FYS4411

Spring 2016

Variational Monte Carlo studies of bosonic systems

John-Anders Stende

Date: March 9, 2016

Abstract

The aim of this project is to use the Variational Monte Carlo (VMC) method to evaluate the ground state energy of a trapped, hard sphere Bose gas for different numbers of particles with a specific trial wave function.

Main findings

Introduction

Demonstrations of Bose-Einstein condensation (BEC) in gases of alkali atoms confined in magnetic traps has gained a lot of interest in the scientific community in recent years. Of interest is for instance the fraction of condensed atoms, the nature of the condensate and the excitations above the condensate.

An important feature of the trapped alkali systems is that they are dilute, i.e. the effective atom size is small compared to both the trap size and the inter-atomic spacing. In this situation the physics is dominated by two-body collisions, well discribed in terms of the s-wave scattering length a of the atoms. The condition for diluteness is defined by the gas parameter $x(\mathbf{r}) = n(\mathbf{r})a^3$, where $n(\mathbf{r})$ is the local density of the system. The theoretical framework of the Gross-Pitaevski equation is valid for $x_{av} \leq 10^{-3}$, but recent experiments have shown that the gas parameter may exceed this value due to the presence of so-called Feshbach resonance. Therefore, other methods like the VMC method may be needed.

In this project we evaluate the ground state energy of a trapped BEC by simulating different numbers of bosons in a harmonic oscillator potential in one, two and three dimensions. The energy is obtained using the VMC method, both with and without importance sampling. We have studied both the interacting and the non-interacting case, i.e. with both an uncorrelated and a correlated trial wave function. The method of blocking is utilized to do statistical analysis on the numerical data. We optimize the variational parameter α using the steepest descent method. The one-body density in the interacting and non-interacting case is alsox computed.

Theory

The trap we use is a spherical (S) or an elliptical (E) harmonic trap in one, two and three dimensions, with the latter given by

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

where ω_{ho} and ω_z defines the trap potential strength in the xy-plane and z-direction respectively. The two-body Hamiltonian is

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

and we reresent the inter-boson interaction by a pairwise, repulsive potential

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

where a is the so-called hard-core diameter of the bosons.

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

$$\Psi_T(\mathbf{R}) = \Psi_T(\mathbf{r}_1, \mathbf{r}_2, \dots \mathbf{r}_N, \alpha, \beta) = \prod_i g(\alpha, \beta, \mathbf{r}_i) \prod_{i<j} f(a, |\mathbf{r}_i - \mathbf{r}_j|), \tag{4}$$

where α and β are variational parameters. The single-particle wave function is proportional to the harmonic oscillator function for the ground state, i.e.,

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

For spherical traps we have $\beta = 1$ and for non-interacting bosons ($a = 0$) we have $\alpha = 1/2a_{ho}^2$. The correlation wave function is

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

Analytical results

The quantity we are aiming to compute is the expectiation value of the so-called local energy

$$E_L(\mathbf{R}) = \frac{1}{\Psi_T(\mathbf{R})} H \Psi_T(\mathbf{R}), \tag{7}$$

We can find closed-form expressions for the local energy with our specific Hamiltonian H and trial wavefunction Ψ_T. Computing the local energy involves a second derivative of Ψ_T, which can be expensive to compute numerically. Analytical expressions are therefore useful, as they can speed up the computations.

First, we find the local energy with only the (spherical) harmonic oscillator potential, that is we set $a = 0$ and $\beta = 1$. During these calcuations we will use natural units, thus $\hbar = m = 1$. For one particle in one dimension we have

$$\Psi_T(x) = e^{-\alpha x^2} \tag{8}$$

and

$$H = -\frac{1}{2}\frac{\partial^2}{\partial x^2} + \frac{1}{2}\omega x^2 \tag{9}$$

The second derivate of the trial wave function is

$$\frac{\partial^2 \Psi_T}{\partial x^2} = 2\alpha e^{-\alpha x^2}(2\alpha x^2 - 1) \tag{10}$$

so that

$$E_L = \frac{1}{\Psi_T} H \Psi_T = \alpha(1 - 2\alpha x^2) + \frac{1}{2}\omega^2 x^2 \tag{11}$$

In three dimensions the double derivative is replaced by the Laplacian when computing the kinetic energy

$$\begin{aligned} \triangledown^2 \Psi_T &= 2\alpha(2\alpha x^2 - 1) + 2\alpha(2\alpha y^2 - 1) + 2\alpha(2\alpha z^2 - 1) && (12) \\ &= 2\alpha(2\alpha r^2 - 3) && (13) \end{aligned}$$

thus the local energy is

$$E_L = -\frac{1}{2} \triangledown^2 \Psi_T + V_{ext} = \alpha(3 - 2\alpha r^2) + \frac{1}{2}\omega^2 r^2 \tag{14}$$

