

Variational Monte Carlo on Elliptical Harmonic Traps with N Bosons

Jonas Boym Flaten

University of Oslo

June 4, 2021

Abstract

Using the Julia language, the method of Variational Monte Carlo (VMC) was implemented for a D -dimensional elliptical harmonic trap system with N bosons, both with and without interaction. 2 variational parameters were tuned using the gradient descent method to obtain an estimate of the ground state energy for traps with $N = \{10, 50, 100\}$ bosons. The results are mainly benchmarked against [1] which considered the same system but implemented the VMC method in C++ and only varies 1 variational parameter. A comparison of the results from two different Monte Carlo sampling methods, as well as statistical refinement of the results, is presented. Importance sampling using Langevin "quantum drift" sampling is shown to be superior to brute-force "random step" sampling. Finally, the effect of boson interaction is discussed.

1 Introduction

As far as current human knowledge goes, Quantum Mechanics seems to be an excellent model of the universe at the smallest scales. In this "picoscopic" regime our world can be thought to consist of fuzzy elementary particles, spreading diffusely through space but exchanging sharp energy quanta through four fundamental mechanisms. One big caveat however is the rate at which a thorough quantum mechanical analysis becomes impossible to conduct. For anything more than 1 particle, the complexity of the system of differential equations to consider quickly becomes overwhelming. Add several interaction mechanisms at once and you are sure to end up with a problem that nobody on Earth has been able to solve analytically (yet).

These difficulties with analytical analysis of quantum many-body systems encourages consideration of statistical and numerical methods. With the currently available computational power in one's own office and the advent of collaborative development of code over the Internet, the field of Computational Physics has emerged as a third alternative to the traditional theoretical and experimental branches of research. Especially within Quantum Mechanics, where the theoreticians face the complexity of coupled differential systems as discussed above while the experimentalists deal with the frailty (and cost) of experiments tuned to the extreme parameters of the quantum world, computational research is in many cases better suited to the task. Through numerical simulations founded on statistical methods and analysis, large quantum mechanical systems can be explored with remarkable precision.

In this report, the computational method of Variational Monte Carlo (VMC) on an elliptical harmonic

trap system containing N bosons is considered, and its implementation in Julia and subsequent results are reported and benchmarked against the previous work of [1] and others on the same system. The system is itself interesting because —noe om bosonisk samfall, noen referanser til forskning på systemet?— as demonstrated by the famous experiment of [2]. VMC treatment of this system is discussed by [3], which in turn serves as the inspiration for the work in this report.

Sections 2 and 3 provide the concrete mathematical model for the system in consideration and the principles leading up to the VMC method. In section 4 the actual Julia implementation as well as the scientific approach to the problem is described. Finally in section 5 the results are presented, discussed and compared to the results of [1].

2 Model

The model system considered in the project is declared in this chapter.

2.1 Quantum system

The quantum system considered here is a D -dimensional quantum trap with an arbitrary number of identical bosons N . In position basis the system Hamiltonian is

$$H[\vec{R}] = \sum_i^N \left(U[\vec{r}_i] - \frac{\hbar^2}{2m} \nabla_i^2 \right) + \sum_i^N \sum_{j>i}^N V[\vec{r}_i, \vec{r}_j] \quad (1)$$

with

$$U[\vec{r}_i] = \frac{m\omega^2}{2} (x_i^2 + y_i^2 + \lambda^2 z_i^2), \quad (2)$$

$$V[\vec{r}_i, \vec{r}_j] = V[\Delta r_{ij}] = \begin{cases} \infty & \text{for } \Delta r_{ij} \leq a \\ 0 & \text{for } \Delta r_{ij} > a \end{cases}. \quad (3)$$

In other words U is an elliptical harmonic oscillator potential, and V describes a hard-sphere interaction between the bosons. In the equations above as well as the rest of this text, \vec{r}_i and $\vec{\nabla}_i$ are the position and spatial derivative of the boson indexed by i . Furthermore $\Delta \vec{r}_{ij} = \vec{r}_i - \vec{r}_j$ is the distance vector between the bosons i and j . Finally \vec{R} is the configuration vector of the system as a whole (containing all the bosonic positions \vec{r}_i).

The parameters of the model are the mass m and characteristic radius a of the bosons, the trap strength ω , as well as the elliptic parameter λ (which potentially makes the trap elliptical in the z -direction). However the variables of the system can be made dimensionless by introducing $\sqrt{\frac{\hbar}{m\omega}}$ as length unit and $\hbar\omega$ as energy unit by splitting

$$\begin{aligned} \vec{r}_i &= \sqrt{\frac{\hbar}{m\omega}} \vec{r}'_i, & \vec{\nabla}_i &= \sqrt{\frac{m\omega}{\hbar}} \vec{\nabla}'_i, \\ H &= \hbar\omega H', & a &= \sqrt{\frac{\hbar}{m\omega}} a'. \end{aligned}$$

The primed variables \vec{r}'_i , $\vec{\nabla}'_i$, H' and a' , are now dimensionless and give the relative size to the corresponding unit. Omitting the primes, the Hamiltonian (1) takes the form

$$H[\vec{R}] = \sum_i^N \frac{1}{2} (U[\vec{r}_i] - \nabla_i^2) + \sum_i^N \sum_{j>i}^N V[\Delta r_{ij}] \quad (4)$$

with

$$U[\vec{r}_i] = x_i^2 + y_i^2 + \lambda^2 z_i^2, \quad (5)$$

$$V[\Delta r_{ij}] = \begin{cases} \infty & \text{for } \Delta r_{ij} \leq a \\ 0 & \text{for } \Delta r_{ij} > a \end{cases}. \quad (6)$$

In the rest of this report, dimensionless variables and equations are considered. Systems of $D \in \{1, 2, 3\}$ dimensions with $N \in \{1, 10, 50, 100\}$ bosons are considered. The respective values $\lambda = 1$ and $\lambda = \sqrt{8}$ will be used to examine spherical and elliptical traps, while the values $a = 0$ and $a = 0.0043$ will be used to examine the non-interacting and interacting case. The last value of a is the characteristic radius of Rb-87 given in [3], which corresponds to a trap strength $\omega \approx 400$ /s if one uses the value $a_{\text{Rb-87}} \approx 5.82$ nm found experimentally in [4].

2.2 Trial state

The variational method discussed in chapter 3 requires a trial state. The bosonic trial state in consideration

has the position basis form

$$\Psi[\vec{R}] = \prod_i^N g[\vec{r}_i] \prod_j^N \prod_{k>j}^N f[\vec{r}_j, \vec{r}_k] \quad (7)$$

with

$$g[\vec{r}_i] = \epsilon^{-\alpha(x_i^2 + y_i^2 + \beta z_i^2)}, \quad (8)$$

$$f[\vec{r}_i, \vec{r}_j] = f[\Delta r_{ij}] = \begin{cases} 0 & \text{for } \Delta r_{ij} \leq a \\ 1 - \frac{a}{\Delta r_{ij}} & \text{for } \Delta r_{ij} > a \end{cases}, \quad (9)$$

Here ϵ is Euler's number and a is the characteristic boson radius introduced in section 2, while α and β are the variational parameters to tune when chasing the ground state energy.

3 Theory

The underlying theory of the project is presented in this chapter.

3.1 The variational method

The Variational Principle of Quantum Mechanics states that the ground state E_G of any quantum system with Hamiltonian operator \mathbf{H} is bounded from above by

$$E_G \leq \frac{\langle \Psi | \mathbf{H} | \Psi \rangle}{\langle \Psi | \Psi \rangle} \quad (10)$$

for any state $|\Psi\rangle$, and the variational method is based on finding a suitable trial state which minimises the bound $\frac{\langle \Psi | \mathbf{H} | \Psi \rangle}{\langle \Psi | \Psi \rangle}$ in order to get an approximation to E_G . For N particles in D dimensions the bound takes the form

$$\frac{\langle \Psi | \mathbf{H} | \Psi \rangle}{\langle \Psi | \Psi \rangle} = \frac{\int \cdots \int \Psi^*[\vec{R}] H[\vec{R}] \Psi[\vec{R}] \delta^{DN} R}{\int \cdots \int |\Psi[\vec{R}]|^2 \delta^{DN} R}$$

in position basis. Introducing now a quantity known as the *local energy* ε as

$$\varepsilon[\vec{R}] = \frac{H[\vec{R}] \Psi[\vec{R}]}{\Psi[\vec{R}]}, \quad (11)$$

the bound can be rewritten to

$$\frac{\langle \Psi | \mathbf{H} | \Psi \rangle}{\langle \Psi | \Psi \rangle} = \int \cdots \int \Pi[\vec{R}] \varepsilon[\vec{R}] \delta^{DN} R,$$

where

$$\Pi[\vec{R}] = \frac{|\Psi[\vec{R}]|^2}{\int \cdots \int |\Psi[\vec{R}]|^2 \delta^{DN} R} \quad (12)$$

is the spatial probability distribution over the configuration space spanned by \vec{R} . But then the Variational Principle (10) can be recast as

$$E_G \leq \langle \varepsilon \rangle_\Pi, \quad (13)$$

and so the variational method becomes a matter of estimating the expected local energy, because this expected value provides an energy bound for the ground

state of the system. At this probabilistic reinterpretation of the Variational Principle, a *Monte Carlo simulation* enters the picture.

Monte Carlo simulations in general are based on drawing a series of random samples from some relevant probability distribution and using statistics to harvest desired results about the system in consideration. In the case of *Variational Monte Carlo* (or VMC), the samples to draw are the local energies ε and the relevant distribution is the spatial distribution Π , defined in equations (11) and (12) above. For this report these quantities are defined with respect to the Hamiltonian H and the trial wavefunction Ψ declared in (4) and (7), and in appendix A.1 the required derivatives of Ψ are calculated to find an analytical expression for the local energy ε . But even though the sample function ε is known and the trial wavefunction Ψ itself is defined, its corresponding distribution Π involves a DN -dimensional integral which quickly becomes impossible to calculate both analytically and numerically.

3.2 The Metropolis algorithm

The *Metropolis algorithm*, first introduced in [5], is an algorithm which makes it possible to sample from just some probability distribution with an unknown constant, and so it is perfectly suited for Variational Monte Carlo. In terms of this report, the algorithm is based on constructing an ergodic Markov chain with Π as its stationary distribution and letting this Markov chain run from some initial configuration, sampling the local energy ε along the way. The longer the Markov chain is allowed to run, the more of the distribution Π is fleshed out, and the better the resulting mean value of ε matches its actual expected value $\langle \varepsilon \rangle_\Pi$. This follows from the *Central Limit Theorem*, which indeed states that for M Monte Carlo cycles (where one cycle is one step in the Markov chain), the expected value of ε is approximated by the sample mean, hereafter denoted simply as the *VMC energy* E_{VMC} :

$$\langle \varepsilon \rangle_\Pi \approx \text{mean } \varepsilon = \frac{1}{M} \sum_{m=1}^M \varepsilon[\vec{R}_m] \equiv E_{\text{VMC}}. \quad (14)$$

The statistical error of this approximation is given by the same theorem to be

$$\begin{aligned} \Delta E_{\text{VMC}} &= \sqrt{\frac{1}{M} \text{van } \varepsilon} \\ &= \sqrt{\frac{1}{M} \left(\frac{1}{M} \sum_{m=1}^C \varepsilon^2[\vec{R}_m] - E_{\text{VMC}}^2 \right)}, \end{aligned} \quad (15)$$

where $\text{van } \varepsilon$ is the sample variance. In these equations, m is an index of each sample and \vec{R}_m is the configuration at that cycle.

