FYS4150 Project 2: Schrödinger's equation for two electrons in a 3-dimensional harmonic oscillator well

Marie Foss, Maria Hammerstrøm

Abstract

We compute the eigenvalues for the 3D harmonic oscillator with and without Coulomb interactions using a brute force Jacobi's method, as well as the numerical package Armadillo. Our results correspond with known analytical solutions.

Github: *https://github.com/mariahammerstrom/Project2*

1 Introduction

The aim of this project is to solve Schrödinger's equation for two electrons in a three-dimensional harmonic oscillator well with and without a repulsive Coulomb interaction. We are first interested in the solution of the radial part of Schrödinger's equation for *one* electron. This equation reads

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

In our case $V(r)$ is the harmonic oscillator potential $(1/2)kr^2$ with $k = m\omega^2$ and E is the energy of the harmonic oscillator in three dimensions. The quantum number l is the orbital momentum of the electron. The oscillator frequency is ω and the energies are

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

with $n = 0, 1, 2, \ldots$ and $l = 0, 1, 2, \ldots$. In this project we use $l = 0$, which means we are studying the ground state.

We will solve Eq. (1) by reformulating it in a discretized form as an eigenvalue equation to be solved with Jacobi's method.

After some substitutions and introducing the dimensionless variable $\rho = (1/\alpha)r$ where α is of dimension length, we can rewrite Eq. (1) as

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

This is the first equation to solve numerically. In three dimensions the eigenvalues for $l = 0$ are $\lambda_0 = 3, \lambda_1 = 7, \lambda_2 = 11, \ldots$.

We use the by now standard expression for the second derivative of a function u

$$u''=\frac{u(\rho+h)-2u(\rho)+u(\rho-h)}{h^2}+O(h^2),$$

where h is our step length. For a given number of steps n_{step}, the step length is defined as

$$h=\frac{\rho_{\max}-\rho_{\min}}{n_{\text{step}}}. \tag{1}$$

Next we define minimum and maximum values for the variable ρ, $\rho_{\min} = 0$ and $\rho_{\max}$, respectively. Define an arbitrary value of ρ as

$$\rho_i=\rho_{\min}+ih \qquad i=0,1,2,\ldots,n_{\text{step}}. \tag{2}$$

Now we can write the Schrödinger equation in a compact way

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

where $V_i = \rho_i^2$ is the harmonic oscillator potential.

Define first the diagonal matrix element

$$d_i=\frac{2}{h^2}+V_i, \tag{3}$$

and the non-diagonal matrix element (which is a mere constant, and are all equal)

$$e_i=-\frac{1}{h^2}. \tag{4}$$

With these definitions the Schrödinger equation takes the following form

$$d_iu_i+e_{i-1}u_{i-1}+e_{i+1}u_{i+1}=\lambda u_i, \tag{5}$$

where u_i is unknown. We can write the latter equation as a matrix eigenvalue problem

$$\begin{pmatrix} d_1 & e_1 & 0 & \ldots & 0 & 0 \\ e_1 & d_2 & e_2 & \ldots & 0 & 0 \\ 0 & e_2 & d_3 & e_3 & \ldots & 0 \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ 0 & \ldots & \ldots & \ldots & d_{n_{\text{step}}-2} & e_{n_{\text{step}}-1} \\ 0 & \ldots & \ldots & \ldots & e_{n_{\text{step}}-1} & d_{n_{\text{step}}-1} \end{pmatrix}\begin{pmatrix} u_1 \\ u_2 \\ \ldots \\ \ldots \\ \ldots \\ u_{n_{\text{step}}-1} \end{pmatrix}=\lambda\begin{pmatrix} u_1 \\ u_2 \\ \ldots \\ \ldots \\ \ldots \\ u_{n_{\text{step}}-1} \end{pmatrix} \tag{6}$$

Secondly, we will consider the Schrödinger equation for *two* electrons by studying two electrons in a harmonic oscillator well which also interact via a repulsive Coulomb interaction.

