Project 2 - FYS3150 Computational Physics

Fredrik Østrem (`fredost`)
Joseph Knutson (`josephkn`)

October 2, 2016

Contents

1 Introduction

In this project we are going to solve Schroedinger's equation for two electrons in a three-dimensional harmonic oscillator well. This will be done with and without a repulsive Coulomb potential while we assume spherical symmetry. Our method consists of the Jacobi method where we take advantage og similarity to solve our problem.

Here we present the solution of the radial part of Schroedinger's equation for one electron

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

In our case $V(r)$ is the harmonic oscillator potential $(1/2)kr^2$ with $k=m\omega^2$.

We substitute $R(r)=(1/r)u(r)$, set $l=0$ and obtain

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

Now we introduce a dimensionless variable $\rho=(1/\alpha)r$ where α is a constant with dimension length. We do this to remove uneccesary factors later. We also set $V(\rho)$ equal to the HO potential$(1/2)k\alpha^2\rho^2$ and rewrite our equation:

$$-\frac{d^2}{d\rho^2}u(\rho)+\frac{mk}{\hbar^2}\alpha^4\rho^2u(\rho)=\frac{2m\alpha^2}{\hbar^2}Eu(\rho).$$

The constant α can now be fixed so that

$$\alpha=\left(\frac{\hbar^2}{mk}\right)^{1/4}.$$

Defining

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

we can rewrite Schroedinger's equation as

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

2 Methods

2.1 Preservation of orthogonality and dot product in a unitary matrix

Assume that $\mathbf{U}$ is an orthogonal transformation. Then by definition, $\mathbf{U}^T\mathbf{U}=\mathbf{I}$, so for any vectors $\mathbf{u}$ and $\mathbf{v}$ we see that

$$(\mathbf{Uu})\cdot(\mathbf{Uv})=(\mathbf{Uu})^T(\mathbf{Uv})=\mathbf{u}^T\underbrace{\mathbf{U}^T\mathbf{U}}_{=\mathbf{I}}\mathbf{v}=\mathbf{u}^T\mathbf{v}=\mathbf{u}\cdot\mathbf{v}$$

Since the dot product of vectors are preserved by $\mathbf{U}$, orthogonality (that is, the dot product being zero) is also preserved.

2.2 Non-interacting case

We have the differential equation

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

where $V(\rho) = \rho^2$, that we want to solve. By using Taylor expansion of $u(\rho)$ and discretizing u and ρ with step length h, we get the equation

$$-\frac{1}{h^2}u_{i-1} + \left(\frac{2}{h^2} - V_i\right)u_i - \frac{1}{h^2}u_{i+1} = \lambda u_i$$

We can write this as a matrix eigenvector equation, $\mathbf{Au} = \lambda\mathbf{u}$, where

$$\mathbf{A} = \begin{bmatrix} d_0 & e_0 & 0 & 0 & \dots & 0 & 0 \\ e_1 & d_1 & e_1 & 0 & \dots & 0 & 0 \\ 0 & e_2 & d_2 & e_2 & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & \dots & \dots & \dots e_{N-2} & d_{N-2} & e_{N-2} \\ 0 & \dots & \dots & \dots & \dots & e_{N-1} & d_{N-1} \end{bmatrix}, \quad d_i = \frac{2}{h^2} + V_i, \quad e_i = -\frac{1}{h^2}$$

We can solve this equation by finding an orthonormal matrix $\mathbf{S}$ and a diagonal matrix $\mathbf{B}$ such that

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

where the diagonal elements of B, b_{ii}, are the eigenvalues λ_i of A, and the row vectors $\mathbf{s}_i$ of $\mathbf{S}$ are the eigenvectors of $\mathbf{A}$. We can see this since, if $\mathbf{y} = \mathbf{e}_i$, then $\mathbf{y}$ is an eigenvector of $\mathbf{B}$ with eigenvalue λ_i; if then $\mathbf{x} = \mathbf{S}^T\mathbf{y}$, then

$$\mathbf{Ax} = \mathbf{S}^T\mathbf{BSx} = \mathbf{S}^T\mathbf{BSS}^T\mathbf{y} = \mathbf{S}^T\mathbf{By} = \mathbf{S}^T\lambda_i\mathbf{y} = \lambda_i\,\mathbf{S}^T\mathbf{y} = \lambda_i\mathbf{x}$$

where $\mathbf{x} = \mathbf{S}^T\mathbf{e}_i$ is the i'th column of $\mathbf{S}^T$, and thus also the i'th row of $\mathbf{S}$.

