocorrelation function goes to 0, so too does e_k. With this, increasing k enough sends e_k negligible. Then the estimation of the variance of the sample mean with a very close approximation

$$var(\bar{X}) = \frac{\sigma_k^2}{n_k}.[4]$$

III. METHOD

Implementation was done in `c++`. Our code was based upon the skeleton code created by by Morten Ledum [5]. We will assume the reader is at least somewhat familiar the structure of this code.

The program bases on the main program to launch the system object and populate the relevant parts of the system as needed. The system holds the objects for the Wavefunction, the Hamiltonian and the sampler classes mainly. The Monte Carlo simulation is performed in the system class, calling on the Hamiltonian and Wavefunction classes as needed for probability and energy estimations. The sampler class logs, prints and saves the data.

A. Basic Monte Carlo

The skeleton class referenced has a random class, which takes care of the initialization and safeties regarding the random number generator (RNG). The generation speed is adequate for this. The random function provides:

- uniform distribution $\in [0, 1]$
- negligible correlation between samples
- sufficiently large period between repeating sequence
- speed of algorithm.

The main Monte Carlo simulation involves repeated runs of the Metropolis sampling rule to test a random-length step in a given direction in the Harmonic Oscillator potential and checking the new to the old wavefunction to ascertain the probability of accepting the move through the metropolis sampling rule.

```
• store $\Psi_{old} = \Psi(\vec{r}_0)$
• for $N_{MC}$ do:
    – generate a random motion $\vec{d_{rand}} = \Delta s \cdot rand$¸ $rand \in [-0.5, 0.5]$.
    – evaluate $\Psi_{new} = \Psi_T(\vec{r} + \vec{d_{rand}})$
    – generate a random value $rand \in [0, 1]$
    – check if: $\frac{\Psi_{new}^2}{\Psi_{old}} \geq rand$
        ∗ set $\Psi_{old} = \Psi_{new}$
        ∗ accept step
    – else
        ∗ keep $\Psi_{old}$.
        ∗ reject step
• calculate energies and statistics
```

We tested the program for:

- dimensions $D = 1, 2, 3$
- particles $P = 1, 10, 100, 500$
- $\alpha = 0.38, 0.42, 0.46, 0.5, 0.54, 0.58, 0.62$
- Frequency $\omega = 1$
- MC cycles $MC = 10^4$
- relative step-lenght = 2
- equilibration fraction: 0.1

B. Importance Sampling

Implementation of importance sampling was done by copying the code for for the Metropolis step, and then changing the lines for change in position and for transition likelihood. We tested three values of Δt, $[0.001, 0.005, 0.01]$ and observed how they affected the result for $n = 1, 10, 100$ particles, $d = 1, 2, 3$ dimensions and $\alpha = 0.46$.

For here on out all calculations were done using IS.

C. Steepest Descent

To calculate the steepest descent we added a clause to the sampler where we made sure that it saved the cumulative value of $\frac{dE}{d\alpha}$ and $E\frac{dE}{d\alpha}$ over all the MC cycles, in addition to the energy. We could then use this to find the change in α, and then repeat. This continues until either 1000 iterations have been reached, or the change in α is smaller than 10^{-6}.

We tested this with $n = 1, 10, 100$ particles, $d = 1, 2, 3$ dimensions, $\Delta t = 0.001$ and the learning rate $\eta = 0.001$. Each SD-iteration used 1000 MC cycles each, and our initial guess of α was 0.45.

D. Resampling

1. *Bootstrap*

For the bootstrap code, We used a basic program, which goes through the given data N times and picks, with replacement k samples to create a new set $\vec{X}^*$ which is then stored as part of a list. This list is then used to check the bias and standard error of the original data. The program also produces a histogram of the "new" sample means generated. According to the theory, this should produce a normal distribution about the "best fit" mean.

The data set used was the `energies_1c.csv` and `energies_1d.csv`

2. *Blocking*

This code was borrowed from the master thesis about the automated blocking method by M. Jonsson [4], and worked with little to no modification.

The data set used was the `energies_1c.csv` and `energies_1d.csv`

E. Parallelizing

To decrease run time we parallelised part of the code using OpenMP. This was done by having each system run in parallel. This means we aren't speeding up individual calculations, but instead running multiple at the same time.

F. Elliptic Trap

To change from a spherical to an elliptic trap we added two arrays, one each for γ and β, with all values set to 1. If a γ and β are provided the last element of the arrays are then set to be equal to the inputted values. When calculating g or the Hamiltonian and are looping over the different dimensions we multiply with the equivalent values in the new arrays.

G. The Repulsive Interaction

For the Hamiltonian we can see that all the repulsive interaction does is make sure particles aren't too close. If their distance is grater than a it has no effect on the energy. All we need to do then is to add a clause to the Metropolis and importance sampling so that we reject any move that would set any particles too close. In addition to this, since we only moving one particle at a time, we only need to check the current particle against all the others, instead of testing all particle pairs.

For the wave function the repulsive interaction is a bit more involved. Here we also have to calculate $f\left(a, |\mathbf{r}_i - \mathbf{r}_j|\right)$ for all the particle pairs as well. However, since we are already testing that $|\mathbf{r}_i - \mathbf{r}_j| > a$ we don't need to worry about that here, and can instead simply assume they are far enough apart.