The Metropolis Markov chain is constructed from two transition probabilities: the proposal distribution $P[\vec{R}'|\vec{R}]$, which gives the probability of proposing a move to some new configuration \vec{R}' when the previous sample was drawn at \vec{R} , and the acceptance probability

$A[\vec{R}'|\vec{R}]$ which gives the probability of actually accepting this same proposed move. Note that while both P and A must take values between 0 and 1, P must be a proper probability distribution in the sense that for all \vec{R}

$$\sum_{\vec{R}'} P[\vec{R}'|\vec{R}] = 1, \quad (16)$$

while A is only some probability function. The Markov matrix then has the transition rates

$$\begin{aligned} M[\vec{R}'|\vec{R}] &= \\ A[\vec{R}'|\vec{R}]P[\vec{R}'|\vec{R}] &+ \delta_{\vec{R}'\vec{R}} \sum_{\vec{R}''} \left(1 - A[\vec{R}''|\vec{R}]\right) P[\vec{R}''|\vec{R}] \end{aligned} \quad (17)$$

where δ is Kronecker's delta, and by letting this matrix act on the target distribution $\Pi[\vec{R}]$ and requiring it to stay the same, the so-called *balance criterion*

$$\begin{aligned} \sum_{\vec{R}'} A[\vec{R}'|\vec{R}]P[\vec{R}'|\vec{R}]\Pi[\vec{R}] \\ = \sum_{\vec{R}'} A[\vec{R}|\vec{R}']P[\vec{R}|\vec{R}']\Pi[\vec{R}'] \end{aligned} \quad (18)$$

follows. If the Markov chain is further required to be ergodic, so that it is ensured to sample from all possible configurations given enough time, then the balance criterion turns into the *detailed balance criterion* in which the sums in (18) are removed. Rewriting the detailed balance criterion, the following constraint is put on the acceptance rate A :

$$\frac{A[\vec{R}'|\vec{R}]}{A[\vec{R}|\vec{R}']} = \frac{P[\vec{R}|\vec{R}']\Pi[\vec{R}']}{P[\vec{R}'|\vec{R}]\Pi[\vec{R}]}. \quad (19)$$

The simplest choice of A to enforce this constraint is the *Metropolis choice*, which is an acceptance probability given by

$$A[\vec{R}'|\vec{R}] = \min \left\{ 1, \frac{P[\vec{R}|\vec{R}']\Pi[\vec{R}']}{P[\vec{R}'|\vec{R}]\Pi[\vec{R}]} \right\}. \quad (20)$$

With this definition of the acceptance probability, the Markov chain is ensured to eventually sample from all corners of the sample space, so that given enough Monte Carlo cycles the sample mean value of ε will converge to its true expected value $\langle \varepsilon \rangle_\Pi$. While almost magical in its simplicity, it is now clear how the Metropolis algorithm can sample from Π without ever calculating the large integral in (12); because only a ratio of Π values appears in the acceptance ratio above, the integral cancels out and is in some sense removed from the problem.

As for the proposal distribution P , it can in principle be any distribution as long as (16) is fulfilled, but the choice will affect both the acceptance ratio to be computed from (20) as well as the number of Monte Carlo cycles required for the VMC energy E_{VMC} to converge. Because the proposal distribution directly affects the moves which are proposed and thus sampled, the choice of proposal distribution corresponds

to choosing a sampling method for the Metropolis algorithm. Note that because the trial wavefunction defined in (7) is a product of functions which depend only on one or two particles each, the computation of the acceptance ratio A simplifies significantly if only one random particle is chosen and moved in each move. In other words, by picking one random particle i with probability $\frac{1}{N}$ and then drawing a move for this particle from some proposal distribution $P[\vec{R}'_i|\vec{R}]$, the expression (20) for the acceptance probability A reduces to

$$A[\vec{R}'_i|\vec{R}] = \min \left\{ 1, \frac{P[\vec{R}'_i|\vec{R}]}{P[\vec{R}_i|\vec{R}]} \prod_{j \neq i} \frac{g^2[\vec{r}'_i] f^2[\Delta r'_{ij}]}{g^2[\vec{r}_i] f^2[\Delta r_{ij}]} \right\}, \quad (21)$$

where the only difference between the configurations \vec{R} and \vec{R}'_i is that the position \vec{r}_i of the randomly chosen particle i has changed to \vec{r}'_i . Hence instead of computing the whole trial wavefunction at each Monte Carlo cycle, only the factors involving particle i need to be computed.

Two sampling methods based on single particle moves will be considered in this report. In the first brute-force "random step" method, a new position for particle i is drawn as

$$\vec{r}'_i = \vec{r}_i + \tilde{\delta}\vec{r}, \quad (22)$$

where $\tilde{\delta}\vec{r}$ is a stochastic vector drawn uniformly from the interval $-\delta s \leftrightarrow \delta s$ in each spatial direction for some fixed step size δs . The proposal distribution P in this case is symmetric,

$$p[\vec{R}'_i|\vec{R}] = p[\vec{R}|\vec{R}'_i],$$

so the acceptance ratio in (21) reduces to

$$A[\vec{R}'_i|\vec{R}] = \min \left\{ 1, \prod_{j \neq i} \frac{g^2[\vec{r}'_i] f^2[\Delta r'_{ij}]}{g^2[\vec{r}_i] f^2[\Delta r_{ij}]} \right\}. \quad (23)$$

3.3 Quantum drift sampling

The second sampling method to be considered, referred to as the "quantum drift" method, is based on the *Langevin equation* – a stochastic differential equation originally introduced to consider Brownian motion. Its original full form is

$$m \frac{\delta^2 \vec{r}}{\delta t^2} = -\gamma \frac{\delta \vec{r}}{\delta t} + \vec{F}[\vec{r}] + \tilde{\vec{B}}, \quad (24)$$

which is nothing else than Newton's second law for a particle subject to a frictional force $-\gamma \frac{\delta \vec{r}}{\delta t}$, an external force field \vec{F} as well as a stochastic Brownian force $\tilde{\vec{B}}$ meant to invoke random fluctuations in the particle's trajectory. In the case of a very strong frictional force the acceleration can be neglected, and the equation reduces to the overdamped Langevin equation

$$\frac{\delta \vec{r}}{\delta t} = \frac{\vec{F}[\vec{r}]}{\gamma} + \frac{\tilde{\vec{B}}}{\gamma}, \quad (25)$$

Using now the length unit set in section 2.1 and introducing a suitable time unit, this equation can be made dimensionless and recast to

$$\frac{\delta \vec{r}}{\delta t} = \frac{1}{2} \vec{Q}[\vec{r}] + \tilde{\vec{W}}, \quad (26)$$

where \vec{Q} is a dimensionless drift term and $\tilde{\vec{W}}$ is a dimensionless stochastic term. Discretising this with the *Euler-Maruyama method* (a modified Euler method for stochastic differential equations) a new algorithm emerges for proposing moves for the random particle i through

$$\vec{r}'_i = \vec{r}_i + \frac{1}{2} \vec{Q}_i[\vec{R}] \delta t + \tilde{\delta}\vec{r} \quad (27)$$

for some fixed small (dimensionless) time step δt and a stochastic vector $\tilde{\delta}\vec{r}$ drawn from a normal distribution with mean zero and deviation $\sqrt{\delta t}$ in each spatial direction. For some consistency with the random step sampling method introduced at the end of section 3.2, δs can be introduced as

$$\delta s = \sqrt{\delta t} \quad (28)$$

to recast the proposal algorithm as

$$\vec{r}'_i = \vec{r}_i + \frac{1}{2} \vec{Q}_i[\vec{R}] \delta s^2 + \tilde{\delta}\vec{r}, \quad (29)$$

in which $\tilde{\delta}\vec{r}$ now is a stochastic vector drawn from a normal distribution with mean zero and deviation δs in each spatial direction.

The proposal distribution P for the algorithm in (29) is not symmetric, but can be found by considering the corresponding *Fokker-Planck equation*, which is a partial differential equation describing the time evolution of the spatial configuration distribution Π when a Langevin equation is invoked on a particle. The Fokker-Planck equation corresponding to the overdamped Langevin equation (26) is also known as the *Smoluchowski equation* and is given by

$$\frac{\delta \Pi}{\delta t} = \frac{1}{2} \vec{\nabla}_i \cdot (\vec{\nabla}_i - \vec{Q}_i) \Pi. \quad (30)$$

The spatial configuration distribution Π is sure to remain constant as long as the drift term \vec{Q}_i takes the form

$$\vec{Q}_i[\vec{R}] = \frac{\vec{\nabla}_i \Pi[\vec{R}]}{\Pi[\vec{R}]}, \quad (31)$$

and the proposal distribution P turns out to be nothing else than the Green function of the Smoluchowski equation (30), which can be calculated to be

$$G[\delta s; \vec{R}'_i|\vec{R}] = \sqrt{\frac{1}{2\pi\delta s^2}}^{DN} \epsilon^{-\frac{(\vec{r}'_i - \vec{r}_i - \frac{1}{2} \vec{Q}_i[\vec{R}]\delta s^2)^2}{2\delta s^2}}. \quad (32)$$

In terms of Variational Monte Carlo the drift term is known as the *quantum drift* and takes the form

$$\vec{Q}_i[\vec{R}] = \frac{2\vec{\nabla}_i \Psi[\vec{R}]}{\Psi[\vec{R}]}. \quad (33)$$

In appendix A.2 an analytical expression for the quantum drift is found using the derivatives of the trial wavefunction Ψ which were already calculated in appendix A.1. Noting that

$$\begin{aligned} & \left(\vec{r}'_i - \vec{r}_i - \frac{1}{2} \vec{Q}_i[\vec{R}] \delta s^2 \right)^2 - \left(\vec{r}_i - \vec{r}'_i - \frac{1}{2} \vec{Q}_i[\vec{R}'] \delta s^2 \right)^2 \\ &= - \left(\vec{r}'_i - \vec{r}_i \right) \cdot \left(\vec{Q}_i[\vec{R}'] + \vec{Q}_i[\vec{R}] \right) \delta s^2 - \frac{\vec{Q}_i^2[\vec{R}'] - \vec{Q}_i^2[\vec{R}]}{4} \delta s^4 \\ &= - \left(\vec{r}'_i - \vec{r}_i + \frac{\vec{Q}_i[\vec{R}'] - \vec{Q}_i[\vec{R}]}{4} \delta s^2 \right) \cdot \left(\vec{Q}_i[\vec{R}'] + \vec{Q}_i[\vec{R}] \right) \delta s^2 \end{aligned}$$

the acceptance probability in (21) reduces in this case of quantum drift sampling to

$$A[\vec{R}'_i|\vec{R}] = \min \left\{ 1, \prod_{j \neq i} \frac{g^2[\vec{r}'_i] f^2[\Delta \vec{r}'_{ij}]}{g^2[\vec{r}_i] f^2[\Delta \vec{r}_{ij}]} e^{-\frac{\vec{Q}_i[\vec{R}'] + \vec{Q}_i[\vec{R}]}{2} \cdot \left(\vec{r}'_i - \vec{r}_i + \frac{\vec{Q}_i[\vec{R}'] - \vec{Q}_i[\vec{R}]}{4} \delta s^2 \right)} \right\} \quad (34)$$

when inserting the Green function in (32) as the proposal distribution P .

As apparent from its definition (33), the quantum drift moves the particles with the distribution gradient, towards configurations of higher probability, and thus ensures that the "most important" configurations are sampled early on. This is why the quantum drift sampling method is also known as *importance sampling*, and it should in general lead to faster convergence of the VMC energy E_{VMC} with respect to the number of Monte Carlo cycles.

3.4 Gradient descent variation

So far only the Monte Carlo sampling itself has been discussed, but an equally important part of Variational Monte Carlo is to find the optimal values for the variational parameters, in the case of this report α and β . After all, estimating an energy bound $\langle \varepsilon \rangle_{\Pi}$ for *some* α and β ridiculously fast and to great precision with quantum drift Monte Carlo is no good if the bound is large compared to other values of α and β . Indeed the whole point of the Variational Principle (10) is that it provides an energy estimate only when the bound is *varied* and *minimised* with respect to the variational parameters. In the following discussion as well as the rest of this report, paired values for α and β are referred to as *variational points*.

The crudest way to minimise the energy bound $\langle \varepsilon \rangle_{\Pi}$ is to estimate it for some range of variational points and pick the lowest value. This "range variation" can only be expected to provide a reasonable energy estimate if the range of variational points are in the vicinity of the true minimum. A more sophisticated method of variation is the so-called *gradient descent variation*, in which the variational gradient (the gradient of $\langle \varepsilon \rangle_{\Pi}$ with respect to α and β) is estimated at some initial variational point and used to guide the values of α and β "downwards" towards regions of lower energy $\langle \varepsilon \rangle_{\Pi}$.

