FYS4411 - Computational Physics II

Project 1

Dorthea Gjestvang
Even Marius Nordhagen

February 28, 2018

Abstract

Write the abstract here

- Github repository containing programs and results are in: `https://github.com/evenmn/FYS4411/tree/master/Project%201`

1 Introduction

Introduction

2 Theory

We study a system of N bosons trapped in a harmonic oscillator with the Hamiltonian given by

$$\hat{H} = \sum_{i}^{N} \left(-\frac{\hbar^2}{2m}\nabla_i^2 + V_{ext}(\vec{r}_i) \right) + \sum_{i<j}^{N} V_{int}(\vec{r}_i, \vec{r}_j) \tag{1}$$

with V_{ext} as the external potential, which is the harmonic oscillator potential, and V_{int} as the interaction term. The interaction will in the first place be ignored, and will be given later.

The wavefunction is on the form

$$\Psi_T(\vec{r}_1, \vec{r}_2, ..., \vec{r}_N, \alpha, \beta) = \prod_{i}^{N} g(\alpha, \beta, \vec{r}_i) \prod_{i<j} f(a, r_{ij}) \tag{2}$$

where $r_{ij} = |\vec{r}_i - \vec{r}_j|$ and g is assumed to be an exponential

$$g(\alpha, \beta, \vec{r}_i) = \exp[-\alpha(x_i^2 + y_i^2 + \beta z_i^2)] \tag{3}$$

which is practical since

$$\prod_i^N g(\alpha, \beta, \vec{r}_i) = \exp[-\alpha(x_1^2 + y_1^2 + \beta z_1^2 + \cdots x_N^2 + y_N^2 + \beta z_N^2)]. \tag{4}$$

α is a variational parameter that we later use to find the energy minimum, and β is a constant. This is also the form of the exact ground state wave function for a harmonic oscillator, so by choosing the correct α, we will find the exact ground state energy (when we The f presented above is the correlation wave function, which is

$$f(a, r_{ij}) = \begin{cases} 0 & r_{ij} \leq a \\ \left(1 - \frac{a}{r_{ij}}\right) & r_{ij} > a. \end{cases} \tag{5}$$

The first case we will take into account, is when $a = 0$, and one might observe that $f = 1$ then. Anyway, ...

We want to calculate the local energy as a function of α, and then use Variational Monte Carlo (VMC) described in section 3.1. For the non-interacting case, the analytical expression is well-known and given by

$$E = \hbar\omega(n + 1/2) \tag{6}$$

where n is the total number of free dimensions, which gonna be an useful benchmark. The local energy is

$$E_L(\vec{r}) = \frac{1}{\Psi_T(\vec{r})} \hat{H} \Psi_T(\vec{r}) \tag{7}$$

which gives the following results considering $a = 0$:

INSERT ANALYTICAL EXPRESSIONS FROM A

For $a \neq 0$ it gets rather more complicated, because we need to deal with the correction wave function as well. By defining

$$f(a, r_{ij}) = \exp\left(\sum_{i<j} u(r_{ij})\right) \tag{8}$$

and doing a change of variables

$$\frac{\partial}{\partial r_k} = \frac{\partial}{\partial r_k}\frac{\partial r_{kj}}{\partial r_{kj}} = \frac{\partial r_{kj}}{\partial r_k}\frac{\partial}{\partial r_{kj}} = \frac{(r_k - r_j)}{r_{kj}}\frac{\partial}{\partial r_{kj}} \tag{9}$$

one will end up with

$$E_L = \sum_k \Bigg(-\frac{1}{2}\Bigg(4\alpha^2\Big(x_k^2 + y_k^2 + \beta^2 z_k^2 - \frac{1}{\alpha} - \frac{\beta}{2\alpha}\Big) \\ - 4\alpha \sum_{j\neq k} (x_k, y_k, \beta z_k) \frac{(\vec{r}_k - \vec{r}_j)}{r_{kj}} u'(r_{kj}) \\ + \sum_{ij\neq k} \frac{(\vec{r}_k - \vec{r}_j)(\vec{r}_k - \vec{r}_i)}{r_{ki} r_{kj}} u'(r_{ki}) u'(r_{kj}) \\ + \sum_{j\neq k} \Big(u''(r_{kj}) + \frac{2}{r_{kj}} u'(r_{kj}) \Big) \Bigg) + V_{ext}(\vec{r}_k) \Bigg). \tag{10}$$

This is not a pretty expression, but .. We could also split up the local energy expression

$$E_{L,i} = -\frac{\hbar^2}{2m} \frac{\nabla_i^2 \Psi_T}{\Psi_T} + V_{ext}(\vec{r}_i) = E_{k,i} + E_{p,i} \tag{11}$$

and calculate the local energy with a numerical approach where the second derivative can be approximated by the three-point formula:

$$f''(x) \simeq \frac{f(x+h) - 2f(x) + f(x-h)}{h^2}. \tag{12}$$

In our case the position is a three dimensional vector, so we need to handle each dimension separately. However, in section 4.1, the CPU time for the analytical and numerical approach are compared.

