Project 2 - FYS3150*

Andreas G. Lefdalsnes
Student: University of Oslo, Department of Physics
email-address: andregl@student.matnat.uio.no

Tellef Storebakken
Student: University of Oslo, Department of Physics
email-address: tellefs@student.matnat.uio.no
(Dated: October 4, 2016)

In this project we solve the Schrodinger equation for two electrons in a 3D harmonic oscillator potential. We solve with and without electron repulsion, and compare the results. To accomplish this we apply a general method of discretizing the domain and reducing the problem to an eigenvalue equation. We thereafter apply Jacobi's rotation algorithm to obtain the eigenvalues of the matrix. We also apply the principles of unit testing by testing the algorithm for some simple problems with known solutions.

I. INTRODUCTION

In this project we aim to solve the Schrodinger equation for two electrons in a 3D harmonic oscillator potential. We will be solving with and without the repulsive Coulomb potential, and comparing the results. For the case of no repulsion we have an analytical expression for the energies, and this will be useful in determining the accuracy of our results. Assume spherical symmetry.

II. THEORY AND METHODS

A. The radial equation

We begin by studying the radial part of Schrodingers' equation for a single electron in a harmonic oscillator potential [1].

$$-\frac{\hbar^2}{2m}(\frac{1}{r^2}\frac{d}{dr}r^2-\frac{l(l+1)}{r^2})R(r)+V(r)R(r)=ER(r) \quad (1)$$

The potential $V(r)=\frac{1}{2}kr^2$ is the harmonic oscillator potential with $k=m\omega^2$ and E is the energy of the electron. ω is the oscillator frequency and the allowed energies are

$$E_{nl}=\hbar\omega(2n+l+\frac{3}{2}) \quad (2)$$

Where the quantum number $n=0,1,2..$ is the energy quantum number and $l=0,1,2...$ is the orbital momentum quantum number. Introducing $R(r)=(1/r)u(r)$ our equation can be rewritten in terms of the second derivative d^2/dr^2:

$$-\frac{\hbar^2}{2m}\frac{d^2}{dr^2}u(r)+(V(r)+\frac{l(l+1)}{r^2}-\frac{\hbar^2}{2m})u(r)=Eu(r) \quad (3)$$

We introduce a dimensionless variable $\rho=(1/\alpha)r$ where alpha is a constant of dimension length and obtain

$$-\frac{\hbar^2}{2m\alpha^2}\frac{d^2}{d\rho^2}u(\rho)+(V(\rho)+\frac{l(l+1)}{\rho^2}\frac{\hbar^2}{2m\alpha^2})u(\rho)=Eu(\rho) \quad (4)$$

In this project we will be interested in the case $l=0$. Now since we are working in spherical coordinates, $r\in[0,\infty)$. Since we require $R(r)$ to go to zero at the boundaries, when we make the substituion $R(r)=(1/r)u(r)=(1/r)u(\alpha\rho)$ we obtain the boundary conditions for $u(\rho)$: $u(0)=u(\infty)=0$.

We insert $V(\rho)=\frac{1}{2}k\alpha^2\rho^2$ and obtain

$$-\frac{\hbar^2}{2m\alpha^2}\frac{d^2}{d\rho^2}u(\rho)+\frac{1}{2}k\alpha^2\rho^2u(\rho)=Eu(\rho) \quad (5)$$

To obtain a simpler expression we multiply by $2m\alpha^2\rho^2/\hbar^2$ and fix α such that

$$\frac{mk}{\hbar^2}\alpha^4=1 \quad (6)$$

and define

$$\lambda=\frac{2m\alpha^2}{\hbar^2}E \quad (7)$$

so we can rewrite our equation as

$$-\frac{d^2}{d\rho^2}u(\rho)+\rho^2u(\rho)=\lambda u(\rho) \quad (8)$$

* Computational Physics, autumn 2016, University of Oslo

To solve this equation we discretize the domain and define minimum and maximum values for ρ, $\rho_{min} = \rho_0 = 0$ and ρ_{max}. ρ_{max} cannot be chosen to be ∞ so we must take care to set it sufficiently large in order to obtain the correct solution. With N mesh points let

$$h = \frac{\rho_{max} - \rho_0}{N} \tag{9}$$

and we obtain a discrete set of values for ρ,

$$\rho_i = \rho_0 + ih \qquad i = 0, 1, 2..., N \tag{10}$$

Replacing the second order derivative by the 2nd order central difference we can write our equation as

$$-\frac{u_{i+1} + u_{i-1} - 2u_i}{h^2} + \rho_i^2 u_i = \lambda u_i \tag{11}$$

Where $u_i = u(\rho_i)$ is the discretized version of our function. We let $V_i = \rho_i^2$ and rewrite this as a matrix equation

$$\boldsymbol{Au} = \lambda \boldsymbol{u} \tag{12}$$

Since the endpoints are known we let A be a matrix of dimension $(n-2)\cdot(n-2)$ and u a vector of dimension (n-2).