For testing we set $\beta = \gamma = 2.82843$, and ran the simulation for $n = 2, 10, 50$ particles in three dimensions. The initial guess of α was 0.71, with $\eta = 0.001$, and 100 MC cycles for each SD-iteration. We kept the rest of the values the same. We also wanted to run the program for 100 particles, but the runtime was too long, so decided against it.

At this point we had some problems with OpenMP, so we turned it off. We believe it might be caused by incompatibility between the OpenMP software and our random number generator.

H. Onebody Densities

We ran the simulation again with the found α from before, but without the repulsive interaction, to compare the result.

IV. RESULTS

A. Basic Monte Carlo

P	D	Accepted Steps	Energy	Time	α
1	1	0.705412	0.507809	0.0249315	0.62
1	1	0.761196	0.523561	0.0259287	0.38
1	1	0.71119	0.503259	0.0259287	0.58
1	1	0.73986	0.50353	0.0269264	0.46
1	1	0.721191	0.501081	0.0259287	0.54
1	1	0.749972	0.510093	0.0269264	0.42
1	1	0.730748	0.5	0.0259287	0.5
1	2	0.625847	1.01544	0.0388944	0.42
1	2	0.584732	1.00222	0.0418869	0.54
1	2	0.596288	1	0.0398938	0.5
1	2	0.610512	1.00176	0.0418869	0.46
1	2	0.645183	1.03842	0.0468734	0.38
1	2	0.569841	1.01033	0.0299199	0.58
1	2	0.555728	1.01859	0.0359044	0.62
1	3	0.562396	1.5761	0.0458771	0.38
1	3	0.524503	1.50447	0.0468748	0.46
1	3	0.545727	1.52329	0.0478715	0.42
1	3	0.504945	1.5	0.0508637	0.5
1	3	0.492166	1.5014	0.0488693	0.54
1	3	0.472164	1.51243	0.0528582	0.58
1	3	0.464829	1.53833	0.0578453	0.62
10	1	0.762696	5.17907	1.37134	0.38
10	1	0.751417	5.0707	1.37931	0.42
10	1	0.73037	5	1.37632	0.5
10	1	0.720191	5.01932	1.3833	0.54
10	1	0.701022	5.13351	1.38031	0.62
10	1	0.739471	5.01208	1.41422	0.46
10	1	0.709768	5.06957	1.42818	0.58
10	2	0.642094	10.3984	2.49433	0.38
10	2	0.608345	10.0321	2.52924	0.46
10	2	0.62567	10.1578	2.57612	0.42
10	2	0.578675	10.0347	2.55816	0.54
10	2	0.593688	10	2.57611	0.5
10	2	0.568608	10.1081	2.78057	0.58
10	2	0.555428	10.2471	2.82844	0.62
10	3	0.50819	15	3.99034	0.5
10	3	0.522314	15.061	4.04918	0.46
10	3	0.560451	15.6215	4.08011	0.38
10	3	0.541582	15.2386	4.18282	0.42
10	3	0.489654	15.04	4.19579	0.54
10	3	0.473619	15.1577	4.03322	0.58
10	3	0.462618	15.3044	4.01328	0.62

TABLE I. Part one of the table detailing the results from the basic brute force Monte Carlo simulation. P is the number of particles, D is the number of dimensions, time is the runtime for that simulation measured in seconds

P	D	Accepted Steps	Energy	Time	α
100	1	0.749874	50.7927	160.467	0.42
100	1	0.728852	50	160.85	0.5
100	1	0.719147	50.151	161.67	0.54
100	1	0.70952	50.5625	160.98	0.58
100	1	0.761406	51.9485	166.104	0.38
100	1	0.738854	50.1854	167.607	0.46
100	1	0.700858	51.1716	168.5	0.62
100	2	0.642579	103.762	350.253	0.38
100	2	0.581634	100.311	350.73	0.54
100	2	0.595603	100	351.64	0.5
100	2	0.610391	100.346	361.496	0.46
100	2	0.62647	101.473	361.945	0.42
100	2	0.569073	101.106	374.239	0.58
100	2	0.556497	102.409	376.473	0.62
100	3	0.560089	155.547	550.086	0.38
100	3	0.540869	152.231	550.064	0.42
100	3	0.50605	150	552.567	0.5
100	3	0.523515	150.491	558.288	0.46
100	3	0.489523	150.489	559.438	0.54
100	3	0.461243	153.54	544.852	0.62
100	3	0.47433	151.675	546.301	0.58
500	1	0.761722	259.418	4559.07	0.38
500	1	0.750238	253.776	4634.83	0.42
500	1	0.739329	250.854	4653.13	0.46
500	1	0.729198	250	4569.79	0.5
500	1	0.719586	250.742	4576.5	0.54
500	1	0.710347	252.747	4562.62	0.58
500	1	0.701429	255.854	4563.7	0.62
500	2	0.64232	519.075	9431.59	0.38
500	2	0.625922	507.663	9486.38	0.42
500	2	0.610461	501.825	9486.86	0.46
500	2	0.595601	500	9341.03	0.5
500	2	0.582079	501.421	9344.81	0.54
500	2	0.569041	505.537	8484.22	0.58
500	2	0.556497	511.699	8549.47	0.62
500	3	0.553451	771.735	11285.3	0.38
500	3	0.53396	757.106	11232.7	0.42
500	3	0.515768	750.56	11239.8	0.46
500	3	0.482569	754.111	11092	0.54
500	3	0.498724	750	11170	0.5
500	3	0.453207	772.671	8116.41	0.62
500	3	0.467413	761.96	8178.95	0.58