In appendix A.3 an analytical expression for the variational gradient is calculated, which turns out to require sampling of the quantities $\frac{\partial \ln \Psi}{\partial \alpha}$ and $\frac{\partial \ln \Psi}{\partial \beta}$ as well as their products with the local energy ε at each Monte Carlo cycle so that the expected values $\left\langle \frac{\partial \ln \Psi}{\partial \alpha_i} \varepsilon \right\rangle_{\Pi}$ and $\left\langle \frac{\partial \ln \Psi}{\partial \beta_i} \varepsilon \right\rangle_{\Pi}$ can be estimated. Then, new, better values of α and β can be found through

$$\alpha' = \alpha - \frac{\partial \langle \varepsilon \rangle_{\Pi}}{\partial \alpha} \delta v, \quad (35)$$

$$\beta' = \beta - \frac{\partial \langle \varepsilon \rangle_{\Pi}}{\partial \beta} \delta v, \quad (36)$$

for some small step size δv . This step size should be tuned so that it is small enough for the gradient descent to converge towards the closest variational minimum, but no smaller as convergence will then be slower. When the gradient $\frac{\partial \langle \varepsilon \rangle_{\Pi}}{\partial \alpha_i}$ is less than some small convergence threshold δg in both directions, approximately optimal values of α and β have been found. The smaller the convergence threshold, the better the approximation.

Because of the finite step size δv and convergence threshold δg , gradient descent variation as described here is only approximate anyway, so there is no need to run too many Monte Carlo cycles at each variational point during descent. Hence a large number of Monte Carlo cycles will only be run at the final, optimal variational point, and so gradient descent variation should both be faster and more accurate than range variation, especially when the optimal variational parameters α and β are not known.

3.5 Block resampling

As described in section 3.2, the calculated sample mean bound E_{VMC} converges towards the true energy bound $\langle \varepsilon \rangle_{\Pi}$ for a large number of Monte Carlo cycles M . However, the statistical error given in (15) is only correct if there is no correlation between the samples. The Metropolis algorithm presented here however, proposes to move only a single particle a short distance with each Monte Carlo cycle. Therefore two configurations following each other in the proposed Markov chain should be strongly correlated; indeed there should be correlation between whole chains of subsequent configurations because the system needs several cycles to "get away" from any current configuration.

A more formal way of measuring correlation in the samples of ε is considering the *sample autocovariance*

$$\text{aucov}_l[\varepsilon] = \frac{1}{M-l} \sum_m^{M-l} \left(\varepsilon[\vec{R}_m] - E_{\text{VMC}} \right) \left(\varepsilon[\vec{R}_{m+l}] - E_{\text{VMC}} \right) \quad (37)$$

for all chain lengths l less than M . As long as this covariance is finite for any l , the samples generated from the Metropolis algorithm are correlated, which means that the statistical error formula (15) should be

replaced by

$$\Delta E_{\text{VMC}} = \sqrt{\frac{1}{M} \left(\text{var}[\varepsilon] + 2 \sum_{l=1}^{M-1} \text{aucov}_l[\varepsilon] \right)}. \quad (38)$$

But this expression involves a double sum in the autocovariance term, which can be quite costly for long Monte Carlo simulations with a high amount of samples. A more efficient method of estimating the correct statistical error, is so-called *block resampling*. Resampling of an experiment in general means extracting better statistical estimates using nothing but the original sample set. The block resampling method is based on the idea that the statistical error of the sample set becomes more apparent if the samples are partitioned into larger blocks and the block averages are used as samples instead. Mathematically, the sample set of ε is transformed with

$$\varepsilon'_m = \frac{\varepsilon_{2m-1} + \varepsilon_{2m}}{2}, \quad (39)$$

for all sample indices m from 1 to $\frac{M}{2}$. The new sample set is half the size of the original sample set, but both the sample mean and the statistical error turns out to be conserved through this "blocking" of the samples. The sample variance however will be different, and it can be shown that the transformation (39) corresponds to "shifting weight" from the autocovariance terms in (38) to the variance term. By blocking multiple times this shifting continues until the samples are no longer correlated, at which point the statistical error formula (38) reduces to the simple form in (15). Continued blocking will eventually shift the error away from the variance term again when the number of samples approaches zero. However, the sample variance turns out to converge strongly towards its maximum and fluctuate around it for a while before declining. Hence blocking can simply be continued until the sample variance declines for the first time. The maximal value of the sample variance can then be used with equation (15) to provide a higher but much more correct estimate of the statistical error for the VMC energy E_{VMC} .

4 Method

As part of the project, a working VMC script was developed in the Julia language, implementing all the methods and formulas introduced in chapter 3. The script is named `QuantumTrapVMC.jl` and can be accessed in the `code` folder at the GitHub page¹ of this project. The script uses names for all variables and parameters consistent with this report, and it is thoroughly commented for easy examination. Its overarching structure is presented in section 4.2 of this chapter, as well as a description of the numerical optimisation performed during development in section 4.3. Finally, section 4.4 regards tuning of the algorithm parameters δv , δg and δs . Firstly though, a brief introduction to the main characteristics of the Julia language is given in section 4.1.

4.1 The Julia language

The Julia language is a fairly new dynamic programming language launched in 2012 and developed initially by computational scientists at the Massachusetts Institute of Technology (MIT), but nowadays by people from all over the world. The goal of the creators was to create a modern high-performance language with "the speed of C" and the "familiar mathematical notation of Matlab" while being "as usable for general programming as Python" and "as easy for statistics as R".[6]

Julia differs from many other popular programming languages in the fact that it is not object-oriented. Instead the user is encouraged to implement polymorphic functions through Julia's *Multiple Dispatch* system, which can then be called and compiled using the native read-eval-print-loop (REPL) on the command line or any supported integrated development environment (IDE). As a dynamic language Julia furthermore relies on garbage-collection (GC) and just-in-time-compilation (JIT), which is user-friendly but can result in slow computation if one is not implementing the algorithms optimally.

At first glimpse, Julia code looks quite similar to Python, Matlab and R, and indeed has similar commands and features in many cases. One distinction however is that Julia supports Unicode, which allows the programmer to use Greek letters and much mathematical notation directly in the code. This is handy when implementing programs based on theoretical methods and formulas such as in this project.

4.2 Structure

The final Julia script `QuantumTrapVMC.jl` was written with the system (4) and the VMC theory described in chapter 3 in mind, and built for flexible usage with the Julia REPL. The parametric values D , N , a and λ of the quantum harmonic trap are organised in `QuantumTrap` structs, which can be passed to the main function `find_VMC_energy` along with an array `Ms` of Monte Carlo cycle numbers to consider. Optional keyword arguments such as `variation` and `sampling` (as well as the related δv , δg and δs) can be passed as well to run VMC with a specific choice of the methods discussed in 3. There are also keyword arguments `αs` and `βs` passing specific values for the variational parameters to consider. For range variation, an array of values for both α and β can be considered and plotted, while for gradient descent variation only one initial variational point should be passed. Finally, the keyword arguments `text_output` and `plot_output` determine what output the script should return to the user.

The script is parallelised through the native Julia package `Distributed`, which sets up T parallel processes on the computer. To this end, a separate function `simulate_n_sample` is declared and distributed to all the processes, hereafter referred to as "worker threads". Equipped with this function, the T worker

¹<https://github.com/flarrek/FYS9411-Project-1>

threads are the actual workhorses of the VMC computations, setting up individual copies of the quantum trap and running $C = \frac{M}{T}$ Monte Carlo cycles using the chosen methods. At each thread, the N particles are scattered initially following a normal distribution.² Sampled values as well as the number of rejected moves are then collected by the main thread, which proceeds to resample the data by blocking, as described in section 3.5. The main thread finally presents the results back to the user through the chosen text and plot output.

The complementary function `compare_VMC_sampling` was implemented to allow comparison of convergence between the random step and quantum drift sampling methods from sections 3.2 and 3.3. It is called similarly to `find_VMC_energy`, except that the keyword argument `sampling` is obviously not specified, and it only allows one specific value for each of the variational parameters α and β .

As already noted, the script is thoroughly commented, and furthermore assertions were added so that the script objects to invalid input and explains to the user what should be adjusted. Hopefully this makes the script fairly easy to use and examine on its own.

4.3 Optimisation

Thorough numerical optimisation was performed during development of the Julia script. In this section, some of the computational bottlenecks of Julia and the VMC theory presented in chapter 3 are discussed, as well as related optimisation. Table 1 shows how various optimisation changes to the code resulted in a total speed-up of about 20 for random step (RS) sampling and 14 for quantum drift (QD) sampling, when compared to the initial code. The times were all measured on a 4-core (8-thread) computer running a reference simulation of 100 million Monte Carlo cycles on a 1D trap with 2 interacting bosons. When the actual results of chapter 5 were calculated however, a 10-core (20-thread) computer was used instead, on which the same reference simulation took only 10 s / 17 s using RS / QD sampling.

Table 1: Comparison of computation times for a reference simulation after various optimisations to the Julia script. 'RS' and 'QD' denotes random step and quantum drift sampling, respectively. The times were all measured on a 4-core (8-thread) computer.

Optimisation	Computation time
	RS / QD
Initial code.	664 s / 946 s
Implement matrix storage.	376 s / 667 s
Remove index slice access.	201 s / 394 s
Devectorise loops.	121 s / 271 s
Parallellise code	
using <code>Threads</code> .	56 s / 120 s
Parallellise code	
using <code>Distributed</code> .	34 s / 69 s

²An alternative scattering of the particles into an initial square lattice was also implemented and can be set through the keyword argument `scattering`, but it lead to the same results and is hence not considered further in this report.

The most demanding computation of the VMC method presented here is the calculation of the local energy ε at each Monte Carlo cycle, calculated with the formula (A18). One should strive to minimise the amount of times this computation is done. Consider as an example the case when a move has been rejected so that the particles stay in the same configuration as in the last cycle. Then there is no point in recalculating the local energy – instead the sampled value from the last cycle should simply be copied. This was implemented in the script from the start.

Because the formula (A18) involves a triple sum over the particle number N , great care should be taken to minimise the required calculations within the sum. Especially the norm calculations of the inter-particle distances Δr_{ij} involved should be handled carefully. There are N^2 such distances at each cycle, a worst-case naive implementation would perform a total of N^4 norm calculations in the double loop and $2N^5$ norm calculations in the triple loop of (A18) if calculated at each loop. A much better approach would be to store the N^2 values of Δr_{ij} in a matrix and simply access the values in the loops. Then one could furthermore note that many of these values would not even have to be recalculated at each cycle – because only one particle is moved at each cycle, indeed only the distances related to this particle need to be updated. Matrix storage like this was implemented for Δr , but also for the vector quantities \vec{q} and $\vec{s}[\Delta r_{ij}] \equiv \vec{s}_{ij}$, as a first optimisation of the code as shown in table 1. The matrix storage is further described in the script comments. When handling matrices in any programming language, one should also keep in mind how matrices are stored in memory and cache. Julia uses the slightly counter-intuitive column-major storage, which means that matrices are stored column-wise in memory. Hence when looping through matrices in Julia, the inner loop should run through the first index of matrices for maximum speed. This is known as *cache-friendly* looping, and was implemented as many places as possible in the script. This is why the matrix quantities are stored as $\Delta R[j, i] = \Delta r_{ij}$ and so on.

The formula (A18) was initially implemented using native vectorised Julia code including index slice access to vectors and the dot product. However, contrary to some other programming languages such as Python, Julia actually optimises loops automatically, and vectorised code often turns out to be slower. Rewriting the vectorised code to plain loops, the simulation ran about thrice as fast for both sampling methods, as apparent in table 1. This lead to quite a speed-up and should be kept in mind for future Julia implementations.

Of course, parallelisation of the script lead to the greatest speed-up of about 3 in the case of random step sampling and 4 in the case of quantum drift sampling. Julia has two native packages for code parallelisation. The `Threads` package sets up T threads on the same computer with shared memory, while the `Distributed` package sets up T separate processes with individ-

ual memory. **Distributed** must hence be used when running calculations on several connected computers. Interestingly, for the simulations of this project the **Distributed** package turned out to be the fastest by far on a single computer as well. This could be due to the shared-memory aspect of **Threads** leading to one thread clearing the cache while another thread is still in need of the loaded numbers – but it might as well be due to bad implementation.

