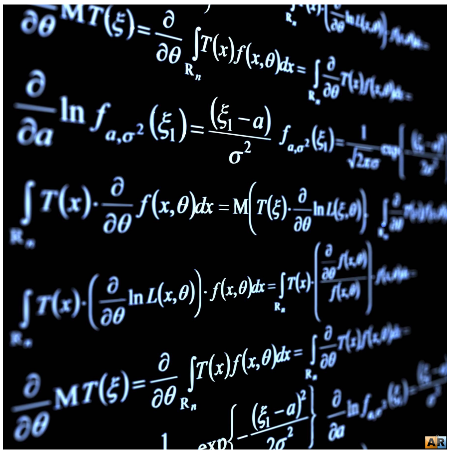

PROJECT 3

FYS3150 - COMPUTATIONAL PHYSICS

Quantum dots

Author:
Vidar SKOGVOLL

November 17, 2014

Abstract

Here is a short summary of the project.

Contents

1 Introduction

Quantum mechanics is an exciting field.

2 Theory

Here is all the theory needed to understand the project.

2.1 The physical system

This is the section explaining the physics of the system. Throughout the project, *natural units* are used ($\hbar = 1$,$c = 1$,$e = 1$,$m_e = 1$) and all energies are in so-called *atomix units* a.u.

2.1.1 The quantum mechanics and the variational principle

2.1.1.1 The quantum mechanics

In this project we will look at a system of N electrons in a so-called *quantum dot.* That is, a two dimensional harmonic oscillator with potential

$$V(\vec{r}) = \frac{1}{2}\omega^2 r^2 \tag{2.1.1}$$

This potential gives rise to a multi-particle Hamiltonian $\hat{H}$ given as the sum of an ordinary Hamiltonian and an electron repulsive part

$$\hat{H} = \sum_{i=1}^{N} \left(-\frac{1}{2}\nabla_i^2 + \frac{1}{2}\omega^2 r_i^2 \right) + \sum_{i<j} \frac{1}{r_{ij}} \tag{2.1.2}$$

Where $r_{ij} = |\vec{r}_i - \vec{r}_j|$ is the distance between the electrons i and j and $r_i = |\vec{r}_i| = \sqrt{x_i^2 + y_i^2}$ when $\vec{r}_i = \begin{pmatrix} x_i \\ y_i \end{pmatrix}$. Our goal in this project is to find the ground eigenstate and energy of this multi-particle Hamiltonian numerically.

2.1.1.2 The variational principle

We will approach this by constructing a real test function $\Psi_T(\vec{r}_0, \vec{r}_1, ..., \vec{r}_{N-1}, \alpha, \beta)$ dependent on two parameters α and β and calculate the expextation value of the hamilton operator $\langle \hat{H} \rangle$. As we know, the orthonormal eigenstates Ψ_i of the Hamiltonian forms a complete basis, so any state, including our test state Ψ_T, can be written as a linear combination of the eigenstates

$$\Psi_T = \sum_i c_i \Psi_i \tag{2.1.3}$$

Inserting this expression into the equation for the expectation value of $\hat{H}$ gives

$$\langle \hat{H} \rangle = \frac{\int \Psi_T \hat{H} \Psi_T d\vec{r}}{\int \Psi_T \Psi_T d\vec{r}} = \frac{\int \left(\sum_i c_i^* \Psi_i^*\right) \hat{H} \left(\sum_i c_i \Psi_i\right) d\vec{r}}{\int \left(\sum_i c_i^* \Psi_i^*\right) \left(\sum_i c_i \Psi_i\right) d\vec{r}} = \frac{\int \left(\sum_i c_i^* \Psi_i^*\right) \left(\sum_i c_i E_i \Psi_i\right) d\vec{r}}{\int \left(\sum_i c_i^* \Psi_i^*\right) \left(\sum_i c_i \Psi_i\right) d\vec{r}}$$
$$= \frac{\sum_i |c_i|^2 E_i}{\sum_i |c_i|} \tag{2.1.4}$$

The energy of the ground state E_0 is smaller than all other E_i's so

$$\frac{\sum_i |c_i|^2 E_i}{\sum_i |c_i|} \geq \frac{\sum_i |c_i|^2 E_0}{\sum_i |c_i|} = E_0 \frac{\sum_i |c_i|^2}{\sum_i |c_i|} = E_0 \tag{2.1.5}$$
$$\langle H \rangle \geq E_0 \tag{2.1.6}$$

This simple observation is called *the variational principle* and is what we will use to narrow our search for the optimal parameters α and β. We will look for the parameters α and β that gives us the smallest value of $\langle \hat{H} \rangle$ and this will be our estimate for the ground state.

