Jacobi's method for solving eigenvalue problems

FYS4150

Written by:

Marius Enga, Patryk Krzyzaniak
Mohamed Ismail and Kristoffer Varslott

Department of Physics UiO

UiO : University of Oslo

October 1, 2019

Contents

I Introduction **3**

II Theory **3**
- i The buckling beam problem 3
- ii Quantum dots in three dimension 5

III Method **8**
- i Jacobi method 8

IV Implementation **10**
- i Overview: Implementation of code 10

V Numerical results **12**
- i Numerical analysis of the Buckling beam problem 12
- ii Numerical analysis of Quantum dots for one and two electrons 14

VI Discussion **16**

VIIConclusion **16**

Abstract

Eigenvalue problems in physics can often be discretized into a linear algebra problem which can be solved by diagonalizing a matrix. Using Jacobi's method to diagonalize, we are able to solve the buckling beam problem and the three-dimensional harmonic oscillator potential with both one and two electrons. We found that Jacobi's method is a brute force method that works good for solving these problems, but it will run slowly for high dimensionality matrices if a high precision in the eigenvalues are desired. Ground state eigenvalues and a couple of the first exited states however are quite accurate, even for low dimensions.

I. Introduction

Solving differential equations is a vital part of physics, where most of the problems encountered is somehow related to differential equations. This article takes a good look at how to solve differential equations by transforming them into eigenvalue problems.

In this project we will develop a code for solving eigenvalue problems. Starting of with a classical mechanics problem with a buckling beam case, we then move on to the quantum mechanical realm with a three-dimensional harmonic oscillator potential with both one and two electrons.

By utilizing our eigenvalue solver, we can solve all these problems by changing the diagonal elements of the tridiagonal Toeplitz matrix. In order to use the eigenvalue solver, the equations have to be scaled according to the different cases that will be studied upon.
Our eigenvalue solver will be implementing the Jacobi's rotation algorithm.

In all the cases we are studying there exists exact analytical solutions for the eigenvalues which we will be comparing our numerical results with. In addition, the implemented eigenvalue solver will be compared to a solver found in the Armadillo [1] [2] package with respect to the CPU time. The analytical solutions for the eigenvalues concerning the two electron scenario were obtained by M. Taut [3].

II. Theory

i. The buckling beam problem

We start of with following differential equation:

$$\gamma \frac{d^2u(x)}{dx^2} = -Fu(x),$$

This second order differential equation is called "The buckling beam problem", which is a classical wave function problem in one dimension. In the equation above $u(x)$ is the vertical displacement of the beam in the y-direction. The beam has length L, $x \in [0, L]$ and F is a force applied at $(L, 0)$ in the direction towards the origin. The parameter γ is a constant defined by properties like the rigidity of the beam. Applying Dirichlet boundary conditions, for then to set $u(0) = u(L) = 0$.

In this specific case two of the parameters γ, F and L are known. An assumption is made where F and L is known, thereafter by creating a dimensional variable $\rho = \frac{x}{L}$ and inserting it as a function of $u(x)$. The equation can be rewritten as:

$$\frac{d^2u(\rho)}{d\rho^2} = -\frac{FL^2}{\gamma}u(\rho) = -\lambda u(\rho) \tag{1}$$

Where the boundaries is as before: $\rho \in [0, 1]$. λ is then defined as $\lambda = FL^2/\gamma$. We need to discretize this equation to obtain the eigenvalues. Writing this second order equation to a discretized form yields:

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

Where the step-length h as been introduced and can is defined as:

$$h = \frac{\rho_N - \rho_0}{N}$$

where ρ_N and ρ_0 is 1 and 0 respectively. Then it is valid to define ρ_i as discretized:

$$\rho_i = \rho_0 + ih \qquad i = 1, 2, \ldots, N.$$

It is possible to rewrite equation 2 as a set of linear equation:

$$\begin{bmatrix} d & a & 0 & 0 & \ldots & 0 \\ a & d & a & 0 & \ldots & 0 \\ 0 & a & d & a & 0 & 0 \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ 0 & \ldots & \ldots & a & d & a \\ 0 & \ldots & \ldots & \ldots & a & d \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ \ldots \\ u_{N-2} \\ u_{N-1} \end{bmatrix} = \lambda \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ \ldots \\ u_{N-2} \\ u_{N-1} \end{bmatrix}$$

$$\mathbf{Au} = \lambda \mathbf{u} \tag{3}$$

The endpoints u_0 and u_N are excluded, and the diagonal is defined as $d = 2/h^2$, whereas the non-diagonal is $a = -1/h^2$