We'll be using the Jacobi method, in which we in multiple steps will remove the off-diagonal elements of $\mathbf{A}$ with orthonormal transformations, until we have a matrix where the off-diagonal elements are sufficiently small. Start with $\mathbf{A}_0 = \mathbf{A}$ and $\mathbf{P}_0 = \mathbf{I}$.

For each step with matrices $\mathbf{A}_n$ and $\mathbf{P}_n$, we find the element a_{kl} that is the largest off-diagonal element in terms of absolute value. We define

$$\tau = \cot 2\theta = \frac{a_{ll} - a_{kk}}{2a_{kl}}$$

and the orthonormal similarity transformation

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

where $(\mathbf{S}_n)_{kk} = (\mathbf{S}_n)_{ll} = \cos\theta$ and $(\mathbf{S}_n)_{kl} = -(\mathbf{S}_n)_{lk} = \sin\theta$.

We then find $\mathbf{A}_{n+1} = \mathbf{S}_n^T \mathbf{A}_n \mathbf{S}_n$ and $\mathbf{P}_{n+1} = \mathbf{S}_n^T \mathbf{P}_n$, and use these for the next step.

Once the off-diagonal elements elements of $\mathbf{A}_n$ are sufficiently small, the diagonal elements of $\mathbf{A}_n$ will contain the eigenvalues of $\mathbf{A}$, and the columns vectors of $\mathbf{P}_n$ will contain the corresponding eigenvectors.

2.3 Unit Testing

We have chosen to implement two tests. Both of us have made a test for checking the preservation of orthogonality. The second test we implemented checks for anomalies in our function which finds the biggest off-diagonal element.

2.3.1 Orthogonality Test

To check if the orthogonality of the vectors are preserved, simply produce the dot product between them. The dot product between two orthonormal vector is always zero:

$$\mathbf{V}_{\lambda 1} \cdot \mathbf{V}_{\lambda 2} = 0$$

On a computer, we rarely end up with zero due to misrepresentation of various numbers. We therefore calculate the dot product and ask if the product is smaller than a tolerance ε:

$$|\mathbf{V}_{\lambda 1} \cdot \mathbf{V}_{\lambda 2}| < \varepsilon$$

If not, we print that the test has failed and maybe stop the program.

We have N eigenvectors when solving our problem for an $N \times N$ matrix. I other words, there are a lot of possible dot products, (N^2). We have each chosen different ways of testing orthogonality between the vectors. Fredrik's unit test produces all N^2 dot products and finds the dot product with the biggest error. This biggest error is then compared with a tolerance.

Joseph's unit test is run every now and then during the similarity transformations. It takes two random column vectors of the unitary matrix S and checks if their dot product is smaller than the tolerance.

2.3.2 Max Off-Diagonal Element Test

Our unit tests regarding the largest off-diagonal value is simpler. We hardcode a symmetrical 5×5 symmetric matrix, as suggested. We then know the maximal off-diagonal value.

To make sure the `Maxoff` function works, we use it to find the max value of the matrix. The value returned should be the biggest value we put into it. Our `if` expression will subtract the returned maximum value from the one we known maximum value of the matrix to check if the answer is within the tolerance 10^{-13} of 0.

2.4 Interacting case

In the interacting case, we had the differential equation

$$-\frac{\partial^2}{\partial\rho^2}\psi(\rho) + \left(\omega_r^2\rho 2 + 1/\rho\right)\psi(\rho) = \lambda\psi(\rho) \tag{1}$$

We used the same code as before, but changed the potential function to $V(\rho) = \omega_r^2\rho^2 + 1/\rho$, and used different parameters $\rho_{\max} = 60$, $N = 800$ and $\omega_r = 0.01, 0.5, 1, 5$.