2.1.1.3 Finding the expectation value of $\hat{H}$

We have

$$\langle \hat{H} \rangle = \frac{\int \Psi_T \hat{H} \Psi_T d\vec{r}}{\int \Psi_T \Psi_T d\vec{r}} = \int \frac{\Psi_T \Psi_T}{\int \Psi_T \Psi_T d\vec{r}} \frac{1}{\Psi_T} \hat{H} \Psi_T d\vec{r} \tag{2.1.7}$$

If we rename probability density function of the particles $\frac{\Psi_T \Psi_T}{\int \Psi_T \Psi_T d\vec{r}} = P(\vec{r})$ and $E_L(\vec{r}) = \frac{1}{\Psi_T} \hat{H} \Psi_T$, then the integral becomes

$$\langle \hat{H} \rangle = \int P(\vec{r}) E_L(\vec{r}) d\vec{r} = \langle E_L \rangle \tag{2.1.8}$$

But we could very well find a minimum of $\langle \hat{H} \rangle$ that is not an eigen energy of the system, i.e. still larger than E_0. To address this problem, let's look at the variance V_{E_L} of $\langle E_L \rangle$.

$$V_{E_L} = \langle E_L^2 \rangle - \langle E_L \rangle^2 = \int P(\vec{r}) \left(\frac{1}{\Psi_T} \hat{H} \Psi_T \right)^2 d\vec{r} - \left(\int P(\vec{r}) \frac{1}{\Psi_T} \hat{H} \Psi_T d\vec{r} \right)^2 \tag{2.1.9}$$

Since

$$\langle (E_L - \langle E_L \rangle)^2 \rangle = \langle E_L^2 - 2E_L \langle E_L \rangle + \langle E_L \rangle^2 \rangle = \langle E_L^2 \rangle - 2\langle E_L \rangle \langle E_L \rangle + \langle E_L \rangle^2$$
$$= \langle E_L^2 \rangle - \langle E_L \rangle^2 = V_{E_L} \tag{2.1.10}$$

We have that if $V_{E_L} = 0$, then

$$\langle (E_L - \langle E_L \rangle)^2 \rangle = 0 \tag{2.1.11}$$
$$0 = \int \left(P(\vec{r}) E_L(\vec{r}) - \int P(\vec{r}) E_L(\vec{r}) d\vec{r} \right)^2 d\vec{r} = \int \left(\frac{1}{\Psi_T} \hat{H} \Psi_T - \int \Psi_T \hat{H} \Psi_T d\vec{r} \right)^2 d\vec{r} \tag{2.1.12}$$

When the Hamiltonian acts on a real function, it gives a real function. And since Ψ_T is real $\frac{1}{\Psi_T}\hat{H}\Psi_T - \int \Psi_T \hat{H}\Psi_T d\vec{r}$ is real. By consequence

$$\left(\frac{1}{\Psi_T}\hat{H}\Psi_T - \int \Psi_T \hat{H}\Psi_T d\vec{r}\right)^2 > 0 \tag{2.1.13}$$

Which means that for the integral in equation 2.1.12 to be zero, the following must be true for all $\vec{r}$

$$\frac{1}{\Psi_T}\hat{H}\Psi_T - \int \Psi_T \hat{H}\Psi_T d\vec{r} = 0 \tag{2.1.14}$$

$$\Psi_T \hat{H}\Psi_T = \int \Psi_T \hat{H}\Psi_T d\vec{r} \tag{2.1.15}$$

The right hand side of this equation is just a real number. Naming this number E gives

$$\frac{1}{\Psi_T}\hat{H}\Psi_T = E \tag{2.1.16}$$

$$\hat{H}\Psi_T = E\Psi_T \tag{2.1.17}$$

Which is nothing but the eigenvalue equation stating that Ψ_T is an eigenstate of $\hat{H}$. This will serve as a test to see if the state we have found when minimizing the expectation value of E_L is an eigenstate of the Hamilton operator. It is just as easily (perhaps easier) shown that if Ψ_T *is* and eigenstate of $\hat{H}$, then the variance of the local energy is 0. This means that if the variance of our test function is *not* 0, then it is *not* an eigenfunction of $\hat{H}$. We can summarize this discussion as follows

$$\boxed{V_{E_L} = 0 \quad \text{if and only if} \quad \Psi_T \text{ is an eigenstate of } \hat{H}} \tag{2.1.18}$$