TABLE II. Part two of the table detailing the results from the basic brute force Monte Carlo simulation. P is the number of particles, D is the number of dimensions, time is the runtime for that simulation measured in seconds

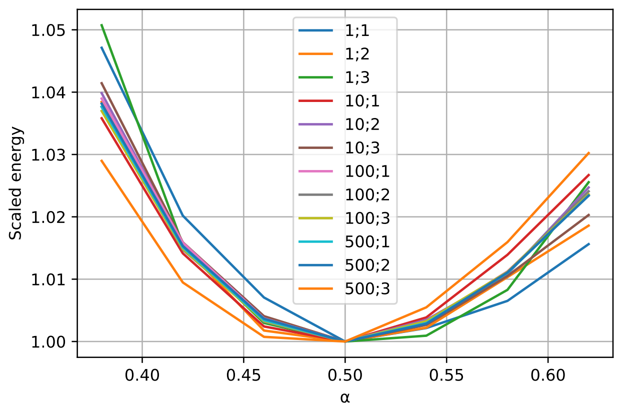

FIG. 1. A plot of the ground state energy against α. The energy has been scaled by the optimal energy for that combination of dimension and energy.

B. Importance Sampling

P	D	Accepted Steps	Energy	Time	Δt
1	1	0.73986	0.50353	0.0229235	0.001
1	1	0.73986	0.50353	0.0279103	0.01
1	1	0.73986	0.50353	0.0269131	0.005
1	2	0.610512	1.00176	0.0378836	0.01
1	2	0.610512	1.00176	0.0388809	0.005
1	2	0.610512	1.00176	0.0418729	0.001
1	3	0.524503	1.50447	0.0488544	0.001
1	3	0.524503	1.50447	0.0488544	0.005
1	3	0.524503	1.50447	0.0508493	0.01
10	1	0.739471	5.01208	1.34439	0.005
10	2	0.608345	10.0321	2.52026	0.001
10	3	0.522314	15.061	3.8936	0.005
10	1	0.739471	5.01208	4.07011	0.001
10	1	0.739471	5.01208	4.30348	0.01
10	2	0.608345	10.0321	8.62296	0.01
10	2	0.608345	10.0321	8.67981	0.005
10	3	0.522314	15.061	11.0684	0.01
10	3	0.522314	15.061	11.0983	0.001
100	1	0.738854	50.1854	141.176	0.005
100	1	0.738854	50.1854	142.308	0.001
100	1	0.738854	50.1854	142.908	0.01
100	2	0.610391	100.346	254.729	0.005
100	2	0.610391	100.346	254.344	0.01
100	2	0.610391	100.346	257.337	0.001
100	3	0.523515	150.491	339.822	0.005
100	3	0.523515	150.491	340.925	0.001
100	3	0.523515	150.491	337.841	0.01

TABLE III. Table detailing the results from the Importance Sampling simulation. P is the number of particles, D is the number of dimensions, time is the runtime for that simulation measured in seconds, and Δt is the variable used by IS.

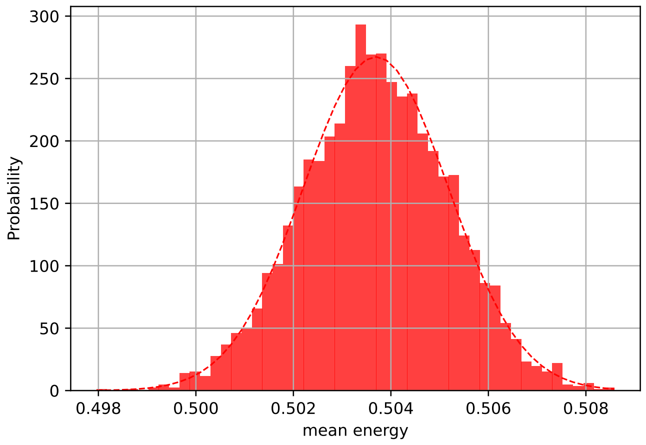

FIG. 2. Histogram of the Bootstrap sample means for 1 particle and 1 dimension in onebody problem with importance sampling, but without a gradient descent functionality. the dotted line represents the best fit of the histogram. The shape is ilustrative of the distribution for other dimensions and particles. the best fit sample mean is at about 0.50005. with a Full Width at Half Maximum of about 0.0039. this leads to a std. deviation of about $\sigma \approx FWHM/2.3 = 0.00169$

C. Steepest Descent

P	D	Accepted Steps	Energy	Time	Found α
1	1	0.731303	0.500049	0.0199464	0.498652
1	2	0.595399	1.00001	0.0239036	0.499533
1	3	0.506167	1.50003	0.053891	0.499643
10	1	0.730003	4.99999	0.85771	0.499896
10	2	0.596266	10.001	1.78922	0.493553
10	3	0.50819	15	2.52725	0.499991
100	1	0.728853	50	75.7935	0.499994
100	2	0.595576	100	147.876	0.500002
100	3	0.50605	150	218.301	0.500002

TABLE IV. Table detailing the results from the simulation with steepest descent. P is the number of particles, D is the number of dimensions, time is the runtime for that simulation measured in seconds, found α is the found best value for α

