UNIVERSITY OF OSLO

FYS3150

COMPUTATIONAL PHYSICS

Eigensolvers A Numerical Study of Eigenvalue Problems

Authors
LARS KRISTIAN SKAAR
BRAGE BREVIG
https://github.uio.no/lkskaar/FYS3150

September 30, 2020

1 Abstract

In this numerical project, we study the Jacobi Rotation Method as an eigenvalue solver. We then devise this solver to study different physical systems. We find that the Jacobi Rotation Method is an inefficient, yet precise eigenvalue solver wherein we have found that the number of iterations for a given dimension n follows $\mathcal{O}(1.8n^2)$ and uses no less than 21 minutes to solve a 1000 x 1000 system. We also find that in quantum harmonic oscillator systems, introducing two electrons causes a large radial probability distribution (the average position of the two is hard to pin down), whilst for such a system larger frequencies result in greater localization.

Contents

2 Introduction

In this numerical study, we tackle eigenvalue problems on the form $A\mathbf{x} = \lambda\mathbf{x}$ by devising the Jacobi Rotation algorithm. Eigenvalue problems frequently occur in the natural sciences and engineering. One might be dealing with the phenomenon of resonance when designing a bridge, or studying n-body interaction problems in the field of quantum mechanics. They are nevertheless an inherent part of the mathematical foundation on which interesting natural phenomena take place. For large systems of equations, or greatly involved eigenvalue problems, there often does not exist analytical solutions. The development of efficient and precise numerical eigenvalue solvers is therefore of great importance and interest to the scientific community. Going forward, we shall present some eigenvalue problems from classical - and quantum mechanics which we wish to solve by the Jacobi Rotation algorithm - an eigenvalue algorithm. We will then elaborate on the implementation of the algorithm itself, before presenting the results and findings in this study. Lastly, we shall discuss these findings in light of their own respect - the results from studying the algorithm, and the physical interpretation of the results obtained through solving different eigenvalue problems.

3 Theory & Background

3.1 The Buckling Beam

As this study deals with eigenvalue problems, let us first illustrate what an eigenvalue problem is by considering a well established example from classical mechanics; the buckling beam problem. Imagine a rigid beam of length L lying along the x-axis, being fixed in its endpoints $x = 0$ and $x = L$. Now, let F be a force acting on the beam at the endpoint $x = L$ toward the origin. The vertical displacement of the beam (i.e. the buckling), $u(x)$, may then be modelled by

$$\begin{cases} \gamma u''(x) = -Fu(x), \; x \in [0, L] \\ u(0) = u(L) = 0 \end{cases} \tag{1}$$

where γ is a constant containing information about the mechanical properties of the beam. Now, defining the dimensionless parameter $\rho = x/L$ and letting $v(\rho) := u(\rho L)$, we may reconstruct the problem such that it is equipped with Dirichlet boundary conditions. As $v''(\rho) = L^2 u''(\rho L)$, we arrive at

$$\begin{cases} -v''(\rho) = \lambda v(\rho), \; \rho \in [0, 1] \\ v(0) = v(1) = 0 \end{cases} \tag{2}$$

with

$$\lambda := \frac{FL^2}{\gamma}$$

The fact that equation (2) is an eigenvalue problem is more easily realized in light of a discrete numerical scheme. Let us first define a mesh grid of n

points such that the value of ρ at each instance i is

$$\rho_i = \rho_0 + ih, \text{ for } i = 0, 1, 2, ..., n$$

with a step size h

$$h = \frac{\rho_n - \rho_0}{n}$$

for which in our case, with endpoints $\rho_0 = 0$ and $\rho_n = 1$, giving $h = 1/n$. We may now define our numerical solution $v_i := v(\rho_i)$ such that

$$\begin{cases} -(1/h^2)(v_{i+1} - 2v_i + v_{i-1}) = \lambda v_i \\ v_0 = v_n = 0 \end{cases} \tag{3}$$