We already know that the solutions $\mathbf{v}$ that we get from the eigenvector solvers are normalized in the sense that

$$\mathbf{v}\cdot\mathbf{v} = \sum_{n=0}^{N} u_i^2 = 1$$

However, we want a solution $\mathbf{v} = c\,\mathbf{u}$ that is normalized in the sense that

$$\mathbf{u}\cdot\mathbf{u} = \sum_{n=0}^{N} hv_i^2 = \sum_{n=0}^{N} hc^2u_i^2 = 1$$

We therefore see that c must be $c = 1/\sqrt{h}$, so we just multiply our solution by this to get the correct normalized quantum state solution.

3 Results

3.1 Eigenvalue solving algorthim

We have implemented two different programs to solve eigenvector systems with symmetric matrices. For parameters we set $N = 5, 10, 15, \ldots, 100$, $\rho_{\max} = 6$ When we run the programs on the non-interacting case, the number of similarity transforms K as a function of the matrix dimensionality N is shown in figure 1. (Note that the two programs use exactly the same number of steps, so there's only one plot.) We have also plotted the best fit curve on the form ax^2, which clearly show $K \propto N^2$.

When we compare the resulting eigenvalue and eigenvectors, we see that the absolute difference between any of the eigenvalues or eigenvectors is at most 1.2×10^{-6}.

In figure 2, we see the time the two algorithm use, and comparing it to the time used by Armadillo's `eig_sym` method. We see that our algorithms both run in less than half a second for $N = 100$, with one using about half the time of the other, but neither can compare to `eig_sym`, which completes in less than 3 ms for the same value of N. In addition, we see that the *unoptimized* version of `code-fredrik/b.cc` (compiled with `-O0` instead of `-Ofast`) is about 6 times slower than the optimized version of the same program, which means that if we want to run our code as fast as possible, we gotta `-Ofast`. 

3.2 Interacting and non-interacting case

In the interacting case, we calculate the wavefunction solutions as described in section 2.4, and we plot the lowest energy eigenvector solutions (normalized) in figure 3 for ω_r values

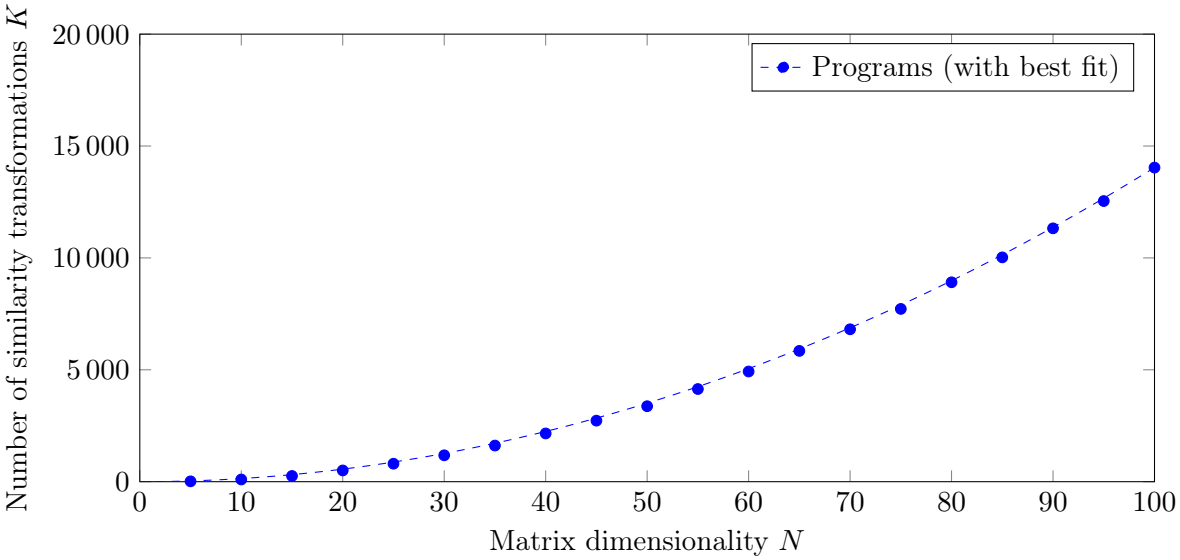

Figure 1: Number of similarity transforms before maximal off-diagonal element is less than 1×10^{-10}. Same for both programs.