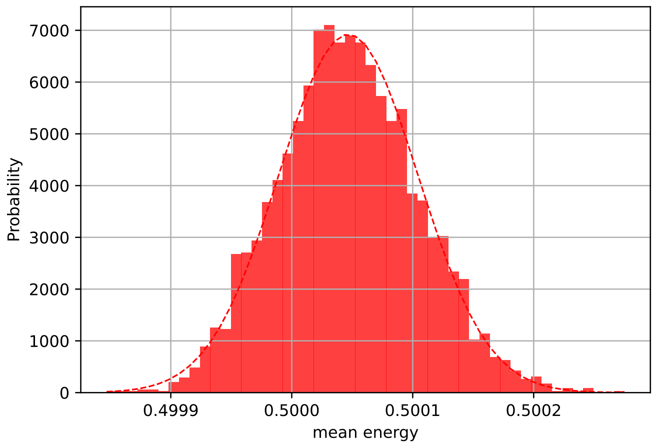

FIG. 3. Histogram of the Bootstrap sample means for 1 particle and 1 dimension in onebody problem with importance sampling, with a gradient descent functionality. the dotted line represents the best fit of the histogram. The peak is around 0.50005 unit-less energy. with a Full Width at Half Maximum of about 0.00013. this leads to a std. deviation of about $\sigma \approx FWHM/2.3 = 0.000056$

D. Repulsive Interaction

P	D	Accepted Steps	Energy	SD Time	Time	Found α
2	3	0.966607	7.44574	3.53161	0.341089	0.780745
10	3	0.608212	34.531	6.60486	95.2771	0.710547
50	3	0.315717	161.824	7361.26	4393.68	0.801405

TABLE V. Table detailing the results from the simulation when the repulsive interaction and elliptical trap. P is the number of particles, D is the number of dimensions, SD time is the runtime for the steepest descent simulation measured in seconds, time is the runtime for that simulation measured in seconds, found α is the found best value for α

E. Onebody Densities

For onebody densities, we ran without the interacting elements to compare to the repulsive interaction.

P	D	Accepted Steps	Energy	Time	α
2	3	0.970941	7.4642	0.16755	0.780745
10	3	0.692499	34.5701	2.96111	0.710547
50	3	0.103483	175.179	66.1967	0.801405

TABLE VI. Table detailing the results from the simulation with the found best α for the repulsive interaction, but when only finding the onebody denseties. P is the number of particles, D is the number of dimensions, time is the runtime for that simulation measured in seconds, found α is the found best value for α

V. DISCUSSION

A. Basic Monte Carlo

Looking at the graph we see the behaviour one would expect. All the simulations find the lowest energy at $\alpha = 0.5$, and the graph the graph grows quadratically as α moves away form 0.5. Interestingly enough we see that the graph grows faster for smaller values of α than for larger ones.

Looking at the tables we see that the energy is in the area around 0.5 times the number of particles times the number of dimensions, and exactly that when $\alpha = 0.5$. This is also the analytical values, so that is a good sign.

We see that the runtime increases exponentially as model complexity increases, as one would expect. We also see that number of accepted steps is more dependent on the dimension than number of particles. We're not quite sure why we get this behaviour. We see this same behaviour throughout the rest of the results as well.

B. Importance Sampling

We can see that the found energy is about the same as with the brute force MC simulation. The same goes for number of accepted steps. The runtime is about the same with 1 particle, longer with 10 particles, and shorter with 100 particles.

The choice of Δt appears to have no impact on the result. For some of the simulations the runtime differs notably, but it does not appear to consistent for which Δt this happens. This is likely caused by a sudden spike in processor usage at the time of the simulation.

C. Steepest Descent

We see that all the αs approach 0.5. This makes sense as this is the point where the energy is the ground state energy is the smallest. The runtime is shorter than for importance sampling. Intuitively one would think the opposite would happen. We believe this is caused by our implementation of the timing being slightly wrong, as it didn't measure the time it took for the steepest descent part, only the simulation at the end with the optimal α. The energy is about 0.5 times the number of particles times the number of dimensions, as one would expect.

D. Repulsive Interaction

We can see that the found α now hovers between about 0.71 and 0.8. The found energy is about 2.2-2.5 times larger than the non-interacting case with the optimal α. The runtime has also increased significantly.

E. Onebody Densities

We see that the found energy is about the same, or even slightly larger than when including the repulsive interaction. This suggest that most of the increase in energy we saw previously came from the inclusion of the elliptical trap. The repulsive interaction might even lower the total energy. This is contrary to what one would expect, and is likely caused by some bug.

VI. CONCLUSION

The brute force Monte Carlo with the naive one-body Hamiltonian and the symmetric HO-potential produces mostly what we would expect in regards to the energies. We also see a very similar proportional change in energy for various numbers of particles in 1, 2 and 3 dimensions. This is as expected for the simplest case. Because there are no interactions in this model, without random statistical errors and floating point round off, the different energy results should overlap when scaled based on the number of dimensions and the number of particles as compared to the 1 dimensional, 1 particle case.

When we implemented importance sampling there seems to be little difference from the brute force method, either in accepted steps, accuracy or calculation time. This would go against the intuition in that the proposed moves should be tailored to the negative gradient in a much larger degree, so as to waste fewer cycles on the uniform distribution of numbers.

The steepest descent addition speeds up the process somewhat, and we see that the α values converge towards what would be expected. The runtime of the program at this stage seems off, and would be a point to examine further. This might be a net fewer cycles running before an optimal α is found to be sent to a more thorough MC simulation of the preferred model values.

