UiO : Department of Physics
University of Oslo

FYS4411: Computational Physics II

Project 1

A Variational Monte Carlo Analysis on Bose-Einstein Condensation in a Trapped Bose Gas

Nicolai Haug Jørn Eirik Betten Aleksandar Davidov

May 29, 2022

Abstract

In this project, we build a variational Monte Carlo method to estimate the ground state energy of an ultracold, dilute Bose gas in harmonic traps. We use a trial wave function composed of a single particle Gaussian with a single variational parameter and a hard sphere Jastrow factor for pair correlations. We consider two Markov chain Monte Carlo sampling algorithms; a random walk Metropolis with Gaussian proposals and a Langevin Metropolis-Hastings with proposals driven according to the gradient flow of the probability density of the trial wave function. The methods are implemented in a Python framework with automatic differentiation, procedures for tuning scale parameters and gradient descent optimizations of the variational parameter. The blocking method, which accounts for the autocorrelation of a Markov chain, is used to calculate the statistical error of the variational energy. The implementation is verified by comparison with closed-form expressions available when we consider a system of non-interacting bosons only. We find that the Langevin Metropolis-Hastings has a more efficient sampling routine taken into account the extra computational costs compared with the random walk Metropolis. The implementation with automatic differentiation is only a small factor slower than the analytical implementation. We also find that the energies per particle of the interacting systems increase with the particle density. Our framework provides a solid foundation for building a more general variational Monte Carlo method that can handle more complex systems.

github.com/nicolossus/FYS4411-Project1

Contents

1. Introduction

Bose-Einstein condensation (BEC), predicted by Einstein (1924; 1925) on the basis of the quantum formulations of Bose (1924), is a state of matter in which a large fraction of an ultracold, dilute gas of bosons simultaneously occupy the lowest accessible quantum state. Almost three quarters of a century after its prediction, BEC was observed in experiments with ultracold, dilute alkali gases confined in magnetic or optical traps, notably vapors of ^{87}Rb (Anderson et al., 1995). A Bose gas used to obtain the BEC is a finite-sized and inhomogeneous system, which makes predicting the properties of the system a many-body problem with a nontrivial solution. Fortunately, the dilute nature of the gas means the physics is dominated by two-body collisions and we can describe the effects of the interaction accurately with just a single physical parameter; the s-wave scattering length, a (Dalfovo et al., 1999). For ^{87}Rb, the s-wave scattering length $a_{\text{Rb}} = 100a_0$, where $a_0 = 0.5292\,\text{Å}$ is the Bohr radius, is usually chosen. In most cases, the confining traps are well approximated by harmonic potentials, and the trap frequency, ω_{ho}, provides a characteristic length of the system $a_{\text{ho}} \equiv [\hbar/(m\omega_{\text{ho}})]^{1/2}$. A typical trap for ^{87}Rb is $a_{\text{ho}} = 1 - 2 \cdot 10^4\,\text{Å}$. For calculations it is thus convenient to use the definite ratio of scattering length to trap length and $a_{\text{Rb}}/a_{\text{ho}} = 4.33 \cdot 10^{-3}$ is usually chosen (see e.g. DuBois and Glyde, 2001). Although the physical interaction can be simply described by the scattering length, we still need the wave function of the system in order to predict its properties. For most problems, the exact wave function is unknown and the crux of a problem resides in the need to define an approximation to the exact wave function. One approach to this problem is to construct a trial wave function which depends on selected variational parameters and captures as much of the physical features of the system as possible. As finding expectation values of quantum mechanical operators involve integrating out $3N$ degrees of freedom, Monte Carlo methods, known for being a particularly attractive choice for multidimensional integrals, become indispensable tools. In particular, the variational Monte Carlo (VMC) method, which is based on the variational principle, consists of finding the values of variational parameters for which the expectation value of the energy is the lowest possible. With these optimal variational parameters, the trial wave function is then an approximation to the ground-state wave function and provides an upper bound of the exact ground-state energy.

In this project, we build a VMC framework with the aim to find an upper bound of the ground state energy of a system of diluted ^{87}Rb trapped in either a spherical or elliptical harmonic oscillator. We consider two Markov chain Monte Carlo (MCMC) sampling algorithms; (i) the common random walk Metropolis (RWM) which explores the local neighborhood of the current state of the Markov chain using proposals from a symmetric probability distribution, and (ii) an algorithm based on the the nonlinear diffusion described by the Fokker-Planck and Langevin equations where the proposals are driven according to the gradient flow of the probability density of the trial wave function. The drift introduced in the second algorithm will typically move the Markov chain faster towards the center of the target distribution and thus mix faster than the plain RWM algorithm. We refer to the second algorithm as Langevin Metropolis-Hastings (LMH) due to its resemblance to Langevin Monte-Carlo type algorithms such as the Metropolis-adjusted Langevin algorithm (MALA) (Grenander and Miller, 1994). In MCMC algorithms, the size of the jumps in the state space of the Markov chain is typically controlled by the scale parameter of the proposal distribution. It is then intuitively clear that we can make an algorithm arbitrarily poor by choosing a scale parameter that is either too small or too large. In our VMC framework, we aim to develop procedures for automatic detection of an optimal scale of the proposal distribution. The optimal set of variational parameter values is the one that minimizes the expectation value

of the energy of the system. An exhaustive search over candidate values in order to find the optimal set is generally computationally expensive or even infeasible, which necessitates the need of using optimization algorithms for this endeavor. By ensuring that the energy functional on the trial wave function becomes a convex functional, we may use methods from convex optimization, such as gradient descent, for finding the optimal variational parameters. Gradient descent methods will be incorporated into our VMC framework as well. Moreover, we will also investigate the use of methods for automatic differentiation to negate the need of closed-form expressions for a particular quantum system under examination.

The project is organized as follows. In Section 2 we first provide the theoretical background of the VMC method before presenting the system under examination and its associated quantities. Next, the specifics of the methodologies, i.e., the MCMC algorithms, the gradient descent algorithms and so forth, used in our VMC framework are given in Section 3. The results are presented and discussed in Section 4, before subsequently they are concluded upon in Section 5. Lastly, an outline of possible continuations of the present work are provided in Section 6.

2. Theoretical Background

2.1. Variational Monte Carlo

Variational Monte Carlo (VMC) is a method applicable for finding an estimate to the ground state energy of a given quantum mechanical system. It proposes a trial wave function, $|\Psi_T\rangle$, and calculates the expected energy, or any other observable given its operator, by making use of a Monte Carlo evaluation of the integral. The trial wave function is designed with care and hopefully resembles the true wave function in architecture. By adding variational parameters in the trial wave function we may make use of the variational principle such that the energy functional on the trial wave function becomes a convex functional. A convex functional has a single global minimum. Therefore, if our trial wave function architecture resembles the architecture of the ground state wave function, we may simply make a search in the variational parameter space for the minima, and find an upper bound estimate of the exact ground state energy, E_0. In the following, we provide the details of the VMC method.

2.1.1. Monte Carlo Integration and Expectation Values

The general motivation to use Monte Carlo integration in quantum physics is its capability to compute multidimensional definite integrals, that is, integrals on the form

$$I = \int \mathrm{d}x_1 \int \mathrm{d}x_2 \ldots \int \mathrm{d}x_n \, f(x_1, x_2, \ldots, x_n),$$

where $f(x_1, x_2, \ldots, x_n)$ is a function in n variables. Fundamental to quantum mechanics is the computation of the expectation value of an observable. In general, for any absolutely continuous stochastic variable X with probability density function (pdf) $p(x)$ over states x, the law of the unconscious statistician states that the expectation value of X is given by

$$\langle X \rangle = \int_{\mathbb{R}} x p(x) \, \mathrm{d}x \,, \tag{2.1}$$

which for quantum systems typically will be a multidimensional integral. The Monte Carlo integration approach to compute the above integral is to sample M possible states x from $p(x)$. The law of large numbers then gives that

$$\langle X \rangle = \lim_{M \to \infty} \frac{1}{M} \sum_{i=1}^{M} x_i p(x_i),$$

such that the expectation value can be approximated by

$$\langle X \rangle \approx \frac{1}{M} \sum_{i=1}^{M} x_i p(x_i). \tag{2.2}$$

The number M is usually referred to as the number of Monte Carlo samples or cycles. The estimation error of Monte Carlo integration is independent of the dimensionality of the integral and decreases as $1/\sqrt{M}$. Monte Carlo integration is therefore more efficient for multidimensional integrals than traditional methods, such as the trapezoidal rule, which typically suffer from the curse of dimensionality.

When applying Monte Carlo integration, the naive Monte Carlo approach is to sample points uniformly on the phase space and evaluate the pdf at those points. Generally, systems only span a tiny portion of their phase space such that $p(x) \simeq 0$ for most states x. The contribution to the expectation value will therefore be negligible for a large portion of the samples, rendering the naive Monte Carlo approach inefficient. There is thus a need for procedures prioritizing sampling toward the values of x that significantly contribute to the expectation value. The family of Markov chain Monte Carlo (MCMC) sampling algorithms have this convergence property and are the predominant choice. The MCMC sampling algorithms used in this project will be discussed in Section 3.1.

For the sake of completeness, we introduce a particularly useful class of expectation values called *moments*. The n-th moment of a stochastic variable X with pdf $p(x)$ is defined as

$$\langle X^n \rangle \equiv \int x^n p(x) \, \mathrm{d}x \,.$$

Moments about the mean of the pdf are called *central moments*, and are used in preference to ordinary moments as they describe the spread and shape of the pdf in a location-invariant manner. The n-th central moment is defined as

$$\langle (X - \langle X \rangle)^n \rangle \equiv \int (x - \langle x \rangle)^n p(x) \, \mathrm{d}x \,.$$

The second central moment, known as the variance, is of particular interest. For a stochastic variable X, the variance is denoted as $\mathrm{Var}(X)$ and given by

$$\mathrm{Var}(X) = \left\langle (X - \langle X \rangle)^2 \right\rangle = \left\langle X^2 \right\rangle - \langle X \rangle^2 \,. \tag{2.3}$$

2.1.2. Review of the Variational Principle

The variational principle (for reference literature see e.g. Lester et al., 1994) states that, given a trial wave function $|\Psi_T\rangle$, the energy functional of the wave function has the ground state energy, E_0, as a lower bound. Any trial wave function can be expanded in terms of the orthonormal basis set of eigenfunctions of the Hamiltonian operator in Hilbert space, as they form a complete set. Denoting the eigenfunctions as $|\Psi_k\rangle$ and the Hamiltonian as H, such that $H|\Psi_k\rangle = E_k|\Psi_k\rangle$, we may expand the trial wave function in this eigenspace

$$|\Psi_T\rangle = \sum_k c_k |\Psi_k\rangle .$$

Making use of the orthonormality of the basis set, the normalization of the trial wave function is given by

$$\langle\Psi_T|\Psi_T\rangle = \sum_k \sum_l c_k c_l \langle\Psi_k|\Psi_l\rangle = \sum_k |c_k|^2.$$

The expected value of the Hamiltonian is then given by

$$\langle\Psi_T| H |\Psi_T\rangle = \sum_k |c_k|^2 E_k$$

and the energy functional can be written as

$$\langle E\rangle = \langle H\rangle = \frac{\langle\Psi_T| H |\Psi_T\rangle}{\langle\Psi_T|\Psi_T\rangle} = \frac{\sum_k |c_k|^2 E_k}{\sum_k |c_k|^2}, \tag{2.4}$$

which, since $E_k \geq E_0 \forall k$, shows that the ground state energy is a lower bound of the energy functional. The variance

$$\mathrm{Var}(E) = \frac{\langle\Psi_T| H^2 |\Psi_T\rangle}{\langle\Psi_T|\Psi_T\rangle} - \left(\frac{\langle\Psi_T| H |\Psi_T\rangle}{\langle\Psi_T|\Psi_T\rangle}\right)^2, \tag{2.5}$$

becomes zero when $|\Psi_T\rangle$ is equal to the (generally not normalized) ground state. Thus, we know we have the exact ground state energy if the variance of the expectation value of the energy is zero.

2.1.3. The Variational Monte Carlo Method

The first step of the VMC method is to formulate a trial wave function, $\Psi_T(\boldsymbol{r};\boldsymbol{\alpha})$, where $\boldsymbol{r} = (\boldsymbol{r}_1, \boldsymbol{r}_2, ..., \boldsymbol{r}_N)$ denotes the spatial coordinates, in total $d \times N$ if we have N particles in d dimensions present, and $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, ..., \alpha_m)$ the variational parameters. The next step is to evaluate the expectation value of the energy given by Equation 2.4. In order to bring Monte Carlo integration (Equation 2.2) to the quantum realm, we need to bring Equation 2.4 to an expression on the form of Equation 2.1. By noting that the trial wave function defines the quantum pdf as

$$p(\boldsymbol{r};\boldsymbol{\alpha}) = \frac{|\Psi_T(\boldsymbol{r};\boldsymbol{\alpha})|^2}{\int \mathrm{d}\boldsymbol{r}\,|\Psi_T(\boldsymbol{r};\boldsymbol{\alpha})|^2} \tag{2.6}$$

and introducing the local quantum operator known as the *local energy*

$$E_L(\boldsymbol{r};\boldsymbol{\alpha}) \equiv \frac{1}{\Psi_T(\boldsymbol{r};\boldsymbol{\alpha})} H\Psi_T(\boldsymbol{r};\boldsymbol{\alpha}), \tag{2.7}$$

we can write the expectation value of the energy as

$$\langle E\rangle = \int \mathrm{d}\boldsymbol{r}\, E_L(\boldsymbol{r};\boldsymbol{\alpha}) p(\boldsymbol{r};\boldsymbol{\alpha}). \tag{2.8}$$

Monte Carlo integration can then be applied, and the approximation takes the form

$$\langle E\rangle \approx \frac{1}{M}\sum_{i=1}^{M} E_L(\boldsymbol{r}_i;\boldsymbol{\alpha}) p(\boldsymbol{r}_i;\boldsymbol{\alpha}). \tag{2.9}$$

The final step of the VMC method is to vary $\boldsymbol{\alpha}$ while re-evaluating the expectation value of the energy until it yields the minimum possible energy. For a convex functional, gradient descent optimization may be used to find the direction in the variational parameter space which lowers the expected value of the energy and update the variational parameters in that direction. We discuss gradient descent optimization in Section 3.3.

2.2. The System

The system under investigation is a trapped Bose gas of alkali atoms, specifically ^{87}Rb. A key feature of alkali systems is that they are dilute, i.e., the volume per atom is much larger than the volume of the atom. The average distance between the atoms is thus much larger than the range of the inter-atomic interaction and the physics is dominated by two-body collisions.

2.2.1. Bosons in a Harmonic Trap

In order to model the Bose gas, we consider atoms trapped in either a spherical (S) or elliptical (E) harmonic oscillator trap:

$$V_{\text{trap}}(\boldsymbol{r}) = \begin{cases} \frac{1}{2}m\omega_{\text{ho}}^2 r^2 & \text{(S)} \\ \frac{1}{2}m[\omega_{\text{ho}}^2(x^2+y^2)+\omega_z^2 z^2] & \text{(E)} \end{cases}. \tag{2.10}$$

Here, ω_{ho}^2 defines the trap potential strength. In the case of an elliptical trap, $\omega_{\text{ho}} = \omega_\perp$ is the trap frequency in the xy plane and ω_z the frequency in the z direction. The atoms are modeled as hard spheres with a diameter proportional to the scattering length, a, that cannot overlap in space, mimicking the strong repulsion that atoms experience at close distances. The two-body interaction between atoms is thus described by the following pairwise, repulsive potential:

$$V_{\text{int}}(\|\boldsymbol{r}_i - \boldsymbol{r}_j\|) = \begin{cases} \infty, & \|\boldsymbol{r}_i - \boldsymbol{r}_j\| \leq a \\ 0, & \|\boldsymbol{r}_i - \boldsymbol{r}_j\| > a \end{cases}, \tag{2.11}$$

where $\|\cdot\|$ denotes the Euclidean distance. The Hamiltonian for a system of N trapped atoms is then given by

$$H = -\frac{\hbar}{2m}\sum_{i=1}^{N}\boldsymbol{\nabla}_i^2 + \sum_{i=1}^{N}V_{\text{trap}}(\boldsymbol{r}_i) + \sum_{i<j}^{N}V_{\text{int}}(\|\boldsymbol{r}_i - \boldsymbol{r}_j\|), \tag{2.12}$$

where the notation $i < j$ under the last summation sign signifies a double sum running over all pairwise interactions once.