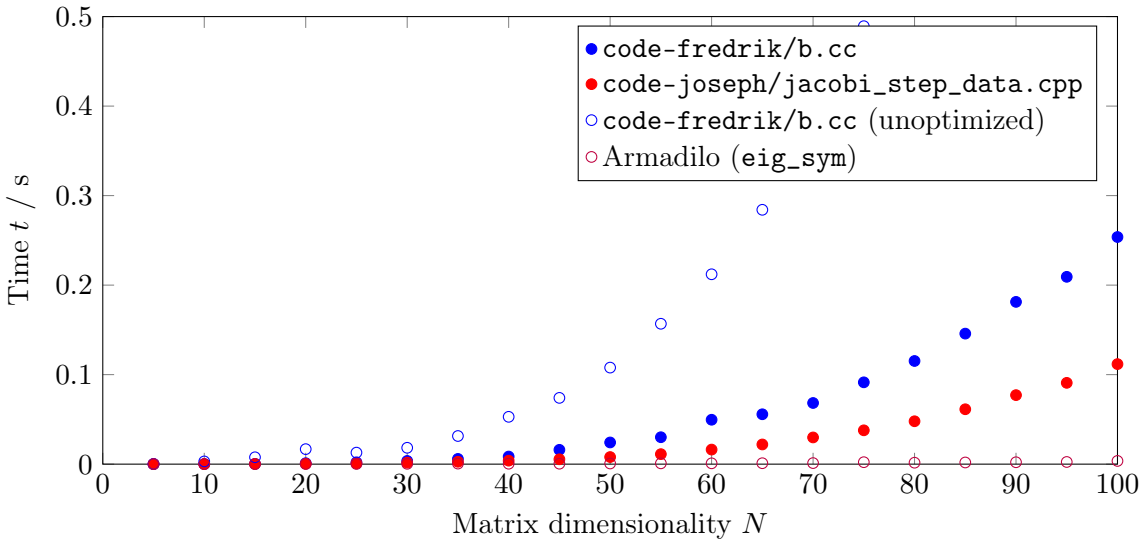

Figure 2: Total computation time as a function of matrix dimensionality N for both programs, compared with Armadillo's `eig_sym` function and an unoptimized version of `code-fredrik/b.cc`.

0.01, 0.5, 1 and 5. In the same figure we also have the plot for the case where there is no repulsion between the electrons.

We see that the shape of the wavefunctions are approximately the same, regardless of ω_r, except that larger values of ω_r give a narrower curve. This means that the electrons are more likely to be found closer to each other for higher ω_r values. We also see that the curve for the non-interacting case closely resembles the curve for $\omega_r = 1$, which makes sense since the non-interacting potential is $V(\rho) = \rho^2$, while in the case $\omega_r = 1$ it is $V(\rho) = \rho^2 + 1/\rho$.