which may be written on the much more compact matrix form

$$\begin{bmatrix} \mathrm{d} & \mathrm{e} & \dots & 0 \\ \mathrm{e} & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \mathrm{e} \\ 0 & \dots & \mathrm{e} & \mathrm{d} \end{bmatrix} \begin{bmatrix} v_1 \\ \vdots \\ \vdots \\ v_{n-1} \end{bmatrix} = \lambda \begin{bmatrix} v_1 \\ \vdots \\ \vdots \\ v_{n-1} \end{bmatrix} \tag{4}$$

with $\mathrm{d} = 2/h^2$ being the main diagonal elements and $\mathrm{e} = -1/h^2$ being the sub - and super diagonal elements. Observe that we exclude the solution vector elements v_0 and v_n which correspond to the value of the solution at the endpoints of our grid.

3.2 A Numerical Scheme for a Three - Dimensional Schrödinger Equation

Although the overarching goal of this numerical project is to study and develop eigensolver algorithms, we shall also pay attention to the possible extensions of our findings. In this respect, we will now move on from studying the "buckling beam" problem, and see how we may change our already established numerical scheme to study quantum mechanical systems. *N*-body problems in quantum mechanics is currently a research area of great interest, and luckily for us, it calls for some clever algorithm architecture.

We ease into this section of the project by considering the Schrödinger equation for a single particle in a three - dimensional harmonic oscillator. The time independent three - dimensional Schrödinger equation reads

$$-\frac{\hbar^2}{2m}\nabla^2\psi(\mathbf{r}) + V(\mathbf{r})\psi(\mathbf{r}) = E\psi(\mathbf{r}) \tag{5}$$

where $\psi(\mathbf{r})$ describes the particle's motion, in part. We shall see later on that we are really only interested in the absolute square of this function, $|\psi(\mathbf{r})|^2$, as this describes the probability distribution of the quantum system. However, let us for now consider in detail the underlying assumptions we make in this case and how they play a role in the construction of the final equation we wish to solve. First and foremost, we assume that the potential is centro - or spherically symmetric which reduces the degrees of freedom in the system to only one; $r \in [0, \infty]$. Also, we know the potential $V(r) = (1/2)m\omega^2 r^2$ as we are dealing with a harmonic oscillator. Rewriting equation (5) to under these assumptions while transforming into spherical coordinates we arrive at

$$-\frac{\hbar^2}{2m}\left(\frac{1}{r^2}\frac{\partial}{\partial r}r^2\frac{\partial}{\partial r}-\frac{\ell(\ell+1)}{r^2}\right)R(r)+\frac{1}{2}m\omega^2r^2R(r)=ER(r) \tag{6}$$

where ℓ denotes the orbital quantum number, which in our case we shall set to zero $\ell = 0$. Devising some clever mathematics, we may scale this equation[1] such that

$$\begin{cases} -z''(\rho)+\rho^2 z(\rho)=\lambda z(\rho),\ \rho\in[0,\infty] \\ z(0)=z(\infty)=0 \end{cases} \tag{7}$$

Equation (7) clearly resembles our initial problem $-v''(x) = \lambda v(x)$, and it comes as no surprise that the numerical schemes are remarkably similar, too. We define a numerical solution $\zeta_i := z(\rho_i)$ with $\rho_i = \rho_0 + ih$ where ρ_0 and h carry the same numerical meaning from the previous scheme. A discrete scheme for the eigenvalue problem (7) therefore takes the form

$$\begin{cases} -(1/h^2)\left(\zeta_{i+1}-2\zeta_i+\zeta_{i-1}\right)+\rho_i^2\zeta_i=\lambda\zeta_i \\ \zeta_0=\zeta_n=0 \end{cases} \tag{8}$$

or on the more compact matrix form

$$\begin{bmatrix} \mathrm{d}' & \mathrm{e}' & \dots & 0 \\ \mathrm{e}' & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \mathrm{e}' \\ 0 & \dots & \mathrm{e}' & \mathrm{d}' \end{bmatrix} \begin{bmatrix} \zeta_1 \\ \vdots \\ \vdots \\ \zeta_{n-1} \end{bmatrix} = \lambda \begin{bmatrix} \zeta_1 \\ \vdots \\ \vdots \\ \zeta_{n-1} \end{bmatrix} \tag{9}$$

now with new, non-constant elements $\mathrm{d}'_i = 2/h^2 + \rho_i^2$ on the main diagonal

[1]For more on this, consult this page.

and off-diagonal elements $e' = -1/h^2$. Here we exclude the solution vector components ζ_0 and ζ_n as before.