2.2.2. Constructing the Trial Wave Function

The trial wave function is chosen to take the form of a Slater-Jastrow wave function:

$$\Psi_T(\boldsymbol{r};\alpha) = \Phi(\boldsymbol{r};\alpha)J(\boldsymbol{r}), \tag{2.13}$$

where α is a variational parameter. The Slater permanent, $\Phi(\boldsymbol{r};\alpha)$, is given by the first N single-particle wave functions chosen to be proportional to the harmonic oscillator function for the ground state:

$$\Phi(\boldsymbol{r};\alpha) = \prod_{i=1}^{N}\phi(\boldsymbol{r}_i;\alpha) = \prod_{i=1}^{N}\exp\Big\{-\alpha\Big(x_i^2 + y_i^2 + \beta z_i^2\Big)\Big\}. \tag{2.14}$$

Here, β could also be a variational parameter. However, in this project we will not treat β as a variational parameter, i.e., we limit ourselves to a single variational parameter, α. For elliptical traps we set $\beta = 2.82843$. For spherical traps we have $\beta = 1$ such that

$$\Phi(\boldsymbol{r};\alpha) = \prod_{i=1}^{N}\exp\Big\{-\alpha|\boldsymbol{r}_i|^2\Big\} \quad \text{(S)}. \tag{2.15}$$

The Jastrow correlation factor, $J(\boldsymbol{r})$, is chosen to be the exact solution of the Schrödinger equation for interacting pairs of hard spheres atoms:

$$J(\boldsymbol{r}) = \prod_{i<j}^{N}f(a, \|\boldsymbol{r}_i - \boldsymbol{r}_j\|), \tag{2.16}$$

where the correlation wave functions are given by

$$f(a, \|\boldsymbol{r}_i - \boldsymbol{r}_j\|) = \begin{cases} 0 & \|\boldsymbol{r}_i - \boldsymbol{r}_j\| \leq a \\ 1 - \frac{a}{\|\boldsymbol{r}_i - \boldsymbol{r}_j\|} & \|\boldsymbol{r}_i - \boldsymbol{r}_j\| > a \end{cases}. \tag{2.17}$$

In practice, with this formulation of the correlation wave functions, the inclusion of the repulsive potential (Equation 2.11) in the Hamiltonian (Equation 2.12) becomes redundant

as the correlation wave functions yield a probability density of zero whenever $\|\boldsymbol{r}_i - \boldsymbol{r}_j\| \leq a$. For non-interacting bosons, $a = 0$.

2.2.3. Drift Force and One-Body Densities

Here, we introduce quantities that will be important moving forward. First is the *drift force* of the system given by

$$\boldsymbol{F}(\boldsymbol{r};\alpha) = \frac{2\boldsymbol{\nabla}\Psi_T(\boldsymbol{r};\alpha)}{\Psi_T(\boldsymbol{r};\alpha)}. \tag{2.18}$$

The drift force comes into play in the Langevin Metropolis-Hastings sampling algorithm (Section 3.1.3) as a part that allows the algorithm to traverse the configuration space efficiently by moving towards regions of high probability density.

The *one-body density*, $\rho(\boldsymbol{r})$, displays the density of particles in configuration space. Its true form is given by the true ground state wave function Ψ_0, where we integrate out all the other particles in the following way

$$\rho(\boldsymbol{r}) = \int_{\boldsymbol{r}_2}\int_{\boldsymbol{r}_3}\cdots\int_{\boldsymbol{r}_N} |\Psi_0|^2 \mathrm{d}\boldsymbol{r}_2\mathrm{d}\boldsymbol{r}_3\ldots\mathrm{d}\boldsymbol{r}_N. \tag{2.19}$$

This is a simplification which we have done because the system of Bose gas contains identical particles. We recognize the spherical symmetry of our system and instead calculate the *radial one-body density*, $\rho(r)$, where $r = \|r\|$. This multidimensional integral is evaluated using Monte Carlo integration (see Section 2.1.1), and the radial one-body density is in practice calculated by making a grid of position values and bins of radial distances. We then count the number of samples from the Monte Carlo simulation that matches each bin, and we obtain the approximated one-body density.

2.2.4. Scaling and Optimization

In our computations, we use natural $\hbar = c = m = 1$ in order to avoid computing with small numbers as they quickly can lead to large numerical errors. Furthermore, inspired by Pfau et al., 2020, we perform computations with the trial wave function in the log domain. The reason for this is two-fold. Firstly, the form of expressions that need to be evaluated becomes easier. Secondly, the use of logarithmic densities in the MCMC algorithms (see Section 3.1) avoid computational over- and underflows as the evaluation of the ratio between the densities in the common formulation of the algorithms is then computed as the difference of the log densities. In the log domain, the trial wave function of Equation 2.13 assumes the form

$$\begin{aligned}\ln\Psi_T(\boldsymbol{r};\alpha) &= \prod_{i=1}^{N}\ln\phi(\boldsymbol{r}_i;\alpha)\prod_{i<j}^{N}\ln f(a,\|\boldsymbol{r}_i-\boldsymbol{r}_j\|)\\ &= \sum_{i=1}^{N}\ln\phi(\boldsymbol{r}_i;\alpha) + \sum_{i<j}^{N}\ln f(a,\|\boldsymbol{r}_i-\boldsymbol{r}_j\|),\end{aligned} \tag{2.20}$$

where

$$\ln \phi(\boldsymbol{r}_i;\alpha) = -\alpha\Big(x_i^2 + y_i^2 + \beta z_i^2\Big).$$

By using the relation $\Psi_T^{-1}\boldsymbol{\nabla}\Psi_T = \boldsymbol{\nabla}\ln\Psi_T$, the drift force of Equation 2.18 can be written as

$$\boldsymbol{F}(\boldsymbol{r};\alpha) = 2\boldsymbol{\nabla}\ln\Psi_T(\boldsymbol{r};\alpha). \tag{2.21}$$

With natural units and in the log domain, the local energy defined by Equation 2.7 is given by

$$\begin{aligned} E_L(\boldsymbol{r};\alpha) &= \Psi_T(\boldsymbol{r};\alpha)^{-1} H \Psi_T(\boldsymbol{r};\alpha) \\ &= -\frac{1}{2}\sum_{i=1}^{N}\Big[\boldsymbol{\nabla}_i^2\ln\Psi_T(\boldsymbol{r};\alpha) + \left(\boldsymbol{\nabla}_i\ln\Psi_T(\boldsymbol{r};\alpha)\right)^2\Big] + \sum_{i=1}^{N} V_{\text{trap}}(\boldsymbol{r}_i), \end{aligned} \tag{2.22}$$

where we have used that $\Psi_T^{-1}\boldsymbol{\nabla}^2\Psi_T = \boldsymbol{\nabla}^2\ln\Psi_T + (\boldsymbol{\nabla}\ln\Psi_T)^2$. The easiest approach to show that the latter relation holds is with the right-hand side as the point of departure;

$$\begin{aligned} \boldsymbol{\nabla}^2\ln\Psi_T + (\boldsymbol{\nabla}\ln\Psi_T)^2 &= \boldsymbol{\nabla}\cdot\boldsymbol{\nabla}\ln\Psi_T + \boldsymbol{\nabla}\ln\Psi_T\cdot\boldsymbol{\nabla}\ln\Psi_T \\ &= \boldsymbol{\nabla}\cdot\left(\frac{1}{\Psi_T}\boldsymbol{\nabla}\Psi_T\right) + \left(\frac{1}{\Psi_T}\boldsymbol{\nabla}\Psi_T\right)\cdot\left(\frac{1}{\Psi_T}\boldsymbol{\nabla}\Psi_T\right) \\ &= -\left(\frac{1}{\Psi_T^2}\boldsymbol{\nabla}\Psi_T\cdot\boldsymbol{\nabla}\Psi_T\right) + \left(\frac{1}{\Psi_T}\boldsymbol{\nabla}^2\Psi_T\right) + \left(\frac{1}{\Psi_T^2}\boldsymbol{\nabla}\Psi_T\cdot\boldsymbol{\nabla}\Psi_T\right) \\ &= \frac{1}{\Psi_T}\boldsymbol{\nabla}^2\Psi_T, \end{aligned}$$

where we again have used the relation $\Psi_T^{-1}\boldsymbol{\nabla}\Psi_T = \boldsymbol{\nabla}\ln\Psi_T$ and the product rule.

As mentioned in the introduction, the harmonic oscillator length $a_{\text{ho}} \equiv [\hbar/(m\omega_{\text{ho}})]^{1/2}$ provides a characteristic length of the system. Without applying natural units, we introduce the scaling $\boldsymbol{r}^* = \boldsymbol{r}/a_{\text{ho}} \Rightarrow \boldsymbol{r} = \boldsymbol{r}^* a_{\text{ho}}$. For brevity, we let $\boldsymbol{r}^* \to \boldsymbol{r}$ so that the preceding notation remains the same. With this scaling, the harmonic oscillator potentials (Equation 2.10) become

$$V_{\text{trap}}(\boldsymbol{r}) = \begin{cases} \frac{1}{2}\hbar\omega_{\text{ho}} r^2 & \text{(S)} \\ \frac{1}{2}\left[\hbar\omega_{\text{ho}}(x^2+y^2) + \hbar\omega_{\text{ho}}\frac{\omega_z^2}{\omega_{\text{ho}}^2} z^2\right] & \text{(E)} \end{cases}.$$

With the energy measured in units $\hbar\omega_{\text{ho}}$, the harmonic oscillator potentials further simplify to

$$V_{\text{trap}}(\boldsymbol{r}) = \begin{cases} \frac{1}{2} r^2 & \text{(S)} \\ \frac{1}{2}[x^2 + y^2 + \gamma^2 z^2] & \text{(E)} \end{cases}, \tag{2.23}$$

where $\gamma = \omega_z/\omega_{\text{ho}}$. As in DuBois and Glyde, 2001 and Nilsen et al., 2005, we will use $\gamma = \beta = 2.82843$ and $a = a_{\text{Rb}}/a_{\text{ho}} = 4.33 \cdot 10^{-3}$.

2.2.5. Closed-Form Expressions for Non-Interacting Bosons

In the following, we derive closed-form expressions for a non-interacting system of N bosons trapped in a d dimensional spherical harmonic oscillator. In this case, the trial wave function is given by

$$\ln \Psi_T(\boldsymbol{r}; \alpha) = -\alpha \sum_{i=1}^{N} |\boldsymbol{r}_i|^2,$$

where $\boldsymbol{r} \in \mathbb{R}^{N \times d}$. In spherical coordinates, the gradient of the trial wave function for particle i is given by the radial derivative

$$\boldsymbol{\nabla}_i = \frac{\partial}{\partial \boldsymbol{r}_i}$$

and the Laplacian by its two radial derivative terms in d dimensions;

$$\boldsymbol{\nabla}_i^2 = \frac{1}{\boldsymbol{r}_i^{d-1}} \frac{\partial}{\partial \boldsymbol{r}_i} \left(\boldsymbol{r}_i^{d-1} \frac{\partial}{\partial \boldsymbol{r}_i} \right).$$

Thus, we have that

$$\boldsymbol{\nabla}_i \ln \Psi_T(\boldsymbol{r}_i; \alpha) = -2\alpha \boldsymbol{r}_i$$

and

$$\boldsymbol{\nabla}_i^2 \ln \Psi_T(\boldsymbol{r}_i; \alpha) = \frac{1}{\boldsymbol{r}_i^{d-1}} \frac{\partial}{\partial \boldsymbol{r}_i} \left(-2\alpha \boldsymbol{r}_i^d \right) = \frac{1}{\boldsymbol{r}_i^{d-1}} \left(-2d\alpha \boldsymbol{r}_i^{d-1} \right) = -2d\alpha.$$

The local energy (Equation 2.22) is then given by

$$E_L(\boldsymbol{r}; \alpha) = -\frac{1}{2} \sum_{i=1}^{N} \left[-2d\alpha + (-2\alpha \boldsymbol{r}_i)^2 \right] + \sum_{i=1}^{N} \frac{1}{2} \boldsymbol{r}_i^2 = Nd\alpha + \sum_{i=1}^{N} \boldsymbol{r}_i^2 \left(\frac{1}{2} - 2\alpha^2 \right). \tag{2.24}$$

As $\sum_i \boldsymbol{r}_i^2$ is always positive, setting $\alpha = 1/2$ gives the minimal local energy;

$$E_L^{\min} = \frac{dN}{2}. \tag{2.25}$$

The the drift force vector (Equation 2.18) is given by

$$\boldsymbol{F}(\boldsymbol{r}; \alpha) = -4\alpha \boldsymbol{r}. \tag{2.26}$$

2.2.6. Closed-Form Expressions for Interacting Bosons

In the case of interacting bosons, we use the following formulation of the trial wave function;

$$\ln \Psi_T(\boldsymbol{r};\alpha) = \sum_{i=1}^{N} \ln \phi(\boldsymbol{r}_i;\alpha) + \sum_{i<j}^{N} u(r_{ij}),$$

where $u(r_{ij}) \equiv \ln f(a, r_{ij})$ and $r_{ij} \equiv \|\boldsymbol{r}_i - \boldsymbol{r}_j\|$. In the ensuing derivations, the aim is to find analytical expressions for the gradient and Laplacian of the trial wave function for a particle k. In order to compute $\boldsymbol{\nabla}_k u(r_{kj})$, we can rewrite the gradient operator as

$$\boldsymbol{\nabla}_k = \boldsymbol{\nabla}_k \frac{\partial r_{kj}}{\partial r_{kj}} = \boldsymbol{\nabla}_k r_{kj} \frac{\partial}{\partial r_{kj}} = \frac{\boldsymbol{r}_k - \boldsymbol{r}_j}{r_{kj}} \frac{\partial}{\partial r_{kj}}.$$

The gradient of the Slater permanent for particle k is simply the gradient of the single-particle element in question. As the gradient of the pairwise correlations only acts on particle k, the only non-zero interactions will involve particle k. Thus, the gradient of the trial wave function for particle k is

$$\begin{aligned} \boldsymbol{\nabla}_k \ln \Psi_T(\boldsymbol{r};\alpha) &= \boldsymbol{\nabla}_k \ln \phi(\boldsymbol{r}_k;\alpha) + \sum_{j\neq k}^{N} \boldsymbol{\nabla}_k u(r_{kj}) \\ &= \boldsymbol{\nabla}_k \ln \phi(\boldsymbol{r}_k;\alpha) + \sum_{j\neq k}^{N} \frac{\boldsymbol{r}_k - \boldsymbol{r}_j}{r_{kj}} u'(r_{kj}), \end{aligned} \tag{2.27}$$

where $u'(r_{kj}) \equiv \frac{\partial}{\partial r_{kj}} u(r_{kj})$ and the summation $\sum_{j\neq k}$ signifies that each interaction with particle k is enumerated only once, and self-interaction is excluded from the summation.

As with the gradient, the Laplacian of the Slater permanent is simply $\boldsymbol{\nabla}_k^2 \ln \phi(\boldsymbol{r}_k;\alpha)$. The Laplacian of the pairwise correlations, on the other hand, is more involved to derive. Our point of departure is

$$\begin{aligned} \boldsymbol{\nabla}_k \sum_{j\neq k}^{N} \boldsymbol{\nabla}_k u(r_{kj}) &= \sum_{j\neq k}^{N} \boldsymbol{\nabla}_k \left(\frac{\boldsymbol{r}_k - \boldsymbol{r}_j}{r_{kj}} u'(r_{kj}) \right) \\ &= \sum_{j\neq k} \left(u'(r_{kj}) \boldsymbol{\nabla}_k \left(\frac{\boldsymbol{r}_k - \boldsymbol{r}_j}{r_{kj}} \right) + \frac{\boldsymbol{r}_k - \boldsymbol{r}_j}{r_{kj}} \boldsymbol{\nabla}_k u'(r_{kj}) \right) \end{aligned} \tag{2.28}$$

where we have used the product rule. The gradient in the first summation term on the right-hand side of Equation 2.28 can be computed using the quotient rule:

$$\boldsymbol{\nabla}_k \frac{\boldsymbol{r}_k - \boldsymbol{r}_j}{r_{kj}} = \frac{\boldsymbol{\nabla}_k (\boldsymbol{r}_k - \boldsymbol{r}_j) r_{kj} - (\boldsymbol{r}_k - \boldsymbol{r}_j) \cdot \boldsymbol{\nabla}_k r_{kj}}{r_{kj}^2},$$

where $\boldsymbol{\nabla}_k (\boldsymbol{r}_k - \boldsymbol{r}_j) = d$, with d denoting the dimensionality, and $\boldsymbol{\nabla} r_{kj} = (\boldsymbol{r}_k - \boldsymbol{r}_j)/r_{kj}$. Thus, we have that

$$\boldsymbol{\nabla}_k \left(\frac{\boldsymbol{r}_k - \boldsymbol{r}_j}{r_{kj}} \right) u'(r_{kj}) = \left(\frac{dr_{kj}}{r_{kj}^2} - \frac{(\boldsymbol{r}_k - \boldsymbol{r}_j)(\boldsymbol{r}_k - \boldsymbol{r}_j)}{r_{kj}^3} \right) u'(r_{kj}) = \frac{d-1}{r_{kj}} u'(r_{kj}).$$

The second summation term on the right-hand side of Equation 2.28 can be computed as follows:

$$\frac{\boldsymbol{r}_k - \boldsymbol{r}_j}{r_{kj}} \boldsymbol{\nabla}_k u'(r_{kj}) = \frac{(\boldsymbol{r}_k - \boldsymbol{r}_j)^2}{r_{kj}^2} \frac{\partial}{\partial r_{kj}} u'(r_{kj}) = u''(r_{kj}),$$

where $u''(r_{kj}) \equiv \frac{\partial^2}{\partial r_{kj}^2} u(r_{kj})$. The completed version of Equation 2.28 then reads

$$\boldsymbol{\nabla}_k \sum_{j \neq k}^{N} \boldsymbol{\nabla}_k u(r_{kj}) = \sum_{j \neq k}^{N} \left(u''(r_{kj}) + \frac{d-1}{r_{kj}} u'(r_{kj}) \right),$$

and the Laplacian of the trial wave function for particle k thus becomes

$$\boldsymbol{\nabla}_k^2 \ln \Psi_T(\boldsymbol{r}; \alpha) = \boldsymbol{\nabla}_k^2 \ln \phi(\boldsymbol{r}_k; \alpha) + \sum_{j \neq k}^{N} \left(u''(r_{kj}) + \frac{d-1}{r_{kj}} u'(r_{kj}) \right). \tag{2.29}$$

Recall that

$$f(a, r_{kj}) = \begin{cases} 0, & r_{kj} \leq a \\ 1 - a/r_{kj}, & r_{kj} > a \end{cases}.$$

For $r_{kj} \leq a$, we then have

$$u'(r_{kj}) = \frac{\partial}{\partial r_{kj}} \ln 0 = \frac{\partial}{\partial r_{kj}} (-\infty) = 0, \tag{2.30}$$

$$u''(r_{kj}) = 0, \tag{2.31}$$

and for $r_{kj} > a$ we have

$$u'(r_{kj}) = \frac{\partial}{\partial r_{kj}} \ln \left(1 - \frac{a}{r_{kj}} \right) = \frac{a}{r_{kj}(r_{kj} - a)}, \tag{2.32}$$

$$u''(r_{kj}) = \frac{a(a - 2r_{kj})}{r_{kj}^2 (r_{kj} - a)^2}. \tag{2.33}$$

The local energy and drift force can then be computed by inserting the above results into Equation 2.22 and Equation 2.18, respectively. As the above expressions are the ones we will implement in our VMC framework, we leave further computations as an exercise for the interested reader.