2.1.2 The test function Ψ_T

We will in this project use the trial wavefunctions of $\vec{r}_i = \begin{pmatrix} x_i \\ y_i \end{pmatrix}$ given by

$$\Psi_T(\vec{r}_0, ..., \vec{r}_{N-1}) = \mathrm{Det}_M(\phi_0, ..., \phi_{N-1}) \cdot \mathrm{J}(\vec{r}_0, ..., \vec{r}_{N-1}) \tag{2.1.19}$$

2.1.2.1 The modified Slater determinant $\mathbf{Det}_M(\phi_0, ..., \phi_{N-1})$

$\mathrm{Det}_M(\phi_0, ..., \phi_{N-1})$ is a modified *Slater determinant* defined as

$$\mathrm{Det}_M(\phi_0, ..., \phi_{N-1}) = |U| \cdot |D|$$

$$\begin{vmatrix} \phi_0(\vec{r}_0) & \phi_2(\vec{r}_0) & ... & \phi_{N-2}(\vec{r}_0) \\ \phi_0(\vec{r}_2) & \phi_2(\vec{r}_2) & ... & \phi_{N-2}(\vec{r}_2) \\ ... & ... & ... & ... \\ \phi_0(\vec{r}_{N-2}) & \phi_2(\vec{r}_{N-2}) & ... & \phi_{N-2}(\vec{r}_{N-2}) \end{vmatrix} \cdot \begin{vmatrix} \phi_1(\vec{r}_1) & \phi_3(\vec{r}_1) & ... & \phi_{N-1}(\vec{r}_1) \\ \phi_1(\vec{r}_3) & \phi_3(\vec{r}_3) & ... & \phi_{N-1}(\vec{r}_3) \\ ... & ... & ... & ... \\ \phi_1(\vec{r}_{N-1}) & \phi_3(\vec{r}_{N-1}) & ... & \phi_{N-1}(\vec{r}_{N-1}) \end{vmatrix} \tag{2.1.20}$$

Where $\phi_i(\vec{r}_i)$ is a wavefunction resembling one of the eigenfunctions of the Hamilton operator for *one* particle in a two dimensional harmonic oscillator, but parameterized by α in the following way:

$$\phi_i(\vec{r}_j) = H_{n_x}(\sqrt{\alpha\omega}x_j)H_{n_y}(\sqrt{\alpha\omega}y_j)\exp(-\alpha\omega(x^2+y^2)/2) \tag{2.1.21}$$

The reason for this modified version of the Slater determinant (whose real form can be explored elsewhere[1]) is that the spin parts of the wavefunctions are not incorporated in the expressions of $\phi_i(\vec{r}_j)$. The result is that if we were to insert these into a regular Slater determinant we would get 0, everytime. The modified Slater determinant Det_M avoids this issue while still conserving some of the most important properties of the Slater determinant.

$n_x(i)$ and $n_y(i)$ corresponds to the quantum numbers needed to "fill up" the system from the lowest energy levels twice (one for each spin configuration). For $i < 12$, the explicit depence of n_x, n_y on i is given in table 2.1.1.

$i =$	0	1	2	3	4	5	6	7	8	9	10	11
$n_x =$	0	0	1	1	0	0	2	2	1	1	0	0
$n_y =$	0	0	0	0	1	1	0	0	1	1	2	2

Table 2.1.1: The explicit dependence of n_x and n_y on i in the construction of the trial wavefunctions.