We start by writing the single-electron equation as

$$-\frac{\hbar^2}{2m}\frac{d^2}{dr^2}u(r) + \frac{1}{2}kr^2u(r) = E^{(1)}u(r),$$

where $E^{(1)}$ stands for the energy with one electron only. For two electrons with *no* repulsive Coulomb interaction, we have the following Schrödinger equation:

$$\left(-\frac{\hbar^2}{2m}\frac{d^2}{dr_1^2} - \frac{\hbar^2}{2m}\frac{d^2}{dr_2^2} + \frac{1}{2}kr_1^2 + \frac{1}{2}kr_2^2\right)u(r_1, r_2) = E^{(2)}u(r_1, r_2).$$

After a series of substitutions, this can be written as

$$-\frac{d^2}{d\rho^2}\psi(\rho) + \omega_r^2\rho^2\psi(\rho) + \frac{1}{\rho} = \lambda\psi(\rho) \tag{7}$$

In this case the potential is:

$$V_i = \omega_r^2\rho^2 + 1/\rho \tag{8}$$

where we will study the cases where the oscillator frequency ω_r is $\omega_r = 0.01$, $\omega_r = 0.5$, $\omega_r = 1$ and $\omega_r = 5$ for the ground state only, that is, the lowest-lying state.

2 Methods

2.1 Jacobi's method

In this project we will use Jacobi's method to find the eigenvalues, which consists of doing a number of similarity transformations

$$\mathbf{S}^T\mathbf{A}\mathbf{S} = \mathbf{B}$$

where $\mathbf{A}$ is the matrix in our given problem, and $\mathbf{S}$ is an $(n \times n)$ orthogonal transformation matrix

$$\mathbf{S} = \begin{pmatrix} 1 & 0 & \dots & 0 & 0 & \dots & 0 & 0 \\ 0 & 1 & \dots & 0 & 0 & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & 0 & \dots \\ 0 & 0 & \dots & \cos\theta & 0 & \dots & 0 & \sin\theta \\ 0 & 0 & \dots & 0 & 1 & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & 0 & \dots \\ 0 & 0 & \dots & 0 & 0 & \dots & 1 & 0 \\ 0 & 0 & \dots & -\sin\theta & \dots & \dots & 0 & \cos\theta \end{pmatrix}.$$

The recipe is to choose θ so that all non-diagonal matrix elements b_{kl} become zero.

In the algorithm we define the quantities $\tan\theta = t = s/c$ with $s = \sin\theta$ and $c = \cos\theta$. By some manipulations, we can write:

$$c = \frac{1}{\sqrt{1+t^2}}. \tag{9}$$

This is used to calculate the difference between matrices $\mathbf{B}$ and $\mathbf{A}$, which we want to minimize:

$$||\mathbf{B} - \mathbf{A}||_F^2 = 4(1-c)\sum_{i=1, i\neq k,l}^{n}(a_{ik}^2 + a_{il}^2) + \frac{2a_{kl}^2}{c^2}.$$

In order to minimize this equation, c should be as close to 1 as possible. c depends on t, which is expressed as:

$$t = -\tau \pm \sqrt{1+\tau^2}. \tag{10}$$

Making t as small as possible, will make c as close to 1 as possible, as seen from Eq. (9). That means choosing the smaller of the roots in Eq. (10).

The **algorithm** for Jacobi's method can be described as follows:

- Choose a tolerance ϵ, typically 10^{-8} or smaller.
- Find matrix element a_{kl} with the largest value and its indices k and l.
- Check that $\max(a_{kl}^2) > \epsilon$ for the off-diagonal matrix elements.
- Compute τ, $\sin\theta$, $\cos\theta$, $\tan\theta$.
- Compute the similarity transformations for this set of values for (k, l), which will create a new matrix, $\mathbf{B} = \mathbf{S}(k, l, \theta)^T\mathbf{A}\mathbf{S}(k, l, \theta)$.
- Continue to follow these steps until $\max(a_{kl}^2) > \epsilon$ is no longer the case.

2.2 Numerical library

In addition to solving the problem in a brute-force way using Jacobi's method, we have also used the numerical library Armadillo, and compared the results from these two methods.