Another observation made during optimisation is that the optimal number of worker threads seemed to be one thread less than the total amount of available threads on the computer. This was the case for both computers used with this project. This makes some sense keeping in mind that there is also a main thread, and although the main thread does not perform as heavy computations as the T worker threads, it would still interfere with the flow of the computations if one of the available threads were to run both the main process and a worker process simultaneously.

4.4 Tuning

The VMC method presented in chapter 3 requires tuning with respect to the gradient descent step size and convergence threshold δv and δg as well as the sampling step size δs . Playing around with the two first, good convergence was found for $\delta v = 0.1$ and $\delta g = 0.01$ in the case of $N = 10$ particles. When increasing to $N = 50$ and $N = 100$ particles however, the gradient descent variation turned out to be more unstable and required a much smaller $\delta v = 0.001$ paired with a larger convergence threshold of $\delta g = 0.1$ to reach convergence at all. Unfortunately, this leads to longer runtime as the script then moves slower through the variational parameter space. When calculating the main results in table 6 in section 5, subsequent gradient descent variations were done with increasing accuracy to reach an adequate optimum. The specific values used during gradient descent for an increasing number of particles can be read from the output included in appendix B. As pointed out in section 3.4, one of the strengths of gradient descent variation is that fewer Monte Carlo cycles can be run at each variational point during descent, and hence the Julia script is set to run a sufficient 10 thousand cycles at each point by default. This number can be specified to something else however, which was done in the last rounds of gradient descent for the main results.

The sampling step size δs sets the average length of each proposed move during the Monte Carlo simulation. Because each move is then judged and accepted or rejected based on the acceptance probability (23) and (34), this step size will directly influence the number of moves accepted. Defining the *acceptance* as the total percentage of accepted moves throughout the simulation, a good way of tuning δs is to consider the acceptance for different values. Choosing `text_output="full"`, the acceptance is printed by the

Julia script, and in table 2 the resulting acceptances for different values of the sampling step size δs is given.

Table 2: Comparison of acceptances for a reference simulation with different values of the sampling step size δs . 'RS' and 'QD' denote random step and quantum drift sampling, respectively.

Step size δs	Acceptance	
	RS	QD
0.1	95%	100%
0.2	89%	100%
0.3	84%	99%
0.4	79%	97%
0.5	74%	94%
0.6	69%	90%
0.7	64%	85%
0.8	59%	77%
0.9	55%	69%

Note that this tuning is not just about getting as high an acceptance as possible. As apparent from the table one could simply set $\delta s = 0.0001$ and obtain 100% acceptance for both sampling methods at all times. But this corresponds to almost not moving the particles at all. Then the configuration would remain approximately the same for the whole simulation, even when running a very high number of Monte Carlo cycles M . In turn, the acceptance ratio of equations (23) and (34) would be about 1 for each proposed move because the probabilities involved would be almost equal – and then of course all moves would be accepted. However since one has essentially sampled the initial configuration M times, one would not obtain a very good energy estimate E_{VMC} .

An ideal acceptance, then, is neither 100% nor 0% but something inbetween. When calculating the results of chapter 5, the value $\delta s = \sqrt{0.4} \approx 0.6325$ was chosen, corresponding to $\delta t = 0.4$. This lead to an acceptance of about 70% for random step sampling and 90% for the more costly quantum drift sampling.

5 Results

Armed with an optimised and tuned Julia script built on a sound theoretical foundation, main results of actual simulations are presented and discussed in this chapter. More results can be found in the **results** folder on GitHub³. The result files follow a naming convention which indicates the system and methods involved. It can be generalised as $\{\text{D}\}\{\text{N}\}(\text{i})(\text{e})\text{-}\{\text{variation}\}\{\text{SAMPLING}\}\{\text{number}\}$, in which **i** is short for "interacting" ($a > 0$) and **e** is short for "elliptical" ($\lambda \neq 1.0$). As two examples, **3D10i_rRS1** denotes the first range-variation, random-step-sampled simulation of a 3-dimensional spherical trap with 10 interacting bosons, while **3D50ie_gdQD2** denotes the second gradient-descent-variation, quantum-drift-sampled simulation of a 3-dimensional elliptical trap with 10 interacting bosons.

³<https://github.com/ffarreke/FYS9411-Project-1>

The script is first benchmarked against exact theoretical results of the non-interacting trap in section 5.1. Then, some results for 3-dimensional traps with 10 bosons are examined in section 5.2. The main results of this report are presented in section 5.3. They are benchmarked and compared mainly to the results of Schøyen and Winther-Larsen[1] from a similar project in 2018, but also the results of DuBois and Glyde[3] by which this whole project was originally inspired.

When not stated otherwise, quantum drift sampling was used. Block resampling was used throughout all the following results.

5.1 Analytical benchmarking

From the analytic expression (A18) for the local energy ε it is easily seen that in the non-interacting case, that is $a = 0$, the whole quantity \vec{s} defined in (A17) vanishes and so only the two first terms of (A18) remain. Furthermore, writing out the quantities U and \vec{q} the second term reduces to

$$\sum_i^N \frac{1}{2} \left(x_i^2 + y_i^2 + \lambda^2 z_i^2 - 4\alpha^2 (x_i^2 + y_i^2 + \beta^2 z_i^2) \right),$$

which means that it, too, vanishes for the optimal values $\alpha = 0.5$ and $\beta = \lambda$. Then, the local energy reduces to a constant, which further means that the actual energy of the state is the same:

$$E = \alpha N \left(D + (\beta - 1) \delta_{D,3} \right). \quad (40)$$

It can actually be shown that the values $\alpha = 0.5$ and $\beta = \lambda$ reduce the trial state (7) to the exact ground state of the system in this case.[1] This analytic result served as a first benchmark for the Julia script. And indeed, running the script for a plethora of non-interacting traps of all dimensions, it produced results that match the formula.

The results are available in the folder `results/analytical benchmarking` on GitHub, but they are not very exciting. Simulations were run for spherical traps with $D \in \{1, 2, 3\}$ dimensions and $N \in \{1, 10, 50, 100\}$ bosons, and results matching (40) with exact precision were obtained from both range-variation, random-step-sampled simulations and gradient-descent-variation, quantum-drift-sampled simulations. Despite in-built block resampling throughout, the statistical errors were never above zero. Hence the Julia script seems to be correctly implemented for non-interacting bosons.

5.2 3-dimensional traps with 10 bosons

Proceeding to check that the script produced expected results with respect to variation, sampling and resampling, four systems of 3-dimensional traps with 10 bosons were simulated. With the naming convention introduced earlier in this chapter, the systems are encoded as 3D10, 3D10e, 3D10i and 3D10ie for readability.

The results of gradient descent variation on all four systems are presented in table 3.

Table 3: Results of gradient descent variation for four 3-dimensional systems with 10 bosons, using step size $\delta v = 0.01$ and threshold $\delta g = 0.001$. The spherical systems 3D10 and 3D10i were varied from initial values $\alpha = 0.4$ and $\beta = 1.1$, while the elliptical systems 3D10e and 3D10ie were varied from initial values $\alpha = 0.4$ and $\beta = 2.8$.

System	Optimal α	Optimal β	VMC energy
3D10	0.4999	1.0003	15.0000 ± 0.0000
3D10i	0.4975	1.0003	15.1527 ± 0.0001
3D10e	0.5001	2.8272	24.1421 ± 0.0000
3D10ie	0.4942	2.8599	24.3975 ± 0.0002

For the non-interacting systems it is reassuring to find that even though gradient descent was deliberately started at a variational point away from the exact optimum, the simulation converges towards the true values $\alpha = 0.5$ and $\beta = 1.0$ or $\beta = \lambda = \sqrt{8} \approx 2.8284$. When it comes to the interacting systems, it is a good sign that the energies are somewhat close to their non-interacting counterparts. Because the characteristic radius $a = 0.0043$ is small and there are few particles in the trap, indeed the interaction should only slightly push the bosons away from the center of the trap, hence increasing the energy just slightly. Note finally that the energy is in general higher for the elliptic trap systems, as the elliptic parameter λ effectively reduces the trap size in the z-direction, making the trap smaller and forcing the bosons against the "potential walls" to a higher degree. The energy results all make physical sense – and comparing the results for 3D10, 3D10e and 3D10ie to [1] they all match the results of Schøyen and Winther-Larsen to 3-digit precision.⁴

Looking at the optimal variational points found with gradient descent in table 3, bosonic interaction seems to *lower* the optimal value of α and *raise* the optimal value of β . The first observation has been made before, and is discussed both in [1] and in the seminal paper [3] by DuBois and Glyde. However the second observation is a novel one. It can not be stated definitely from these few gradient descent results alone, especially not in the spherical case when β is "raised" by the same minuscule amount for both traps considered. But it is an observation to examine further.

Building on the results of table 3, range variation of energy is plotted for the non-interacting system 3D10 in figure 1. As expected, it displays a clear minimum at the optimal value $\alpha = 0.5$. Range variation plotted for the elliptical system 3D10e turned out similarly and is not included here.

⁴Results for system 3D10i are not presented in [1].

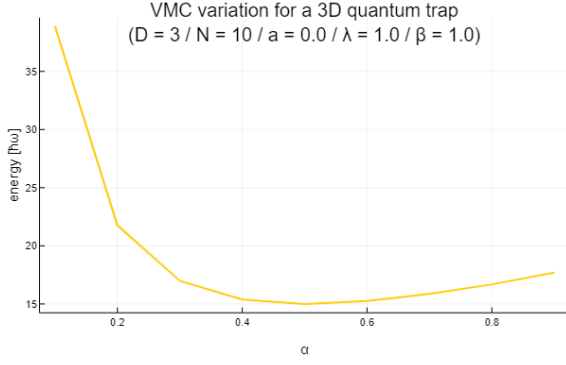


Figure 1: Range variation for system 3D10 around the optimal value for α , with the value for β fixed at the optimal 1.0. As expected, there is a clear minimum at $\alpha = 0.5$.

More interesting are the range variation results for the interacting systems given in figures 2 and 3, which were varied around the optimal values for α and β found with the gradient descent variation above. The resulting values for α , β and E_{VMC} are given in table 4.

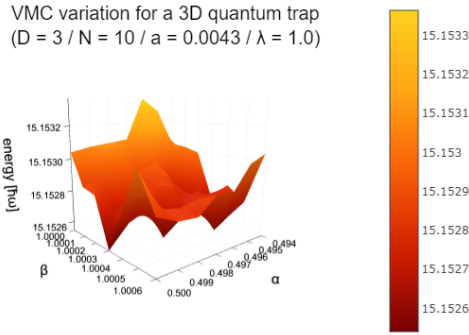


Figure 2: Range variation for the spherical interacting system 3D10i around the optimal values for α and β found with gradient descent variation (table 3). The minimum is less clear in this case.

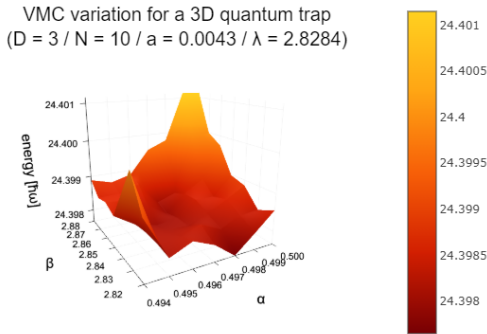


Figure 3: Range variation for the elliptical interacting system 3D10ie around the optimal values for α and β found with gradient descent variation (table 3). The minimum is also here less clear.

Table 4: Results of range variation for the 3-dimensional systems with 10 interacting bosons, based on the gradient descent results from table 3.