2.1.2.2 The Jastrow factor $\mathbf{J}(\vec{r}_0, ..., \vec{r}_{N-1})$

$\mathrm{J}(\vec{r}_0, ..., \vec{r}_{N-1})$ is a so-called *Jastrow factor*, which represents the electron repulsion part of the wavefunction, defined as

$$\boxed{\begin{array}{c} \mathrm{J}(\vec{r}_0, ...\vec{r}_{N-1}) = \prod_{i<j}^{N} \exp\left(\dfrac{a_{ij}r_{ij}}{1+\beta r_{ij}}\right) \\ \text{where} \quad r_{ij} = |\vec{r}_i - \vec{r}_j| \quad \text{and} \quad a_{ij} = \begin{cases} 1/3 & \text{if spin(i) and spin(j) are parallell} \\ 1 & \text{if spin(i) and spin(j) are anti-parallell} \end{cases} \end{array}} \tag{2.1.22}$$

From the expression of the Jastrow factor, we see that it is zero whenever $r_{ij} = 0$ for any pair $i \neq j$. This is what we want from such a factor, namely that the wavefunction is zero whenever there is no distance between two particles (i.e. $r_{ij} = 0$).

2.1.2.3 Closed form expression of the local energy

To evaluate the local energy

$$E_L(\vec{r}) = \frac{1}{\Psi_T}\hat{H}\Psi_T = \frac{1}{\Psi_T}\left(\sum_i -\frac{1}{2}\nabla_i^2 + \sum_i V(\vec{r}_i)\right)\Psi_T = \sum_i V(\vec{r}_i) - \frac{1}{2}\sum_i \frac{1}{\Psi_T}\nabla_i^2\Psi_T \tag{2.1.23}$$

[1] http://en.wikipedia.org/wiki/Slater_determinant

We need a lot of computational power. This is mainly due to the fact that we need to compute the sum of the laplacian operators on each particle. This is an easy task to do "brute force", but if we were able to find an analytical expression for the local energy, it would possibly simplify calculations by alot. Let's look at one of the terms in the laplacian sum, naming it LSP (Laplacian sum part)

$$LSP = \frac{1}{\Psi_T}\nabla_i^2\Psi_T \tag{2.1.24}$$

Inserting the trial wavefunction expression (equation 2.1.19) into the latter gives

$$\frac{1}{\Psi_T}\nabla_i^2\Psi_T = \frac{1}{\mathrm{Det}_M \mathrm{J}}\nabla_i^2(\mathrm{Det}_M \mathrm{J}) \tag{2.1.25}$$

The function Det_M is a product of two matrice-determinants $|U|$ and $|D|$ where $|U|$ handles all the particles assigned spin up and $|D|$ the ones assigned spin down. Particle i has either spin up or down, so let $|S_i|$ denote the matrix determinant which handles particle i and $|S_{j\neq i}|$ denote the one that doesn't, then

$$\frac{1}{\Psi_T}\nabla_i^2\Psi_T = \frac{1}{|S_i||S_{j\neq i}|\mathrm{J}}\nabla_i^2(|S_i||S_{j\neq i}|\mathrm{J}) = \frac{1}{|S_i||S_{j\neq i}|\mathrm{J}}|S_{j\neq i}|\nabla_i^2(|S_i|\mathrm{J}) = \frac{1}{|S_i|\mathrm{J}}\nabla_i^2(|S_i|\mathrm{J}) \tag{2.1.26}$$

Using the product rule of the laplacian operator gives

$$\boxed{\frac{1}{\Psi_T}\nabla_i^2\Psi_T = \frac{\nabla_i^2|S_i|}{|S_i|} + \frac{\nabla_i^2\mathrm{J}}{\mathrm{J}} + 2\frac{\nabla_i\mathrm{J}}{\mathrm{J}}\cdot\frac{\nabla_i|S_i|}{|S_i|}} \tag{2.1.27}$$

