Many-body He^4 boson with Restricted Boltzmann Machine

University of Oslo

Kari Eriksen

Desember 2019

Abstract

Abstract goes here

Contents

Dedication

Declaration

I declare that..

Acknowledgements

I want to thank...

Chapter 1

Introduction

Part I

Theory

Chapter 2

Many-body quantum theory

2.1 The Bose-Einstein Condensation

2.2 Ansatz for the wavefunction and the hamiltonian

We are studying a system of two electrons confined in a harmonic oscillator trap described by the Hamiltonian

$$\hat{H} = \sum_{i=1}^{N} \left(-\frac{1}{2}\nabla_i^2 + \frac{1}{2}\omega^2 r_i^2 \right) + \sum_{i<j} \frac{1}{r_{ij}} \tag{2.1}$$

where the first sum is the standard harmonic oscillator part and the last is the interacting part between the electrons and N represent the number of particles. ω is the oscillator frequency of the trap and r_i is the position of particle i, whereas r_{ij} is the distance between the particles and given as $r_{ij} = |\mathbf{r_i} - \mathbf{r_j}|$.
Our wave function is given from the energy of the restricted Boltzmann machine, see (**??**), which is the joint energy functional between the visible and hidden nodes. From the marginal probability of the joint probability distribution, see (**??**), we get our wave equation.

$$\begin{aligned} \Psi(X) &= F_{rbm}(X) && (2.2)\\ &= \frac{1}{Z}\exp\left(-\sum_{i}^{M} \frac{(X_i - a_i)^2}{2\sigma^2} \right) \prod_{j}^{N} \left(1 + \exp\left(b_j + \sum_{i}^{M} \frac{X_i \omega_{ij}}{\sigma^2} \right) \right) && (2.3) \end{aligned}$$

Here Z is the partition function, X_i represents the visible nodes running up to M, and a_i and b_j are the biases described in the section below, (**??**), where number of hidden nodes j runs up to N. ω_{ij} is an $M \times N$ matrix holding the weights connecting the visible nodes with the hidden and σ is the standard deviation of the noise in our model.

2.3 Interaction

Chapter 3

VMC and DMC

3.1 Markoc chains

3.2 Variational Monte Carlo

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

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

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

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

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

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

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

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

Then the expectation value of the local energy is given by

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

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

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

3.3 The Metropolis Algorithm

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

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

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

$$P_i^{(n)} = \sum_j \left[P_j^{(n-1)} T_{j\rightarrow i} A_{j\rightarrow i} + P_j^{(n-1)} T_{i\rightarrow j}(1 - A_{i\rightarrow j}) \right]. \tag{3.1}$$

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

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

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

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

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

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

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

Algorithm 1: Monte Carlo with Metropolis-Hastings

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

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

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

$$\frac{p_i T_{i\to j}}{p_j T_{j\to i}} = \frac{|\Psi_T(\mathbf{R}_p)|^2}{|\Psi_T(\mathbf{R})|^2} \tag{3.2}$$

3.4 Importance Sampling

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

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

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

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

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

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

Now equation **??** from Metropolis algorithm changes to

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

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

3.5 Gradient Descent

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

$$\alpha_{i+1} = \alpha_i - \gamma \frac{\partial \langle E_L(\alpha_i) \rangle}{\partial \alpha_i} \tag{3.3}$$

$$\gamma_i = (\alpha_{i+1} - \alpha_i) \cdot \frac{1}{E_{L}{}^{\alpha}_{i+1} - E_{L}{}^{\alpha}_{i}} \tag{3.4}$$

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

$$E_{l}{}^{\alpha}_{i} = 2\left(\overline{\frac{\psi^\alpha}{\psi} E_L}^{\,i} - \overline{\frac{\psi^\alpha}{\psi}}\,\overline{E_L}^{\,i}\right) \tag{3.5}$$

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

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

[2]The equation was retrieved from the lecture notes

$$\frac{\psi^\alpha}{\psi} = \frac{\psi \prod_i^N (x_i^2 + y_i^2 + \beta z_i^2)}{\psi} \tag{3.6}$$

$$= \prod_i^N (x_i^2 + y_i^2 + \beta z_i^2) \tag{3.7}$$

This holds for both the interactive and non-interactive case since the correlation wavefunction, $f(r_i, r_j)$, does not have any dependence on α
In the program the only addition to make is the computation of two additional expecation values[3]. Empirically we've also seen that we can reduce the number of MC-cycles quite drastically and still get a good convergence to the optimal α. In summary gradient descent and other associated methods allows for a radical speedup in estimation of the optimal values of the variational parameters. The implementation of this method can be found in the file `main_f.cpp`.

3.6 Blocking Method

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

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

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

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

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

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

[3]implemented in `vmc.cpp`

3.7 Numerical differentiation

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

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

Algorithm 2: Numerical differentiation of the second order of the trial wavefunction on a system **R**

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

Chapter 4

Machine Learning: RBM/DBM

4.1 Restricted Boltzmann Machine

Restricted Boltzmann machine is a energy-based generative model which include hidden and visible variables (ref to Metha). It consists of a two-layer network with a two dimensional matrix W_{ij} telling how strong the connections between the hidden and visible nodes are. The energy related to a configuration of the nodes is what forms the basis of our model. It is given in (**??**).

$$E(\mathbf{v}, \mathbf{h}) = -\sum_i a_i(v_i) - \sum_\mu b_j(h_j) - \sum_{ij} W_{ij} v_i h_j \tag{4.1}$$

Here $a_i(v_i)$ and $b_j(h_j)$ are bias functions of the visible and hidden layers which we are allowed to choose our self. In our case we want the visible layer to take a continuous form and the hidden layer to be binary, (Gaussian-binary), meaning a_i and b_j takes the following from

$$a_i(v_i) = \frac{v_i^2}{2\sigma_i^2}, \quad b_j(h_j) = b_j h_j.$$

We are working with restricted Boltzmann machine meaning there is no connection between nodes within layers, only between layers. Also this network is a generative one so we want our network to learn a probability distribution. We begin with the joint probability distribution of the visible and hidden nodes is given in (**??**) where Z is the partition function, see (**??**).

$$F_{rbm}(\mathbf{X}, \mathbf{h}) = \frac{1}{Z} e^{-E(\mathbf{X}, \mathbf{h})} \tag{4.2}$$

$$Z = \int\int \frac{1}{Z} e^{-E(\mathbf{X}, \mathbf{h})} d\mathbf{x} d\mathbf{h} \tag{4.3}$$

From (**??**) we can marginalize over all the hidden units and get the distribution over the visible units. And as mentioned above this is what we use to represent our wave function.

$$F_{rbm}(\mathbf{X}) = \sum_{\mathbf{h}} F_{rbm}(\mathbf{X}, \mathbf{h}) \tag{4.4}$$

$$= \frac{1}{Z} \sum_{\mathbf{h}} e^{-E(\mathbf{X}, \mathbf{h})} \tag{4.5}$$

Since we have no data to train we use the technique of reinforcement learning. We feed the network with input using Monte Carlo method and based on the variational principle we find the ground state of the system by seeking the configuration that gives the lowest quantum mechanical energy. According to this principle we change the weights and biases by gradient descent method and hopefully the network will converge towards the correct state.

4.2 Gibbs Sampling

Part II

Implementation and results

Chapter 5

Implementation

Chapter 6

Results

Chapter 7

Conclusion

Appendix A

Appendix Title