3. Methodology

3.1. Sampling Algorithms

The system under investigation consists of N bosons in d dimensions, which gives a $N \times d$ dimensional configuration space. As mentioned in Section 2.1.1, sampling points uniformly on the configuration space will, in general, require a sizeable number of samples to obtain an accurate estimate. A more efficient approach is to iteratively sample points such that at each step we expect to sample from a distribution that becomes closer to the target distribution, $p(x)$.

3.1.1. Markov Chain Monte Carlo

Markov chain Monte Carlo (MCMC) methods (for reference literature see e.g. Ross, 2014 or Gelman et al., 2014) constitute a class of computational sampling algorithms that are particularly well-suited for sampling from high-dimensional distributions. The name MCMC is the combination of two properties: *Monte Carlo* and *Markov chain*. Monte Carlo is the practice of estimating statistical properties of a distribution by examining random samples from the distribution, whereas the Markov chain property is a sequential process in which the probability of transitioning to the next state is only dependent on the current state. The property that the probability of transitioning to the next state is conditional solely on the present state is known as the Markov property. Hence, the key idea of MCMC is to generate a sequence $\{X_t : t \geq 0\}$ of random samples X_t, where the index $t \in \mathbb{N}$ is interpreted as time, with the Markov property,

$$\mathrm{P}(X_{t+1} = x' \mid X_0 = x_0, X_1 = x_1, ..., X_t = x) = \mathrm{P}(X_{t+1} = x' \mid X_t = x) = p(x' \mid x),$$

where $p(x' \mid x)$ denotes the probability of transitioning from any given state x to any other given state x'. Vital to the method's success, however, is not the Markov property but rather the construction of transition probabilities, $p(x' \mid x)$, for which the Markov chain converges to a unique stationary distribution $\pi(x)$ such that $\pi(x) = p(x)$. A sufficient, but not necessary, condition for the existence of a stationary distribution is *detailed balance*, which requires that each transition is reversible;

$$\pi(x)p(x' \mid x) = \pi(x')p(x \mid x'). \tag{3.1}$$

The uniqueness of a the stationary distribution is guaranteed by *ergodicity* of a Markov chain. A Markov chain is ergodic if it has an aperiodic state, which implies that all remaining states are also aperiodic, and every state is positive recurrent, which means that the expected amount of time for returning to the same state is finite.

For the MCMC algorithms discussed in the next sections it is useful to decompose the transition probability $p(x' \mid x)$ as

$$p(x' \mid x) = q(x' \mid x)a(x' \mid x),$$

where $q(x' \mid x)$ is a proposal distribution that proposes a move to state x' from x and $a(x' \mid x)$ is the acceptance probability, i.e., the probability of accepting the proposed transition $x \to x'$. The proposal distribution $q(x' \mid x)$ can in principle be chosen completely arbitrarily, although

algorithms may be limited to the use of proposal distributions with certain properties, such as being symmetric. Typically, the proposal distribution is chosen among a set of well-known and tractable distributions, so that sampling from the distribution is straightforward. The proposal distribution $q(x' \mid x)$ should share the same support as the target distribution $p(x)$.

Practically, MCMC algorithms are generally implemented with multiple Markov chains starting from arbitrary points in the configuration space. In the random walk MCMC algorithms presented in the subsequent sections, the Markov chains then stochastically explore the configuration space according to the algorithm and tend to move toward regions of higher probability density. After a sufficient number of (time) steps, the Markov chain reaches the stationary distribution, i.e., the distribution does not depend on the position within the chain and the samples are just meandering around a stationary point. Only after convergence to the stationary distribution is the sampler guaranteed to be sampling from the target distribution. In performing MCMC sampling, the difficulty usually does not reside in constructing a Markov chain with the desired properties but rather in determining the number of steps needed for the chain to converge to the stationary distribution within an admissible error.

3.1.2. Random Walk Metropolis

The random walk Metropolis (RWM) algorithm (Metropolis et al., 1953) is one of the most established MCMC sampling methods. In its original form, the RWM algorithm is an adaptation of a random walk that explores the local neighborhood of the current state of a Markov chain using a symmetrical proposal distribution. The RWM algorithm was later generalized to the more general case of using non-symmetrical proposal distributions in the Metropolis-Hastings (MH) algorithm (Hastings, 1970). The MH algorithm can be derived by using the condition for detailed balance (Equation 3.1) as the point of departure:

$$p(x)p(x' \mid x) = p(x')p(x \mid x')$$

or

$$p(x)q(x' \mid x)a(x' \mid x) = p(x')q(x \mid x')a(x \mid x'),$$

which can be written as

$$\frac{a(x' \mid x)}{a(x \mid x')} = \frac{p(x')q(x \mid x')}{p(x)q(x' \mid x)}.$$

The condition for detailed balance is satisfied by setting

$$a(x' \mid x) = \min\left(1, \frac{p(x')q(x \mid x')}{p(x)q(x' \mid x)}\right), \tag{3.2}$$

which is known as the Metropolis criterion. Here, we have set $a(x \mid x') = 1$, but note that setting $a(x' \mid x) = 1$ instead also satisfies the condition. If the proposal distribution is symmetric, i.e., $q(x' \mid x) = q(x \mid x')\,\forall(x, x')$, then the Metropolis criterion simplifies to

$$a(x' \mid x) = \min\left(1, \frac{p(x')}{p(x)}\right). \tag{3.3}$$

Both the normal and uniform distributions are examples of symmetric distributions.

The Metropolis criterion is a rule that decides whether to accept or reject a proposed move. The quality of the proposed state is evaluated by computing its relative probability density, as normalization constants will cancel each other out, and comparing it to the relative probability density of the current state. If the probability density of the proposed state is higher than that of the current state, the proposed state is accepted and becomes the next state in the chain. If the probability density of the proposed state is lower than that of the current state, the proposed state is accepted with a probability determined by the ratio of probability densities. In this way, the states in the Markov chain tend to move toward the highest probability regions, but can still move away from these high-probability regions. In order to implement the acceptance/rejection rule in a computer program, we generate a uniform random number $u \sim \mathrm{U}(0,1)$ after computing $a(x' \mid x)$ for a given proposal x'. If $u \leq a(x' \mid x)$, we accept the proposal. On the other hand, if $u > a(x' \mid x)$, we reject the proposal.

In terms of our quantum system, we denote the current state, or configuration, by $\boldsymbol{r}$ and a proposed state by $\boldsymbol{r}'$. As proposal distribution for the RWM algorithm we choose a normal distribution with the location parameter specified by the current state $\boldsymbol{r}$ and a common scale parameter σ, $q(\boldsymbol{r}' \mid \boldsymbol{r}) = \mathrm{N}(\boldsymbol{r}' \mid \boldsymbol{r}, \sigma)$. We remark that expressing the normal distribution in terms of the scale, or standard deviation, σ and not the more common squared scale, or variance, σ^2, is deliberate. In computer programs, the normal distribution is typically parametrized by the scale parameter. By following this convention our methodology and associated equations will more directly translate to computer code. The scale parameter σ is a tuning parameter that we increase or decrease if the acceptance rate of the Markov chain simulation is too high or low, respectively (see Section 3.1.4). Recall that the target quantum probability density is given by Equation 2.6, such that the RWM algorithm's Metropolis criterion (Equation 3.3) then becomes

$$a(\boldsymbol{r}' \mid \boldsymbol{r}) = \min\left(1, \frac{|\Psi_T(\boldsymbol{r}'; \alpha)|^2}{|\Psi_T(\boldsymbol{r}'; \alpha)|^2}\right).$$

As we perform computations with the trial wave function in the log domain, the above ratio of densities simplifies to the difference of the log densities:

$$\ln\left(a(\boldsymbol{r}' \mid \boldsymbol{r})\right) = \min\left(0, 2\ln\Psi_T(\boldsymbol{r}'; \alpha) - 2\ln\Psi_T(\boldsymbol{r}; \alpha)\right). \tag{3.4}$$

Algorithm 1 outlines a simple implementation of the RWM algorithm in our VMC study. In our VMC framework the sampler will be more complex (see Section 3.6), but the general idea is encapsulated by the algorithm.

Algorithm 1 Random Walk Metropolis

Inputs:
Initial configuration $\boldsymbol{r}_0$
Variational parameter α
Number of Monte Carlo cycles M

$t = 0$
repeat
  Sample proposals $\boldsymbol{r'} \sim \mathrm{N}(\boldsymbol{r}_t, \sigma)$
  Calculate the Metropolis criterion, a, given by Equation 3.4
  Sample $u \sim \ln(\mathrm{U}(0,1))$
  if $u \leq a$ **then**
    Accept proposal $\boldsymbol{r}_{t+1} \leftarrow \boldsymbol{r'}$
  else
    Reject proposal $\boldsymbol{r}_{t+1} \leftarrow \boldsymbol{r}_t$
  Sample the local energy E_L^t given by Equation 2.22
  t ← t + 1
until t=M-1
Compute the expectation value of the energy $\langle E \rangle = \frac{1}{M} \sum_t E_L^t$

3.1.3. Langevin Metropolis-Hastings

By guiding the proposals according to the gradient flow of the probability density of the trial wave function, the sampling quality can be increased. The following algorithm, which we have dubbed Langevin Metropolis-Hastings (LMH), is based on the Fokker-Planck and Langevin equations for Brownian particles. In statistical mechanics, it is well-established that the diffusion coefficient is related to the mean squared displacement of a Brownian particle in the stationary state, and hence the diffusion equation can be set for the probability density of a Brownian particle. The Langevin equation describes Brownian motion, i.e., the apparently random movement of a particle in a fluid due to constant collisions with other particles. The Fokker-Planck equation governs the time evolution of the probability density for the velocity of Brownian particles under the influence of a diffusion generated by interactions and drift generated by the friction in the fluid. Although the original intention of the Fokker-Planck and Langevin equations were to describe Brownian motion, they can also govern the behavior of a system, such as a Markov chain, in presence of a random noise and its evolution towards a stationary state. In the LMH algorithm, the dynamics introduced by the Fokker-Planck and Langevin equations typically make the Markov chain converge faster than the RWM algorithm towards the highest density regions of the target quantum probability distribution.

By considering a system of N Brownian particles subject to a force $\boldsymbol{F}(\boldsymbol{r})$, the Fokker-Planck equation takes the form of the Smoluchowski diffusion equation

$$\frac{\partial p(\boldsymbol{r},t)}{\partial t} = \sum_{i=1}^{N} D\boldsymbol{\nabla}_i \cdot (\boldsymbol{\nabla}_i - \boldsymbol{F}(\boldsymbol{r}_i))p(\boldsymbol{r},t), \tag{3.5}$$

where $p(\boldsymbol{r},t)$ denotes the time-dependent probability density, D is the diffusion coefficient and $\boldsymbol{F}(\boldsymbol{r}_i)$ is interpreted as the ith component of a drift term (drift velocity) caused by an

external potential. For simplicity, we will in the following denote $p(\boldsymbol{r}, t) \rightarrow p$ and $\boldsymbol{F}(\boldsymbol{r}_i) \rightarrow \boldsymbol{F}_i$. To find the stationary state of the probability density, we set $\frac{\partial p}{\partial t} = 0$ which yields

$$\boldsymbol{\nabla}_i^2 p = p\boldsymbol{\nabla}_i \boldsymbol{F}_i + \boldsymbol{F}_i \boldsymbol{\nabla}_i p.$$

For the above equation to be independent of the configuration $\boldsymbol{r}$ and yield zero for all particles i, the drift vector should be of the form $\boldsymbol{F}_i = g\boldsymbol{\nabla}_{\mathrm{i}} p$ such that

$$\boldsymbol{\nabla}_i^2 p = p\boldsymbol{\nabla}_i g(\boldsymbol{\nabla}_i p)^2 + pg\boldsymbol{\nabla}_i^2 p + g(\boldsymbol{\nabla}_i p)^2,$$

which results in $g = \frac{1}{p}$. This defines the gradient flow of the probability density of the trial wave function as $\boldsymbol{F} = 2\Psi_T^{-1}\boldsymbol{\nabla}\Psi_T$, which is the drift force defined by Equation 2.21.

The motion in configuration space of the Brownian particles governed by the Fokker-Planck equation is given by the Langevin equation. In MCMC calculations, the proposed new state $\boldsymbol{r'}$ comes from the solution of the Langevin equation

$$\frac{\partial \boldsymbol{r}}{\partial t} = \mathrm{D}\boldsymbol{F}(\boldsymbol{r}(t)) + \eta, \tag{3.6}$$

where η is a stochastic force distributed with mean zero and variance $2D$, where the diffusion coefficient $D = \frac{1}{2}$ (in natural units) which follows from the Schrödinger equation. The solution of the Langevin equation can be obtained by using Euler's method, which gives

$$\boldsymbol{r'} = \boldsymbol{r} + \mathrm{D}\boldsymbol{F}(\boldsymbol{r})\Delta t + \boldsymbol{\xi}\sqrt{\Delta t}, \tag{3.7}$$

where Δt is the time step and $\boldsymbol{\xi}$ here is a multivariate standard normal random variable. Usually, $\boldsymbol{\xi}$ is defined as a multivariate normal random variable with mean $\mathbf{0}$ and common variance $2D\Delta t$ which becomes only Δt since $D = \frac{1}{2}$. In Equation 3.7, however, the spread of the distribution is defined by the scale parameter $\sqrt{\Delta t}$, which will have to be tuned in order to obtain the desired acceptance rate of the LMH algorithm (see Section 3.1.4). A good choice for the proposal distribution, $q(\boldsymbol{r'} \mid \boldsymbol{r})$, is the Green's function of the Fokker-Planck equation in the short-time approximation:

$$G(\boldsymbol{r'}, \boldsymbol{r}, \Delta t) = \frac{1}{(4\pi \mathrm{D}\Delta t)^{\frac{dN}{2}}} \exp\left[-\frac{(\boldsymbol{r'} - \boldsymbol{r} - \mathrm{D}\Delta t\boldsymbol{F}(\boldsymbol{r}))^2}{4\mathrm{D}\Delta t}\right], \tag{3.8}$$

where d is the dimensionality. As the proposal distribution in Equation 3.8 is non-symmetric, the Metropolis criterion is given by Equation 3.2, that is

$$a(\boldsymbol{r'} \mid \boldsymbol{r}) = \min\left(1, \frac{p(\boldsymbol{r'})G(\boldsymbol{r}, \boldsymbol{r'}, \Delta t)}{p(\boldsymbol{r})G(\boldsymbol{r'}, \boldsymbol{r}, \Delta t)}\right). \tag{3.9}$$

Our implementation of the LMH algorithm will be in the logarithmic domain, such that

$$\ln\left(a(\boldsymbol{r'} \mid \boldsymbol{r})\right) = \min(0, \ln\left(p(\boldsymbol{r'})\right) + \ln\left(G'(\boldsymbol{r}, \boldsymbol{r'}, \Delta t)\right) - \ln\left(p(\boldsymbol{r})\right) - \ln\left(G'(\boldsymbol{r'}, \boldsymbol{r}, \Delta t)\right)), \tag{3.10}$$

where G' denotes the Green's function without the normalization constant.

An overview of the Langevin Metropolis-Hastings sampling algorithm is listed in Algorithm 2, where, for simplicity and generality, also here only the general idea is encapsulated by the algorithm.