System	Optimal α	Optimal β	VMC energy
3D10i	0.4970	1.0006	15.1525 ± 0.0001
3D10ie	0.4940	2.8600	24.3976 ± 0.0007

The found VMC energies are almost the same as the ones found with gradient descent, but this is due to the energy being approximately the same in the whole variational areas on the plots as much as due to anything else; while the range variation for 3D10ie finds approximately the same minimum to 3-digit precision, the range variation for 3D10i actually finds a *new* minimum when compared to the results from gradient descent variation in table 3. Furthermore both plots reveal a chaotic energy surface across α and β , indicating that gradient descent could easily get caught in local minima for these systems. Hence, the apparent raising of β discussed above might just be due to the numerical imprecision which is clearly in play here. The effect of bosonic interaction on the optimal values of α and β will be examined further in section 5.3, in which elliptical traps with 10, 50 and 100 interacting bosons are simulated.

As another test of expected results, the convergence of sampling methods for the interacting systems are plotted and compared in figures 4 and 5. In both plots it is clear that quantum drift sampling converges faster than random step sampling. Already at $M = 10^4$ Monte Carlo cycles, the QD-sampled VMC energy has converged pretty well. Looking also at the statistical error ribbons, the (block-resampled) error becomes smaller in the case of QD sampling when compared to RS sampling for $M > 10^3$. It hence seems that the quantum drift method indeed samples the important parts of configuration space earlier on, as discussed in section 33.

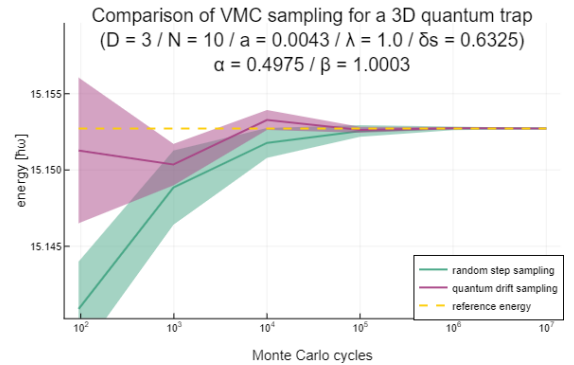


Figure 4: Comparison of sampling methods for the spherical interacting system (3D10i) at the optimal values of α and β found with gradient descent variation (table 3).

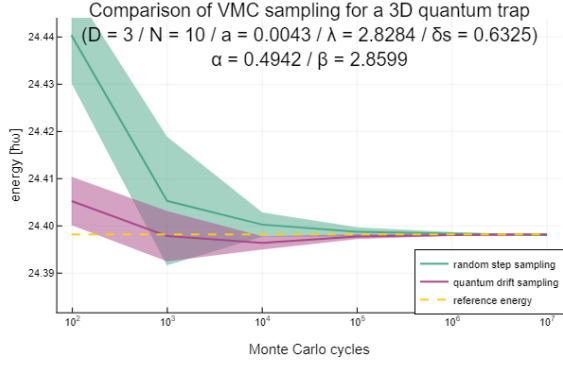


Figure 5: Comparison of sampling methods for the elliptical interacting system (3D10ie) around the optimal values for α and β found with gradient descent variation (table 3).

Convergence of sampling methods for the non-interacting system 3D10 is plotted in figure 6. As expected after the discussion in section 5.1, both methods show definite convergence with no error, because the local energy is simply constant in this case. The plot for 3D10e turned out similarly.⁵

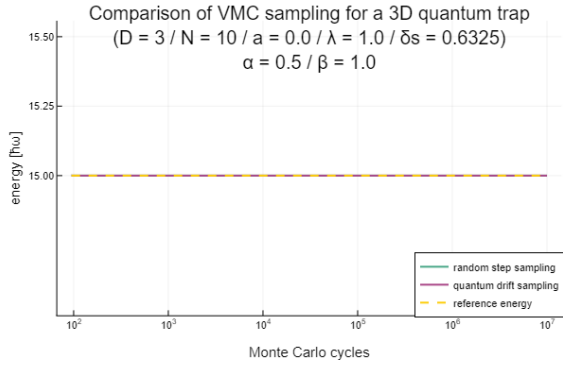


Figure 6: Comparison of sampling methods for the spherical non-interacting system (3D10) around the optimal values for α and β . Both methods converge from start.

As a conclusion to this section, block resampling of statistical error for system 3D10ie is plotted in figure 7. The resampled error grows with the number of blockings until reaching a maximum (in this case at 8), just as described in section 3.5.

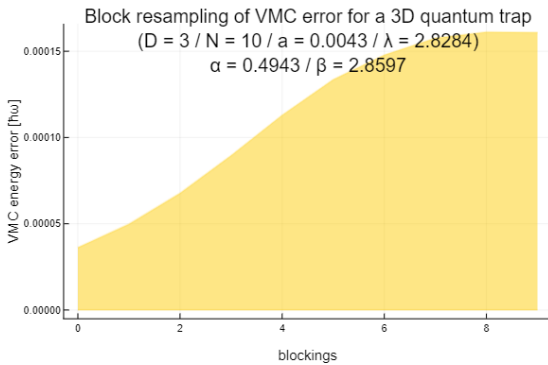


Figure 7: Block resampling of the statistical error for the elliptical interacting system (3D10ie).

Having obtained output from the script which is mostly as expected, the main results of this report are presented in the next section.

5.3 3-dimensional elliptical traps with 10, 50 and 100 interacting bosons

The gradient descent results for 3D traps with 10 bosons in table 3 seemed to indicate that the optimal value for the elliptic variational parameter β is raised when particle interaction is introduced to the system. The lowering of the optimal value for α is reported [1], but since they did not vary β , instead keeping it constant at $\beta = \lambda = \sqrt{8} \approx 2.8284$, they cannot strengthen the claim. Furthermore, the variational plots of figures 2 and 3 display a chaotic energy surface for interacting systems which indicates the possibility of being trapped in a local minimum which could in the worst scenario be a result of statistical or numerical imprecision, rather than being a true minimum at all.

To examine the impact of particle interaction on β , elliptical traps with 10, 50 and 100 (hereafter referred to as 3D10ie, 3D50ie and 3D100ie) interacting bosons are considered in deeper detail. The results of Schøyen and Winther-Larsen for these systems are presented in table 5.

Table 5: Results of Schøyen and Winther-Larsen[1] for elliptical interacting systems, varying only the variational parameter α .

System	VMC energy
3D10ie	24.3986 ± 0.0003
3D50ie	127.2852 ± 0.0077
3D100ie	266.2904 ± 0.0297

Starting at the optimal variational point of [1], which was found to be $\alpha = 0.49744$ and $\beta = \lambda = \sqrt{8} \approx 2.8284$, the three systems were gradient descended with the Julia script, varying both α and β to search for an even lower VMC energy. All systems were gradient descended using the initial values $\delta s = 0.0001$ and $\delta g = 0.1$ given in section 4.4. As expected, the larger systems 3D50ie and 3D100ie were found to be more unstable with respect to gradient descent; while the smallest system was varied 4 successive times with increasing precision in δv and δg , the larger systems were only varied 3 and 2 successive times, respectively, as gradient descent variation started to diverge. The full Julia output from the search is included in appendix B.2, but the results are concisely summed up in table 6. Interestingly, the raising of β is demonstrated also in these results, and furthermore the raising increases with the number of bosons N . In fact, the relative raising of β seems to be approximately equal to the relative lowering of α when compared to the non-interacting optimal values $\alpha = 0.5$ and $\beta = 2.8284$, with respective relative changes 1%, 5% and 9% for 10, 50 and 100 bosons.

⁵Actually, the plot for 3D10e turned out almost similarly, except for some strange numerical behaviour for some values of M , as can be seen in the folder `results` on GitHub. This is thought to be due to catastrophic cancellation or something of the kind.

Table 6: Results of subsequent gradient descent for elliptical interacting systems, varying both variational parameters α and β . Convergence plots and full Julia output of the search is included in appendix B.

System	Optimal α	Optimal β	VMC energy
3D10ie	0.4943	2.8597	24.3979 ± 0.0000
3D50ie	0.4743	2.9813	127.2168 ± 0.0004
3D100ie	0.4581	3.0957	265.9246 ± 0.0023

The results of table 6 were controlled by running a final 100 million Monte Carlo cycles at the found optimal point, as shown in the output of appendix . Convergence plots for all three values are given in section B.1, in which it becomes clear that the VMC energy converges already at 100 thousand Monte Carlo cycles for all three systems. This strengthens the validity of the results, as the last gradient descent was run with 100 thousand Monte Carlo cycles at each variational point. It also indicates that it is sufficient really to run only $M = 10^6$ Monte Carlo cycles in future simulations of these systems, which would of course greatly reduce the computation time.

As one last verification of the Julia script developed here, simulations for all three systems were also run at the optimal point $\alpha = 0.49744$ and $\beta = \lambda = \sqrt{8}$ found by Schøyen and Winther-Larsen, with the same number of Monte Carlo cycles, $M = 2^{21}$ and a sampling step size $\delta s = \sqrt{0.1}$ corresponding to their $\delta t = 0.1$. These parametric values are shown in table 7 to reproduce their energy estimates to 1-digit precision, statistical errors also matching in magnitude.

Table 7: Reproduction of the results of Schøyen and Winther-Larsen[1] for elliptical interacting systems, using the same parametric values as them.

System	VMC energy
3D10ie	24.3986 ± 0.0003
3D50ie	127.2818 ± 0.0063
3D100ie	266.3434 ± 0.0328

Having demonstrated overall consistent results with [1], one should of course also benchmark against other work, but there are not too many other viable results from the exact system of this report to be found. The seminal paper by DuBois and Glyde[3] considers a greater range of systems, and does not explicitly present results for the three systems considered in this section. However, rescaling their figure 2 as done below, one clearly sees that the results found in this report, as well as the report of Schøyen and Winther-Larsen[1], both fall within the discernible range of their calculations.

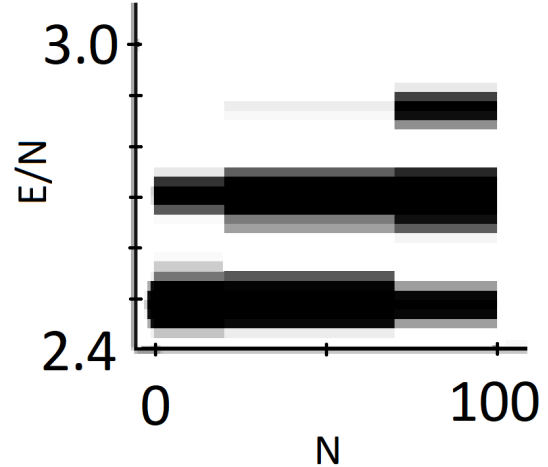


Figure 8: A rescaled presentation of the results of DuBois and Glyde[3]. Their results discernably matches the results of this report as well as [1].

Note that the energy in figure 8 is given per particle, so it should be multiplied respectively by 10, 50 and 100. In [3] they used a Variational Monte Carlo approach similar to the one presented here, and it is reassuring to see that they obtain results of a similar scale to the results in tables 5 and 6.

6 Conclusion

In this report, the theoretical and computational development of a Variational Monte Carlo script written in the Julia language was presented, and it was run to both reproduce well-known results and produce novel results regarding the elliptic variational parameter β . For elliptical traps with 10, 50 and 100 interacting bosons, varying both α and β resulted in slightly lower energy estimates than only varying α as done by Schøyen and Winther-Larsen. This lowering encourages further analysis of the optimal β which is demonstrated here to be raised by introducing particle interaction to the system. However as the change in optimal β is relatively small for the numbers of bosons considered here, the results of this report also validates the usual approximation $\beta = \lambda$ at least for systems with fewer than 100 particles.

Computational optimisation was done to the written Julia script with respect to storage, memory-handling and parallelisation, resulting in a total speed-up of more than 14. As the script was developed as a first-time experience with the Julia language, there is sure to be many more optimisations which can further increase the speed of simulations. That said however, Julia performance better than C++ has seldom been reported, and only in special cases.

Despite the Julia script of this project being written with a specific bosonic harmonic trap system in mind, one could fairly easily extend it to perform Variational Monte Carlo on a broader range of quantum systems. To this end, numerical differentiation could be implemented to calculate the local energy for an arbitrary

system, although as demonstrated in [1] such calculations are prone to be very slow compared to analytical calculations such as the one derived in this report. The script could also be extended to sample and estimate other system observables of interest.