3 Methods

3.1 Variational Monte Carlo

Variational Monte Carlo (VMC) is a widely used method for approximating the ground state of a quantum system. The method is based on Markov chains, and move a particle (or a set of particles) one step for each cycle, i.e.

$$\vec{R}_{new} = \vec{R} + r \cdot \text{step} \tag{13}$$

Both the direction and the change in position are randomly chosen, so with a plain VMC implementation the particles will move randomly and independently of each other. We are going to use the Metropolis algorithm in addition to the VMC, which accepts or rejects moves based on the probability ratio between the old and the new position. This makes the system approach the most likely state, and the idea is that after a certain number of cycles the system will be in the most likely state.

3.2 Metropolis Algorithm

As mentioned above the task of the Metropolis algorithm is to move the system against the most likely state. The original algorithm, here named brute force, is the simplest one, and does not deal with the transition probabilities. The modified Metropolis-Hastings algorithm includes, on the other hand, the transition probabilities and will be slightly more time consuming per cycle. We expect the latter to converge faster to the most likely state.

The foundation of the Metropolis algorithm is that the probability for a system to undergo a transition from state i to state j is given by the transition probability multiplied by the acceptance probability

$$W_{i\to j} = T_{i\to j} \cdot A_{i\to j} \tag{14}$$

where $T_{i\to j}$ is the transition probability and $A_{i\to j}$ is the acceptance probability. Built on this, the probability for being in a state i at time (step) n is

$$P_i^{(n)} = \sum_j \left[P_j^{(n-1)} T_{j\to i} A_{j\to i} + P_i^{(n-1)} T_{i\to j}(1 - A_{i\to j}) \right] \tag{15}$$

since this can happend in two ways. One can start in this state i at time $n-1$ and be rejected or one can start in another state j at time $n-1$ and complete an accepted move to state i. In fact $\sum_j T_{i\to j} = 1$, so we can rewrite this as

$$P_i^{(n)} = P_i^{(n-1)} + \sum_j \left[P_j^{(n-1)} T_{j\to i} A_{j\to i} - P_i^{(n-1)} T_{i\to j} A_{i\to j} \right]. \tag{16}$$

When the times goes to infinity, the system will approach the most likely state and we will have $P_i^{(n)} = p_i$, which requires

$$\sum_j \left[p_j T_{j\to i} A_{j\to i} - p_i T_{i\to j} A_{i\to j} \right] = 0. \tag{17}$$

Rearranging, we obtain a quite useful results

$$\frac{A_{j\to i}}{A_{i\to j}} = \frac{p_i T_{i\to j}}{p_j T_{j\to i}} \tag{18}$$

3.2.1 Brute force

In the brute force Metropolis algorithm we want to check if the new position is more likely than the current position, and for that we calculate the probabilities $P(\vec{R}) = |\Psi_T(\vec{R})|^2$ for both positions. We get rid off the transition

probabilities setting $T_{i\to j} = T_{j\to i}$, and then end up with the plain ratio

$$w = \frac{P(\vec{R}_{new})}{P(\vec{R})} = \frac{|\Psi_T(\vec{R}_{new})|^2}{|\Psi_T(\vec{R})|^2}. \tag{19}$$

w will be larger than one if the new position is more likely than the current, and smaller than one if the current position is more likely than the new one. Metropolis handle this by accepting if the ratio w is larger than a random number r in the interval $[0, 1]$, and rejecting if not:

$$\text{New position: } \begin{cases} \text{accept} & \text{if} \quad w > r \\ \text{reject} & \text{if} \quad w \leq r. \end{cases} \tag{20}$$

3.2.2 Importance sampling

The important sampling technique is often refered to as Metropolis-Hastings algorithm. The approach is the same as for the brute force Metropolis algorithm, but we will end ut with a slightly more complicated acceptation criteria. To understand the details, we need to begin with the Fokker-Planck equation, which describes the time-evolution of the probability density function $P(R, t)$. In one dimension it reads

$$\frac{\partial P(R,t)}{\partial t} = D\frac{\partial}{\partial R}\left(\frac{\partial}{\partial R} - F\right)P(R,t). \tag{21}$$

where F is the drift force and D is the diffusion coefficient. Even though the probability density function can give a lot of useful information, an equation describing the motion of a such particle would be more appropriate for our purposes. Fortunately this equation exists, and satisfies the Fokker-Planck equation. The Langevin equation is given by

$$\frac{\partial R(t)}{\partial t} = DF(R(t)) + \eta \tag{22}$$

where η can be considered as a random variable. One can solve this differetial equation by applying the forward Euler method and introducing gaussian variables ξ

$$R_{new} = R + DF(R)\Delta t + \xi\sqrt{\Delta t} \tag{23}$$

which will be used to update the position. Moreover we also need to update the acceptance criteria since we no longer ignore the transition probabilities.