It is possible to find analytical expression for all these terms, and that has been already been done in a master thesis written by Jørgen Høgberget [3]. The arguments will not be repeated but the results are as follows (with names added for code reference)

$$NSS = \frac{\nabla_i|S_i|}{|S_i|} = \sum_{k=0}^{N/2}\left[(S^{-1})_{ki}\cdot\nabla_i\phi_{2k}(\vec{r}_i)\right] \tag{2.1.28a}$$

$$N2SS = \frac{\nabla_i^2|S_i|}{|S_i|} = \sum_{k=0}^{N/2}\left[(S^{-1})_{ki}\cdot\nabla_i^2\phi_{2k}(\vec{r}_i)\right] \tag{2.1.28b}$$

$$NJJ = \frac{\nabla_i\mathrm{J}}{\mathrm{J}} = \sum_{k\neq i}\frac{a_{ik}}{r_{ik}}\frac{\vec{r}_i-\vec{r}_k}{(1+\beta r_{ik})^2} \tag{2.1.28c}$$

$$N2JJ = \frac{\nabla_i^2\mathrm{J}}{\mathrm{J}} = \left|\frac{\nabla_i J}{J}\right|^2 + \sum_{k\neq i}\frac{a_{ik}}{r_{ik}}\frac{1-\beta r_{ik}}{(1+\beta r_{ik})^3} \tag{2.1.28d}$$

$\nabla_i\phi_k(\vec{r}_i)$ and $\nabla_i^2\phi_k(\vec{r}_i)$ scan be found simply by inserting and taking the derivative of the Hermite polynomials. The explicit formulas for ϕ_k for k in the range 0 to 11 is given in table 2.1.2 and are also taken from the master thesis by Jørgen Høgberget.

k	(n_x, n_y)	$\phi_k(\vec{r})$	$\nabla_i\phi_k(\vec{r}_i) = [\nabla_x, \nabla_y]$	$\nabla_i^2\phi_k(\vec{r}_i)$
0	(0,0)	1	$-\left[l^2x, l^2y\right]$	$l^2(l^2r^2-2)$
2	(1,0)	$2lx$	$-2l\left[(lx-1)(lx+1), l^2xy\right]$	$2l^3x(l^2r^2-4)$
4	(0,1)	$2ly$	$-2l\left[l^2xy, (ly-1)(ly+1)\right]$	$2l^3y(l^2r^2-4)$
6	(2,0)	$4l^2x^2-2$	$-2\left[l^2x(2l^2x^2-5), l^2y(2l^2x^2-1)\right]$	$2l^2(l^2r^2-6)(2l^2x^2-1)$
8	(1,1)	$4l^2xy$	$-4l^2\left[y(lx-1)(lx+1), x(ly-1)(ly+1)\right]$	$4l^4xy(l^2r^2-6)$
10	(0,2)	$4l^2y^2-1$	$-2\left[l^2x(2l^2y^2-1), l^2y(2l^2y^2-5)\right]$	$2l^2(l^2r^2-6)(2l^2y^2-1)$

Table 2.1.2: Table of derivatives of $\phi_k(\vec{r}_i)$ where $l = \sqrt{\alpha\omega}$. The factor $e^{-\frac{1}{2}l^2r^2}$ is ommitted from all expressions. The formula for $k+1$ is the same as for k if k is even (e.g. $\phi_0 = \phi_1$).

2.1.3 Discretization

2.1.4 The virial theorem

2.2 The numerical foundation

This is the section explaining the numerical theory upon which the project is built.

2.2.1 Monte Carlo simulations

A Monte Carlo simulation is a way of solving a mathematical or physical problem by generating a random (or pseudorandom [2]) sequence of numbers and evaluating some quantity on the assumption that our the random sequence of numbers is representative of the domain from which the quantity is evaluated. An example is evaluating the area of the unit circle by randomly placing points in a $[-1,1]\times[-1,1]$ grid and find the fraction points whose distance to the origin is ≤ 1 and multiply this fraction by the area of the grid (i.e. 4). Such a simple Monte Carlo simulation can give the result as shown in figure 2.2.1.

However, the method is not confined to this sort of problem, but can be applied to a variety of mathematical and physical problems. In this report, the method, through the Metropolis algorithm (see section 2.2.2) has been applied to a quantum mechanical system.

2.2.2 The Metropolis algorithm

The Metropolis algorithm is a method which cleverly employs a stoichastic approach in order to quickly estimate certain mathematical objects. The method is explained at lengths elsewhere[2], but in this section we will look at an example which captures the main idea of the method.

Suppose we have a PDF[3] $P(x)$ in a domain $[a,b]$ for which we want to calculate the expectation value $\langle g\rangle$ of some function $g(x)$. The integral we need to solve is then

[2] No electronic random number generator of today is truly random. The sequence of numbers generated will repeat itself after a long period. These periods however, are increadibly long and we will for this report consider the random number generators to be truly random.