Algorithm 2 Langevin Metropolis-Hastings

Inputs:
Initial configuration $\boldsymbol{r}_0$
Variational parameter α
Number of Monte Carlo cycles M

$t = 0$
repeat
  Compute drift force $\boldsymbol{F}(\boldsymbol{r}_t, \alpha)$ (Equation 2.21)
  Sample proposals $\boldsymbol{r'} \sim \boldsymbol{r}_t + D\boldsymbol{F}(\boldsymbol{r}_t, \alpha)\Delta t + \mathrm{N}\left(\mathbf{0}, \sqrt{\Delta t}\right)$, cf. Equation 3.7
  Compute drift force $\boldsymbol{F}(\boldsymbol{r}', \alpha)$ (Equation 2.21)
  Calculate the Metropolis criterion, a, given by Equation 3.10
  Sample $u \sim \ln(\mathrm{U}(0, 1))$
  if $u \leq a$ **then**
    Accept proposal $\boldsymbol{r}_{t+1} \leftarrow \boldsymbol{r'}$
  else
    Reject proposal $\boldsymbol{r}_{t+1} \leftarrow \boldsymbol{r}_t$
  Sample the local energy E_L^t given by Equation 2.22
  t $\leftarrow$ t + 1
until t=M-1
Compute the expectation value of the energy $\langle E \rangle = \frac{1}{M} \sum_t E_L^t$

3.1.4. Optimal Scale Parameter

In applying the RWM and LMH algorithms discussed above, it is necessary to choose a proposal distribution $q(\boldsymbol{r}' \mid \boldsymbol{r})$. Typically, q is chosen among a set of well-known and tractable distributions, with the purpose of making random generation and evaluations of probabilities straightforward. Many distributions are specified by a scale parameter, σ, and the scale parameter thus becomes a tuning parameter which determines the jumps of an algorithm in the phase space. For small σ, the algorithm will propose small jumps. Consequently, almost all proposed states will be accepted but the algorithm will explore the phase space slowly and thus may take a long time to converge to the stationary distribution. On the other hand, if σ is large, the algorithm will propose large jumps. In this case, the jumps will likely end in regions where the probability density is small and most proposed states will be rejected. However, it is reasonable to assume there exist values of σ in between these extremes where the algorithm performs optimally. (Roberts and Rosenthal, 2001) characterize optimal values of σ for several MCMC algorithms in terms of the optimal acceptance rate of a particular algorithm. In our VMC framework, we will implement an adaptive procedure for tuning the scale parameters based on a desired acceptance rate. For RWM algorithm, we tune the scale parameter such that the algorithm has an acceptance rate between 20-50%, which is the convention in many other MCMC frameworks such as `PyMC` (Salvatier et al., 2016). The Metropolis-adjusted Langevin algorithm (MALA) (Grenander and Miller, 1994) has a larger optimal acceptance rate than RWM. Due to the LMH algorithm's resemblance to MALA, we

use the convention for the latter also for LMH and set the desired acceptance rate between 40-80%.

Implementation-wise, we have to record the acceptance rate over a given interval of MC cycles. If the acceptance rate is too high, we increase the scale by a factor. Conversely, if the acceptance rate is too low, we decrease the scale by a factor. This is reiterated until the acceptance rate is within the desired range or if a set maximum number of MC cycles is exceeded. In the adaptive tuning of the scale parameter, the properties of the algorithm is altered. As it most likely will not converge to the stationary distribution during tuning, we do not use recorded quantities in this phase for the final sampling results, such as the expectation value of the energy.

3.2. Blocking

Unlike conventional Monte Carlo integration where the random samples are statistically independent, those in MCMC are autocorrelated. Consequently, the autocorrelation should be taken into account in the calculation of statistical errors of the estimated statistics. Here, we discuss the *blocking* method (Flyvbjerg and Petersen, 1989) which can be used to address the autocorrelation of a Markov chain. In the statistical theory of the design of experiments, *blocking* is the arranging of experimental units in groups (blocks) that are similar to one another. Typically, a blocking factor is a source of variability that is not of primary interest to the experimenter. An example of a blocking factor might be the sex of a patient; by blocking on sex, this source of variability is controlled for, thus leading to greater accuracy. In probability theory the blocks method consists of splitting a sample into blocks (groups) separated by smaller subblocks so that the blocks can be considered almost independent. The blocks method helps proving limit theorems in the case of dependent random variables. The blocking method does something similar, although instead of separating the samples with subblocks, it merges subsequent samples in the time series by taking the average. It provides an alternative method for estimating the variance $\mathrm{Var}(\hat{\theta})$ for the estimator $\hat{\theta} = \overline{X}$, where $\overline{X}$ denotes the mean of the dataset X. It can be compared to dependent bootstrapping but with a much lower computational complexity of $\mathscr{O}(M)$. This allows for speedups compared to other methods with a big-O of $\mathscr{O}(M^2)$ or $\mathscr{O}(M \log M)$. Further advantages includes effective scaling for large number of observations, M. Whereas for large M, dependent bootstrapping scales poorly, blocking does not, in fact it becomes more accurate for larger M.

Assume that we have a sample size of $M = 2^d$ for any $d \in \mathbb{N}$ and an M-tuple $X = (X_1, X_2, \ldots, X_M)$ of stationary time series which we assume are *asymptotically uncorrelated*, which means that the correlation between two samples rapidly decays as the number of time steps between them increases. This now let us define the *blocking transformation*. The idea here is to take the mean of a subsequent pair of elements from X and form a new sample X_1. This process is repeated d times, which is why $M = 2^d$ is required, yielding $X_0, \ldots, X_{d-1}$ containing the subsequent averages of observations. This gives for each iteration a dataset of size $M_k = 2^{d-k}$, with the number of observations with each iteration. We can recursively define X_k as

$$
\begin{aligned}
(X_0)_k &\equiv (X)_k \\
(X_{i+1})_k &\equiv \frac{1}{2}((X_i)_{2k-1} + (X_i)_{2k}) \qquad \text{for all} \qquad 1 \leq i \leq d-1
\end{aligned}
$$

The sample X_k is X_0 subjected to k blocking transformations. The variance in the mean of X_k is consequently defined by

$$Var(\overline{X}_k) = \frac{\sigma_k^2}{M_k} + \underbrace{\frac{2}{M_k}\sum_{h=1}^{M_k-1}\left(1-\frac{h}{M_k}\right)\gamma_k(h)}_{\equiv e_k} = \frac{\sigma_k^2}{M_k} + e_k \quad \text{if} \quad \gamma_k(0) = \sigma_k^2$$

as result of the blocking transformation. Where e_k is the truncation error due to correlations between samples defined as

$$e_k = \frac{2}{M_k}\sum_{h=1}^{M_k-1}\left(1-\frac{h}{M_k}\right)\gamma_k(h),$$

and M_k denotes the number of samples in $(\boldsymbol{X})_k$, $h = |i-j|$ is the number of time steps between the samples which we calculate the autocovariance between, while $\gamma_k(h)$ is the autocovariance for the kth sample.

It can be shown (for a proof of the convergence of the blocking method and an automatic implementation, see (Jonsson, 2018)) that $Var\left(\overline{X}_k\right) = Var\left(\overline{X}\right)$ for all $0 \leq k \leq d-1$ when the truncation error is included in the calculations. The sequence $\{e_k\}_{k=0}^{d-1}$ is decreasing, and, given enough samples (large enough d), it can be as made as small as we want. Thus, given enough blocking transformations, the true sample variance, $Var\left(\overline{X}\right)$, will be well approximated by the variance of $\boldsymbol{X}_k$ for a sufficiently large k. The sample variance

$$Var\left(\overline{X}\right) \approx \frac{\sigma_k^2}{n_k}, \tag{3.11}$$

and we can make this approximation arbitrarily good.

3.3. Gradient Descent Optimization

The variational principle ensures convexity in the energy functional with respect to the variational parameters. A gradient descent method converges to the only minima when a functional is convex. Therefore, assuming proper estimation of the energy functional and its gradient with respect to the variational parameter α, a gradient descent method assures convergence to the ground state of the system.

Gradient descent utilises the fact that the negative of the gradient of a function points in the direction of steepest descent. The multivariate Newton-Raphson iterative method for finding the minima of a multivariate functional $f : \mathbb{R}^N \to \mathbb{R}$ is

$$\boldsymbol{x}^{\mathrm{i}+1} = \boldsymbol{x}^{\mathrm{i}} - \left[\mathrm{H}_{f(\boldsymbol{x}^{\mathrm{i}})}\right]^{-1}\boldsymbol{\nabla}_{\boldsymbol{x}^{\mathrm{i}}} f(\boldsymbol{x}^{\mathrm{i}}) \tag{3.12}$$

where $\boldsymbol{x}$ are the variatonal parameters, $\mathrm{H}_{f(\boldsymbol{x}^{\mathrm{i}})}$ is the Hessian of the function and $\boldsymbol{\nabla}_{\boldsymbol{x}^{\mathrm{i}}} f(\boldsymbol{x}^{\mathrm{i}})$ is the gradient of the function, both with respect to the variational parameters at iteration i. This iteration scheme converges quadratically for convex functionals. However, evaluation of the inverse of the Hessian is often computationally costly, and is not needed to reach convergence. In many applications (machine learning i.e.), the inverse of the Hessian is often replaced with a single scalar, the *learning rate*, often denoted as η. The simplified iterative scheme

$$\boldsymbol{x}^{\mathrm{i}+1} = \boldsymbol{x}^{\mathrm{i}} - \eta\boldsymbol{\nabla}_{\boldsymbol{x}^{\mathrm{i}}} f(\boldsymbol{x}^{\mathrm{i}}) \tag{3.13}$$

is the simple gradient descent method. The simple gradient descent method is guaranteed to converge (when the functional is convex) if the learning rate is smaller than some threshold value. If it is larger than the threshold value, it diverges. On the other hand, if the learning rate is too small, the number of iterations needed for finding the minima of the functional may be very large. We need to tune the learning rate to be in that sweet spot such that the functional converges, and converges within an acceptable time-frame.

In machine learning the functional to be minimized is often called a cost (or loss) function. For our system, we want to minimize the energy functional

$$\langle E \rangle = \int \mathrm{d}\boldsymbol{r} E_L(\boldsymbol{r};\boldsymbol{\alpha}) p(\boldsymbol{r};\boldsymbol{\alpha}), \tag{3.14}$$

with respect to the variational parameters $\boldsymbol{\alpha}$. The gradient with respect to the variational parameters of the energy functional becomes

$$\boldsymbol{\nabla}_{\boldsymbol{\alpha}} \langle E \rangle = 2(\langle \boldsymbol{\nabla}_{\boldsymbol{\alpha}} \ln \Psi_T E_L \rangle - \langle \boldsymbol{\nabla}_{\boldsymbol{\alpha}} \ln \Psi_T \rangle \langle E_L \rangle), \tag{3.15}$$

and we need to sample this equation in the MCMC. A derivation of Equation 3.15 can be found in Appendix A.2. Three integrals are needed to evaluate this expression, and we evaluate them by sampling the local gradient with respect to the variational parameters of the trial wave function

$$\left\langle \frac{\boldsymbol{\nabla}_{\boldsymbol{\alpha}} \Psi}{\Psi} \right\rangle \approx \frac{1}{M} \sum_{i=1}^{M} \frac{\boldsymbol{\nabla}_{\boldsymbol{\alpha}} \Psi_T(\boldsymbol{r};\boldsymbol{\alpha})}{\Psi_T(\boldsymbol{r};\boldsymbol{\alpha})} p(\boldsymbol{r};\boldsymbol{\alpha}), \tag{3.16}$$

the energy

$$\langle E \rangle \approx \frac{1}{M} \sum_{i=1}^{M} E_L(\boldsymbol{r};\boldsymbol{\alpha}) p(\boldsymbol{r};\boldsymbol{\alpha}), \tag{3.17}$$

and their product

$$\left\langle \frac{\boldsymbol{\nabla}_{\boldsymbol{\alpha}} \Psi}{\Psi} E \right\rangle \approx \frac{1}{M} \sum_{i=1}^{M} E_L(\boldsymbol{r};\boldsymbol{\alpha}) p(\boldsymbol{r};\boldsymbol{\alpha}) \frac{\boldsymbol{\nabla}_{\boldsymbol{\alpha}} \Psi(\boldsymbol{r};\boldsymbol{\alpha})}{\Psi(\boldsymbol{r};\boldsymbol{\alpha})} p(\boldsymbol{r};\boldsymbol{\alpha}). \tag{3.18}$$

The variational parameters is then updated by the iterative scheme

$$\boldsymbol{\alpha}^{(k+1)} = \boldsymbol{\alpha}^{(k)} - \eta \boldsymbol{\nabla}_{\boldsymbol{\alpha}} \langle E \rangle , \tag{3.19}$$

where the superscript k denotes how many times the variational parameters have been updated, until convergence in $\boldsymbol{\alpha}$. A typical stopping criterion for the optimization algorithm would be to check if the L2-norm of the difference between the kth and $k+1$th iteration is less than some tolerance, ϵ. If

$$\left\| \boldsymbol{\alpha}^{(k+1)} - \boldsymbol{\alpha}^{(k)} \right\|_2^2 \leq \epsilon \tag{3.20}$$

we stop the iterative scheme, and return the variational parameters $\boldsymbol{\alpha}^{(k+1)}$ as our optimal parameters.

Since we are only looking at a single variational parameter, α, we could have found the second derivative of the expected value of the energy with respect to α. However, for further work regarding a more general wave function, we stayed with a gradient descent method that can optimize an arbitrary number of variational parameters with relatively low computational cost.

3.3.1. ADAM

The simple gradient descent method will converge under the right circumstances, but may not be the fastest and most efficient optimizing scheme. In a multidimensional variational parameter space, the negative of the gradient will typically not point directly to the minima, and typically we end up moving from one side of a 'valley' in the cost function to the other. To make this scheme more efficient *momentum* is added, which is a weighted addition of the previous gradients to the calculation. This way the gradient and the momentum will point more along the valley towards the minima, and less 'across the valley'. Another implementation which improves the choice of step length in each dimension (specifically for non-convex functions), is the Root Mean Squared Propagation (RMSProp). A third improvement, mostly designed to escape local minima of the cost function, is a stochastic batching of the data, often referred to as Stochastic Gradient Descent. For more information on optimization algorithms, we refer you to chapter 8 of (Goodfellow et al., 2016).

We have implemented ADAM (derived from 'adaptive moments') as our optimizer. The ADAM optimizer (Kingma and Ba, 2017) is a method that utilises momentum and RMSProp, both with bias corrections. The algorithm keeps track of the first and second moments of the gradient, then bias corrects them, for then to use the second moment as an RMSProp rescaler of the gradient with momentum. β_1 and β_2 are exponential decay factors for the their respective moments. By appropriating this into our iteration scheme for finding the minima of the expected energy, we can update our variational parameters, $\boldsymbol{\alpha}$, as follows

$$
\begin{aligned}
\boldsymbol{g}_t &= \nabla_{\boldsymbol{\alpha}} \langle E \rangle (\boldsymbol{\alpha}_t) \\
\boldsymbol{m}_t &= \beta_1 \boldsymbol{m}_{t-1} + (1-\beta_1)\boldsymbol{g}_t \\
\boldsymbol{s}_t &= \beta_2 \boldsymbol{s}_{t-1} + (1-\beta_2)\boldsymbol{g}_t \cdot \boldsymbol{g}_t \\
\hat{\boldsymbol{m}}_t &= \frac{\boldsymbol{m}_t}{1-\beta_1^t} \\
\hat{\boldsymbol{s}}_t &= \frac{\boldsymbol{s}_t}{1-\beta_2^t} \\
\boldsymbol{\alpha}_{t+1} &= \boldsymbol{\alpha}_t - \eta_t \frac{\hat{\boldsymbol{m}}_t}{\sqrt{\hat{\boldsymbol{s}}_t} + \epsilon}
\end{aligned}
$$

where $\boldsymbol{m}_t = \mathbb{E}[g_t]$ and $\boldsymbol{s}_t = \mathbb{E}[g_t^2]$ are the running average of both the first and second moment of the gradient. $\epsilon \sim 10^{-8}$ is a regularization constant to prevent divergence and singularities. In the update equation the fraction is performed element-wise.

3.4. Parallelization

Parallelizing a single Markov chain is not a straightforward task. However, MCMC sampling with multiple chains is so-called *embarrassingly parallelizable*, which means there is little to no effort to divide the workload as the computations are independent. We will therefore implement procedures where we let multiple chains sample independently in parallel. Note that each chain still needs sufficient time to converge towards the stationary distribution, i.e., each will still need a sufficient number of Monte Carlo cycles to propagate in its workload.

3.5. Automatic Differentiation

Automatic differentiation is a tool which uses numerical methods to accurately calculate the derivatives of functions, with many applications, maybe most notably in the backpropagation algorithm in machine learning. A derivative of a function can be determined by finding the derivative of every operation in the function and we can use the chain rule to calculate the derivative. In backpropagation the deivative is calculated by reversing all the operations, but it can just as easily be calculated from input to output.