Adding in the repulsive interaction and the elliptical trap causes the ground state energy of the system to increase at a greater rate than without. This seems to be an intuitive result, given the increased space each particle now takes, the total energy for all the particles to fit needs to accommodate the repulsive forces. Thus, the energy increases.

Returning to the one-body problem, we see a greater or equal energy for the system as compared to the repulsive potential. This could be the difference between the circular and elliptical oscillator potential forming the trap. It could perhaps also be an issue with the simulation, weighing the trap over the repulsion. Further study of this phenomenon would be recommended.

Further study could also be done to determine the exact nature of the issue with OpenMP we encountered. Steps could also likely be taken to decrease runtime, allowing for larger sample sizes/a higher amount of particles to be run in a reasonable time frame.

Appendix A: Some math

1. Formulas

We have

$$H = \sum_i^N \left(\frac{-\hbar^2}{2m} \nabla_i^2 + V_{ext}\left(\mathbf{r}_i\right) \right) + \sum_{i<j}^N V_{int}\left(\mathbf{r}_i, \mathbf{r}_j\right), \tag{A1}$$

and

$$\Psi_T(\mathbf{r}) = \Psi_T\left(\mathbf{r}_1, \mathbf{r}_2, \ldots \mathbf{r}_N, \alpha, \beta\right) \tag{A2}$$

$$= \left[\prod_i g\left(\alpha, \beta, \mathbf{r}_i\right)\right] \left[\prod_{j<k} f\left(a, |\mathbf{r}_j - \mathbf{r}_k|\right)\right]. \tag{A3}$$

Here we have

$$g\left(\alpha, \beta, \mathbf{r}_i\right) = \exp\left[-\alpha\left(x_i^2 + y_i^2 + \beta z_i^2\right)\right], \tag{A4}$$

and

$$f\left(a, |\mathbf{r}_i - \mathbf{r}_j|\right) = \begin{cases} 0 & |\mathbf{r}_i - \mathbf{r}_j| \le a \\ \left(1 - \frac{a}{|\mathbf{r}_i - \mathbf{r}_j|}\right) & |\mathbf{r}_i - \mathbf{r}_j| > a \end{cases} \tag{A5}$$

2. Local energy

First we are to find the local energy

$$E_L(\mathbf{r}) = \frac{1}{\Psi_T(\mathbf{r})} H \Psi_T(\mathbf{r}) \tag{A6}$$

when $a = 0$, $\beta = 1$ and we discard totally the two-body potential. We can see that $a = 0 \Rightarrow f = 1$, and $g\left(\alpha, 0, \mathbf{r}_i\right) = \exp\left(-\alpha {r_i}^2\right)$. We thus have

$$\Psi_T(\mathbf{r}) = \prod_i e^{-\alpha {r_i}^2}. \tag{A7}$$

$$\begin{aligned} E_L(\mathbf{r}) &= \frac{1}{\Psi_T(\mathbf{r})} \left[\sum_i^N \left(\frac{-\hbar^2}{2m}\nabla_i^2 + V_{ext}\left(\mathbf{r}_i\right)\right)\right] \left[\prod_j e^{-\alpha {r_j}^2}\right] \\ &= \frac{1}{\Psi_T(\mathbf{r})} \left[\sum_i^N \left(\frac{-\hbar^2}{2m}\nabla_i^2 + \frac{1}{2} m {\omega_{ho}}^2 {r_i}^2\right)\right] \left[\prod_j e^{-\alpha {r_j}^2}\right] \\ &= \frac{1}{\Psi_T(\mathbf{r})} \sum_i^N \left(\frac{-\hbar^2}{2m}\nabla_i^2 \prod_j e^{-\alpha {r_j}^2} + \frac{1}{2} m {\omega_{ho}}^2 {r_i}^2 \prod_j e^{-\alpha {r_j}^2}\right) \\ &= \sum_i^N \left(\frac{1}{\Psi_T(\mathbf{r})} \frac{-\hbar^2}{2m} \left[\nabla_i^2 e^{-\alpha {r_i}^2}\right] \prod_{j\neq i} e^{-\alpha {r_j}^2} + \frac{1}{2} m {\omega_{ho}}^2 {r_i}^2\right) \end{aligned}$$

Here we can use that

$$\nabla_i^2 e^{-\alpha {r_i}^2} = 2\alpha e^{-\alpha {r_i}^2} \left(2\alpha {\mathbf{r}_i}^2 - d\right), \tag{A8}$$

where d is the number of dimensions.

$$\begin{aligned} E_L(\mathbf{r}) &= \sum_i^N \left(\frac{1}{\Psi_T(\mathbf{r})} \frac{-\hbar^2}{2m} \left[2\alpha e^{-\alpha {r_i}^2} \left(2\alpha {\mathbf{r}_i}^2 - d\right)\right] \prod_{j\neq i} e^{-\alpha {r_j}^2} \right. \\ &\qquad \left. + \frac{1}{2} m {\omega_{ho}}^2 {r_i}^2\right) \\ &= \sum_i^N \left(\frac{1}{\Psi_T(\mathbf{r})} \frac{-\hbar^2 \alpha \left(2\alpha {\mathbf{r}_i}^2 - d\right)}{m} \prod_j e^{-\alpha {r_j}^2} + \frac{1}{2} m {\omega_{ho}}^2 {r_i}^2\right) \\ &= \sum_i^N \left(\frac{-\hbar^2 \alpha}{m} \left(2\alpha {\mathbf{r}_i}^2 - d\right) + \frac{1}{2} m {\omega_{ho}}^2 {r_i}^2\right) \end{aligned}$$