[3] Probability Distribution Function

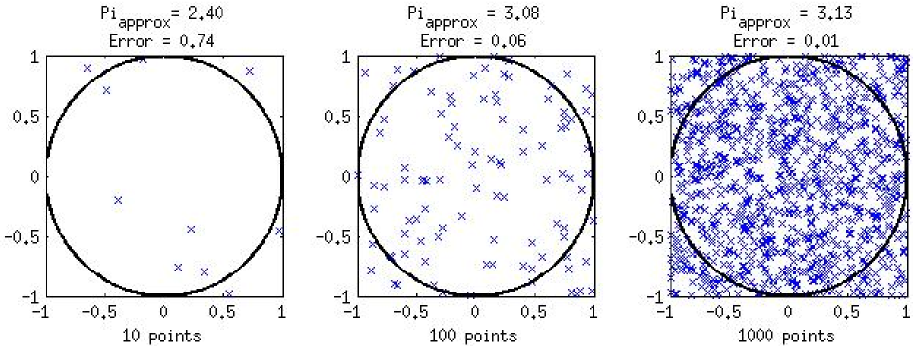

Figure 2.2.1: The results from a very simple Monte Carlo simulation of estimating the circle constant π. The precision increases with the number of points.

$$\langle g \rangle = \int_a^b P(x) g(x) dx \tag{2.2.1}$$

This integral can be approximated as follows

$$\int_a^b P(x) g(x) dx \approx \frac{b-a}{N} \sum_i P(x_i) g(x_i) \equiv I \tag{2.2.2}$$

Where x_i are some uniformly chosen values in the interval $[a, b]$. Now, imagine instead of picking values x_i uniformly and weighing them by multiplying $g(x)$ with $P(x)$ instead chose the values of $\tilde{x_i}$ from the PDF $P(x)$ and calculated the quantity $\tilde{I}$ given by

$$\tilde{I} = \frac{1}{N} \sum_i g(\tilde{x_i}) \tag{2.2.3}$$

It can be shown mathematically that for large enough N, these two quantities I and $\tilde{I}$ approach the same value. The problem with such an approach is that we need the precise expression for the PDF $P(x)$ and a robust algorithm for choosing random values from it. With the Metropolis algorithm however, we can use this approach *without* knowing the precise expression of the PDF and the relevant values from the domain come naturally.

The algorithm requires that we are able to calculate $\tilde{P}(x)$, an unnormalized version of $P(x)$ (i.e. some function $aP(x)$ proportional to $P(x)$). This may seem like a very strong requirement, but in many applications, as in this project, this is a much easier task than to calculate the precise PDF. The algorithm goes as follows. Starting with a position x choose a new trial position x_p by

$$x_p = x + \Delta x \tag{2.2.4}$$

Where Δx is a random step according to some rule (see subsections). Then generate a probability criteria s, a random number between zero and one. If

$$\frac{P(x_p)}{P(x)} = \frac{aP(x_p)}{aP(x)} = \frac{\tilde{P}(x_p)}{\tilde{P}(x)} \equiv w \geq s \tag{2.2.5}$$

We accept the trial position as our new x and if not we reject it. If we choose new values of x_i in this manner, the collection of x_i's will in fact reflect the PDF $P(x)$, which was what we needed in order to use equation 2.2.3. Note how equation 2.2.5 doesn't require us to have the exact form of the probability distribution function, only a function $\tilde{P}(x)$ proportional to it.

The intuition behind the algorithm is that for each new position x_i we generate is drawn towards the part of the domain where $P(x)$ is bigger. To see this, we note that if $P(x_p) > P(x)$ then $\frac{P(x_p)}{P(x)} > 1$ which is always bigger than $s \in [0, 1]$ and the new move is always accepted. Whereas if $P(x_p) < P(x)$, the move might be rejected. This allows new values of x_i to be chosen from where $P(x)$ is big, but at the same time allows values with lower values of $P(x)$ to be chosen. Which is what we expect from a PDF. The fact that for a large number M of such steps, the values x_i picked actually reflects the PDF requires some more mathematics, and once again we refer to the lecture notes of the course [2].