We decided to use JAX (Bradbury et al., 2018), a machine learning framework developed by Google Research teams, for automatically calculating the needed gradients. Our ambition is to make a Variational Monte Carlo method which can take an arbitrary trial wave function, and automatically carry out the computations needed to find a good approximation to its ground state energy. Due to the inner workings of JAX, our formulation of the trial wave function in the interacting case is not straightforward to automatically differentiate. In this project we limit ourselves to implement automatic differentiation for the non-interacting case to investigate its viability.

3.6. The Variational Monte Carlo Sampler

Our VMC framework is implemented as a Python package and can be found here https://github.com/nicolossus/FYS4411-Project1. The Python code is written in a highly vectorized manner, primarily by using functionality offered by NumPy (Harris et al., 2020) and JAX. The VMC framework has implementations of all the methods previously discussed. We have implemented a base class for a sampler with all methods except for the specific sampling step of an algorithm. Both the RWM and LMH sampler then inherits the base class and only needs to implement one step of the respective algorithm. Our VMC sampler have three phases; (i) tuning phase, (ii) optimization phase, and (iii) sampling phase. In the tuning phase, the sampler searches for the optimal scale parameter. Similarly, in the optimization phase, the sampler searches for the optimal variational parameter. Finally, in the sampling phase the sampler obtains energy samples used to compute the expectation value. Listing 1 outlines the interface of our VMC framework with explanatory comments included. The `sample` method, in particular, is designed to be flexible and let the user adjust the sampler's knobs.

```
import vmc

# Initialize a system; here we use the class for a spherical
# harmonic oscillator with non-interacting bosons (SHONIB)
system = vmc.SHONIB()

# Set MCMC method; either RWM or LMH. Both take the system
# object as constructor argument
rwm = vmc.RWM(system)

# Perform sampling
res = rwm.sample(
    nsamples,                # no. of energy samples to obtain
    initial_positions,       # initial spatial configuration
    alpha,                   # (initial) variational parameter
    nchains=1,               # no. of Markov chains
    seed=None,               # seed can be set for reproducibility
    tune=True,               # whether to tune scale parameters
    tune_iter=10000,         # maximum tuning cycles
    tune_interval=500,       # cycles between each tune update
    tol_tune=1e-5,           # tolerance level used to stop tune early
    optimize=True,           # whether to optimize alpha
    max_iter=50000,          # maximum optimize cycles
    batch_size=500,          # cycles in a batch used in optimization
    gradient_method='adam',  # specify optimization method
    eta=0.01,                # optimizer's learning rate
    tol_optim=1e-5,          # tolerance used to stop optimizer early
    early_stop=True,         # whether to use early stopping or not
    log=True,                # whether to show logger and progress bar
    logger_level="INFO",     # the level of logger
    **kwargs                 # kwargs depends on the MCMC method;
)                            # set scale for RWM and dt for LMH

# The results are returned in a pandas.DataFrame. They can
# be saved directly by using the sampler's `.to_csv` method
rwm.to_csv("filename.csv")
```

Listing 1: *Example usage of the VMC framework.*

4. Results and Discussion

4.1. Validating the VMC Framework

Figure 4.1 shows the VMC computations of the expected energy and the corresponding variance with both the analytical and automatic differentiation approaches for a system of $N = 1, 10, 100, 500$ non-interacting bosons in a spherical harmonic oscillator. We here use a grid search for the optimal variational parameter which is computationally expensive and generally infeasible for a non-coarse grid. However, in this case where we know the exact values, a grid search with a grid containing the exact value can be used to validate the sampling approach and implementation. As can be seen in the figure, the exact energy $\langle E \rangle / N = 3/2\,\hbar\omega_{\text{ho}}$ with $\text{Var}(E)/N = 0$ at $\alpha = 1/2$ are found by both approaches, which indicates that the sampler works well and the implementation of the system is correct. The figure also illustrates the convex nature of the variational parameter optimization problem. In this study we optimize the variational parameter with respect to the expectation value of the energy, as described in Section 3.3. However, the figure indicates that the variance might be more optimal as the optimization target since it tends to yield rather large values for non-optimal variational parameter values.

Table 4.1 tabulates the computed expectation values of the energy using the same spatial configuration of interacting bosons in a spherical harmonic oscillator with two different implementations. One implementation is based on closed-form expressions in the linear domain whereas the other on closed-form expressions in the logarithmic domain (see Section 2.2.6). As can be seen in the table, the expected energy calculations are identical for a given system size. As the implemented closed-form expressions found by different routes are identical, this further indicates the validity of the implementation.

Table 4.1: *Comparison between the implementation of analytical expressions for a system of interacting of bosons in a spherical harmonic oscillator. One is performed in the linear domain and the other in the log domain.*

# of bosons	$\langle E \rangle$ linear domain	$\langle E \rangle$ log domain
2	3.001	3.001
5	7.512	7.512
10	15.052	15.052
20	30.236	30.236
50	76.497	76.497
100	155.915	155.915
200	324.771	324.771
500	897.372	897.372
1000	2,095.517	2,095.517

Figure 4.2 shows the three phases of the sampler for both the RWM and LMH algorithms used on a system of $N = 100$ bosons in a spherical trap both with and without interactions. First, the sampler tunes the proposal scale parameter by increasing or decreasing it according to a look-up table (see **`vmc/utils/sampler_utils.py`** in the source code) that tries to achieve a target acceptance rate. Then, the sampler search for the optimal variational parameter by employing gradient descent optimization, specifically, the ADAM optimizer discussed in Section 3.3.1. The updates in the tuning and optimization phases are done after a set interval of MCMC steps. Both the tuning and optimization phase also employs early

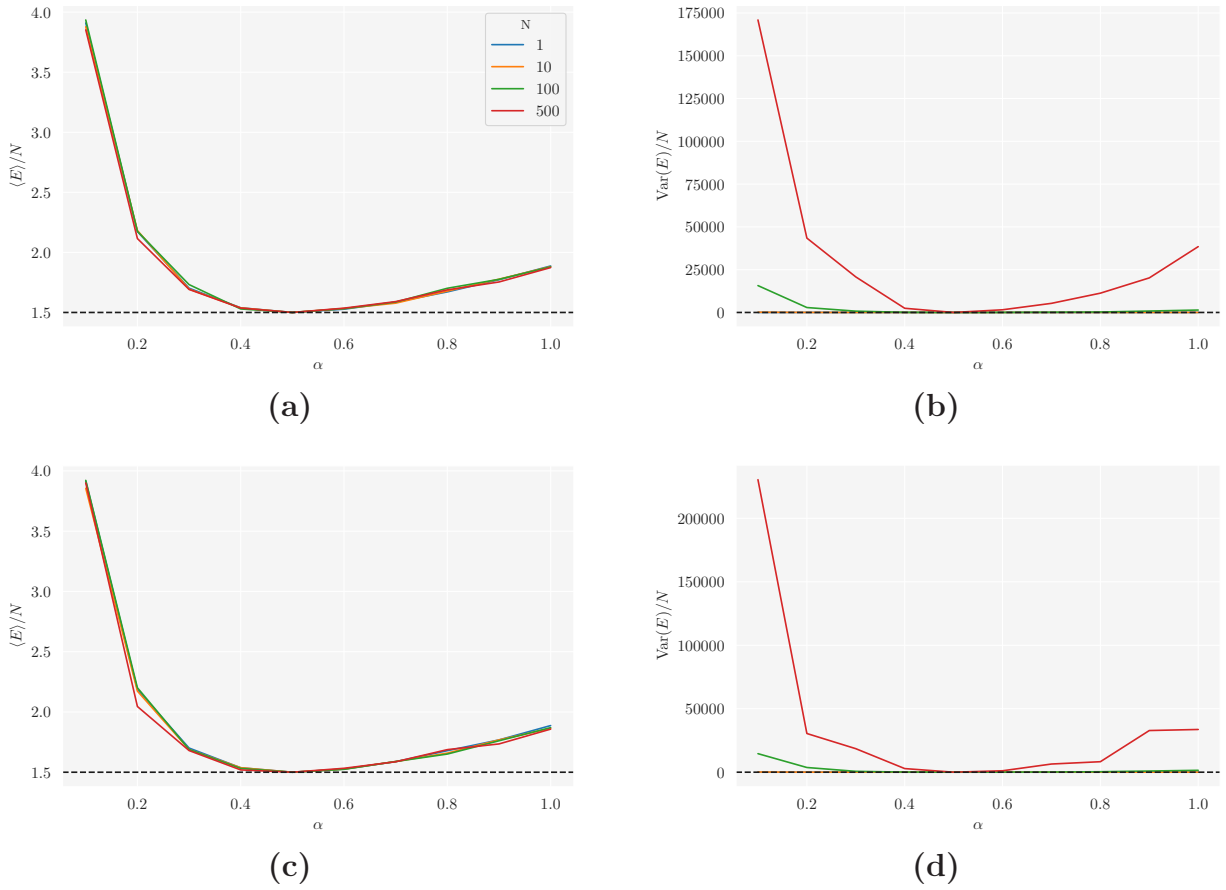


Figure 4.1: *The expected value of the energy, $\langle E \rangle$, and variance, $\mathrm{Var}(E)$, scaled by the number of bosons, N, in a grid search for the optimal variational parameter α. The system is a spherical harmonic oscillator with non-interacting bosons and the grid of variational parameters $\alpha \in [0.1, 1.0]$ with step size 0.1. In **(a)** and **(b)** the expectation values are computed with the closed-form expressions, whereas in **(c)** and **(d)** the computations are done with automatic differentiation through JAX. Each computed point in the grid is the average over 16 tuned Markov chains with $20,000$ energy samples used to calculate the expectation values in each.*

stopping, i.e., the phase will end if there is no change in values within a specified tolerance level between consecutive tuning or optimization intervals. Finally, the energy is sampled in what is hopefully the stationary state of the Markov chain. As can be seen in the figure, the different Markov chains all seem to converge well towards a stationary state by the time the sampler reaches the sampling phase.

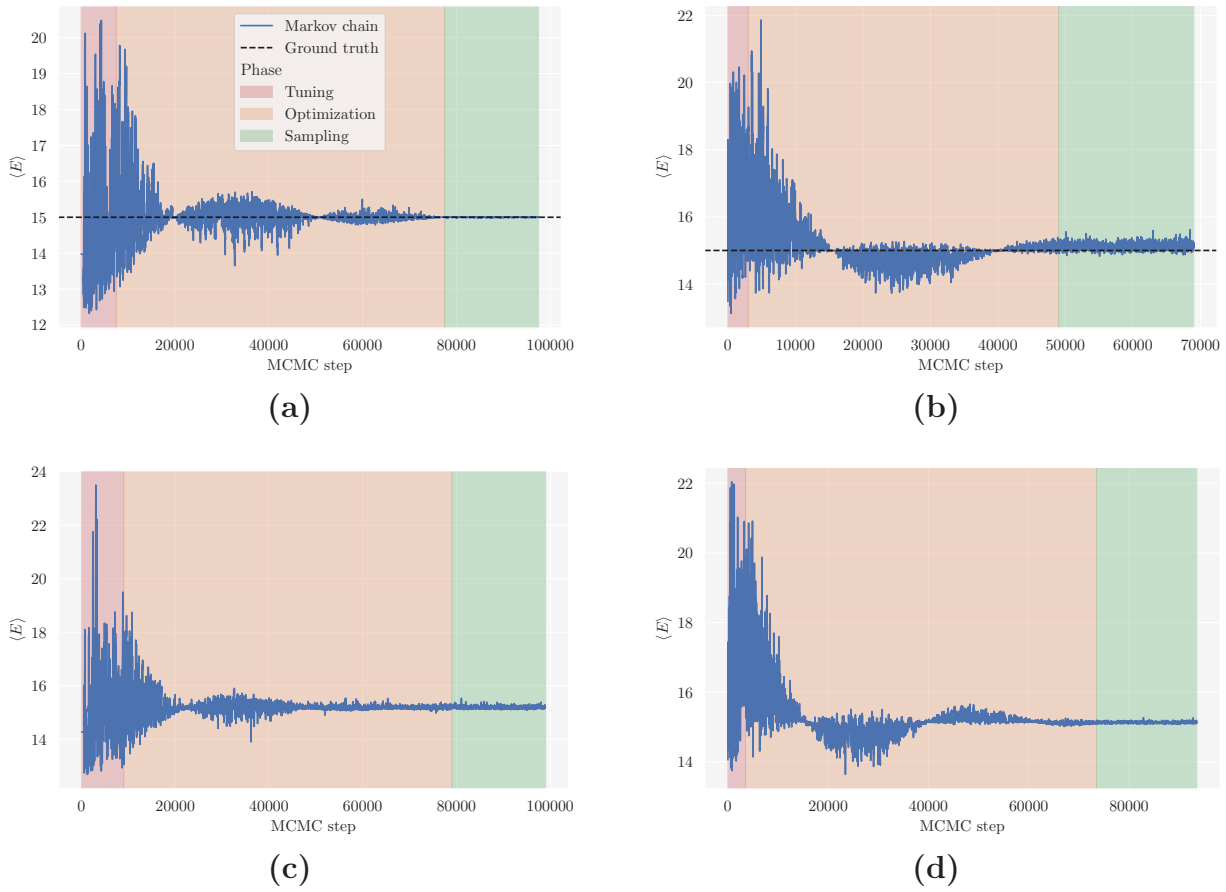


Figure 4.2: *The time evolution of a Markov chain in the three phases of the sampler; (i) tuning of the proposal scale parameter; (ii) gradient descent optimization of the variational parameter; and (iii) sampling of the variational energy. The updates in the tuning and optimization phase are done on a batch of samples, here 500 for tuning and 1000 for optimization. If the updated value remains the same between consecutive batches, the sampler will end the phase early. Here, the system consists of $N = 100$ bosons in a spherical harmonic oscillator both with and without interactions. The initial variational parameter is set to $\alpha_0 = 0.4$. In* ***(a)*** *and* ***(b)*** *we see the RWM and LMH algorithm on a system of non-interacting bosons, respectively. In* ***(c)*** *and* ***(d)*** *we see the RWM and LMH algorithm on a system of interacting bosons, respectively.*

4.2. Comparing the RWM and LMH algorithms

Figure 4.3 shows the average energies, average statistical error and the variance in the energy, against the number of samples, M, for the RWM and LMH sampling algorithms on a spherically symmetric harmonic oscillator system of 100 particles. The average energy per particle over 16 chains is close to the exact energy for both the RWM and LMH samplers. The statistical error per particle in the RWM chains are almost a factor of 10 larger than for the LMH chains, and the variance in the energy per particle shows that there are much larger fluctuations in the energy using the RWM algorithm than the LMH. This indicates that guiding the proposals according to the gradient flow of the probability density of the trial wave function, in fact does increase the sampling quality by a factor.

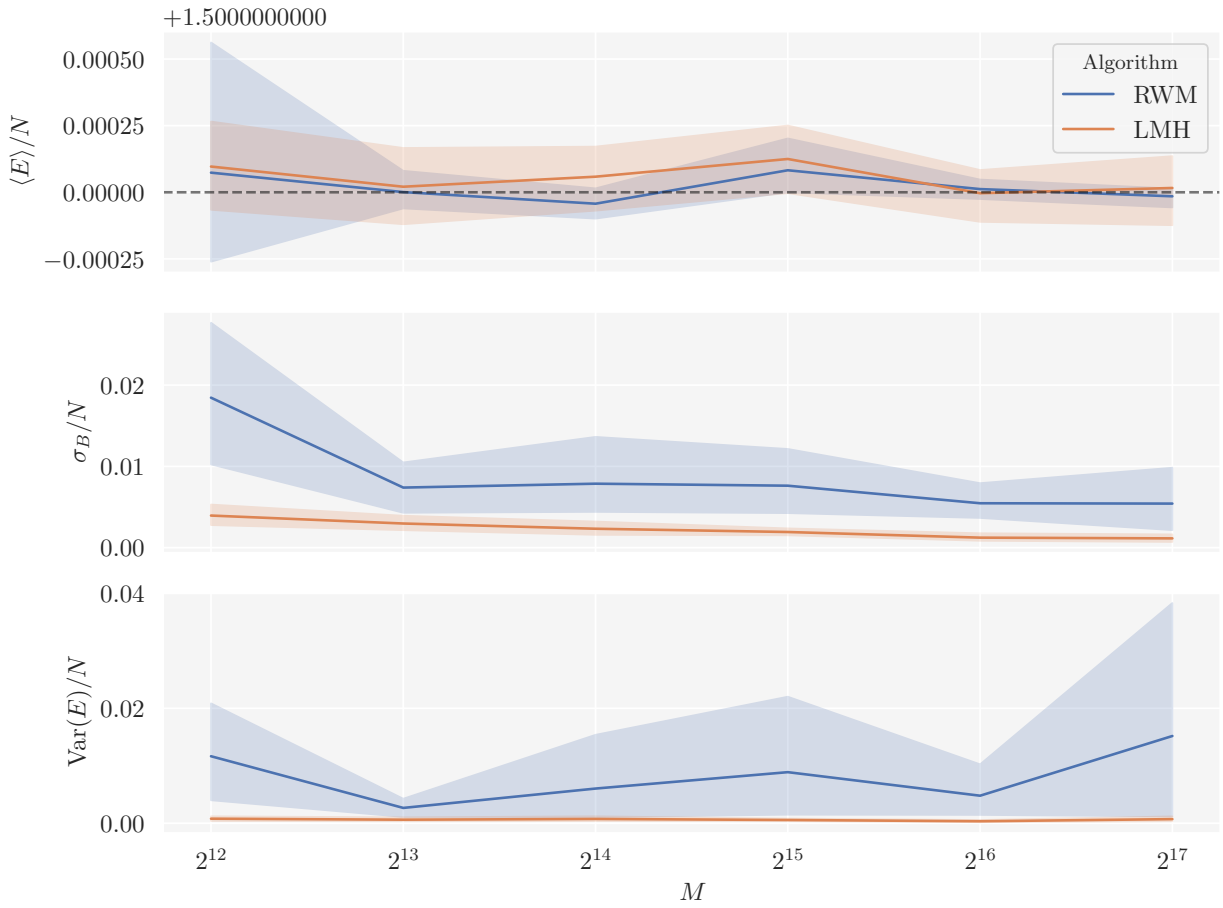


Figure 4.3: *The sampled expected energy per particle* ($\langle E \rangle /N$)*, the standard error* (σ_B/N) *and the variance* ($\mathrm{Var}(E)/N$) *against the number of samples,* M*. The system is a spherical harmonic oscillator, with an initial* α*-value of* 0.4*. For every MCMC-chain run (*16 *per* M*-value), there were* $10,000$ *tuning cycles, and* α *was optimized over* $50,000$ *cycles. The solid lines indicates the average expected value over the* 16 *chains, while the shaded region denotes the standard error of the mean. The blue lines and regions displays the results from the RWM algorithm, while the orange lines shows the corresponding results from the LMH algorithm.*

Figure 4.4 shows the evolution of variational parameter during the optimization phase. Here, the learning rate is $\eta = 0.05$ (see Figure A.1 and Figure A.2 in Appendix A.1 for a justification) and all 16 Markov chain are given the same initial $\alpha_0 = 0.4$. The blue and orange transparent lines show the evolution for each chain whereas the opaque blue and solid lines show the average across the chains for the RWM and LMH algorithms, respectively. As can be seen in the figure, the optimization phase is not directly affected by which algorithm is used. As long as both algorithms have found an optimal scale for its proposal distribution, the number of optimization epochs needed to converge close to the optimal variational parameter is effectively the same.