We now turn our attention to N particles, with the following wavefunction and Hamiltonian

$$\Psi_T(\mathbf{R}) = \prod_i e^{-\alpha r_i^2} \tag{15}$$

$$H = \sum_i^N \left(-\frac{1}{2}\nabla_i^2 + \frac{1}{2}\omega^2 r_i^2 \right) \tag{16}$$

The first term of the k-th Laplacian of this wavefunction is

$$\triangledown_k^2 \prod_i e^{-\alpha r_i^2} = 2\alpha(2\alpha x_k^2 - 1) \prod_i e^{-\alpha r_i^2} \tag{17}$$

and when we divide with Ψ_T to obtain the the local energy we end up with

$$E_L = \sum_i^N \left(\alpha(3 - 2\alpha r_i^2) + \frac{1}{2}\omega^2 r_i^2 \right) \tag{18}$$

For one dimension the expression is

$$E_L = \sum_i^N \left(\alpha(1 - 2\alpha x_i^2) + \frac{1}{2}\omega^2 x_i^2 \right) \tag{19}$$

It is also useful to compute the analytical expression for the drift force F to be used in importance sampling

$$F = \frac{2\nabla\Psi_T}{\Psi_T}. \tag{20}$$

The gradient of Ψ_T is

$$\begin{aligned} \nabla\Psi_T &= (-2\alpha x, -2\alpha y, -2\alpha z)e^{-\alpha r^2} && (21) \\ &= -2\alpha e^{-\alpha r^2}\mathbf{r} && (22) \end{aligned}$$

Dividing by the wavefunction yields

$$F = -4\alpha\mathbf{r} \tag{23}$$

Find local energy for the whole system...

Methods

We use the *Variational Monte Carlo* (VMC) method in this project to obtain the ground state energy for our bosonic system. VMC applies the *variational principle* from quantum mechanics

$$E_0 \leq \frac{\langle \Psi_T | H | \Psi_T \rangle}{\langle \Psi_T | \Psi_T \rangle} \tag{24}$$

which states that the ground state energy is always less or equal than the expectation value of our Hamiltonian H for any trial wavefunction Ψ_T. VMC consists in choosing a trial wavefunction depending on one or more variational parameters, and finding the values of these parameters for which the expectation value of the energy is the lowest possible. The main challenge is to compute the multidimensional integral

$$\frac{\langle \Psi_T | H | \Psi_T \rangle}{\langle \Psi_T | \Psi_T \rangle} = \frac{\int d\mathbf{R} \Psi_T^*(\mathbf{R}, \boldsymbol{\alpha}) H(\mathbf{R}) \Psi_T(\mathbf{R}, \boldsymbol{\alpha})}{\int d\mathbf{R} \Psi_T^*(\mathbf{R}, \boldsymbol{\alpha}) \Psi_T(\mathbf{R}, \boldsymbol{\alpha})} \tag{25}$$

where $\mathbf{R}$ is the positions of all the particles and $\boldsymbol{\alpha}$ is the set of variational parameters. Traditional integration methods like Gauss-Legendre methods are too computationally expensive, therefore other methods are needed.

Monte Carlo integration

Monte Carlo integration employs a non-deterministic approach to evaluate multidimensional integrals like (25), or in general

$$I = \int_\Omega f(\mathbf{x}) d\mathbf{x} \tag{26}$$

Instead of using an explicit integration scheme, we sample points

$$\mathbf{x}_1 \ldots \mathbf{x}_N \in \Omega \tag{27}$$

according to some rule. The naive approach, called brute force Monte Carlo, is to use N uniform samples. The integral can then be approximated as the average of the function values at these points

$$I \approx \frac{1}{N} \sum_{i=1}^{N} f(\mathbf{x}_i) \tag{28}$$

The brute force method is however not very efficient, as it samples an equal amount of points in all regions of Ω, including those where f is zero.