$$\boldsymbol{Au} = \begin{pmatrix} \frac{2}{\hbar^2} + V_1 & -\frac{1}{\hbar^2} & 0 & \cdots & \cdots & 0 \\ -\frac{1}{\hbar^2} & \frac{2}{\hbar^2} + V_2 & -\frac{1}{\hbar^2} & 0 & \cdots & \cdots \\ 0 & -\frac{1}{\hbar^2} & \frac{2}{\hbar^2} + V_3 & -\frac{1}{\hbar^2} & 0 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & \cdots & -\frac{1}{\hbar^2} & \frac{2}{\hbar^2} + V_{N-3} & -\frac{1}{\hbar^2} \\ 0 & \cdots & \cdots & \cdots & -\frac{1}{\hbar^2} & \frac{2}{\hbar^2} + V_{N-2} \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ \\ u_{N-2} \end{pmatrix} = \lambda \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ \\ u_{N-2} \end{pmatrix} \tag{13}$$

where $u_0 = u_{N-1} = 0$.

B. Coulomb Interaction

For two electrons with no Coulomb interaction in a harmonic oscillator potential the Schroedinger equation can be written as

$$(-\frac{\hbar^2}{2m}\frac{d^2}{dr_1^2} - \frac{\hbar^2}{2m}\frac{d^2}{dr_2^2} + \frac{1}{2}kr_1 2 + kr_2^2)u(r_1, r_2) = Eu(r_1, r_2) \tag{14}$$

With no interaction the two-electron wavefunction $u(r_1, r_2)$ can be written as a product of two single-electron wavefunction. Introducing the relative coordinate $\boldsymbol{r} = \boldsymbol{r_1} - \boldsymbol{r_2}$ and the center of mass coordinate $\boldsymbol{R} = \frac{1}{2}(\boldsymbol{r_1} + \boldsymbol{r_2})$ the radial equation becomes

$$(-\frac{\hbar^2}{m}\frac{d^2}{dr^2} - \frac{\hbar^2}{4m}\frac{d^2}{dR2} + \frac{1}{4}kr^2 + kR^2)u(r, R) = Eu(r, R) \tag{15}$$

the solution can be separated as $u(r, R) = \psi(r)\phi(R)$ and the energy is a sum of the relative energy E_r and center-of-mass energy E_R

$$E = E_r + E_R \tag{16}$$

adding the repulsive Coulomb interaction

$$V(r_1, r_2) = \frac{\beta e^2}{|\boldsymbol{r_1} - \boldsymbol{r_2}|} = \frac{\beta e^2}{r} \tag{17}$$

where $\beta e^2 = 1.44$ eVnm. The relative motion Schroedinger equation becomes

$$(-\frac{\hbar^2}{m}\frac{d^2}{dr^2} + \frac{1}{4}kr^2 + \frac{\beta e^2}{r})\psi(r) = E\psi(r) \tag{18}$$

introducing $\rho = r/\alpha$, defining a new frequency

$$\omega_r^2 = \frac{1}{4}\frac{mk}{\hbar^2}\alpha^4 \tag{19}$$

fixing α

$$\alpha\frac{m\beta e^2}{\hbar^2} = 1 \tag{20}$$

and defining

$$\lambda = \frac{m\alpha^2}{\hbar^2}E \tag{21}$$

and we obtain Schroedinger's equation

$$-\frac{d^2}{d\rho^2}\psi(\rho) + \omega_r^2\rho^2\psi(\rho) + \frac{1}{\rho} = \lambda\psi(\rho) \tag{22}$$

which using the methods described in the previous section can be solved numerically as an eigenvalue problem.