The non-interacting system performs simple calculations in a highly efficient manner as it is easy to vectorise the calculations. Figure 4.5 shows run times for 20000 tune cycles, 50000 optimization cycles and 2^{15} sampling cycles for the Random Walk Metropolis and Langevin Metropolis-Hastings sampling algorithms, and using the analytical and numerical

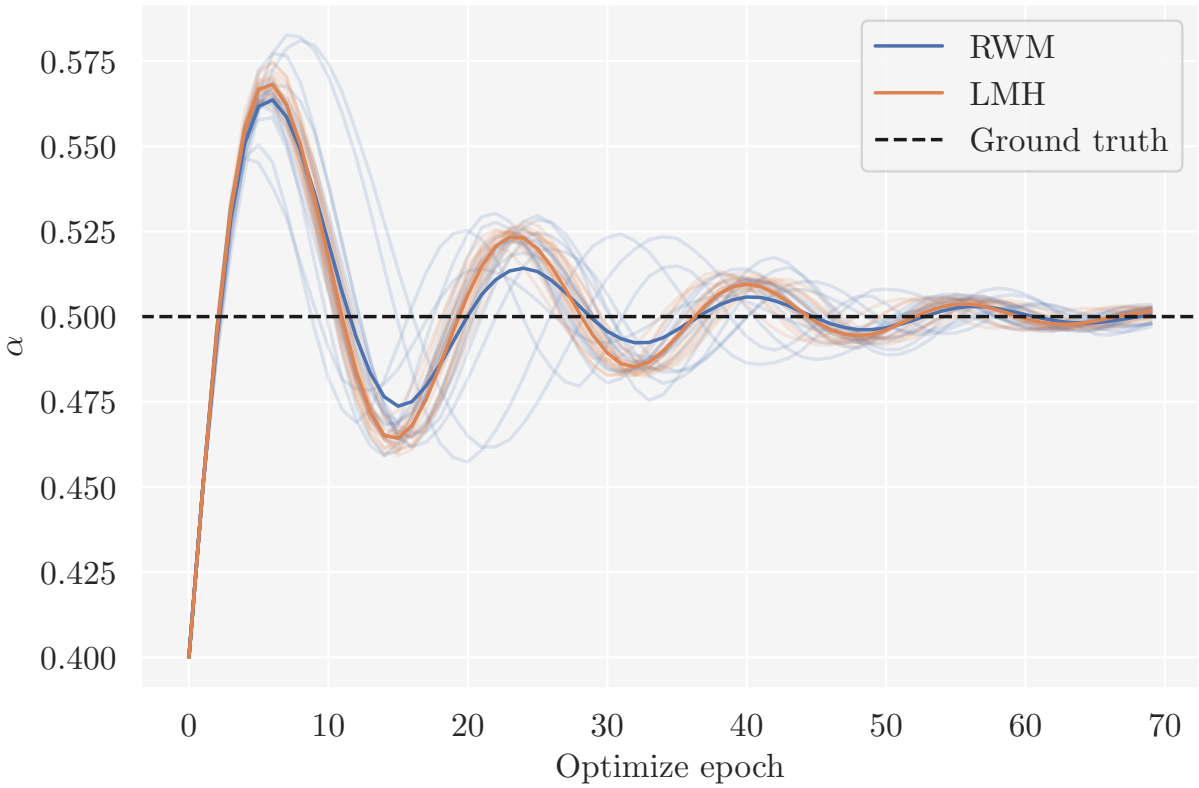


Figure 4.4: *Evolution of the variational parameter during the optimization. The transparent lines show the evolution for each chain and the solid line the average. The blue lines show the evolution with the RWM algorithm and the orange the LMH algorithm.*

wave functions. There were 10 runs for every $N \in [1, 10, 50, 100, 500]$ for every different sampler, and the error bars display the standard deviation of the 10 samples. Adding more particles to the system only slightly increases the sampling time. The RWM algorithm is faster than the LMH algorithm, and the numerical implementation is slower than the analytical wave function. It seems that for the analytical samplers the LMH seems to take a rough factor of 4 more time than its RWM counterpart, but for the numerical computations (which needs to numerically calculate the Green's function for every step) it was a factor between 6-8 longer run time for the LMH sampler.

The two comparisons in Figure 4.3 and Figure 4.5 show that the number of samples to achieve the same quality sampling is quite different between the two approaches, and the LMH sampling algorithm seems to be more effective, even with the compuational costs taken into account.

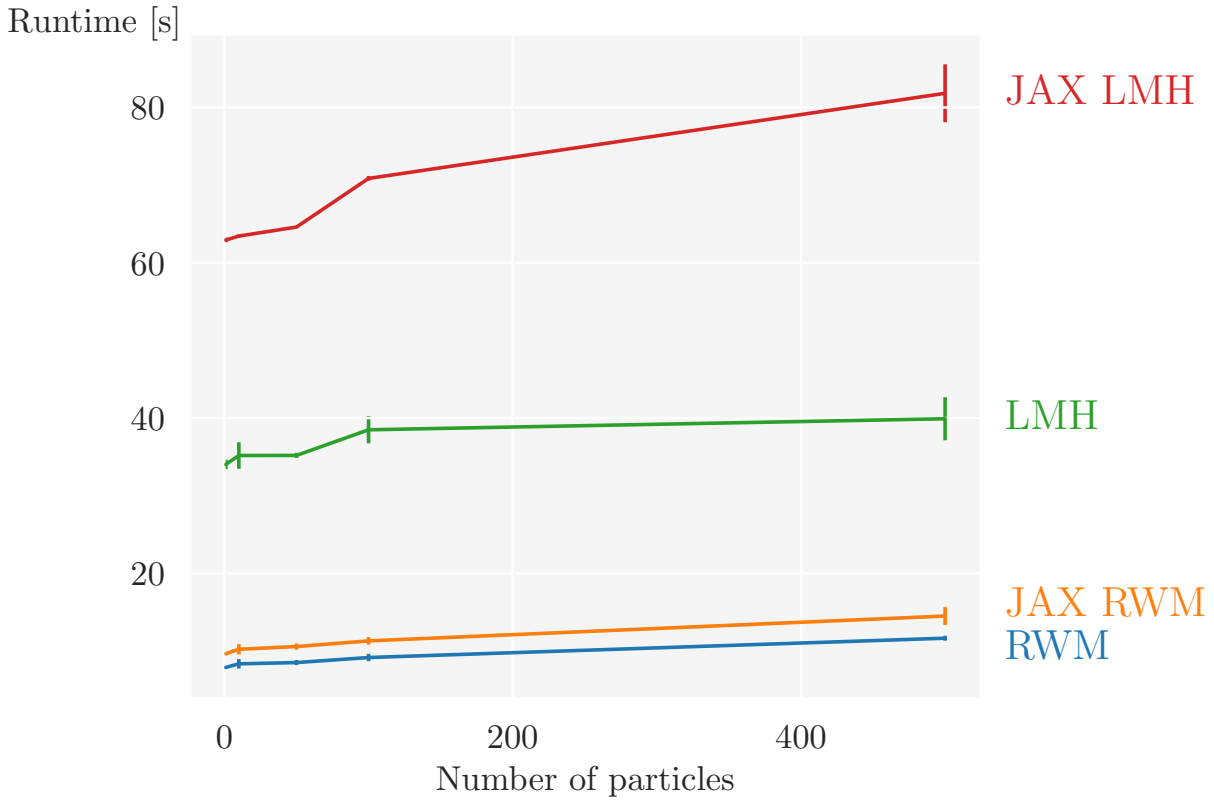
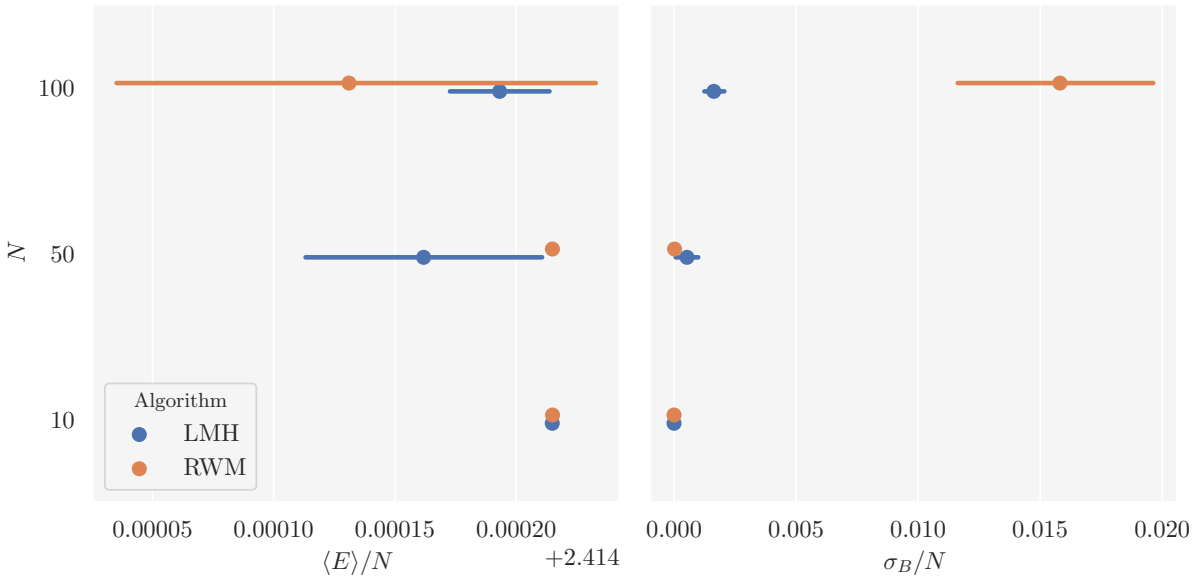


Figure 4.5: *Runtimes for Random Walk metropolis and Langevin Metropolis-Hastings algorithms on spherical non-interacting systems with an analytical and numerical wave function (JAX). The error bars extend one standard deviation from the mean.*

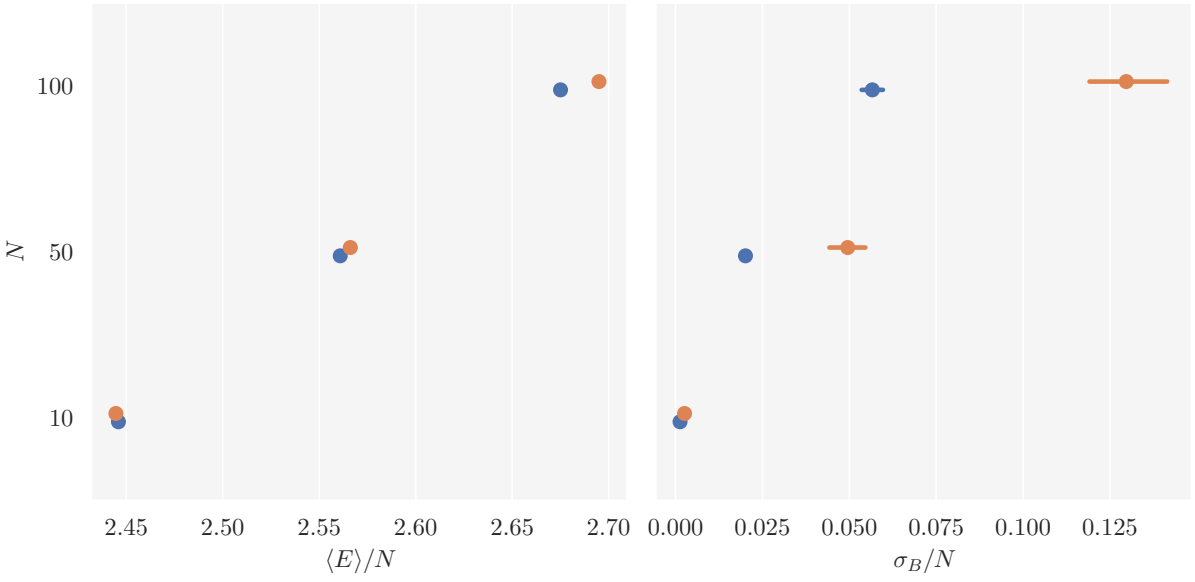
4.3. Estimations of the Ground State Energy

The elliptical potential described in Equation 2.10 yields a ground state energy per particle of $E_0/N = 2.414215\hbar\omega_0$ for non-interacting particles. Figure 4.6 displays the mean energies and mean standard error (points), together with their spread (one standard error of the mean confidence interval) of 16 threads of sampling for each sampling algorithm. The parameters for the sampling runs were all set to a maximum of $20,000$ tuning cycles, with a tuning interval of 1000. The initial alpha value was set to 0.5 and the threads had all a maximum of $50,000$ optimization cycles, with an update at every $1,000$th cycle. The number of samples collected were 2^{15}.

When the particles are interacting, the energy increases as a function of particle density in the trap. The collected mean energies together with the mean statistical error is tabulated in Table 4.2. The interacting bosons in the trap have a higher ground state than the non-interacting particles, and it increases with the number of particles. When the number of bosons in the trap increases, the density does too, meaning that the average distances to the nearest bosons decrease. Thus, we expect (and have seen in (DuBois and Glyde, 2001)) that the energy will continue to rise if we further increase the density. There is also the question of the quite large gap in average expected energies between the interacting systems when $N = 100$ between the RWM and LMH sampling algorithms. This may have to do with the low number of samples produced by the chains, and the LMH algorithm has proven more



(a)



(b)

Figure 4.6: *Sampled expected energies of the elliptical trap where the particles are non-interacting (**a**) and interacting (**b**). The sampled average energy per particle over 16 chains against the number of particles are plotted to the left, and in the right windows the average statistical error per particle given by the blocking method is plotted. The stretch in the points shows spread in value by one standard error of the mean. The sample size is $M = 2^{15}$, and there were a maximum of $20,000$ tuning cycles and $50,000$ optimization cycles, which could be less would the early stopping criterion be met.*

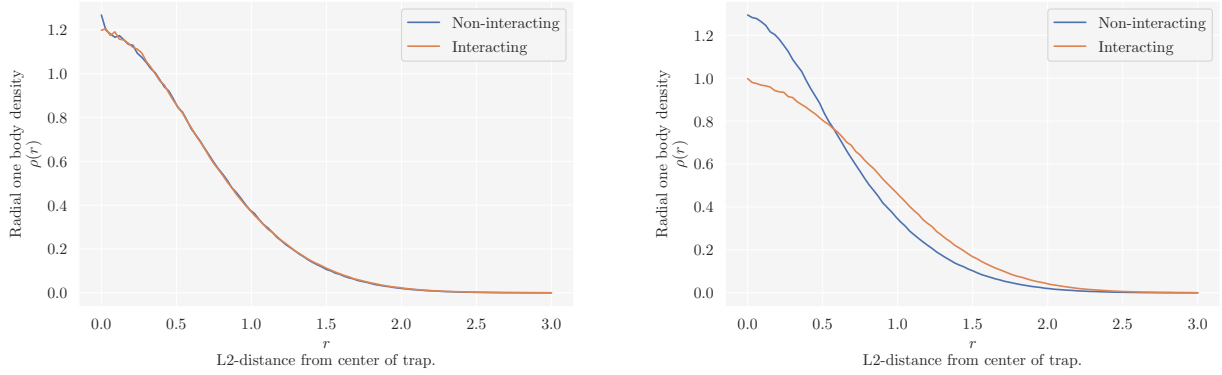
effective in its choice of proposals. However, both values are well within one statistical error of each other, and may simply be due to the stochasticity of the procedures.