Metropolis algorithm

A more clever approach than the brute force method is to sample points according to the probability distribution (PDF) defined by f. Such a PDF is in general difficult to obtain, thus we can't sample directly from it. The solution is the Metropolis algorithm, which is a method to obtain random samples from a PDF for which direct sampling is difficult. These sample values are produced iteratively, with the distribution of the next sample being dependent only on the current sample value, thus making the sequence of samples into a Markov chain. We define $\mathbf{P}_i^{(n)}$ to be the probability for finding the system in state i at step n. The Metropolis algorithm is as follows:

- Sample a possible new state j with some probability $T_{i \to j}$
- Accept the new state with probability $A_{i \to j}$ and use it as the next sample, or recect the new state with probability $1 - A_{i \to j}$ and use state i as sample again

The transition probability T and the acceptance probability A must fulfill the principle of detailed balance

$$\frac{A_{i \to j}}{A_{j \to i}} = \frac{p_i T_{i \to j}}{p_j T_{j \to i}} \tag{29}$$

which ensures that $\mathbf{P}_i^{(n \to \infty)} \to p_i$, i.e. we end up at the correct distribution regardless of what we begin with.

The particles undergo a random walk under the guidance of the Metropolis algorithm. Defining the PDF

$$P(\mathbf{R}) = \frac{|\Psi_T(\mathbf{R})|^2}{\int |\Psi_T(\mathbf{R})|^2 d\mathbf{R}} \tag{30}$$

and the local energy (7), the integral (25) can be rewritten as

$$\langle E_L \rangle = \int P(\mathbf{R}) E_L(\mathbf{R}) d\mathbf{R} \tag{31}$$

and we see that our problem amounts to finding the expectation value of the local energy E_L on the PDF P. The VMC method approximates this integral as

$$\langle E_L \rangle \approx \frac{1}{N} \sum_{i=1}^{N} P(\mathbf{R}_i, \boldsymbol{\alpha}) E_L(\mathbf{R}_i, \boldsymbol{\alpha}) \tag{32}$$

where N is the number of Monte Carlo cycles and $\mathbf{R}_i$ is the position of the particles at step i. The integral $\int |\Psi_T(\mathbf{R})|^2 d\mathbf{R}$ is very difficult to compute, but the Metropolis algorithm only needs a *ratio* of probabilities to decide if a move is accepted or not. This can be seen if we rewrite (29) as

$$\frac{p_j}{p_i} = \frac{T_{i\to j} A_{i\to j}}{T_{j\to i} A_{j\to i}} \tag{33}$$

In our case $p_j = P(\mathbf{R}_j)$ and $p_i = P(\mathbf{R}_i)$. The simplest form of the Metropolis algorithm, called brute force Metropolis, is to assume that the transition probability $T_{i\to j}$ is symmetric, implying that $T_{i\to j} = T_{j\to i}$; the ratio of probabilities (33) thus equals the ratio of acceptance probabilities. This leads to a description of the Metropolis algorithm where we accept or reject a new move by calculating the ratio

$$w = \frac{|\Psi_T(\mathbf{R}_j)|^2}{|\Psi_T(\mathbf{R}_i)|^2} \tag{34}$$

If $w \geq s$, where s is a random number $s \in [0, 1]$, the new position is accepted, else we stay at the same place. We now have the full machinery of the Monte Carlo approach to obtain the ground state energy of our bosonic system:

- Fix the number of Monte Carlo steps and choose the initial positions $\mathbf{R}$ and variational parameters $\boldsymbol{\alpha}$. Also set the step size $\Delta\mathbf{R}$ to be used when moving from $\mathbf{R}_i$ to $\mathbf{R}_j$.
- Initialize the local energy
- Choose a random particle
- Calculate a trial position $\mathbf{R}_j = \mathbf{R}_i + r\dot{\Delta}\mathbf{R}$ where r is a random variable $r \in [0, 1]$
- Use the Metropolis algorithm to accept or reject this move by calculating the ratio (34). If $w \geq s$, where s is a random number $s \in [0, 1]$, the new position is accepted, else we stay at the same place.
- If the step is accepted, set $\mathbf{R} = \mathbf{R}_j$ for the chosen particle
- Update the local energy

When the Monte Carlo sampling is finished, we calculate the mean local energy, which is our approximation of the ground state energy of the system. The Metropolis algorithm is implemented as follows:

Listing 1: Brute Forde Metropolis algorithm

```
int particle = Random::nextInt(m_numberOfParticles);      // choose random particle
int dimension = Random::nextInt(m_numberOfDimensions);  // choose random dimension
double change = (Random::nextDouble()*2-1)*m_stepLength;  // propose change

// get old wavefunction
double waveFuncOld = m_waveFunction->evaluate(m_particles);

// adjust position
m_particles[particle]->adjustPosition(change, dimension);

// get new wavefunction
double waveFuncNew = m_waveFunction->evaluate(m_particles);

// accept/reject new position using Metropolis algorithm
double ratio = pow(waveFuncNew, 2) / pow(waveFuncOld, 2);
```

```
if (ratio >= Random::nextDouble()) {
    //cout << m_particles[particle]->getPosition()[0] << endl;
    return true;
}
else {
    // correct position change
    m_particles[particle]->adjustPosition(-change, dimension);
    return false;
}
```