3.3 The Two Electron Problem

Going forward, we shall see how we may study the case where two interacting electrons are confined in a three-dimensional harmonic oscillator. To avoid cluttering this section of the article with too much theory, we shall rather focus on the resulting radial equation[2] describing the two-electron system and its numerical scheme. The radial equation for two electrons in a harmonic oscillator may be cast into a dimensionless equation

$$\begin{cases} -\psi''(\rho) + \omega^2\rho^2\psi(\rho) + (1/\rho)\psi(\rho) = \lambda\psi(\rho), \;\; \rho \in [0, \infty] \\ \psi(0) = \psi(\infty) = 0 \end{cases} \tag{10}$$

where ω represents the oscillator frequency. As before, we define our numerical solution to equation (10) $\varphi_i = \psi(\rho_i)$ such that

$$\begin{cases} -(1/h^2)\left(\varphi_{i+1} - 2\varphi_i + \varphi_{i-1}\right) + \omega^2\rho_i^2\varphi_i + (1/\rho_i)\varphi_i = \lambda\varphi_i \\ \varphi_0 = \varphi_n = 0 \end{cases} \tag{11}$$

At this point, we should recognize that we are yet again just adding linear terms to the differential equation, which transform the diagonal elements

[2]For more on this, consult this page.

of the matrix A when solving the system as a matrix equation

$$\begin{bmatrix} \mathrm{d}^\star & \mathrm{e}^\star & \dots & 0 \\ \mathrm{e}^\star & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \mathrm{e}^\star \\ 0 & \dots & \mathrm{e}^\star & \mathrm{d}^\star \end{bmatrix} \begin{bmatrix} \varphi_1 \\ \vdots \\ \vdots \\ \varphi_{n-1} \end{bmatrix} = \lambda \begin{bmatrix} \varphi_1 \\ \vdots \\ \vdots \\ \varphi_{n-1} \end{bmatrix}$$

where $\mathrm{d}^\star = 2/h^2 + \omega^2\rho_i^2 + 1/\rho_i$ and $\mathrm{e}^\star = -1/h^2$.

3.4 Preservation of orthogonality

Jacobi's Rotation Method (described below) is a way of diagonalizing matrices by unitary transformations. As we are dealing with eigenvalue problems, it is only natural to require that such a transform preserves the orthogonality of the computed eigenvectors. Luckily, it does. To see this, consider a set of basis vectors in $\mathbb{R}^n$

$$\mathbf{v}_i = \begin{bmatrix} v_i^{(1)} \\ v_i^{(2)} \\ \vdots \\ v_i^{(n)} \end{bmatrix}$$

where the superscripts (k) represent the k-th component of the i-th basis vector. We assume that these basis vectors are orthogonal such that

$$\mathbf{v}_i^T \mathbf{v}_j = \delta_{ij}$$

where δ_{ij} is the Krönecker - delta

$$\delta_{ij} = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases}$$

Consider then a matrix operator U which is unitary such that $U^T U = I$ where I is the identity matrix. Now, let $\mathbf{w}_i = U\mathbf{v}_i$ represent the set of transformed basis vectors. For these to be orthogonal, we must require that $\mathbf{w}_i^T \mathbf{w}_j = \delta_{ij}$, too. Computing this dot product leads to

$$\mathbf{w}_i^T \mathbf{w}_j = (U\mathbf{v}_i)^T U\mathbf{v}_j = \mathbf{v}_i^T U^T U \mathbf{v}_j$$

$$\mathbf{w}_i^T \mathbf{w}_j = \mathbf{v}_i^T \mathbf{v}_j$$

$$\rightarrow \mathbf{w}_i^T \mathbf{w}_j = \delta_{ij}$$

showing that the basis $\mathbf{w}_i$ is orthogonal, too.