Furthermore we have $\alpha = 1/2a_{ho}^2$, where $a_{ho} \equiv (\hbar/m\omega_{ho})^{\frac{1}{2}}$, giving $\alpha = \frac{m\omega_{ho}}{2\hbar}$. If we insert this we get

$$\begin{aligned} E_L(\mathbf{r}) &= \sum_i^N \left(\frac{-\hbar^2}{m} \frac{m\omega_{ho}}{2\hbar} \left(\frac{m\omega_{ho}}{\hbar} {\mathbf{r}_i}^2 - d\right) + \frac{1}{2} m {\omega_{ho}}^2 {r_i}^2\right) \\ &= \sum_i^N \left(\frac{-\hbar\omega_{ho}}{2} \left(\frac{m\omega_{ho}}{\hbar} {\mathbf{r}_i}^2 - d\right) + \frac{1}{2} m {\omega_{ho}}^2 {r_i}^2\right) \\ &= \sum_i^N \left(\frac{1}{2}\hbar\omega_{ho} d - \frac{m{\omega_{ho}}^2}{2} {\mathbf{r}_i}^2 + \frac{1}{2} m {\omega_{ho}}^2 {r_i}^2\right) \\ &= \sum_i^N \frac{1}{2}\hbar\omega_{ho} d = \underline{\underline{\frac{1}{2}\hbar\omega_{ho} d N}} \end{aligned}$$

3. Drift Force

$$F = \frac{2\nabla\Psi_T}{\Psi_T} = \frac{2\sum_i \nabla_i \Psi_T}{\Psi_T} \tag{A9}$$

$$\begin{aligned} \nabla_i \Psi_T &= \nabla_i \prod_j e^{-\alpha {r_j}^2} = \nabla_i \prod_j e^{-\alpha\left({x_j}^2 + {y_j}^2 + {z_j}^2\right)} \\ &= \left(-2\alpha x_i e^{-\alpha {r_i}^2}, -2\alpha y_i e^{-\alpha {r_i}^2}, -2\alpha z_i e^{-\alpha {r_i}^2}\right) \prod_{j\neq i} e^{-\alpha {r_j}^2} \\ &= -2\alpha\left(x_i, y_i, z_i\right) \prod_j e^{-\alpha {r_j}^2} = -2\alpha \mathbf{r}_i \Psi_T \end{aligned}$$

$$F_i = \frac{2\nabla_i \Psi_T}{\Psi_T} = \frac{-4\alpha \mathbf{r}_i \Psi_T}{\Psi_T} = -4\alpha \mathbf{r}_i \tag{A10}$$

$$F = \underline{\underline{-4\alpha\mathbf{r}}} \tag{A11}$$

4. First derivative of particle k

We want to show that

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

We have:

$$\begin{aligned}\nabla_k \Psi_T(\mathbf{r}) =&\nabla_k \left[\prod_{i} \phi\left(\mathbf{r}_i\right)\right] \exp\left(\sum_{j<m} u\left(r_{jm}\right)\right)\\ &= \left[\nabla_k \prod_{i} \phi\left(\mathbf{r}_i\right)\right] \exp\left(\sum_{j<m} u\left(r_{jm}\right)\right)\\ &+ \left[\prod_{i} \phi\left(\mathbf{r}_i\right)\right] \nabla_k \exp\left(\sum_{j<m} u\left(r_{jm}\right)\right)\end{aligned}$$

We can use:

$$\begin{aligned}&\nabla_k \exp\left(\sum_{j<m} u\left(r_{jm}\right)\right)\\ &= \exp\left(\sum_{j<m, j\neq k} u\left(r_{jm}\right)\right) \nabla_k \exp\left(\sum_{l\neq k} u\left(r_{jm}\right)\right)\\ &= \exp\left(\sum_{j<m} u\left(r_{jm}\right)\right) \sum_{l\neq k} \nabla_k u\left(r_{jm}\right)\end{aligned}$$

And:

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

Putting that together we get:

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

5. Second derivative of particle k

We want to show that

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

First:

$$\begin{aligned}\frac{1}{\Psi_T(\mathbf{r})}\nabla_k^2 \Psi_T(\mathbf{r}) = \frac{1}{\Psi_T(\mathbf{r})}\nabla_k^2 \phi\left(\mathbf{r}_k\right) \left[\prod_{i\neq k} \phi\left(\mathbf{r}_i\right)\right] \exp\left(\sum_{j<m} u\left(r_{jm}\right)\right)\\ + \frac{1}{\Psi_T(\mathbf{r})} 2\nabla_k \phi\left(\mathbf{r}_k\right) \left[\prod_{i\neq k} \phi\left(\mathbf{r}_i\right)\right] \exp\left(\sum_{j<m} u\left(r_{jm}\right)\right) \sum_{l\neq k} \nabla_k u\left(r_{jm}\right)\\ + \frac{1}{\Psi_T(\mathbf{r})} \left[\prod_{i} \phi\left(\mathbf{r}_i\right)\right] \nabla_k \exp\left(\sum_{j<m} u\left(r_{jm}\right)\right) \sum_{l\neq k} \nabla_k u\left(r_{jm}\right)\\ =\frac{\nabla_k^2 \phi\left(\mathbf{r}_k\right)}{\phi\left(\mathbf{r}_k\right)} + 2\frac{\nabla_k \phi\left(\mathbf{r}_k\right)}{\phi\left(\mathbf{r}_k\right)} \sum_{j<m} \nabla_k u\left(r_{jm}\right) + \left(\sum_{l\neq k} \nabla_k u\left(r_{jm}\right)\right)^2\\ + \sum_{l\neq k} \nabla_k^2 u\left(r_{jm}\right)\end{aligned}$$