Steepest descent method

We turn now to the problem of finding the variational parameters that minimizes the expectation value of the local energy $\langle E_L(\mathbf{R},\alpha)\rangle$. This project considers a trial wavefunction with only one variational parameter α. There are many optimization algorithms to choose from, we have chosen the Steepest descent method due to its simplicity. This method finds a local minimum of a function by taking steps proportional to the negative gradient of the function at a given point, i.e. where the function has the steepest descent. The algorithm is as follows:

- Choose an initial α_0 and step length γ_0.
- For $i \geq 0$: Compute $\alpha_{i+1} = \alpha_i - \gamma_i \frac{d\langle E_L(\mathbf{R},\alpha)\rangle}{d\alpha}$
- Continue until a maximum number of steps are performed or $|\alpha_{i+1} - \alpha_i|$ is less than some tolerance

$\langle E_L(\mathbf{R},\alpha)\rangle$ is as we have seen a multidimensional integral, and the derivative w.r.t. α is not easily computed. Let us define

$$\bar{E}_\alpha = \frac{d\langle E_L(\alpha)\rangle}{d\alpha} \tag{35}$$

and

$$\bar{\Psi}_T = \frac{d\Psi_T(\alpha)}{d\alpha} \tag{36}$$

Using the chain rule and the hermicity of the Hamiltonian it can be shown that

$$\bar{E}_\alpha = 2\left(\Big\langle \frac{\bar{\Psi}_T}{\Psi_T(\alpha)} E_L(\alpha)\Big\rangle - \Big\langle \frac{\bar{\Psi}_T}{\Psi_T(\alpha)}\Big\rangle \langle E_L(\alpha)\rangle\right) \tag{37}$$

thus we need the expectation values of

$$\frac{\bar{\Psi}_T}{\Psi_T(\alpha)} E_L(\alpha) \tag{38}$$

and

$$\frac{\bar{\Psi}_T}{\Psi_T(\alpha)} \tag{39}$$

The complete VMC method then amounts to the following:

- Make initial guess α_0
- Run 10^4-10^5 Metropolis steps, sample (38) and (39)
- Compute (37)
- Compute new α using the Steepest descent method

The above steps are repeated until the value of α is sufficiently accurate, before a new round of Metropolis steps are run, this time with many cycles (10^6-10^8). We then obtain our approximation for the ground state energy of the system.

The Steepest descent method is implemented as follows:

Listing 2: The Steepest Descent method

```
void SteepestDescent::optimize(double initialAlpha) {

    int maxNumberOfSteps = 30;
    double tolerance = 0.001;
    double oldAlpha = initialAlpha;
    for (int i=0; i < maxNumberOfSteps; i++) {

        // make initial state
        m_system->getInitialState()->setupInitialState();

        // set value of alpha
        m_system->getWaveFunction()->setAlpha(oldAlpha);

        // run metropolis steps
        m_system->runMetropolisSteps((int) 1e4, false, false, false);

        // compute derivative of exp. value of local energy w.r.t. alpha
        double localEnergyDerivative = 2 *
                            ( m_system->getSampler()->getWaveFunctionEnergy() -
                              m_system->getSampler()->getWaveFunctionDerivative() *
                              m_system->getSampler()->getEnergy() );

        cout << "localEnergyDerivative = " << localEnergyDerivative << endl;

        // compute new alpha
        double newAlpha = oldAlpha - m_stepLengthOptimize*localEnergyDerivative;
        cout << "newAlhpa = " << newAlpha << endl;
        cout << "oldAlpha = " << oldAlpha << endl;
        cout << std::abs(newAlpha-oldAlpha) << endl;
        if ( std::abs(newAlpha - oldAlpha) < tolerance ) {
            break;
        }
        // before new iteration
        oldAlpha = newAlpha;
    }
    cout << "Optimal alpha = " << oldAlpha << endl;

    // run many Metropolis steps with the optimal alpha

    // make initial state
    m_system->getInitialState()->setupInitialState();

    // set value of alpha
    m_system->getWaveFunction()->setAlpha(oldAlpha);

    // run metropolis steps
    m_system->runMetropolisSteps((int) 1e7, false, false, false);
}
```