4 Numerical Methods and Implementation

4.1 Jacobi's Rotation Method

In this project, we are concerned with solving eigenvalue problems, and to do this we devise and study the Jacobi Rotation Method. To better understand inner mechanics of this rather subtle method, consider first a simple matrix $A \in \mathbb{R}^{2x2}$

$$A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}$$

Assume also that this matrix is symmetric. Consider now a 2x2 rotation matrix $R \in \mathbb{R}^{2x2}$

$$R = \begin{bmatrix} c & -s \\ s & c \end{bmatrix}$$

where $c = \cos\vartheta$ and $s = \sin\vartheta$. Jacobi's Rotation Algorithm calls for a similarity transformation of A such that $A' = RAR^T$ where A' is symmetric and similar to A. As the similarity transformation computes to

$$RAR^T = \begin{bmatrix} c^2a_{11} - cs(a_{12} + a_{21}) + s^2a_{22} & c^2a_{12} + cs(a_{11} - a_{22}) - s^2a_{21} \\ c^2a_{21} + cs(a_{11} - a_{22}) - s^2a_{12} & c^2a_{22} + cs(a_{12} + a_{21}) + s^2a_{11} \end{bmatrix}$$

or more compactly, assuming still that the initial matrix A is symmetric

$$A' = \begin{bmatrix} c^2a_{11} - 2csa_{12} + s^2a_{22} & (c^2 - s^2)a_{12} + cs(a_{11} - a_{22}) \\ (c^2 - s^2)a_{12} + cs(a_{11} - a_{22}) & c^2a_{22} + 2csa_{12} + s^2a_{11} \end{bmatrix}$$

The grand idea behind Jacobi's Rotation algorithm is to require that all off-diagonal elements are transformed to zero. We should first observe that from A' we may derive the definition

$$a_{12}\cos^2\vartheta + \sin\vartheta\cos\vartheta(a_{11} - a_{22}) := 0 \tag{12}$$

Using trigonometric identities, we arrive at a more compact definition

$$\tau := \cot 2\vartheta := \frac{a_{11} - a_{22}}{2a_{12}}$$

Defining also $t := \tan\vartheta$ and $c := 1/\sqrt{1+t^2}$ we arrive at a much more elegant form of equation (12)

$$t^2 + 2\tau t - 1 := 0 \tag{13}$$

which may be easily implemented on a computer to solve for the angles ϑ which transform the off-diagonal elements to zero. The algorithm is generalized to n-dimensional systems such as the ones introduced in the Theory section by introducing the Givens rotation matrix

$$G(i,j,\vartheta) = \begin{bmatrix} 1 & \dots & 0 & \dots & 0 & \dots & 0 \\ \vdots & \ddots & \vdots & & \vdots & & \vdots \\ 0 & \dots & c & \dots & -s & \dots & 0 \\ \vdots & & \vdots & \ddots & \vdots & & \vdots \\ 0 & \dots & s & \dots & c & \dots & 0 \\ \vdots & & \vdots & & \vdots & \ddots & \vdots \\ 0 & \dots & 0 & \dots & 0 & \dots & 1 \end{bmatrix}$$

where $c = \cos\vartheta$ and $s = \sin\vartheta$ as before. The similarity transformation $A' = GAG^T$ gives rise to a system of matrix entries

$$\begin{cases} a'_{ii} = c^2 a_{ii} - 2cs a_{ij} + s^2 a_{ii} \\ a'_{jj} = s^2 a_{ii} + 2cs a_{ij} + c^2 a_{jj} \\ a'_{ij} = a'_{ji} = \left(c^2 - s^2\right) a_{ij} + cs\left(a_{ii} + a_{jj}\right) \\ a'_{iq} = a'_{qi} = c a_{iq} - s a_{jq} \quad q \neq i,j \\ a'_{jq} = a'_{qj} = s a_{iq} + c a_{jq} \quad q \neq i,j \\ a'_{qp} = a_{qp} \quad q,p \neq i,j \end{cases} \tag{14}$$

The general algorithm performs a sweep where it finds and targets the largest off-diagonal entry[3], 'rotates' this entry by solving equation (13) such that it becomes sufficiently small and updates all entries by their definitions in system (14). Notice, however, that this could cause entries of value zero to transform into new, non-zero valued entries. This implication forces the algorithm to perform sweeps and similarity transformations iteratively until all off-diagonal entries are transformed into some value which is smaller than a given tolerance, rendering it highly inefficient.[4] In summary, one may consider the algorithm to work on the basis of localizing, transforming and repeating.