In [8], the very system considered in this report was treated, analytically and numerically, using Gross-Pitaevskii minimisation as well as Variational Monte Carlo, to obtain results for the ground state and vortex states of an elliptical trap with $N = 500$ interacting bosons with a large characteristic radius of $a = 0.15155$. Were the Julia script of this project to

be further optimised or run on a supercomputer, it could produce results for their trap system which could both verify their ground state estimates, but also lower the estimates further by raising β as demonstrated.

While the script is ready for new adventures, the main goal of this project was to get acquainted with the Variational Monte Carlo method, computational optimisation as well as the Julia language. In that regard it was a success. Braced and inspired by the computational lessons learned here, I will probably use both VMC and Julia again in future research.

References

- [1] Øyvind S. Schøyen and Sebastian G. Winther-Larsen. Variational Monte Carlo on Bosonic Systems. Project report in FYS4411 – Computational Physics II, University of Oslo, April 2018.
- [2] M. Anderson, J. Ensher, M. Matthews, C. Wieman, and E Cornell. Observation of Bose-Einstein Condensation in a Dilute Atomic Vapor. *Science*, 269(5221):198–201, July 1995.
- [3] J. L. DuBois and H. R. Glyde. Bose-Einstein condensation in trapped bosons: A variational monte carlo analysis. *Physical Review A*, 63(2), Jan 2001.
- [4] H. M. J. M. Boesten, C. C. Tsai, J. R. Gardner, D. J. Heinzen, and B. J. Verhaar. Observation of a shape resonance in the collision of two cold ^{87}Rb atoms. *Physical Review A*, 55:636–640, Jan 1997.
- [5] Nicholas Metropolis, Arianna W. Rosenbluth, Marshall N. Rosenbluth, Augusta H. Teller, and Edward Teller. Equation of State Calculations by Fast Computing Machines. *The Journal of Chemical Physics*, 21(6):1087–1092, 1953.
- [6] Jeff Bezanson, Stefan Karpinski, Viral B. Shah, and Alan Edelman. Why We Created Julia, Feb 2012. Electronic article at <https://julialang.org/blog/2012/02/why-we-created-julia/>. Accessed on 02.06.2021.
- [7] F. Dalfovo and S. Stringari. Bosons in anisotropic traps: ground state and vortices. *Physical Review A*, 53(4):2477–2485, Apr 1996.
- [8] J. K. Nilsen, J. Mur-Petit, M. Guilleumas, M. Hjorth-Jensen, and A. Polls. Vortices in atomic Bose-Einstein condensates in the large-gas-parameter region. *Physical Review A*, 71(5), May 2005.

APPENDIX

A Calculations

Some of the longer analytical calculations of the project are presented here.

A.1 Local energy ε

Equation (11) defines the local energy ε through the Hamiltonian H given in (4) and the trial wavefunction Ψ given in (7). The potential terms U og V in the Hamiltonian give direct contributions to the local energy, but the kinetic term with its second derivative $\vec{\nabla}_i^2 \Psi$ needs to be considered further. To this end, write

$$\Psi[\vec{R}] = G[\vec{R}]F[\vec{R}] \quad (\text{A1})$$

with

$$G[\vec{R}] = \prod_i^N g[\vec{r}_i] \quad (\text{A2})$$

$$F[\vec{R}] = \prod_j^N \prod_{k>j}^N f[\Delta r_{jk}] \quad (\text{A3})$$

and note that

$$\vec{\nabla}_i^2 \Psi = \vec{\nabla}_i \cdot \left(\vec{\nabla}_i G F + G \vec{\nabla}_i F \right) = \vec{\nabla}_i^2 G F + 2 \vec{\nabla}_i G \cdot \vec{\nabla}_i F + G \vec{\nabla}_i^2 F. \quad (\text{A4})$$

Hence the derivatives of both G and F must be calculated in order to get an analytical expression for the local energy ε . The derivatives of G are

$$\begin{aligned} \vec{\nabla}_i G &= \vec{\nabla}_i g[\vec{r}_i] \prod_{j \neq i}^N g[\vec{r}_j] \\ \implies \vec{\nabla}_i^2 G &= \vec{\nabla}_i^2 g[\vec{r}_i] \prod_{j \neq i}^N g[\vec{r}_j], \end{aligned}$$

and because the derivatives of g are

$$\vec{\nabla}_i g[\vec{r}_i] = -2\alpha \left(\vec{x}_i + \vec{y}_i + \beta \vec{z}_i \right) g[\vec{r}_i] \quad (\text{A5})$$

$$\begin{aligned} \implies \vec{\nabla}_i^2 g[\vec{r}_i] &= -2\alpha \left(\left(D + (\beta - 1)\delta_{D,3} \right) g[\vec{r}_i] + \left(\vec{x}_i + \vec{y}_i + \beta \vec{z}_i \right) \cdot \vec{\nabla}_i g[\vec{r}_i] \right) \\ &= 2\alpha \left(2\alpha \left(\vec{x}_i + \vec{y}_i + \beta \vec{z}_i \right)^2 - \left(D + (\beta - 1)\delta_{D,3} \right) \right) g[\vec{r}_i] \end{aligned} \quad (\text{A6})$$

where δ is the Kronecker delta, these are quite simply given by

$$\vec{\nabla}_i G = -2\alpha \left(\vec{x}_i + \vec{y}_i + \beta \vec{z}_i \right) G, \quad (\text{A7})$$

$$\vec{\nabla}_i^2 G = 2\alpha \left(2\alpha \left(\vec{x}_i + \vec{y}_i + \beta \vec{z}_i \right)^2 - \left(D + (\beta - 1)\delta_{D,3} \right) \right) G. \quad (\text{A8})$$

To find the derivatives of F , rewrite the product in (A3) to the exponential form

$$F[\vec{R}] = \epsilon^{\sum_j^N \sum_{k>j}^N \ln f[\Delta r_{jk}]} \quad (\text{A9})$$

and note that

$$\begin{aligned}
\vec{\nabla}_i F &= \sum_{j \neq i}^N \left(\frac{\vec{\nabla}_i f[\Delta r_{ij}]}{f[\Delta r_{ij}]} \right) F \\
\Rightarrow \vec{\nabla}_i^2 F &= \sum_{j \neq i}^N \left(\vec{\nabla}_i \cdot \left[\frac{\vec{\nabla}_i f[\Delta r_{ij}]}{f[\Delta r_{ij}]} \right] \right) F + \sum_{j \neq i}^N \left(\frac{\vec{\nabla}_i f[\Delta r_{ij}]}{f[\Delta r_{ij}]} \right) \cdot \vec{\nabla}_i F \\
&= \sum_{j \neq i}^N \left(\frac{\vec{\nabla}_i^2 f[\Delta r_{ij}]}{f[\Delta r_{ij}]} - \frac{(\vec{\nabla}_i f[\Delta r_{ij}])^2}{f^2[\Delta r_{ij}]} \right) F + \sum_{j \neq i}^N \sum_{k \neq i}^N \left(\frac{\vec{\nabla}_i f[\Delta r_{ij}] \cdot \vec{\nabla}_i f[\Delta r_{ik}]}{f[\Delta r_{ij}] f[\Delta r_{ik}]} \right) F \\
&= \sum_{j \neq i}^N \left(\frac{\vec{\nabla}_i^2 f[\Delta r_{ij}]}{f[\Delta r_{ij}]} + \sum_{\substack{k \neq i \\ k \neq j}}^N \frac{\vec{\nabla}_i f[\Delta r_{ij}] \cdot \vec{\nabla}_i f[\Delta r_{ik}]}{f[\Delta r_{ij}] f[\Delta r_{ik}]} \right) F.
\end{aligned}$$

The derivatives of f are

$$\vec{\nabla}_i f[\Delta r_{ij}] = \frac{a \Delta \vec{r}_{ij}}{\Delta r_{ij}^3} \quad (\text{A10})$$

$$\begin{aligned}
\Rightarrow \vec{\nabla}_i^2 f[\Delta r_{ij}] &= \frac{aD}{\Delta r_{ij}^3} - \frac{3a \Delta \vec{r}_{ij}^2}{\Delta r_{ij}^5} \\
&= \frac{a(D-3)}{\Delta r_{ij}^3} \quad (\text{A11})
\end{aligned}$$

as long as all $\Delta r_{ij} > a$. Now since, by the definition (9),

$$\Delta r_{ij} f[\Delta r_{ij}] = \Delta r_{ij} - a$$

it follows that

$$\vec{\nabla}_i F = \sum_{j \neq i}^N \left(\frac{a \Delta \vec{r}_{ij}}{\Delta r_{ij}^2 (\Delta r_{ij} - a)} \right) F, \quad (\text{A12})$$

$$\vec{\nabla}_i^2 F = \sum_{j \neq i}^N \left(\frac{a(D-3)}{\Delta r_{ij}^2 (\Delta r_{ij} - a)} + \sum_{\substack{k \neq i \\ k \neq j}}^N \frac{a^2 \Delta \vec{r}_{ij} \cdot \Delta \vec{r}_{ik}}{\Delta r_{ij}^2 (\Delta r_{ij} - a) \Delta r_{ik}^2 (\Delta r_{ik} - a)} \right) F. \quad (\text{A13})$$

Then in total equation (A4) takes the form

$$\begin{aligned}
\vec{\nabla}_i^2 \Psi &= 2\alpha \left(2\alpha (\vec{x}_i + \vec{y}_i + \beta \vec{z}_i)^2 - (D + (\beta - 1)\delta_{D,3}) \right) \Psi \\
&\quad - 4\alpha (\vec{x}_i + \vec{y}_i + \beta \vec{z}_i) \cdot \sum_{j \neq i}^N \left(\frac{a \Delta \vec{r}_{ij}}{\Delta r_{ij}^2 (\Delta r_{ij} - a)} \right) \Psi \\
&\quad + \sum_{j \neq i}^N \left(\frac{a(D-3)}{\Delta r_{ij}^2 (\Delta r_{ij} - a)} + \sum_{\substack{k \neq i \\ k \neq j}}^N \frac{a^2 \Delta \vec{r}_{ij} \cdot \Delta \vec{r}_{ik}}{\Delta r_{ij}^2 (\Delta r_{ij} - a) \Delta r_{ik}^2 (\Delta r_{ik} - a)} \right) \Psi \\
\Rightarrow \frac{\vec{\nabla}_i^2 \Psi}{\Psi} &= 2\alpha \left(2\alpha (\vec{x}_i + \vec{y}_i + \beta \vec{z}_i)^2 - (D + (\beta - 1)\delta_{D,3}) \right) \\
&\quad + \sum_{j \neq i}^N \frac{a(D-3) - 4\alpha (\vec{x}_i + \vec{y}_i + \beta \vec{z}_i) \cdot a \Delta \vec{r}_{ij}}{\Delta r_{ij}^2 (\Delta r_{ij} - a)} \\
&\quad + \sum_{j \neq i}^N \sum_{\substack{k \neq i \\ k \neq j}}^N \frac{a^2 \Delta \vec{r}_{ij} \cdot \Delta \vec{r}_{ik}}{\Delta r_{ij}^2 (\Delta r_{ij} - a) \Delta r_{ik}^2 (\Delta r_{ik} - a)} \quad (\text{A14})
\end{aligned}$$

and by introducing the quantities

$$\vec{q}[\vec{r}_i] = -4\alpha(\vec{x}_i + \vec{y}_i + \beta\vec{z}_i), \quad (\text{A15})$$

$$d[\Delta r_{ij}] = \Delta r_{ij}^2(\Delta r_{ij} - a), \quad (\text{A16})$$

$$\vec{s}[\Delta \vec{r}_{ij}] = \frac{a\Delta \vec{r}_{ij}}{2d[\Delta r_{ij}]}, \quad (\text{A17})$$

it follows that the total analytic expression for the local energy is

$$\begin{aligned} \varepsilon[\vec{R}] = & \alpha N \left(D + (\beta - 1)\delta_{D,3} \right) + \sum_i^N \frac{1}{2} \left(U[\vec{r}_i] - \frac{1}{4}\vec{q}^2[\vec{r}_i] \right) \\ & - \sum_i^N \sum_{j \neq i}^N \left(\frac{a(D-3)}{2d[\Delta r_{ij}]} + \vec{q}[\vec{r}_i] \cdot \vec{s}[\Delta \vec{r}_{ij}] \right) - 2 \sum_i^N \sum_{j \neq i}^N \sum_{\substack{k \neq i \\ k \neq j}}^N \vec{s}[\Delta \vec{r}_{ij}] \cdot \vec{s}[\Delta \vec{r}_{ik}] \end{aligned} \quad (\text{A18})$$

as long as all $\Delta r_{ij} > a$. This last restriction is why the hard-sphere interaction energy V is omitted altogether; for configurations in which $\Delta r_{ij} \leq a$ for some i and j , the trial wavefunction Ψ in (7) is defined to be zero (because $f[\Delta r_{ij}]$ is zero), which in turn means that the Metropolis algorithm will never sample such configurations, and so the fact that the local energy would be infinite for such configurations (because V is infinite) is not a problem.