3 Results

3.1 Harmonic oscillator

In the case of the harmonic oscillator there is an analytical solution for the lowest eigenvalues. For $l = 0$ these are $\lambda_0 = 3, \lambda_1 = 7, \lambda_2 = 11, \dots$ Solving this numerically will not lead to the exact values, but we can get very close to this. We have computed the eigenvalues using our algorithm as described in the previous section, as well as using the Armadillo package.

In order to get the lowest three eigenvalues with four leading digits, we need around $n_{\text{step}} = 200$. Then the lowest three eigenvalues are: $\lambda_0 = 2.999, \lambda_1 = 6.99$ and $\lambda_2 = 10.99$ using $\rho_{\max} = 10.0$.

The eigenvalues will depend on our choice for $\rho_{\max}$. By testing we find that a lower $\rho_{\max}$ makes the eigenvalues more precise, say $\rho_{\max} = 4.0$, but gets worse for values lower than this, making the eigenvalues larger than they should be. For values higher than 10.0 the eigenvalues also are bad, becoming too small.

The number of similarity transformation needed for the non-diagonal matrix elements to become zero depends on the dimensionality of the matrix. An upper limit to the allowed number of

similarity transformations is in the range $12n^3 - 20n^3$ according to the lecture notes. We have set the limit to $50n^3$ in our code to be on the safe side. When running the code, the number of transformations is counted. For different dimensionalities we find (for $\rho_{\max} = 10.0$):

n	No. of iterations	Time (s)
5	11	0
10	41	0
20	176	0
50	1727	0
100	11315	0
200	61308	17

where the time is for our algorithm. Using Armadillo resulted in 0 seconds for all of the above n-values. Both methods give similar results for the eigenvalues. See `selected_results.txt` to see output examples.

We want to estimate the number of transformations and extract a behavior as function of the dimensionality of the matrix. The relationship between n and number of iterations are plotted in Fig. 1 where the blue line represents the output from our code and the red line is an approximation of $n^{2.1}$ we found to be a nice fit.

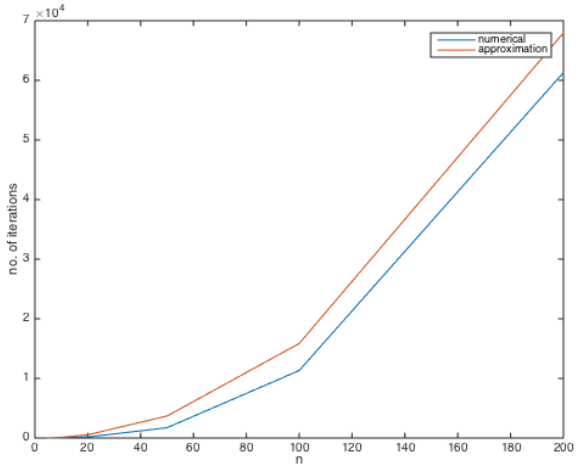

Figure 1: Number of iterations as a function of n, approximated by $n^{2.1}$ (blue).

From comparisons with analytical solutions, we see that our program is working well and we can proceed with the next potential of interest.

3.2 Harmonic oscillator with Coulomb interactions

Next we want to study the harmonic oscillator with Coulomb interactions. For specific oscillator frequencies there is an analytical answer to Eq. (7), described in the article by M. Taut, Phys. Rev. A 48, 3561 - 3566 (1993). Thus, we can compare our numerical solution with the analytical solution, given as:

$$\begin{aligned} V_0 &= \frac{3}{2}\left[\frac{\omega_r}{2}\right]^{2/3} \\ \omega_e &= \sqrt{3}\omega_r \\ \epsilon_m^{'} &= V_0 + \omega_e[m + \frac{1}{2}], \, m = 0, 1, \ldots \end{aligned} \tag{11}$$

where $\epsilon_m^{'}$ gives the analytical eigenvalues.