Using the Chain rule we can separate the r_{kl} and the r_k we want to differentiate by.

$$\begin{aligned}\nabla_k u(r_{kl}) &= \nabla_k(r_{kl}) \frac{\partial}{\partial r_{kl}} u(r_{kl})\\ &= \frac{(r_k - r_l)}{r_{kl}} \frac{d}{dr_{kl}} u(r_{kl}) = \frac{(r_k - r_l)}{r_{kl}} u'(r_{kl})\end{aligned}$$

Which gives:

$$\begin{aligned}{\nabla_k}^2 u\left(r_{k,i}\right) =&\nabla_k \frac{(\mathbf{r}_k - \mathbf{r}_i)}{r_{ki}} u'\left(r_{ki}\right)\\ &=\frac{(\mathbf{r}_k - \mathbf{r}_i)}{r_{ki}} \nabla_k u'\left(r_{ki}\right) + u'\left(r_{ki}\right) \nabla_k \frac{(\mathbf{r}_k - \mathbf{r}_i)}{r_{ki}}\\ &=\frac{(\mathbf{r}_k - \mathbf{r}_i)}{r_{ki}} \cdot \frac{(\mathbf{r}_k - \mathbf{r}_i)}{r_{ki}} \frac{\partial}{\partial r_{ki}} u'\left(r_{ki}\right)\\ &\quad + u'\left(r_{ki}\right) \frac{r_{ki} \nabla_k \left(\mathbf{r}_k - \mathbf{r}_i\right) - \left(\mathbf{r}_k - \mathbf{r}_i\right) \nabla_k r_{ki}}{r_{ki}^2}\\ &=u''\left(r_{ki}\right) + u'\left(r_{ki}\right) \frac{r_{ki} \cdot \nabla_k \mathbf{r}_k - \left(\mathbf{r}_k - \mathbf{r}_i\right) \frac{(\mathbf{r}_k - \mathbf{r}_i)}{r_{ki}}}{r_{ki}^2}\\ &=u''\left(r_{ki}\right) + u'\left(r_{ki}\right) \frac{3r_{ki} - r_{ki}}{r_{ki}^2}\\ &=u''\left(r_{ki}\right) + \frac{2}{r_{ki}} u'\left(r_{ki}\right)\end{aligned}$$

Putting this together we get:

$$\frac{1}{\Psi_T(\mathbf{r})}\nabla_k^2\Psi_T(\mathbf{r}) = \frac{\nabla_k^2\phi\left(\mathbf{r}_k\right)}{\phi\left(\mathbf{r}_k\right)} + 2\frac{\nabla_k\phi\left(\mathbf{r}_k\right)}{\phi\left(\mathbf{r}_k\right)}\left(\sum_{j<m}\frac{(r_k-r_l)}{r_{kl}}u'(r_{kl})\right)$$
$$+\sum_{i\neq k}\sum_{j\neq k}\frac{(r_k-r_i)(r_k-r_j)}{r_{ki}r_{kj}}u'(r_{ki})u'(r_{kj})$$
$$+\sum_{j\neq k}(u''(r_{kj})+\frac{2}{r_{kj}}u'(r_{kj}))$$

6. Second derivative of particle k for Hamiltonian

We are to find an expression for

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

We have:

$$\begin{aligned}
u(r_{kj}) &= \ln(f(r_{kj}))\\
u'(r_{kj}) &= \frac{1}{f(r_{kj})}f'(r_{kj})\\
u''(r_{kj}) &= \frac{f(r_{kj})f''(r_{kj})-f'(r_{kj})^2}{f(r_{kj})^2}\\
f(a,r_{kj}) &= 1-\frac{a}{r_{kj}}\\
f'(r_{kj}) &= \frac{a}{r_{kj}^2}\\
f''(a,r_{kj}) &= -2\frac{a}{r_{kj}^3}
\end{aligned}$$

which gives

$$\begin{aligned}
u'(r_{kj}) &= \frac{1}{1-\frac{a}{r_{kj}}}\frac{a}{r_{kj}^2} = \frac{a}{r_{kj}(r_{kj}-a)}\\
u''(r_{kj}) &= \left(\frac{a}{r_{kj}^2-ar_{kj}}\right)' = -\frac{a}{(r_{kj}^2-ar_{kj})^2}\left(2r_{kj}-a\right)\\
&= \frac{-2ar_{kj}+a^2}{r_{kj}^2(r_{kj}-a)^2}.
\end{aligned}$$

We also have

$$\begin{aligned}
\phi_k &= \exp\left[-\alpha(x_i^2+y_i^2+\beta z_i^2)\right]\\
\nabla_k\phi_k &= 2\alpha\phi_k(x_k,y_k,\beta z_k)\\
{\nabla_k}^2\phi_k &= 2\alpha\phi_k(2\alpha(x^2+y^2+\beta z^2)+\beta+2).
\end{aligned}$$