C. Jacobi's rotation method

For a real symmetric $n \cdot n$ matrix $\boldsymbol{A}$, we have n eigenvalues λ_i and there exists a real orthogonal matrix $\boldsymbol{S}$ such that [2]

$$\boldsymbol{S^T A S} = diag(\lambda_1, \lambda_2, ..., \lambda_n) \tag{23}$$

In general a similarity transform

$$\boldsymbol{B = S^T A S} \qquad \boldsymbol{S^T S = I} \tag{24}$$

will have the same eigenvalues as $\boldsymbol{A}$, but different eigenvectors. If we have an orthogonal basis v_i such that

$$\boldsymbol{v_j^T v_i = \delta_{ij}} \tag{25}$$

a unitary transformation $\boldsymbol{U^T U = I}$ will preserve the orthogonality of the basis vectors, such that if $w_i = Uv_i$

$$\boldsymbol{w_j^T w_i = (U v_j)^T (U v_i) = v_j^T U^T U v_i = \delta_{ij}} \tag{26}$$

The idea of Jacobi's method is to perform a series of similarity transformations such that we receive a diagonal matrix $\boldsymbol{D}$ with the eigenvalues of $\boldsymbol{A}$ on the diagonal.

Consider a $N \times N$ matrix

$$S = \begin{pmatrix} 1 & 0 & \cdots & 0 & 0 \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \cdots & 0 & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & 0 & \cdots \\ 0 & 0 & \cdots & \cos(\theta) & 0 \cdots & 0 & \sin(\theta) \\ 0 & 0 & \cdots & 0 & 1 \cdots & 0 & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & 0 & \cdots \\ 0 & 0 & \cdots & 0 & 0 \cdots & 1 & 0 \\ 0 & 0 & \cdots & -\sin(\theta) & 0 \cdots & 0 & \cos(\theta) \end{pmatrix} \tag{27}$$

with $\boldsymbol{S^T = S^{-1}}$. It performs a plane rotation around an angle θ in n-dimensional space. We apply a similarity transformation $\boldsymbol{S^T A S}$ to our matrix $\boldsymbol{A}$ such that the largest non-diagonal elements $a_{kl} = a_{lk}$ become zero. For a real symmetric matrix we thus reduce the norm of the offdiagonal elements

$$\text{off}(\boldsymbol{A}) = \sqrt{\sum_{i=1}^{n} \sum_{j=1, j \neq i}^{n} a_{ij}^2} \tag{28}$$

and after a sufficient number of iterations we are guaranteed a matrix

$$\boldsymbol{S_N^T S_{N-1}^T .... S_1^T A S_1 .... S_{N-1} S_N = diag(\lambda_1, ..., \lambda_N)} \tag{29}$$

Numerically it is sufficient that the largest non-diagonal element is smaller than some tolerance ϵ

$$max(a_{ij}^2) \leq \epsilon \tag{30}$$

Convergence is guaranteed with the Jacobi method, however the number of floating point operations is quite large, $\mathcal{O}(12n^3)$ operations in order to zero out non-diagonal elements.

D. Unit testing

In order to test the accuracy of our methods we can introduce some simple unit tests. For example we could check that our implementation of Jacobi's method for a simple symmetric matrix with known eigenvalues produces the exact results. For our problem the 3 lowest eigenvalues are known in the case of no interaction, and it is easy to test for example if our algorithm produces the lowest eigenvalue to some precision.
In addition, the solutions to the Schroedinger equation form an orthogonal basis. In the previous subsection I show that an orthogonal transformation preserves orthogonality, so we can test our algorithm by checking to see if the resulting eigenvectors are orthogonal.
Finally, we test our function to find the maximum off-diagonal value of a matrix by testing it on a simple 3x3 symmetric matrix.

III. RESULTS AND DISCUSSION

A. Programming precision

First we checked how many mesh points N we needed. We ran the program for a system where we knew the lowest eigenvalue and wanted the difference to be less than 10^{-4}. When doing this for a known problem where the lowest eigenvalue should have been $\lambda_{theory} = 3$ we found that we needed $N = 200$ mesh points to get this precision.
We also checked how many similarity transformations we needed before all the non-diagonal elements were essentially zero. In the program we checked this by adding a counter which checked how many times we rotated matrix elements. We found that with $N = 200$ mesh points, we needed approximately 66600 transformations. Our tolerance when doing this was that the diagonal elements had to be lower than $tol = 10^{-8}$.

Now that we found our right mesh point precision, we timed our algorithm and compared it to the $c++$ library Armadillo. In Table (1) the times are included.