For $n = 200$ and $\rho_{\max} = 50.0$ we find:

ω_r	0.01	0.5	1.0	5.0
λ_0	0.105773	2.22507	4.0372	16.9105
λ_1	0.141504	4.11139	7.81445	34.4305
λ_2	0.178008	6.017	11.5845	49.9979

ω_r	0.01	0.5	1.0	5.0
$\epsilon_0^{'}$	0.105041	2.05658	3.62193	14.1863
$\epsilon_1^{'}$	0.139682	3.78863	7.08603	31.5068
$\epsilon_2^{'}$	0.174323	5.52068	10.5501	48.8273

We see that the analytic and numerical eigenvalues correspond quite nicely for all ω_r, though the analytical eigenvalues are generally lower in value than the numerical eigenvalues.

We calculate the eigenvectors for the ground state using Armadillo's `eigsym` function. The eigenvectors represents the wave function. A plot of the wave function for two electrons as a function of relative coordinates ρ and different values of ω_r is plotted in Fig.2. We see that a stronger oscillator potential ω_r creates a deeper potential well, as expected. ρ represents the distance between the two electrons, so we see that for a larger oscillator potential, the electrons will be closer.

For the case of $\omega_r = 0.01$, the chosen value of ρ_{max} is important. ρ_{max} needs to be quite large to make the plot come out right in this case. We landed on $\rho_{max} = 40$.

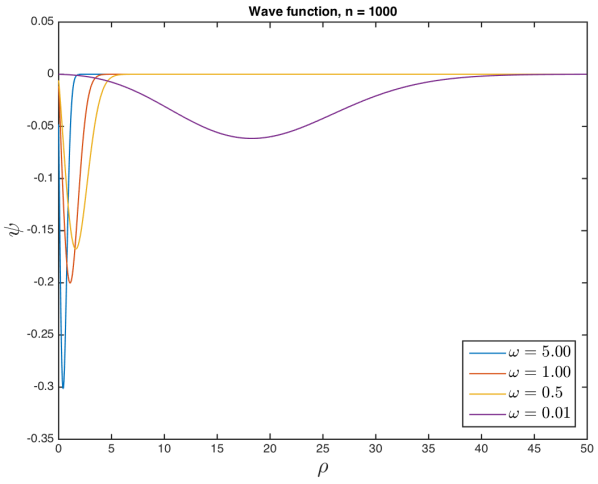

Figure 2: The wave function.

By plotting the squared of the eigenvector values, we get the probability distribution, shown in Fig. 3. The figure shows the probability that the electrons will be in a given distance ρ. For a smaller oscillator potential ω_r the electrons are more likely to be far apart, and vice versa.

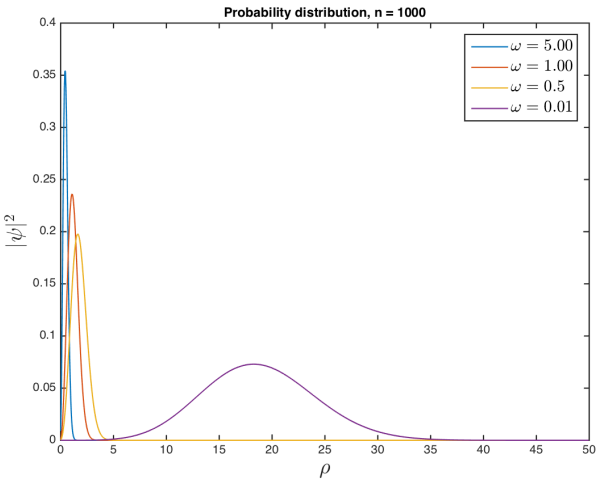

Figure 3: The normalized probability distribution.

3.3 Unit tests

In our code we have used several tests to make sure that the results are sound. The tests include:

- Checking that the rotation matrix only makes one rotation when $n = 2$.
- Checking our results for the eigenvalues with analytical solutions.

4 Conclusions

The number of iterations needed to find the eigenvalues of an $n \times n$ matrix was found to be approximately $n^2.1$. For higher values of ω_r, the relative distance between the two electrons decreased.

5 List of codes

The codes developed for this project are:

`main.cpp` – main program, C++
`plotting.m` – plotting program, MATLAB