A.2 Quantum drift \vec{Q}

Equation (33) defines the quantum drift of the system. From the formulas (A7) and (A12) of section A.1 it follows that

$$\begin{aligned} \vec{\nabla}_i \Psi &= \vec{\nabla}_i G F + G \vec{\nabla}_i F \\ &= -2\alpha(\vec{x}_i + \vec{y}_i + \beta\vec{z}_i) \Psi + \sum_{j \neq i}^N \left(\frac{a\Delta \vec{r}_{ij}}{\Delta r_{ij}^2(\Delta r_{ij} - a)} \right) \Psi \\ \Rightarrow \frac{\vec{\nabla}_i \Psi}{\Psi} &= -2\alpha(\vec{x}_i + \vec{y}_i + \beta\vec{z}_i) + \sum_{j \neq i}^N \left(\frac{a\Delta \vec{r}_{ij}}{\Delta r_{ij}^2(\Delta r_{ij} - a)} \right), \end{aligned} \quad (\text{A19})$$

which means that the analytic expression for the quantum drift becomes

$$\vec{Q}_i[\vec{R}] = \vec{q}[\vec{r}_i] + 4 \sum_{j \neq i}^N \vec{s}[\Delta \vec{r}_{ij}]. \quad (\text{A20})$$

with the quantities introduced in (A15)–(A17).

A.3 Variational gradient $\frac{\partial \langle \varepsilon \rangle_{\Pi}}{\partial \alpha_i}$

The energy bound $\langle \varepsilon \rangle_{\Pi}$ introduced in section 3.1 is given by

$$\langle \varepsilon \rangle_{\Pi} = \frac{\int \dots \int \Psi^2[\vec{R}] \varepsilon[\vec{R}] \delta^{DN} R}{\int \dots \int \Psi^2[\vec{R}] \delta^{DN} R} \quad (\text{A21})$$

in the case of a real trial state Ψ such as the one defined in (7) and considered here. Denoting the variational parameters as α_i for $i \in \{1, 2\}$ such that $\alpha_1 = \alpha$ and $\alpha_2 = \beta$, it is relevant for gradient descent variation as discussed in section 3.4 to calculate the variational gradient $\frac{\partial \langle \varepsilon \rangle_{\Pi}}{\partial \alpha_i}$. As both Ψ and ε depends on α_i , the gradient is

$$\begin{aligned} \frac{\partial \langle \varepsilon \rangle_{\Pi}}{\partial \alpha_i} &= \frac{\int \dots \int \left(2\Psi \frac{\partial \Psi}{\partial \alpha_i} \varepsilon + \Psi^2 \frac{\partial \varepsilon}{\partial \alpha_i} \right) \delta^{DN} R}{\int \dots \int \Psi^2 \delta^{DN} R} - \frac{\int \dots \int \Psi^2 \varepsilon \delta^{DN} R}{\int \dots \int \Psi^2 \delta^{DN} R} \frac{\int \dots \int 2\Psi \frac{\partial \Psi}{\partial \alpha_i} \delta^{DN} R}{\int \dots \int \Psi^2 \delta^{DN} R} \\ &= \frac{\int \dots \int \left(2\Psi^2 \frac{\partial \Psi}{\Psi \partial \alpha_i} \varepsilon + \Psi^2 \frac{\partial \varepsilon}{\partial \alpha_i} \right) \delta^{DN} R}{\int \dots \int \Psi^2 \delta^{DN} R} - \frac{\int \dots \int \Psi^2 \varepsilon \delta^{DN} R}{\int \dots \int \Psi^2 \delta^{DN} R} \frac{\int \dots \int 2\Psi^2 \frac{\partial \Psi}{\Psi \partial \alpha_i} \delta^{DN} R}{\int \dots \int \Psi^2 \delta^{DN} R} \\ &= \int \dots \int \left(2\Pi \frac{\partial \ln \Psi}{\partial \alpha_i} \varepsilon + \Pi \frac{\partial \varepsilon}{\partial \alpha_i} \right) \delta^{DN} R - \int \dots \int \Pi \varepsilon \delta^{DN} R \int \dots \int 2\Pi \frac{\partial \ln \Psi}{\partial \alpha_i} \delta^{DN} R \\ &= 2 \left\langle \frac{\partial \ln \Psi}{\partial \alpha_i} \varepsilon \right\rangle_{\Pi} - 2 \left\langle \frac{\partial \ln \Psi}{\partial \alpha_i} \right\rangle_{\Pi} \langle \varepsilon \rangle_{\Pi} + \left\langle \frac{\partial \varepsilon}{\partial \alpha_i} \right\rangle_{\Pi}. \end{aligned} \quad (\text{A22})$$

However the expected value of the variational derivative $\frac{\partial \varepsilon}{\partial \alpha_i}$ turns out to be zero,

$$\begin{aligned} \left\langle \frac{\partial \varepsilon}{\partial \alpha_i} \right\rangle_{\Pi} &= \frac{\int \dots \int \Psi^2 \frac{\partial}{\partial \alpha_i} \left[\frac{H[\Psi]}{\Psi} \right] \delta^{DN} R}{\int \dots \int \Psi^2 \delta^{DN} R} \\ &= \frac{\int \dots \int \left(H \left[\frac{\partial \Psi}{\partial \alpha_i} \right] \Psi - H[\Psi] \frac{\partial \Psi}{\partial \alpha_i} \right) \delta^{DN} R}{\int \dots \int \Psi^2 \delta^{DN} R} = \frac{\langle \Psi | \mathbf{H} \left| \frac{\partial \Psi}{\partial \alpha_i} \right\rangle - \langle \Psi | \mathbf{H}^\dagger \left| \frac{\partial \Psi}{\partial \alpha_i} \right\rangle}{\langle \Psi | \Psi \rangle} = 0, \end{aligned} \quad (\text{A23})$$

so the variational gradient reduces to

$$\frac{\partial \langle \varepsilon \rangle_{\Pi}}{\partial \alpha_i} = 2 \left\langle \frac{\partial \ln \Psi}{\partial \alpha_i} \varepsilon \right\rangle_{\Pi} - 2 \left\langle \frac{\partial \ln \Psi}{\partial \alpha_i} \right\rangle_{\Pi} \langle \varepsilon \rangle_{\Pi}. \quad (\text{A24})$$

The expected values in this expression can be estimated by sampling the quantities

$$\frac{\partial \ln \Psi}{\partial \alpha} [\vec{R}] = - \sum_i^N (x_i^2 + y_i^2 + \beta z_i^2), \quad (\text{A25})$$

$$\frac{\partial \ln \Psi}{\partial \beta} [\vec{R}] = - \sum_i^N \alpha z_i^2, \quad (\text{A26})$$

as well as their product with the local energy ε , at each Monte Carlo cycle and then calculate their average when the averages of $\langle \varepsilon \rangle$ and $\langle \varepsilon^2 \rangle$ are calculated.

B Output

Some of the output from the Julia script developed for the project is included here for reference. The rest can be found in the folder `results` on GitHub⁶.

B.1 Convergence of results for the systems 3D10ie, 3D50ie and 3D100ie

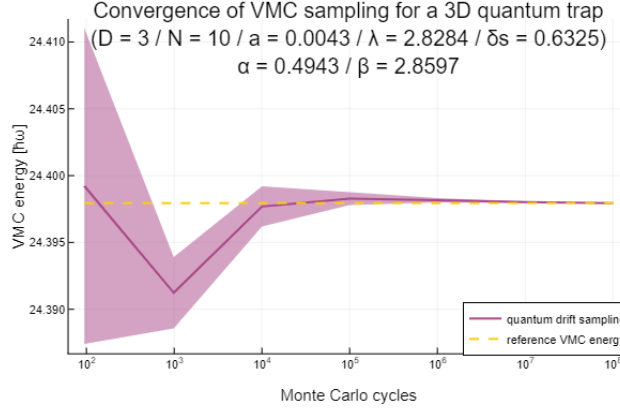


Figure 9: Convergence of VMC energy for the elliptical trap with 10 interacting bosons (3D10ie).

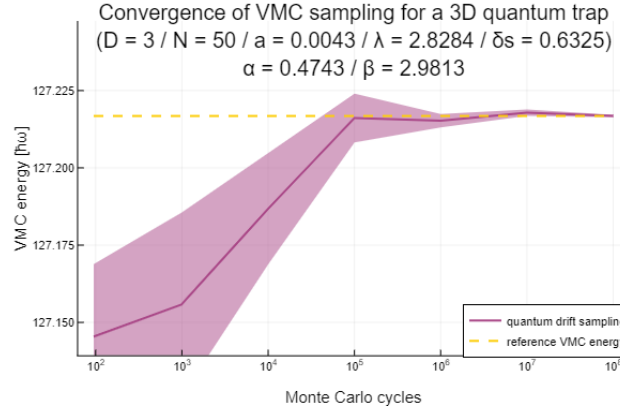


Figure 10: Convergence of VMC energy for the elliptical trap with 50 interacting bosons (3D50ie).

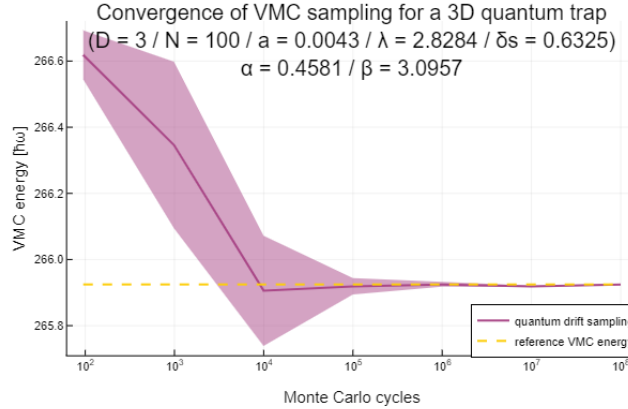


Figure 11: Convergence of VMC energy for the elliptical trap with 100 interacting bosons (3D100ie).