2.2.2.1 Brute force Metropolis

If we have no information about the physical nature of the system a reasonable way to model Δx is the following

$$\Delta x = r \Delta x_0 \tag{2.2.6}$$

Where r is a random number between 0 and 1 and Δx_0 is a predefined step length. The step length Δx_0 is affecting the effectiveness of the algorithm in two contradicting ways. A small step length increases the probability of each suggested move x_p being accepted, but weakens the ergodicity[4] of the method. Increasing the acceptance probability reduces the amount of times we need to evaluate the probability ratio w, but also increses the amount of Monte Carlo simulations needed in order to get a representative collection of x_i's. It can be argued that a good balance between these two aspects is to achieve an acceptance ratio (i.e. the ratio between accepted and rejected moves) of around 0.5.

This is called the "Brute Force Metropolis algorithm".

2.2.2.2 Importance sampling

If our current position x is in a region where the probability distribution is important, i.e. has a large value, a small step Δx would be favorable. This is because we want to sample many points in this region, which is what a small step allows. In contrast, if the current position x is in an unimportant region, we want a large step Δx since we don't mind moving a bit farther from the region we're in. The brute force approach produced a step independent of the PDF value in each point, which resulted in an optimal acceptance ratio of around 0.5. If we could introduce some sort of rule which adjusts the step Δx according to the value of the PDF in the point we currently are, this could allow us to achieve an acceptance rate of around 0.9 with the same ergodicity.

To make such a rule, we need to use our physical understanding of the system. One way of doing so is to consider the points to move as a random walker would where the resulting probability is equal to the PDF we're treating. Doing this, and invoking the Fokker-Planck and Langevin equations [5] , it can be shown that the choice of Δx is as follows

[4] The way in which the walker is able to reach all positions within a finite number of steps.

[5] Once again we refer to the lecture notes [2] for a more detailed explanation.

$$\Delta x = DF(x)\delta t + \eta \tag{2.2.7}$$

Where D is the diffusion term, $F(x)$ is a drift term which is responsible for pulling the particle towards regions where the PDF is important and η is a gaussian random number.

Using this approach is what we will call the "Metropolis algorithm with importance sampling".

2.2.2.3 The metropolis algorithm for our wavefunction

As discussed in section 2.1.1 we need to solve the integral

$$\langle E_L \rangle = \int P(\vec{r}) E_L(\vec{r}) d\vec{r} \tag{2.2.8}$$

Where we have a trial function

$$\Psi_T(\vec{r}_0, ..., \vec{r}_{N-1}, \alpha, \beta) \tag{2.2.9}$$

dependent on 2 trial parameters α and β where $\vec{r}_i = \begin{pmatrix} x_i \\ y_i \end{pmatrix}$. This is exactly the kind of problem the Metropolis algorithm can solve and the explicit algorithm for calculating $\langle E_L \rangle$ and $\langle E_L^2 \rangle$ is given in algorithm 2.1.

If we move only one particle at a time, each new trial position will be given by

$$\boxed{\mathbf{r_p} = \begin{pmatrix} x_0 & x_1 & ... & x_i & + \Delta\vec{r} & ... & x_{N-1} \\ y_0 & y_1 & & y_i & & & y_{N-1} \end{pmatrix}} \tag{2.2.10}$$

Where $\mathbf{r} = \begin{pmatrix} x_0 & x_1 & ... & x_{N-1} \\ y_0 & y_1 & ... & y_{N-1} \end{pmatrix}$. If we are using the brute force approach, then $\Delta\vec{r}$ is simply given by

$$\boxed{\Delta\vec{r} = \Delta r \cdot \vec{\text{rand}}} \tag{2.2.11}$$

Where Δr is a predefined step length and $\vec{\text{rand}}$ is a random 2-vector with elements between -1 and 1.