This gives:

$$\begin{aligned}
\frac{{\nabla_k}^2\phi_k}{\phi_k} &= 2\alpha(2\alpha(x^2+y^2+\beta z^2)+\beta+2)\\
\frac{\nabla_k\phi_k}{\phi_k} &= 2\alpha(x_k,y_k,\beta z_k)\\
\frac{(r_k-r_j)}{r_{kj}}u'(r_{kj}) &= \frac{(r_k-r_j)}{r_{kj}}\frac{a}{r_{kj}(r_{kj}-a)} = \frac{a(r_k-r_j)}{r_{kj}^3-r_{kj}^2a}\\
u''+\frac{2}{r_{kj}}u' &= \frac{-2ar_{kj}+a^2}{r_{kj}^2(r_{kj}-a)^2}+\frac{2a}{r_{kj}^2(r_{kj}-a)}\\
&= \frac{-2ar_{kj}+a^2+2a(r_{kj}-a)}{r_{kj}^2(r_{kj}-a)^2} = 0
\end{aligned}$$

$$\begin{aligned}
&\frac{(r_k-r_i)(r_k-r_j)}{r_{ki}r_{kj}}u'(r_{ki})u'(r_{kj})\\
&= \frac{(r_k-r_i)(r_k-r_j)}{r_{ki}r_{kj}}\frac{a^2}{r_{ki}r_{kj}(r_{ki}-a)(r_{kj}-a)}\\
&= \frac{a^2}{(r_k-r_i)(r_k-r_j)(r_{ki}-a)(r_{kj}-a)}
\end{aligned}$$

If we put everything together we get:

$$\begin{aligned}
\frac{1}{\Psi_T(r)}\nabla_k^2\Psi_T(r) =& 2\alpha\left(2\alpha\left(x_k^2+y_k^2+\beta^2z_k^2\right)+\beta+2\right)\\
&+4\alpha\left(x_k,y_k,\beta z_k\right)\sum_{j\neq k}\frac{a(r_k-r_j)}{r_{kj}^3-r_{kj}^2a}\\
&+\sum_{i\neq k}\sum_{j\neq k}\frac{a^2}{(r_k-r_i)(r_k-r_j)(r_{ki}-a)(r_{kj}-a)}\\
=& 2\alpha\left(2\alpha\left(x_k^2+y_k^2+\beta^2z_k^2\right)+\beta+2\right)\\
&+4\alpha\left(x_k,y_k,\beta z_k\right)\sum_{j\neq k}\frac{a(r_k-r_j)}{r_{kj}^3-r_{kj}^2a}\\
&+\left(\sum_{j\neq k}\frac{a}{(r_k-r_j)(r_{kj}-a)}\right)^2
\end{aligned}$$

which can be put directly into the Hamiltonian.

7. New units

We introduce length in units of a_{ho} and energy in units $\hbar\omega_{ho}$.

$$
\begin{aligned}
H =& \sum_i^N \left(\frac{-\hbar^2}{2m} \nabla_i^2 + \frac{1}{2} m \left[\omega_{ho}^2 \left(x^2 + y^2 \right) + \omega_z^2 z^2 \right] \right) \\
&+ \sum_{i<j}^N V_{int} \left(\mathbf{r}_i, \mathbf{r}_j \right) \\
\to& \sum_i^N \frac{1}{2\hbar\omega_{ho}} \left(-\frac{\hbar^2}{m{a_{ho}}^2} \nabla_i^2 + m{a_{ho}}^2 \left[\omega_{ho}^2 \left(x^2 + y^2 \right) + \omega_z^2 z^2 \right] \right) \\
&+ \sum_{i<j}^N V_{int} \left(\mathbf{r}_i, \mathbf{r}_j \right) \\
=& \sum_i^N \frac{1}{2} \left(-\nabla_i^2 + x^2 + y^2 + \frac{m{a_{ho}}^2}{\hbar{\omega_{ho}}^2} \omega_z^2 z^2 \right) + \sum_{i<j}^N V_{int} \left(\mathbf{r}_i, \mathbf{r}_j \right) \\
=& \sum_i^N \frac{1}{2} \left(-\nabla_i^2 + x^2 + y^2 + \gamma^2 z^2 \right) + \sum_{i<j}^N V_{int} \left(\mathbf{r}_i, \mathbf{r}_j \right)
\end{aligned}
$$

where

$$
\gamma^2 = \frac{m{a_{ho}}^2}{\hbar{\omega_{ho}}^2} \omega_z^2 = \frac{\hbar}{m\omega_{ho}} \frac{m}{\hbar\omega_{ho}} \omega_z^2 = \frac{\omega_z^2}{\omega_{ho}^2} \Rightarrow \gamma = \frac{\omega_z}{\omega_{ho}}
$$

[1] Henrik Flyvbjerg and Henrik Gordon Petersen. Error estimates on averages of correlated data. *The Journal of Chemical Physics*, 91(1):461–466, 1989.

[2] Morten Hjorth-Jensen. Computational physics 2: computational quantum mechanics.

[3] Morten Hjorth-Jensen. *Computational Physics Lecture Notes Fall 2015*. 2015.

[4] Marius Jonsson. Standard error estimation by an automated blocking method. *Physical Review E*, 98(4):043304, 2018.

[5] Morten Ledum. Simple variational monte carlo solve for fys4411.