⁶<https://github.com/flarrek/FYS9411-Project-1>

B.2 Gradient descent variation of the systems 3D10ie, 3D50ie and 3D100ie

```
julia> @time find_VMC_energy(QuantumTrap(3,10);as=[0.49744])

Finding the VMC energy for an elliptical 3D quantum trap with 10 interacting particles.

Quantum trap parameters: D = 3 / N = 10 / a = 0.0043 /  $\lambda$  = 2.8284
Algorithm methods: Gradient descent variation / Normal scattering / Quantum drift sampling
Algorithm parameters:  $\delta v$  = 0.0001 /  $\delta g$  = 0.1 /  $\delta s$  = 0.6325

Gradient descending from the initial variational point ( $\alpha$  = 0.4974 /  $\beta$  = 2.8284),
running 9994 Monte Carlo cycles at each point ...
Running 999989 Monte Carlo cycles at the final variational point ( $\alpha$  = 0.4974 /  $\beta$  = 2.8284) ...
VMC simulation finished!

Optimal  $\alpha$ : 0.4974
Optimal  $\beta$ : 2.8284
VMC energy: 24.3983  $\pm$  0.0003

2.474557 seconds (14.87 k allocations: 116.097 MiB, 64.44% gc time)

julia> @time find_VMC_energy(QuantumTrap(3,10);as=[0.4974], $\beta$ s=[2.8284], $\delta v$ =0.001, $\delta g$ =0.01)

Finding the VMC energy for an elliptical 3D quantum trap with 10 interacting particles.

Quantum trap parameters: D = 3 / N = 10 / a = 0.0043 /  $\lambda$  = 2.8284
Algorithm methods: Gradient descent variation / Normal scattering / Quantum drift sampling
Algorithm parameters:  $\delta v$  = 0.001 /  $\delta g$  = 0.01 /  $\delta s$  = 0.6325

Gradient descending from the initial variational point ( $\alpha$  = 0.4974 /  $\beta$  = 2.8284),
running 9994 Monte Carlo cycles at each point ...
Running 999989 Monte Carlo cycles at the final variational point ( $\alpha$  = 0.4958 /  $\beta$  = 2.8464) ...
VMC simulation finished!

Optimal  $\alpha$ : 0.4958
Optimal  $\beta$ : 2.8464
VMC energy: 24.3982  $\pm$  0.0003

829.111017 seconds (6.90 M allocations: 1008.172 MiB, 95.59% gc time)

julia> @time find_VMC_energy(QuantumTrap(3,10);as=[0.4958], $\beta$ s=[2.8464], $\delta v$ =0.01, $\delta g$ =0.001)

Finding the VMC energy for an elliptical 3D quantum trap with 10 interacting particles.

Quantum trap parameters: D = 3 / N = 10 / a = 0.0043 /  $\lambda$  = 2.8284
Algorithm methods: Gradient descent variation / Normal scattering / Quantum drift sampling
Algorithm parameters:  $\delta v$  = 0.01 /  $\delta g$  = 0.001 /  $\delta s$  = 0.6325

Gradient descending from the initial variational point ( $\alpha$  = 0.4958 /  $\beta$  = 2.8464),
running 9994 Monte Carlo cycles at each point ...
Running 999989 Monte Carlo cycles at the final variational point ( $\alpha$  = 0.4945 /  $\beta$  = 2.8584) ...
VMC simulation finished!

Optimal  $\alpha$ : 0.4945
Optimal  $\beta$ : 2.8584
VMC energy: 24.3976  $\pm$  0.0002

175.911023 seconds (1.46 M allocations: 302.958 MiB, 95.31% gc time)

julia> @time find_VMC_energy(QuantumTrap(3,10),[10^5,10^6];as=[0.4945], $\beta$ s=[2.8584], $\delta v$ =0.01, $\delta g$ =0.001)

Finding the VMC energy for an elliptical 3D quantum trap with 10 interacting particles.

Quantum trap parameters: D = 3 / N = 10 / a = 0.0043 /  $\lambda$  = 2.8284
Algorithm methods: Gradient descent variation / Normal scattering / Quantum drift sampling
Algorithm parameters:  $\delta v$  = 0.01 /  $\delta g$  = 0.001 /  $\delta s$  = 0.6325

Gradient descending from the initial variational point ( $\alpha$  = 0.4945 /  $\beta$  = 2.8584),
running 9997 Monte Carlo cycles at each point ...
Running 999989 Monte Carlo cycles at the final variational point ( $\alpha$  = 0.4943 /  $\beta$  = 2.8597) ...
VMC simulation finished!

Optimal  $\alpha$ : 0.4943
Optimal  $\beta$ : 2.8597
VMC energy: 24.3976  $\pm$  0.0002

79.456373 seconds (676.69 k allocations: 657.257 MiB, 85.98% gc time)

julia> @time find_VMC_energy(QuantumTrap(3,10),[10^e for e in 2:8];as=[0.4943], $\beta$ s=[2.8597],variation="range",plot_output="convergence")

Finding the VMC energy for an elliptical 3D quantum trap with 10 interacting particles.

Quantum trap parameters: D = 3 / N = 10 / a = 0.0043 /  $\lambda$  = 2.8284
Algorithm methods: Range variation / Normal scattering / Quantum drift sampling
Algorithm parameters:  $\delta s$  = 0.6325

Running 9999983 Monte Carlo cycles at the variational point ( $\alpha$  = 0.4943 /  $\beta$  = 2.8597) ...
VMC simulation finished!

Optimal  $\alpha$ : 0.4943
Optimal  $\beta$ : 2.8597
VMC energy: 24.3979  $\pm$  0.0

95.581292 seconds (85.96 k allocations: 8.716 GiB, 1.65% gc time, 0.02% compilation time)
```

Figure 12: Output from gradient descent variation and subsequent refinement of the elliptical trap with 10 interacting bosons (3D10ie).

```

julia> @time find_VMC_energy(QuantumTrap(3,50);as=[0.49744])

Finding the VMC energy for an elliptical 3D quantum trap with 50 interacting particles.

Quantum trap parameters: D = 3 / N = 50 / a = 0.0043 /  $\lambda$  = 2.8284
Algorithm methods: Gradient descent variation / Normal scattering / Quantum drift sampling
Algorithm parameters:  $\delta v$  = 0.0001 /  $\delta g$  = 0.1 /  $\delta s$  = 0.6325

Gradient descending from the initial variational point ( $\alpha$  = 0.4974 /  $\beta$  = 2.8284),
running 9994 Monte Carlo cycles at each point ...
Running 999989 Monte Carlo cycles at the final variational point ( $\alpha$  = 0.4784 /  $\beta$  = 2.9499) ...
VMC simulation finished!

Optimal  $\alpha$ : 0.4784
Optimal  $\beta$ : 2.9499
VMC energy: 127.2219  $\pm$  0.0067

3605.351353 seconds (18.66 M allocations: 2.465 GiB, 58.00% gc time)

julia> @time find_VMC_energy(QuantumTrap(3,50);as=[0.4784], $\beta$ s=[2.9499], $\delta v$ =0.001, $\delta g$ =0.01)

Finding the VMC energy for an elliptical 3D quantum trap with 50 interacting particles.

Quantum trap parameters: D = 3 / N = 50 / a = 0.0043 /  $\lambda$  = 2.8284
Algorithm methods: Gradient descent variation / Normal scattering / Quantum drift sampling
Algorithm parameters:  $\delta v$  = 0.001 /  $\delta g$  = 0.01 /  $\delta s$  = 0.6325

Gradient descending from the initial variational point ( $\alpha$  = 0.4784 /  $\beta$  = 2.9499),
running 9994 Monte Carlo cycles at each point ...
Running 999989 Monte Carlo cycles at the final variational point ( $\alpha$  = 0.4738 /  $\beta$  = 2.9811) ...
VMC simulation finished!

Optimal  $\alpha$ : 0.4738
Optimal  $\beta$ : 2.9811
VMC energy: 127.2164  $\pm$  0.0024

865.881236 seconds (4.19 M allocations: 655.300 MiB, 55.53% gc time)

julia> @time find_VMC_energy(QuantumTrap(3,50),[10^5,10^6];as=[0.4738], $\beta$ s=[2.9811], $\delta v$ =0.001, $\delta g$ =0.01)

Finding the VMC energy for an elliptical 3D quantum trap with 50 interacting particles.

Quantum trap parameters: D = 3 / N = 50 / a = 0.0043 /  $\lambda$  = 2.8284
Algorithm methods: Gradient descent variation / Normal scattering / Quantum drift sampling
Algorithm parameters:  $\delta v$  = 0.001 /  $\delta g$  = 0.01 /  $\delta s$  = 0.6325

Gradient descending from the initial variational point ( $\alpha$  = 0.4738 /  $\beta$  = 2.9811),
running 99997 Monte Carlo cycles at each point ...
Running 999989 Monte Carlo cycles at the final variational point ( $\alpha$  = 0.4743 /  $\beta$  = 2.9813) ...
VMC simulation finished!

Optimal  $\alpha$ : 0.4743
Optimal  $\beta$ : 2.9813
VMC energy: 127.2133  $\pm$  0.0022

93.143263 seconds (63.22 k allocations: 164.140 MiB, 6.97% gc time)

julia> @time find_VMC_energy(QuantumTrap(3,50),[10^5 for e in 2:8];as=[0.4743], $\beta$ s=[2.9813],variation="range",plot_output="convergence")

Finding the VMC energy for an elliptical 3D quantum trap with 50 interacting particles.

Quantum trap parameters: D = 3 / N = 50 / a = 0.0043 /  $\lambda$  = 2.8284
Algorithm methods: Range variation / Normal scattering / Quantum drift sampling
Algorithm parameters:  $\delta s$  = 0.6325

Running 9999983 Monte Carlo cycles at the variational point ( $\alpha$  = 0.4743 /  $\beta$  = 2.9813) ...
VMC simulation finished!

Optimal  $\alpha$ : 0.4743
Optimal  $\beta$ : 2.9813
VMC energy: 127.2168  $\pm$  0.0004

5043.042183 seconds (88.61 k allocations: 8.731 GiB, 0.03% gc time, 0.00% compilation time)

```

Figure 13: Output from gradient descent variation and subsequent refinement of the elliptical trap with 50 interacting bosons (3D50ie).

```

julia> @time find_VMC_energy(QuantumTrap(3,100);as=[0.49744])

Finding the VMC energy for an elliptical 3D quantum trap with 100 interacting particles.

Quantum trap parameters: D = 3 / N = 100 / a = 0.0043 /  $\lambda$  = 2.8284
Algorithm methods: Gradient descent variation / Normal scattering / Quantum drift sampling
Algorithm parameters:  $\delta v$  = 0.0001 /  $\delta g$  = 0.1 /  $\delta s$  = 0.6325

Gradient descending from the initial variational point ( $\alpha$  = 0.4974 /  $\beta$  = 2.8284),
running 9994 Monte Carlo cycles at each point ...
Running 999989 Monte Carlo cycles at the final variational point ( $\alpha$  = 0.4574 /  $\beta$  = 3.0957) ...
VMC simulation finished!

Optimal  $\alpha$ : 0.4574
Optimal  $\beta$ : 3.0957
VMC energy: 265.9434  $\pm$  0.0255

12698.236659 seconds (15.49 M allocations: 2.066 GiB, 13.65% gc time)

julia> @time find_VMC_energy(QuantumTrap(3,100),[10^5,10^6];as=[0.4574], $\beta$ s=[3.0957], $\delta v$ =0.0001, $\delta g$ =0.1)

Finding the VMC energy for an elliptical 3D quantum trap with 100 interacting particles.

Quantum trap parameters: D = 3 / N = 100 / a = 0.0043 /  $\lambda$  = 2.8284
Algorithm methods: Gradient descent variation / Normal scattering / Quantum drift sampling
Algorithm parameters:  $\delta v$  = 0.0001 /  $\delta g$  = 0.1 /  $\delta s$  = 0.6325

Gradient descending from the initial variational point ( $\alpha$  = 0.4574 /  $\beta$  = 3.0957),
running 9997 Monte Carlo cycles at each point ...
Running 999989 Monte Carlo cycles at the final variational point ( $\alpha$  = 0.4581 /  $\beta$  = 3.0957) ...
VMC simulation finished!

Optimal  $\alpha$ : 0.4581
Optimal  $\beta$ : 3.0957
VMC energy: 265.9159  $\pm$  0.009

1970.812048 seconds (265.72 k allocations: 330.552 MiB, 1.36% gc time)

julia> @time find_VMC_energy(QuantumTrap(3,100),[10^e for e in 2:8];as=[0.4581], $\beta$ s=[3.0957],variation="range",plot_output="convergence")

Finding the VMC energy for an elliptical 3D quantum trap with 100 interacting particles.

Quantum trap parameters: D = 3 / N = 100 / a = 0.0043 /  $\lambda$  = 2.8284
Algorithm methods: Range variation / Normal scattering / Quantum drift sampling
Algorithm parameters:  $\delta s$  = 0.6325

Running 9999983 Monte Carlo cycles at the variational point ( $\alpha$  = 0.4581 /  $\beta$  = 3.0957) ...
VMC simulation finished!

Optimal  $\alpha$ : 0.4581
Optimal  $\beta$ : 3.0957
VMC energy: 265.9246  $\pm$  0.0023

44930.463445 seconds (108.85 k allocations: 8.760 GiB, 0.00% gc time, 0.00% compilation time)

```

Figure 14: Output from gradient descent variation and subsequent refinement of the elliptical trap with 100 interacting bosons (3D100ie).