4.2 Implementation and Structure

As we now have discussed the mathematics and idea behind the algorithm we are devising to solve eigenvalue problems, we find it will be useful to explain some of the thought processes behind the implementation and structure of the algorithm itself. The aforementioned first step in the algorithm is to target the maximum off-diagonal matrix entry. This is very easily implemented as shown below. Take note of that the following code snippets are not written in actual C++ syntax, they are all illustrative. Follow the link to our GITHUB repository on the first page of this article to view all working programs.

[3]This is implemented via code in a stand-alone function `maxoffdiag()`

[4]More on this in the Results and Discussion sections.

```
        double maximum_offdiag = 0.0

        for (int i = 0; i < n; i++)
                for (int j = i + 1; j < n; j++)
                        double aij = fabs(A(i,j))
                        if (aij > maximum_offdiag)
                                maximum_offdiag = aij
```

Here A is the matrix(ces) we wish to diagonalize. This implementation assumes that the input matrices are symmetric, as one may observe in the second `for` - loop where the column index $j = i + 1$ forces the program to only sweep through the upper triangular entries in the matrix.

Going forward, the next step is to rotate `maximum_offdiag` so that it is transformed approximately to zero using the function `jacobi_rotate()`. Refer to our repository to view the code for this particular function. The last step is to repeat this process until all off-diagonal elements are approximately zero, implemented by the function `jacobi_method` as

```
                double max_itr = n * n * n;
                double tol = 1e-8;
                int iterations = 0;
                double max = maxoffdiag();
                clock_t start, finish;
                start = clock();

                while (max > tol && iterations <= max_itr) {
                        max = maxoffdiag();
                        jacobi_rotation();
                        iterations++;
                }
                finish = clock();
```

After calling `jacobi_method` the matrix A is approximately diagonalized. All the while we want to compute the eigenvalues, it is also of interest to compute the eigenvectors associated to the spectral decomposition of A given by $A = R\Lambda R^T$ with Λ as the diagonal matrix containing the eigenvalues of A on its main diagonal. To find the eigenvector matrix R we first declare an initial matrix $R^{(1)}$ as an identity matrix in which we store the computed eigenvectors, and then go on to perform transformations on R in pace with the transformations of A itself, such that R approximates an analytical matrix $\mathcal{R}$ containing analytical solutions.

4.3 Unit Tests

To ensure that the program works well during its design, we have implemented a few benchmark tests for each unit of the program, so - called unit tests. These are meant to give us, the designers, or any users insight into what may have caused errors, may they occur. They are more importantly there to ensure that, if the program is executable, the output makes mathematical and physical sense. Examples of these test are one which makes sure that the input matrix is symmetric (it must be for the algorithm to work), one that tests the precision of the computed eigenvalues and lastly one which tests the orthogonality of the computed eigenvectors. The last mentioned test is very important as we must make sure the program "carries" the mathematics of the initial problem with it.

5 Results

Table 1: Tabular overview of how many iterative sweeps and transformations that are performed for a given dimensionality n, and their associated elapsed times.

n	No. of iterations	Elapsed Execution Time [s]
5	32	$5.00 \cdot 10^{-6}$
10	138	$3.10 \cdot 10^{-5}$
50	4374	$8.94 \cdot 10^{-3}$
100	17602	$1.2 \cdot 10^{-1}$
200	70845	$1.83 \cdot 10^{0}$
400	286674	$2.88 \cdot 10^{1}$
800	1153443	$4.58 \cdot 10^{2}$
1000	1813201	$1.28 \cdot 10^{3}$

Table 1 gives a tabular overview of how many iterations the algorithm performs to solve the "buckling beam" problem for different system dimensions n. It also presents how much time has elapsed just calling the `jacobi_method()` function.