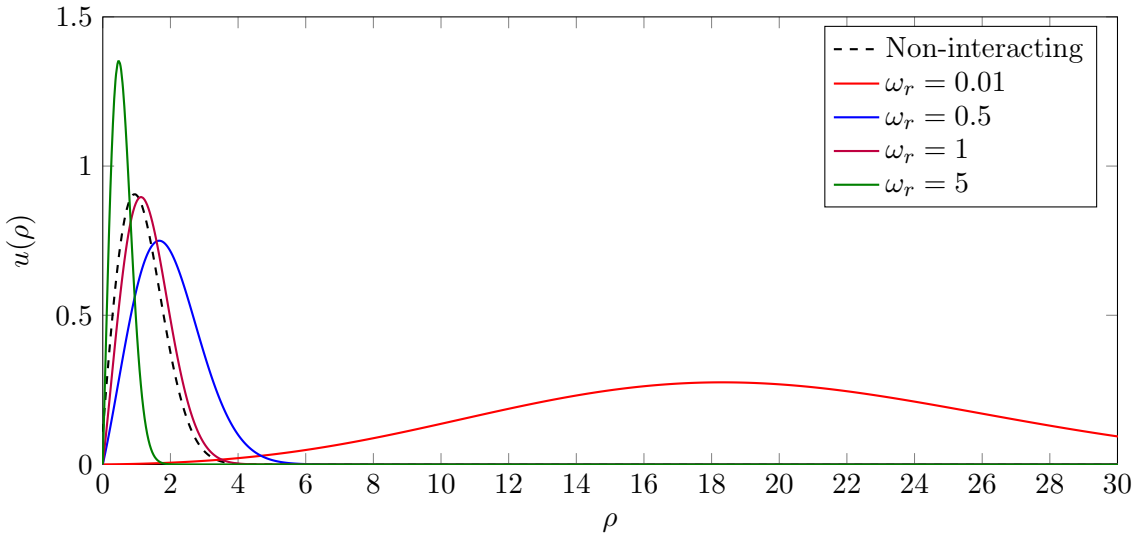

Figure 3: Normalized numerical solutions of (1) for different values of ω_r, with $N = 800$ and $\rho_{\max} = 60$.

4 Conclusions

We have looked at the wavefunctions of two electrons in interacting and non-interacting systems, and by using Poisson's equation we have turned the Schrödinger equation into an eigenvalue problem, and solved that numerically using our own implementations of the Jacobi method.

The implementations of the algorithms we used are sufficiently fast to handle cases up to at least $N = 100$ within reasonabe time, but are much slower than the native methods provided by linear algebra libraries such as Armadillo.

We saw that the ground state wavefunction solutions to the eigenvalue equation in the interacting case was affected by the "frequency" of the system $\omega_r = \frac{1}{2}\sqrt{mk}\alpha^2/\hbar$, and that this in turn made interacting electron pairs with a higher frequency tend to stay closer to each other. We also saw that in the case where the electrons didn't interact, the wavefunction was very similar to that of the interacting electrons with $\omega_r = 1$.

A Appendix

All files used in this project can be found at `https://github.com/frxstrem/fys3150/tree/master/project2`. The following code files are used:

b)
- `code-fredrik/b.cc`
  - *Implements method for solving eigenvector equations with symmetric matrices, and specifically solving the case where there is no Coulomb interaction between the electrons. Saves the data about number of steps, time per step and absolute error compared to Armadillo's `eig_sym` to file b.dat.*
- `code-fredrik/comp_eig.hh`
  - *Contains function `comp_eig` used to compare two solutions to an eigenvector problem, and returns the maximal absolute difference between either the eigenvalues or eigenvectors.*
- `code-joseph/jacobi.cpp`
  - *Solves the eigenvalue problem for two non interacting electrons.*
- `code-joseph/jacobi_step_data.cpp`
  - *Same as jacobi.cpp, but writes the eigenvalues and vectors to file.*

c)
- `code-fredrik/b.test.cc`
  - *Contains the unit tests for `code-fredrik/b.cc`.*

d,e)
- `code-fredrik/d.cc`
  - *Same as `code-fredrik/b.cc`, except also handles the case where there is Coulomb interaction between the electrons. Saves the plots for the ground wavefunction solutions to files `d-*.dat`.*
- `code-joseph/jacobi_interact.cpp`
  - *This one contains my unit tests at the bottom. It also solves the eigenvalue problem fors the two electrons, but this time with a potential that assumes Coulomb interaction. Writes the eigenvalues and vectors to file.*
- `code-joseph/jacobi.py`
  - *Gathers data from the interacting electrons solution (`jacobi_interact.cpp`) in order to plot the wavefunctions of the ground state.*

B Bibliography

[1] Computational Physics I FYS3150. *Project 2*. 2016. URL: https://github.com/CompPhysics/ComputationalPhysics/blob/master/doc/Projects/2016/Project2/project2_2016.pdf.

[2] Conrad Sanderson and Ryan Curtin. "Armadillo: a template-based C++ library for linear algebra." In: *Journal of Open Source Software* vol. 1 (2016), p. 26.