From Table (1) we can see that the method we have used for finding the eigenvalues is very inefficient compared to the method Armadillo is using. We have then

TABLE (I) Time comparison

Method	**Time [s]**
Jacobi algorithm	13.73
Armadillo	0.02

used the armadillo-function $eig\_sym(A)$ which also takes and symmetric matrix, just like we have assumed in our Jacobi-method. The difference is while we assume the whole matrix is filled with elements, eig_sym takes advantage of the fact that we have a sparse matrix (we have a tridiagonal matrix where the rest of the elements are zeros).

B. Coulomb interaction

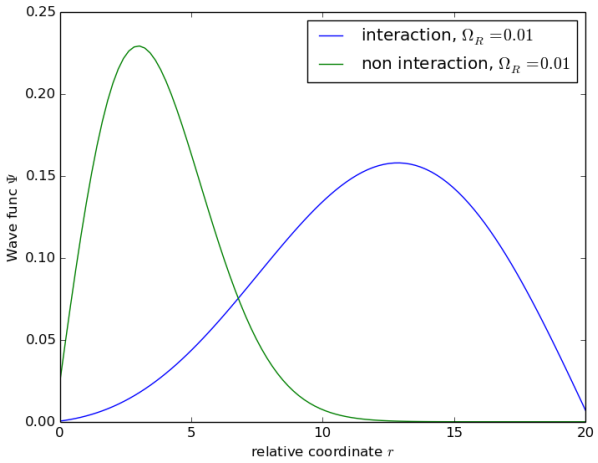

FIG. (1)

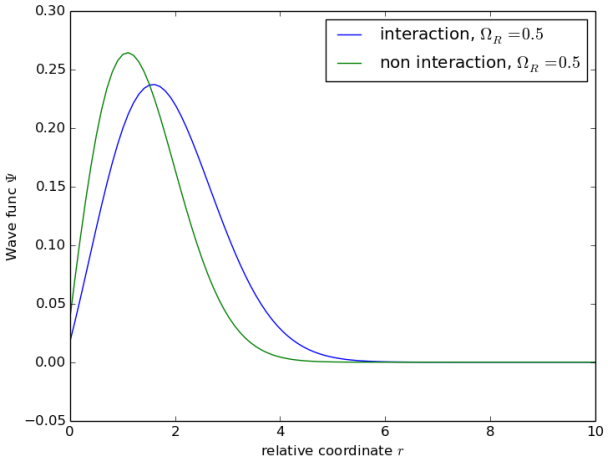

FIG. (2)

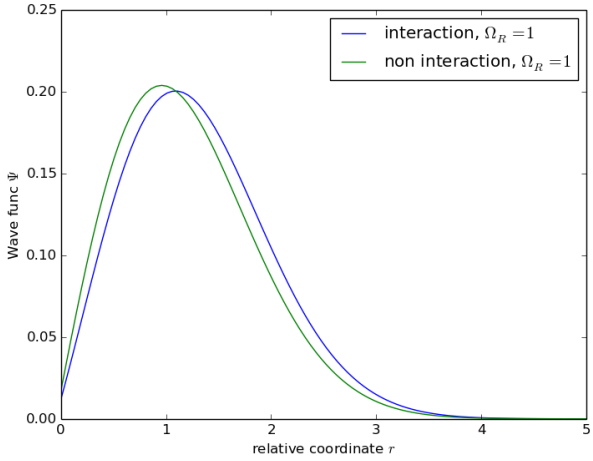

FIG. (3)

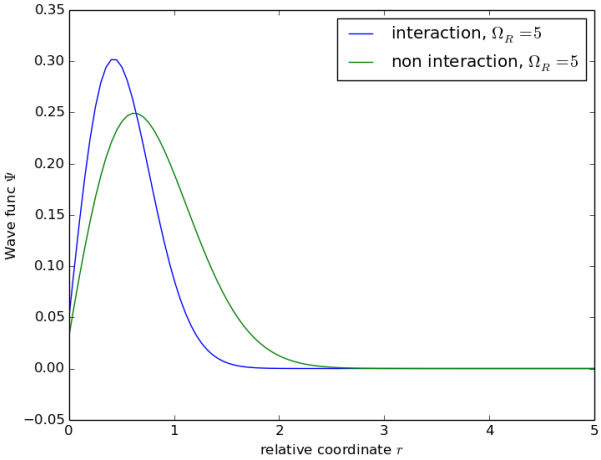

FIG. (4)

C. Tests

We were to implement some unit-tests in to our program.

[1] All theory in this project adapted from FYS3150 Project 2 (Fall 2016) $< link >$.
[2] See M.H. Jensen, Computational Physics: Lecture Notes Fall 2015, ch. 7.3, available at $< link2 >$.