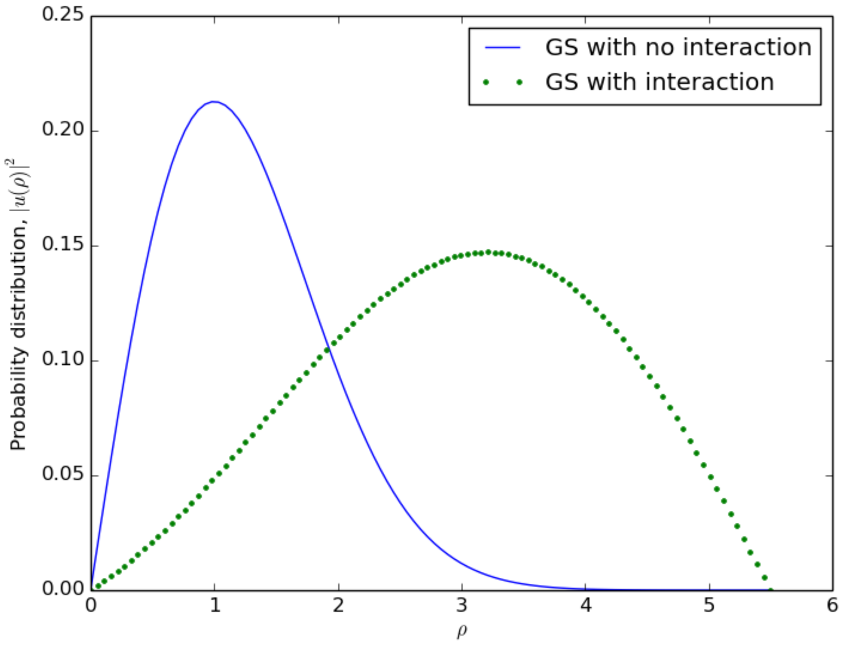

Figure 1: GS of the one-electron - and two-electron system.

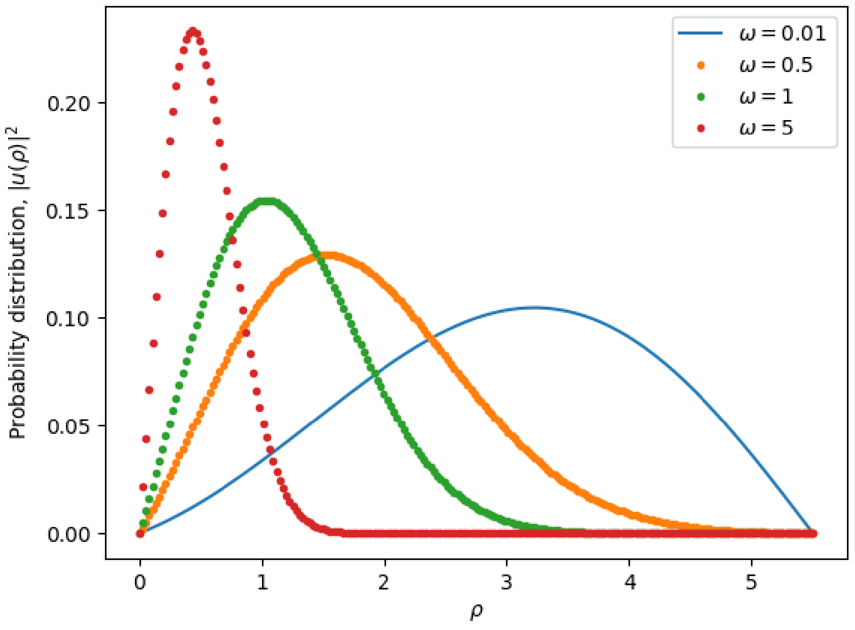

Figure 2: GS of the two-electron system for different frequencies ω.