Table 4.2: *Tabulated sampling results of* 16 *sampling runs for each system and algorithm. The number of samples was* 2^{15}*, with a maximum of* 50,000 *optimization cycles and a maximum of* 20,000 *tuning cycles. The system was a harmonic oscillator with an elliptical potential described by Equation 2.10, with varying number of bosons. The hard-sphere diameter,* a*, is set to* 0 *if there are no interactions between the bosons, while* $a = a_{\mathrm{Rb}} = 0.00433$ *if the particles are interacting.*

Measurement	LMH ($a = 0$)	RWM ($a = 0$)	LMH ($a = a_{Rb}$)	RWM ($a = a_{Rb}$)
$\langle E \rangle / (N = 10)[\hbar\omega_0]$	2.41421	2.41421	2.4459	2.4446
$\langle E \rangle / (N = 50)[\hbar\omega_0]$	2.4142	2.4142	2.5609	2.5661
$\langle E \rangle / (N = 100)[\hbar\omega_0]$	2.4142	2.4141	2.6750	2.6950
$\sigma_{\mathrm{B}}(N = 10)[\hbar\omega_0]$	$2 \cdot 10^{-7}$	$3 \cdot 10^{-7}$	$1 \cdot 10^{-3}$	$3 \cdot 10^{-3}$
$\sigma_{\mathrm{B}}(N = 50)[\hbar\omega_0]$	$5 \cdot 10^{-4}$	$2 \cdot 10^{-5}$	$2 \cdot 10^{-2}$	$5 \cdot 10^{-2}$
$\sigma_{\mathrm{B}}(N = 100)[\hbar\omega_0]$	$2 \cdot 10^{-3}$	$2 \cdot 10^{-2}$	$6 \cdot 10^{-2}$	$1 \cdot 10^{-1}$

4.4. One-Body Densities

Figure 4.7 shows the sampled one-body densities for 10 and 100 particles in a spherical harmonic oscillator with the interactions turned on and off. As the number of particles increases, the density of particles increases, and the interactions between the particles become a larger factor. For 10 particles, the one body densities with and without interactions are almost identical. This means that for 10 particles, the interactions do not have a large impact on the system. For 100 particles, the radial one body densities for the interacting and non-interacting cases differ more. The particles in the interacting case are more spread out in the 2-dimensional space than the non-interacting case, which is expected, due to the repulsiveness of the Jastrow correlation factor.



(a) **(b)**

Figure 4.7: ***(a)*** *Comparison between the radial one body densities for* 10 *particles with and without interactions (Jastrow factor* $a = 0.00433$*) in 2-dimensional space.* ***(b)*** *Comparison between the radial one body densities for* 100 *particles with and without interactions (Jastrow factor* $a = 0.00433$*) in 2-dimensional space.*

5. Conclusion

In this study we have investigated the variational Monte Carlo (VMC) method for estimating the ground state energy of an ultracold, dilute Bose gas in both a spherical and elliptical harmonic trap. We have used a trial wave function composed of a single particle Gaussian with a single variational parameter and a hard sphere Jastrow factor for pair correlations. We have built a VMC framework in Python with the Markov chain Monte Carlo sampling algorithms *random walk Metropolis* (RWM) and *Langevin Metropolis-Hastings* (LMH). The framework contained procedures for tuning the scale of the proposal distributions and gradient descent optimization of the variational parameter. Moreover, we also implemented automatic differentiation that can be applied to systems of non-interacting bosons.

We found that the size of the statistical error of the variational energy found with the RWM algorithm is larger than for the LMH algorithm, given the same number of energy samples. This means that the RWM algorithm is slower than the LMH in terms of the total number of MC cycles needed to achieve a result with similar uncertainty. The statistical errors in Figure 4.3 demonstrate that LMH asymptotically mix considerably faster than RWM, and thus would need fewer samples to find the ground state energy of the system with a given precision. In order to achieve the same statistical error in the RWM, the number of samples needs to be considerably higher. The increased number of accepted proposals in LMH compared to RWM also decrease the correlation between the samples. However, in terms of computation time the RWM algorithm performs better than LMH, as each step of the algorithm is fast compared to LMH. Figure 4.5 shows that the computational cost of the LMH is, by a significant factor, larger than for RWM, which is due to the calculations of the gradients and Green's function in the step function. Taking both the accuracy, computational time and also the comparable convergence rates in Figure 4.4 into account, the LMH algorithm seems to be the better choice of the two sampling algorithms in terms of computational efficiency for estimation of expectation values.

We were successful in employing automatic differentiation to the system of non-interacting bosons. This comes at a significant computational cost, as illustrated by Figure 4.5, but the advantage is that it mitigates the need for closed-form expressions.

We found the analytical ground state of the non-interacting boson system to be $\frac{3}{2}\hbar\omega_{\text{ho}}$ for the sperical trap, and $2.414215\hbar\omega_{\text{ho}}$ for the elliptical trap. When the interactions simulated by the Jastrow correlation factor were included, we found an increase in the energy per particle proportional to the particle density. We calculated energies for three interacting system sizes, $N = [10, 50, 100]$, and found the ground state energies per particle for the elliptical systems to be $(2.446 \pm 0.001)\hbar\omega_{\text{ho}}$,$(2.56 \pm 0.02)\hbar\omega_{\text{ho}}$, $(2.68 \pm 0.06)\hbar\omega_{\text{ho}}$, respectively. For the spherical systems we found there to be similar increases in energy due to interactions of the particles, and they are tabulated in Table A.1.

The one-body densities reveal that, when the bosons are interacting, the state of the system is dependent on the density of bosons in the trap. One should think that when the interacting particles are close to one another, they interact with more frequency and strength, thus having a larger impact on the system state. If the interactions between the bosons are attractive, we therefore expect to observe a narrowing of the one-body density, and a spreading if the interactions were repulsive. We observe in Figure 4.7 that the one-body density is more spread out in configuration space, which corresponds to repulsive interactions, and an increase in the spread as a function of the density.

6. Future Work

In the present work we have built an automated VMC framework for bosonic systems. Building off of the present work, an interesting avenue for further research would be to generalize the automatic differentiation procedures to also apply to interacting systems. Moreover, the framework should be fairly easy to expand to be able to handle fermionic systems as well. There are also partially unanswered questions regarding the difference in the correlations between the samples generated by the two sampling algorithms, like an estimate for a factor of additional samples needed to generate the same statistical error using the RWM, when we know the statistical error for LMH with a given samples size, or the number of steps between uncorrelated samples for each algorithm. An implementation of an unsupervised machine learning method, like a *restricted Boltzmann machine* based on the same sampler, will be implemented for further analysis of the bosonic system in project 2.

References

Einstein, Albert (1924). "Quantentheorie des einatomigen idealen Gases (Quantum theory of monatomic ideal gases)". In: *Zeitschrift für Physik* 26, pp. 261–267 (cit. on p. 1).

– (1925). "Quantentheorie des einatomigen idealen Gases. Zweite Abhandlung (Quantum theory of monatomic ideal gases, part two)". In: *SBd Preuss. Akad. Wiss. Ber* 1.3 (cit. on p. 1).

Bose, Satyendra Nath (1924). "Plancks Gesetz und Lichtquantenhypothese (Planck's Law and Light Quantum Hypothesis)". In: *Zeitschrift für Physik* 26, pp. 178–181 (cit. on p. 1).

Anderson, M. H., J. R. Ensher, M. R. Matthews, C. E. Wieman, and E. A. Cornell (1995). "Observation of Bose-Einstein Condensation in a Dilute Atomic Vapor". In: *Science* 269.5221, pp. 198–201 (cit. on p. 1).

Dalfovo, Franco, Stefano Giorgini, Lev P. Pitaevskii, and Sandro Stringari (1999). "Theory of Bose-Einstein condensation in trapped gases". In: *Rev. Mod. Phys.* 71 (3), pp. 463–512 (cit. on p. 1).

DuBois, J. L. and H. R. Glyde (2001). "Bose-Einstein condensation in trapped bosons: A Variational Monte Carlo analysis". In: *Phys. Rev. A* 63 (2), p. 023602 (cit. on pp. 1, 8, 29).

Grenander, Ulf and Michael I. Miller (1994). "Representations of Knowledge in Complex Systems". In: *Journal of the Royal Statistical Society. Series B (Methodological)* 56.4, pp. 549–603 (cit. on pp. 1, 17).

Lester, W. A., B. L. Hammond, and P. J. Reynolds (1994). *Monte Carlo Methods in Ab Initio Quantum Chemistry*. World Scientific Lecture and Course Notes in Chemistry. Singapore: World Scientific (cit. on p. 4).

Pfau, David, James S. Spencer, Alexander G. D. G. Matthews, and W. M. C. Foulkes (2020). "Ab initio solution of the many-electron Schrödinger equation with deep neural networks". In: *Phys. Rev. Research* 2 (3), p. 033429 (cit. on p. 7).

Nilsen, J. K., J. Mur-Petit, M. Guilleumas, M. Hjorth-Jensen, and A. Polls (2005). "Vortices in atomic Bose-Einstein condensates in the large-gas-parameter region". In: *Phys. Rev. A* 71 (5), p. 053610 (cit. on p. 8).

Ross, Sheldon M. (2014). *Introduction to Probability Models*. 11th ed. Amsterdam, The Netherlands: Academic Press (cit. on p. 12).

Gelman, Andrew, John B. Carlin, Hal S. Stern, David B. Dunson, Aki Vehtari, and Donald B. Rubin (2014). *Bayesian Data Analysis*. 3rd ed. Texts in statistical science. Boca Raton, Fl: CRC Press (cit. on p. 12).

Metropolis, Nicholas, Arianna W Rosenbluth, Marshall N Rosenbluth, Augusta H Teller, and Edward Teller (1953). "Equation of State Calculations by Fast Computing Machines". In: *The Journal of chemical physics* 21.6, pp. 1087–1092 (cit. on p. 13).

Hastings, W. K (1970). "Monte Carlo sampling methods using Markov chains and their applications". In: *Biometrika* 57.1, pp. 97–109 (cit. on p. 13).

Roberts, Gareth O. and Jeffrey S. Rosenthal (2001). "Optimal Scaling for Various Metropolis-Hastings Algorithms". In: *Statistical Science* 16.4, pp. 351–367 (cit. on p. 17).

Salvatier, John, Thomas V. Wiecki, and Christopher Fonnesbeck (2016). "Probabilistic programming in Python using PyMC3". In: *PeerJ. Computer science* 2, e55. ISSN: 2376-5992. DOI: 10.7717/peerj-cs.55 (cit. on p. 17).

Flyvbjerg, H. and H. G. Petersen (1989). "Error estimates on averages of correlated data". In: *The Journal of Chemical Physics* 91.1, pp. 461–466. DOI: 10.1063/1.457480. eprint: https://doi.org/10.1063/1.457480. URL: https://doi.org/10.1063/1.457480 (cit. on p. 18).

Jonsson, Marius (2018). "Standard estimation by an automated blocking method". In: *Physical Review* (cit. on p. 19).

Goodfellow, Ian, Yoshua Bengio, and Aaron Courville (2016). *Deep Learning*. http://www.deeplearningbook.org. MIT Press (cit. on p. 21).

Kingma, Diederik P. and Jimmy Ba (2017). *Adam: A Method for Stochastic Optimization*. arXiv: 1412.6980 [cs.LG] (cit. on p. 21).

Bradbury, James et al. (2018). *JAX: composable transformations of Python+NumPy programs*. Version 0.2.5. URL: http://github.com/google/jax (cit. on p. 22).

Harris, Charles R. et al. (2020). "Array programming with NumPy". In: *Nature* 585.7825, pp. 357–362 (cit. on p. 22).

A. Appendix

A.1. Additional Results

Optimzer examination on a spherical harmonic oscillator system without interactions between the bosons.

Figure A.1 displays the evolution of the variational parameter, α, from ten different initial starting values in the range $\alpha \in [0.1, 1.0]$ using the RWM sampling algorithm. For most initial α-values the ADAM optimizer converges to the optimal variational parameter for the spherical harmonic oscillator, $\alpha^{\text{opt}} = 0.5$, although we see that the convergence is a little too slow for the smallest initial α when the learning rate is 0.01, and the learning rate should maybe be adjusted to $\eta = 0.05$ for faster convergence. When $\eta = 0.5$ we see that the initial values close to $\alpha = 0.1$ have a hard time converging within the epoch interval. Eventually maybe these also converge to the right value, as the gradients seem to have an easier time finding the optimal variational parameter from $\alpha > \alpha^{\text{opt}}$, however it seems the gradient at an α-value far enough away from the optimal seems to have a weak gradient. Figure A.2 displays the same examinations, only using the LMH sampling algorithm instead. The two plots display very similar results. The only remark to add is that more initial α-values seem to have a hard time converging when $\eta = 0.5$.

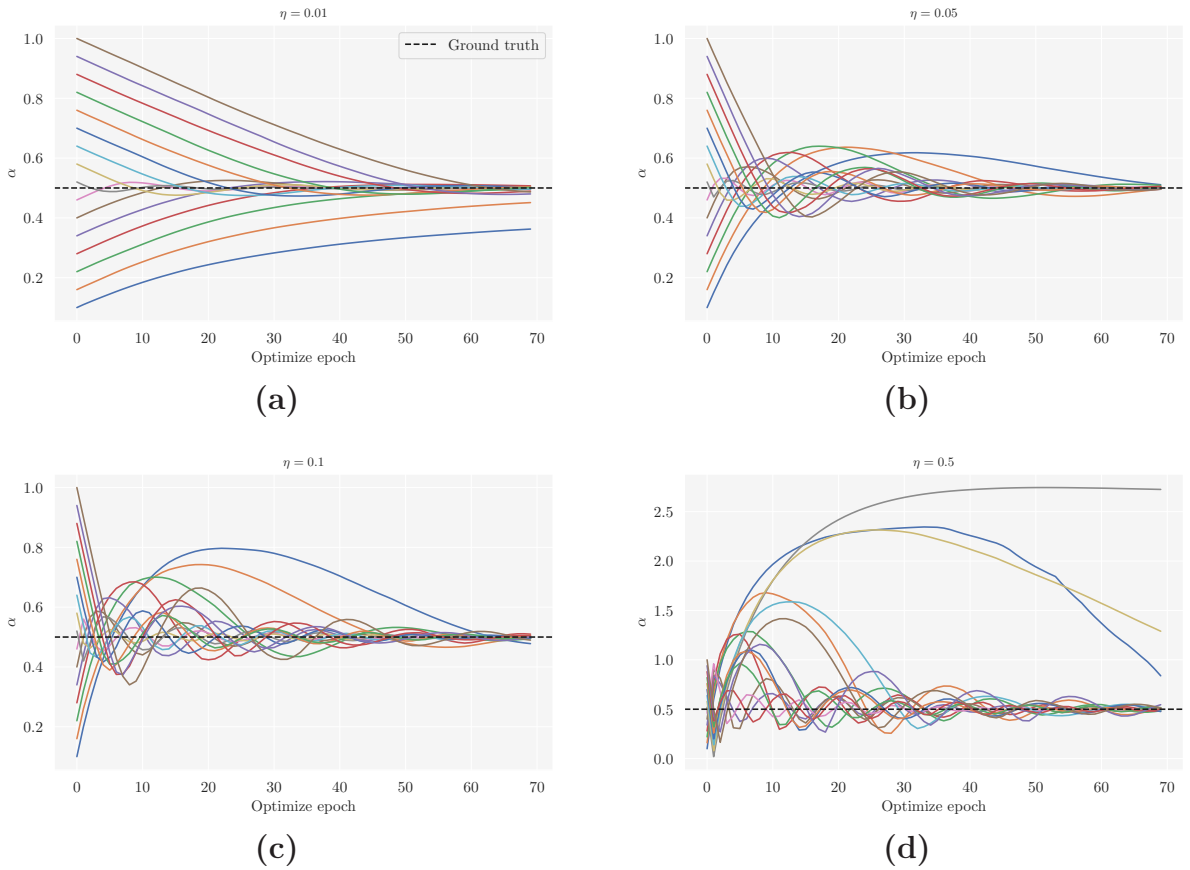


Figure A.1: *Optimization runs with multiple learning rates, η, examining the convergence of the variational parameter, α, starting from initial alpha $\in [0.1, 1.0]$ with ten equidistant values, as a function of optimization epochs. The sampling algorithm is the RWM.*

Figure A.2 is similar to Figure A.1, only here we use the LMH algorithm. Comparing Figure A.1 and Figure A.2 we see no faster convergence for the LMH sampling algorithm than for the RWM sampling algorithm.