If we want to implement importance sampling however, we need expressions for the terms in equation 2.2.7. These terms can be shown [1] to be

$$D = \frac{1}{2} \tag{2.2.12}$$

Which stems from the fact that the drift is caused by kinetic energy in front of which is a factor $\frac{1}{2}$ and

$$F = 2\frac{1}{\Psi_T}\nabla_i \Psi_T \tag{2.2.13}$$

Data:
An initial position matrix $\mathbf{r} = (\vec{r}_0, \vec{r}_1 ... \vec{r}_{N-1}) = \begin{pmatrix} x_0 & x_1 & ... & x_{N-1} \\ y_0 & y_1 & ... & y_{N-1} \end{pmatrix}$
A method of chosing the step $\quad \text{Method}(\mathbf{r}) = \Delta\vec{r} = \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix}$
Result:
The expectation value of the local energy: $\langle E_L \rangle$
The expectation value of the local energy squared: $\langle E_L^2 \rangle$

begin
  $\textit{cumulative\_local\_energy} = E_L(\mathbf{r}, \alpha, \beta)$ // Initialization
  $\textit{cumulative\_local\_energy\_squared} = 0$
  $\textit{counter} = 0$
  while $\textit{counter} < M$ **do**
    $i = \text{randint}(0, 1, ..., N-1)$ // Choose random element index
    $\Delta\vec{r} = \text{Method}(\mathbf{r})$ // Create a random two-dimensional step
    $\mathbf{r_p} = \begin{pmatrix} x_0 \; x_1 & ... & x_i + \Delta\vec{r} & ... & x_{N-1} \\ y_0 \; y_1 & & y_i & & y_{N-1} \end{pmatrix}$ // Create a trial position matrix
    $s = \text{randint}(0,1)$ // Generate a probability criteria
    $w = |\psi(\alpha, \beta, \mathbf{r_p})|^2 / |\psi(\alpha, \beta, \mathbf{r})|^2$ // Calulate the probability ratio
    if $w \geq s$ **then**
      $\vec{r} = \vec{r}_p$
      $E_L(\mathbf{r}, \alpha, \beta) = \frac{1}{\Psi_T(\mathbf{r}, \alpha, \beta)} \hat{H} \psi_T(\mathbf{r}, \alpha, \beta)$ // Calculate the local energy
    end
    $\textit{cumulative\_local\_energy} \mathrel{+}= E_L(\mathbf{r}, \alpha, \beta)$ // Update *cumulative_local_energy*
    $\textit{cumulative\_local\_energy\_squared} \mathrel{+}= E_L(\mathbf{r}, \alpha, \beta)^2$
    // Update *cumulative_local_energy_squared*
    $\textit{counter} \mathrel{+}= 1$ // Update *counter*
  end
  Calculate $\langle E_L \rangle = \frac{\textit{cumulative\_local\_energy}}{M}$
  Calculate $\langle E_L^2 \rangle = \frac{\textit{cumulative\_local\_energy\_squared}}{M}$
end

Algorithm 2.1: The metropolis algorithm used for finding the expecation value of the local energy and the expecation value of the local energy squared. "Method" refers to either the brute force approach or importance sampling.

The formula for using importance sampling when choosing the trial position $\mathbf{r_p}$ for our wave-function is thus

$$\boxed{\Delta\vec{r} = \left(\frac{1}{\Psi_T} \nabla_i \Psi_T \right) \delta t + \eta} \tag{2.2.14}$$

We can rewrite this, as in section 2.1.2.3

$$\frac{1}{\Psi_T} \nabla_i \Psi_T = \frac{1}{|S_i| \mathrm{J}} \nabla_i (|S_i| \mathrm{J}) = \frac{1}{|S_i| \mathrm{J}} \left(|S_i| \nabla_i \mathrm{J} + \mathrm{J} \nabla_i |S_i| \right) \tag{2.2.15}$$

$$\boxed{\frac{1}{\Psi_T}\nabla_i\Psi_T = \frac{\nabla_i|S_i|}{|S_i|} + \frac{\nabla_i \mathrm{J}}{\mathrm{J}}} \tag{2.2.16}$$

Expressions we can find both from numerical differenciation and the close form expressions in equations 2.1.28. In this report, we will use both these approaches and compare the CPU time needed.

3 Results and discussion

Section listing results and discussing them.

3.1 Subsection

The same as the method section.

4 Conclusion

References

[1] Morten Hjorth-Jensen. Additional slides for monte carlo project. http://www.uio.no/.../montecarloaddition.pdf, 2014.

[2] Morten Hjorth-Jensen. *Computational Physics - Lecture Notes Fall 2014*. August 2014.

[3] Jørgen Høgberget. Quantum monte-carlo studies of generalized many-body systems, June 2013.