With the Fokker-Planck equation as base, the transition probabilities are given by Green's function

$$\begin{aligned} T_{R\to R_{new}} &= G(R_{new}, R, \Delta t) \\ &= \frac{1}{(4\pi D\Delta t)^{3N/2}} \exp[-(R_{new} - R - D\Delta t F(R))^2/4D\Delta t] \end{aligned} \tag{24}$$

and the acceptance criteria becomes

$$r < \frac{G(R, R_{new}, \Delta t)|\Psi_T(R_{new})|^2}{G(R_{new}, R, \Delta t)|\Psi_T(R)|^2}. \tag{25}$$

4 Results

4.1 CPU-time

For the brute force Metropolis algorithm we developed both an analytical and a numerical method to calculate the local energy. In table (1) we present the results from these calculations and the performance. All the measurements are done in three dimensions with $1e6$ Monte Carlo cycles. a is fixed to zero.

Table 1: The local energy calculated numerically for $a = 0$ and without repulsive interaction. The number of Monte Carlo cycles is fixed to 1e6, and the performance is presented along with the local energies. The analytical local energy is given by $E_L = 0.5 \cdot N \cdot d$ where N is the number of particles and d is the number of dimensions, in our case set to 3.

	Analytical		**Numerical**	
N	$\langle E_L \rangle$ $[\hbar\omega_{HO}]$	CPU-time [s]	$\langle E_L \rangle$ $[\hbar\omega_{HO}]$	CPU-time [s]
1	1.50000	0.146392	1.49999	0.426018
10	15.000	11.8992	14.9999	38.0378
100	150.00	8326.16		
500				

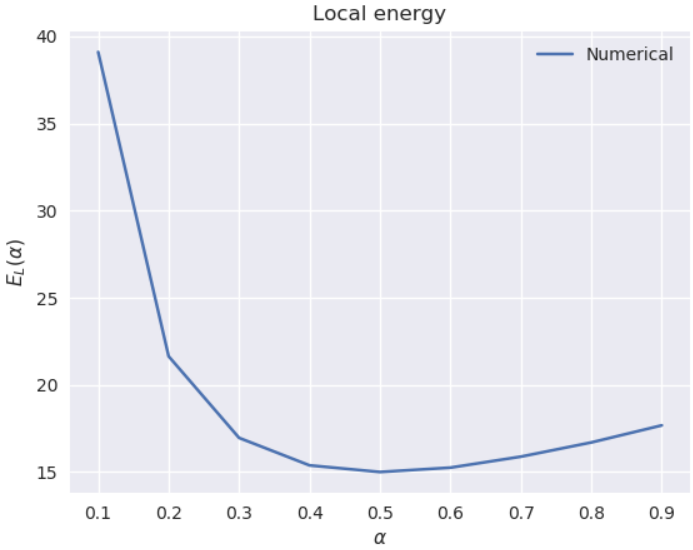

Figure 1: Add caption

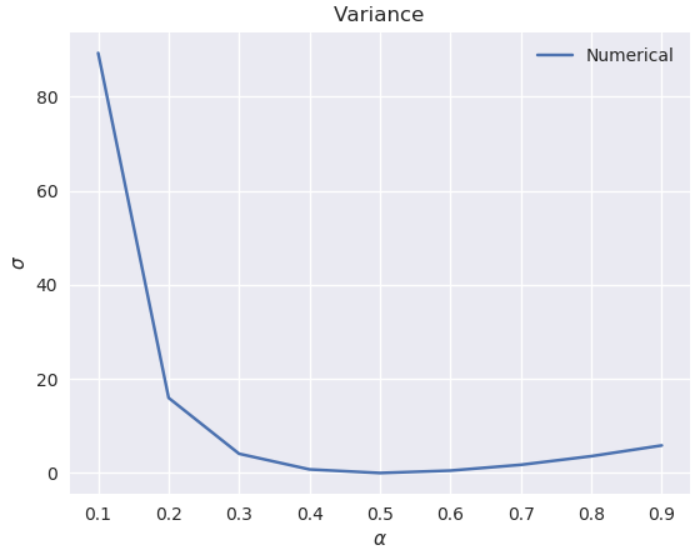

Figure 2: Add caption

5 Discussion

6 Conclusion