6 Discussion

6.1 The Algorithm

From table 1 it becomes very clear that, as was mentioned in the section on implementation, that the Jacobi Rotation Method is highly inefficient. For only eight hundred points of integration the program has a run time of nearly eight minutes, and for a thousand points of integration the program uses an astounding 21 minutes to execute the method. One should also observe that the number of required iterations with respect to the given dimensionality n follows $\mathcal{O}(1.8n^2)$

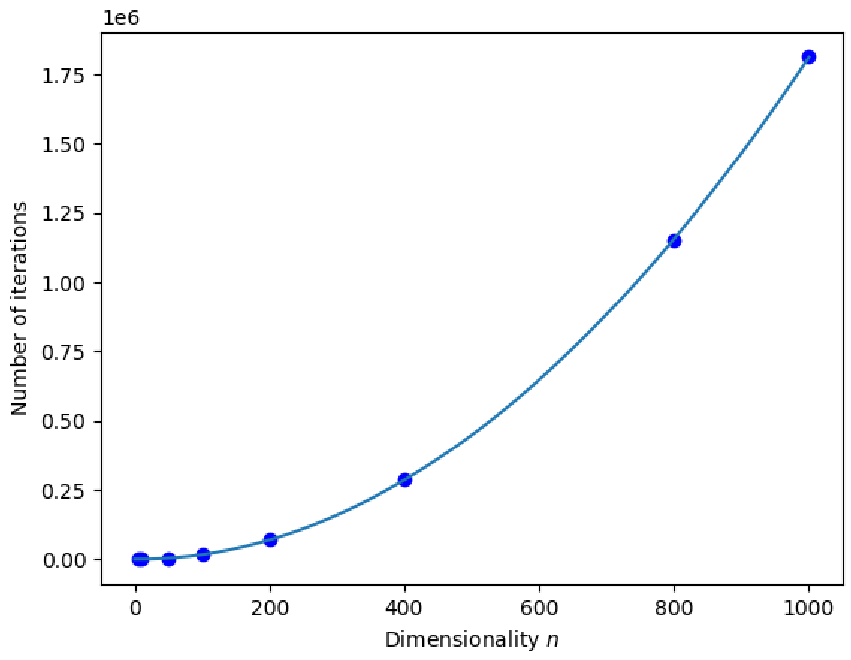

Figure 3: A plot showing the evolution of number of iterations for a given n.

Here we have devised a library `numpy.polyfit()` do determine the gradient of the fitted polynomial.

Although the program is numerically inefficient, it is nevertheless quite precise for low n, a conclusion we draw from having used the test function `TEST_eigval()` to ensure that our computed eigenvalues are comparable to a benchmark tester using Armadillo's `eig_sym()` with a given tolerance.

6.2 The Physical Systems

Figure 1 describes the physical state of a one-electron - and a two-electron system. Each plot describes the *radial probability distribution* of the systems. We see that, for one electron in a 3D harmonic oscillator, the distribution is much sharper than it is for two electrons, meaning that a single electron is more localized than a system consisting of two.

In figure 2 we see plots of the ground state in a 3D harmonic oscillator occupied by two electrons and how the radial probability distribution changes as the oscillator frequency increases. Higher frequencies localizes the electrons more as seen by the smaller distributions, whilst lower frequencies results in a much larger distribution. One should from this conclude that higher frequencies causes the two electrons to have more well defined positions.

7 Conclusive Remarks

In this study, we have found and presented that the Jacobi Rotation Method is a highly inefficient eigenvalue solver using a total number of iterations following $\mathcal{O}(1.8n^2)$ as a result of having to perform iterative transformations on off-diagonal elements to diagonalize a matrix within a given tolerance. Nevertheless, the algorithm finds tolerably precise eigenvalues for low dimensions n. We have also used the eigenvalue solver practically to solve the Schrödinger equation for one and two electrons in an harmonic oscillator well, and have made to findings about such systems; for one, introducing two electrons causes wider distributions in their radial probabilities, and secondly that for a two-electron system, increasing the frequency of the oscillator results in a more well defined average position.

8 References

Hjorth-Jensen, Morten. *Computational Physics: Lecture Notes Fall 2015*. PDF file. August 2015. `https://github.com/CompPhysics/ComputationalPhysics/blob/master/doc/Lectures/lectures2015.pdf`