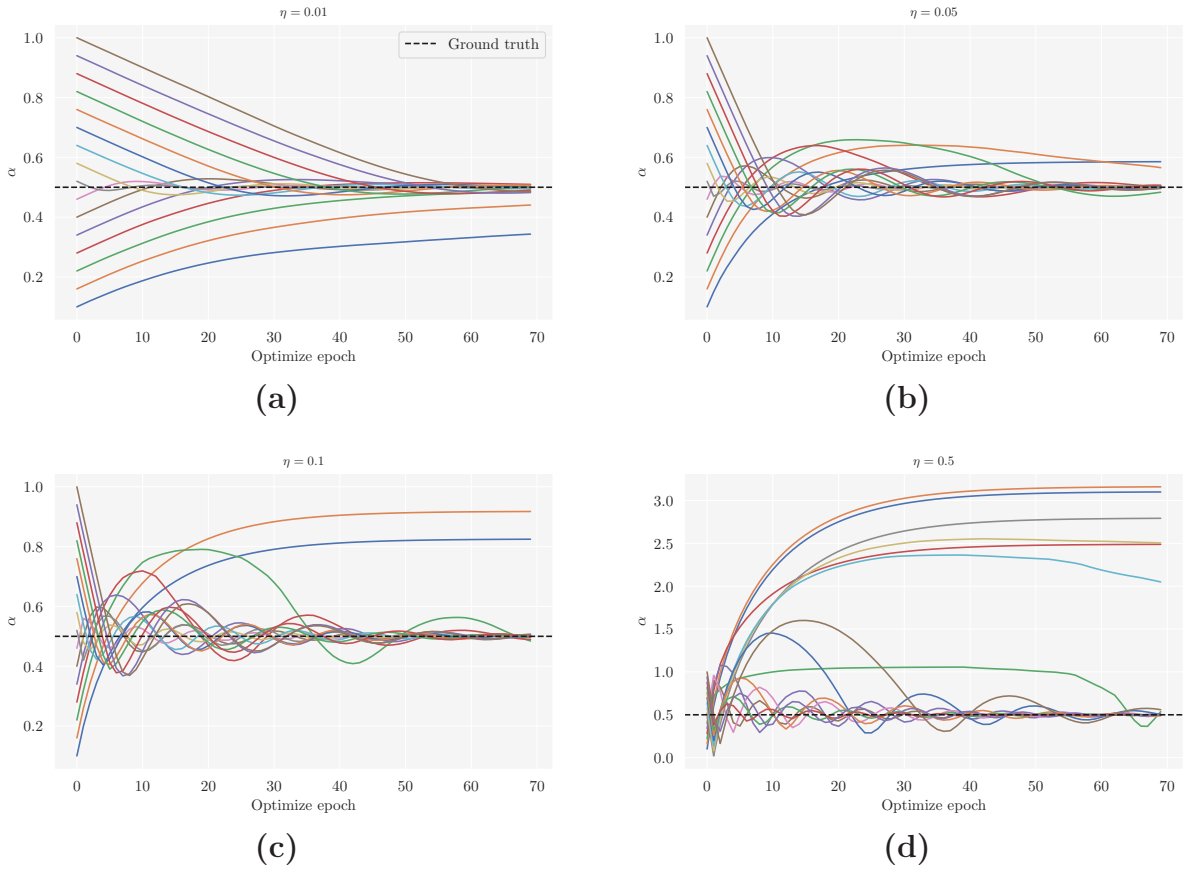


Figure A.2: *Optimization runs with multiple learning rates, η, examining the convergence of the variational parameter, α, starting from initial $\alpha \in [0.1, 1.0]$ with ten equidistant values, as a function of optimization epochs. The sampling algorithm is the LMH.*

Energy comparisons for interacting particles in a spherical harmonic oscillator In Figure A.3 variational grid searches for the non-interacting case, and $10, 50, 100$ particles with interactions are shown. The y-axis displays the energy per particle. A 95% confidence interval is shaded in light blue. The minimal energy per particle values together with their standard error, σ, are tabulated in Table A.1.

Number of particles	Interactions	$\frac{\langle E \rangle_{\min}}{N}$	σ	α^{opt}
50	Off	1.500	$0 \cdot 10^{0}$	0.5
10	On	1.51731	$8 \cdot 10^{-5}$	0.5
50	On	1.59490	$3 \cdot 10^{-4}$	0.5
100	On	1.66912	$3 \cdot 10^{-4}$	0.5

Table A.1: *The minimal energies of the grid searches for $10, 50, 100$ particles, together with the standard error and the optimal α values. The interactions column displays whether or not the particles are interacting with one another.*

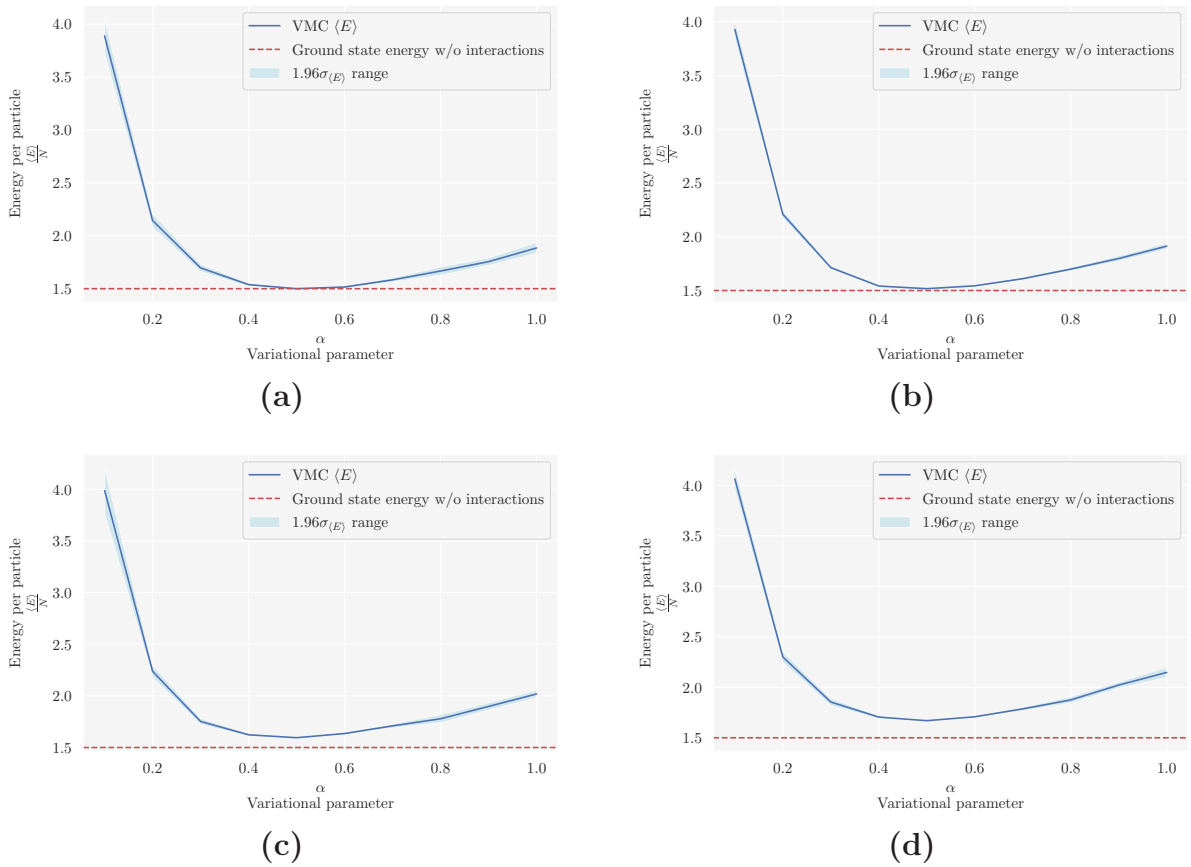


Figure A.3: ***(a)*** *Random Walk Metropolis with 50 non-interacting particles.* ***(b)*** *Random Walk Metropolis with ten interacting particles.* ***(c)*** *Random Walk Metropolis with 50 interacting particles.* ***(d)*** *Random Walk Metropolis with 100 interacting particles.*

As you can see in Table A.1, the minimal energy per particle is increasing with the number of particles when they are interacting. The density of particles within the potential increases with the number of particles unleashed in the same potential. This leads to more interactions between the particles, and thus a higher energy. Another important quantity, the standard error, is represented with the 95% confidence interval $[\mu \pm 1.96\sigma]$ by the shaded light blue in Figure A.3. The standard error is relatively small, but it should be possible to see the convex nature of it. It is decreasing from $\alpha = 0.1 \rightarrow \alpha = 0.5$, where it increases again.

Figure A.4 displays the VMC calculations of the grid searches in Figure A.3 all together in one plot. This way it is easier to see the increase in energy per particle as a function of the number of interacting particles.

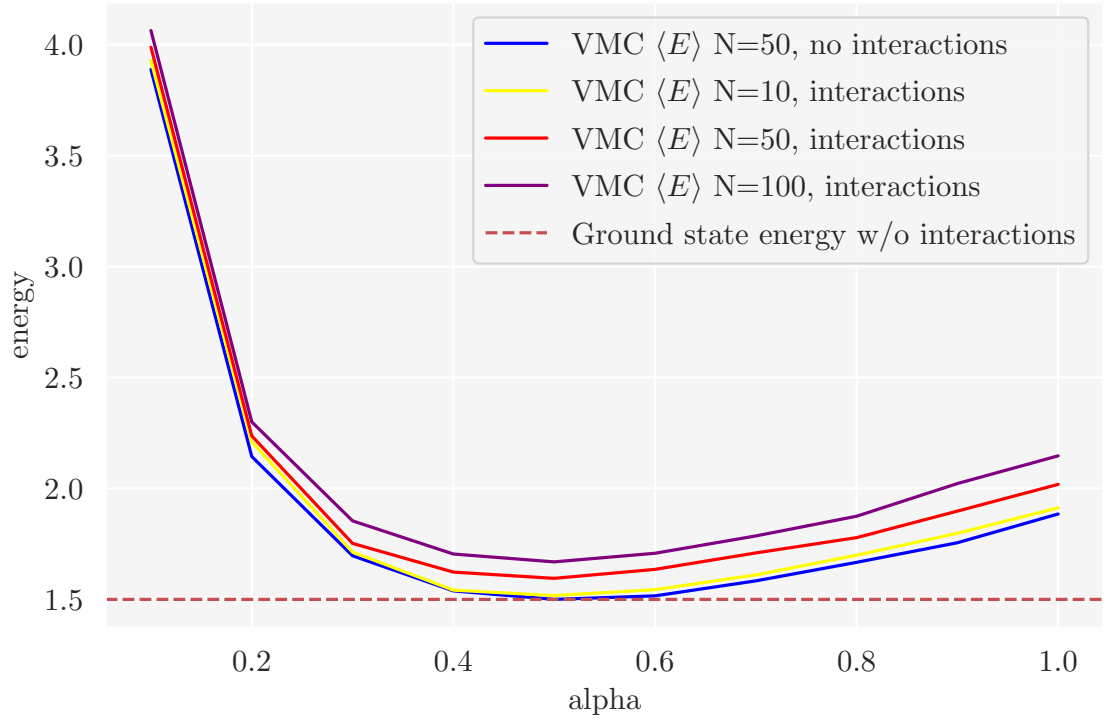


Figure A.4: *Random Walk Metropolis energy per particle comparison between* $N = 10, 50, 100$ *interacting particles, and* $N = 50$ *non-interacting particles.*

A.2. Additional Derivations

Error in standard Monte Carlo approximation

The mean square error ϵ_M of the Monte Carlo approximation can be written as

$$\mathscr{E}_M(\mathbb{E}[f(X)], E_M[f]) = \sqrt{\mathbb{E}[(\mathbb{E}[f(X)] - E_M[f])^2]}. \tag{A.1}$$

First we recognize the fact that the expecation value of the Monte Carlo approximation is equal to the true expectation value, $\mathbb{E}[f(X_m)] = \mathbb{E}[f(X)] \quad \forall m \in [1, M] \in \mathbb{N}$, which must be true as X and X_m are identically distributed. We denote the true expecation value, $\mathbb{E}[f(X)] = \mu$. We square $\mathscr{E}_M$

$$\begin{aligned}
\mathscr{E}_M^2 &= \mathbb{E}[(\mathbb{E}[f(X)] - E_M[f])^2] \\
&= \mathbb{E}[\mathbb{E}[f(X)] - 2\mathbb{E}[f(X)]E_M[f] + E_M[f]^2] \\
&= \mathbb{E}[f(X)] - 2\mathbb{E}[f(X)]\mathbb{E}[E_M[f]] + \mathbb{E}[E_M[f]^2] \\
&= \mu^2 - 2\mu\frac{1}{M}\sum_{m=1}^{M}\mathbb{E}[f(X_m)] + \frac{1}{M^2}\sum_{m=1}^{M}\sum_{n=1}^{M}\mathbb{E}[f(X_m)f(X_n)].
\end{aligned}$$

We acknowledge the fact that $f(X_m)$ and $f(X_n)$ are indepent variables, and thus

$$\begin{aligned}
\mathscr{E}_M^2 &= \mu^2 - 2\mu\frac{1}{M}\sum_{m=1}^{M}\mathbb{E}[f(X_m)] + \frac{1}{M^2}\sum_{m=1}^{M}\sum_{n=1}^{N}\mathbb{E}[f(X_m)]\mathbb{E}[f(X_n)] \\
&= \mu^2 - 2\mu\frac{M\mu}{M} + \frac{1}{M^2}\sum_{m=1}^{M}(\sum_{m\neq n}\mu^2 + \mathbb{E}[f(X)^2]) \\
&= -\mu^2 + \frac{1}{M^2}((M^2-M)\mu^2 + M\mathbb{E}[f(X)^2]) \\
&= -\frac{\mu^2}{M} + \frac{1}{M}\mathbb{E}[f(X)^2] \\
&= \frac{\mathbb{E}[f(X)^2] - \mathbb{E}[f(X)]^2}{M} = \frac{\mathrm{Var}[f(X)]}{M}
\end{aligned}$$

which leads to

$$\mathscr{E}_M = \frac{\sqrt{\mathrm{Var}[f(X)]}}{\sqrt{M}}.$$

Derivation of the gradient of the local energy with respect to the variational parameters.

The expected value of the energy is

$$\langle E\rangle = \int \mathrm{d}\boldsymbol{r} E_L(\boldsymbol{r};\boldsymbol{\alpha})p(\boldsymbol{r};\boldsymbol{\alpha}), \tag{A.2}$$

which can be rewritten as

$$\langle E\rangle = \frac{\int \mathrm{d}\boldsymbol{r}\Psi^* H\Psi}{\int \mathrm{d}\boldsymbol{r}\Psi^*\Psi}. \tag{A.3}$$

Now we want to take the derivative of this expression with respect to the variational parameters $\boldsymbol{\alpha}$.

$$\begin{aligned}
\boldsymbol{\nabla}_{\boldsymbol{\alpha}}\langle E\rangle &= \frac{\int \mathrm{d}\boldsymbol{r}(\boldsymbol{\nabla}_{\boldsymbol{\alpha}}\Psi^* H\Psi + \Psi^* H\boldsymbol{\nabla}_{\boldsymbol{\alpha}})}{\int \mathrm{d}\boldsymbol{r}\Psi^*\Psi} - \frac{(\int \mathrm{d}\boldsymbol{r}\Psi^* H\Psi)(\int \mathrm{d}\boldsymbol{r}[\Psi\boldsymbol{\nabla}_{\boldsymbol{\alpha}}\Psi^* + \Psi^*\boldsymbol{\nabla}_{\boldsymbol{\alpha}}\Psi])}{\left(\int \mathrm{d}\boldsymbol{r}\Psi^*\Psi\right)^2} \\
(*) &= \frac{\int \mathrm{d}\boldsymbol{r}(\boldsymbol{\nabla}_{\boldsymbol{\alpha}}\Psi^* H\Psi + \Psi[H\boldsymbol{\nabla}_{\boldsymbol{\alpha}}\Psi]^*)}{\int \mathrm{d}\boldsymbol{r}\Psi^*\Psi} - 2\langle E\rangle\left\langle\frac{\boldsymbol{\nabla}_{\boldsymbol{\alpha}}\Psi}{\Psi}\right\rangle \\
&= 2\left\langle E\frac{\boldsymbol{\nabla}_{\boldsymbol{\alpha}}\Psi}{\Psi}\right\rangle - 2\langle E\rangle\left\langle\frac{\boldsymbol{\nabla}_{\boldsymbol{\alpha}}\Psi}{\Psi}\right\rangle,
\end{aligned}$$

where we assume hermiticity in the Hamiltonian in (*). Thus, the gradient with respect to the variational parameters becomes

$$\boldsymbol{\nabla}_{\boldsymbol{\alpha}}\langle E\rangle = 2\left(\left\langle E\frac{\boldsymbol{\nabla}_{\boldsymbol{\alpha}}\Psi}{\Psi}\right\rangle - \langle E\rangle\left\langle\frac{\boldsymbol{\nabla}_{\boldsymbol{\alpha}}\Psi}{\Psi}\right\rangle\right). \tag{A.4}$$