This eigenvalue problem has analytical eigenpairs, with eigenvalues given as

$$\lambda_j = d + 2a\cos\left(\frac{j\pi}{N+1}\right) \qquad j = 1, 2, \ldots N. \tag{4}$$

ii. Quantum dots in three dimension

Here we look at electrons moving in a three-dimensional harmonic oscillator potential, where they repel each other via Coulomb interaction. We assume that there is a spherical symmetry for simplifying our problem. We start of with radial part of the Schroedinger's equation for one electron:

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

Where $V(r)$ is the harmonic oscillator potential set as $(1/2)kr^2$ with $k=m\omega^2$ and E is the energy of the harmonic oscillator in three dimensions. The oscillator frequency is ω and the energies are

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

with $n=0,1,2,\dots$ and $l=0,1,2,\dots$.

Where n and l are the quantum numbers, giving us quantized energies. Since we have made a transformation to spherical coordinates it means that $r\in[0,\infty)$. The quantum number l is the orbital momentum of the electron. Then we substitute $R(r)=(1/r)\psi(r)$ and obtain

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

The boundary conditions are $u(0)=0$ and $u(\infty)=0$.
We introduce a dimensionless variable as we did for the buckling beam problem: $\rho=(1/\alpha)r$ where α is a constant with dimension length and get

$$-\frac{\hbar^2}{2m\alpha^2}\frac{d^2}{d\rho^2}\psi(\rho)+\left(V(\rho)+\frac{l(l+1)}{\rho^2}\frac{\hbar^2}{2m\alpha^2}\right)\psi(\rho)=E\psi(\rho).$$

Another simplification we will do is to set the azimuthal quantum number l = 0. Inserting $V(\rho)=(1/2)k\alpha^2\rho^2$ and setting l = 0 we get:

$$-\frac{\hbar^2}{2m\alpha^2}\frac{d^2}{d\rho^2}\psi(\rho)+\frac{k}{2}\alpha^2\rho^2\psi(\rho)=E\psi(\rho).$$

We multiply thereafter with $2m\alpha^2/\hbar^2$ on both sides we end up with:

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

We can fix the constant α so that $\frac{mk}{\hbar^2}\alpha^4 = 1$,. This gives us a definition for λ:

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

By implementing this into the Schroedinger's equation we get:

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

We will again want to convert this into a discretized form, we then end up:

$$-\frac{\psi_{i+1} - 2\psi_i + \psi_{i-1}}{h^2} + \rho_i^2\psi_i = \lambda\psi_i$$

Finally we convert this into a set of linear equation on matrix form, where it consist of eigenvectors with corresponding eigenvalues where the eigenvectors is operated on with a tridiagonal matrix:

$$\begin{bmatrix} d_1 & e_1 & 0 & 0 & \dots & 0 \\ e_1 & d_2 & e_2 & 0 & \dots & 0 \\ 0 & e_2 & d_3 & e_3 & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & \dots & e_{N-3} & d_{N-2} & e_{N-2} \\ 0 & \dots & \dots & \dots & e_{N-2} & d_{N-1} \end{bmatrix} \begin{bmatrix} \psi_1 \\ \psi_2 \\ \dots \\ \dots \\ \dots \\ \psi_{N-1} \end{bmatrix} = \lambda \begin{bmatrix} \psi_1 \\ \psi_2 \\ \dots \\ \dots \\ \dots \\ \psi_{N-1} \end{bmatrix}.$$

$$\boldsymbol{H}\boldsymbol{\psi} = \lambda\boldsymbol{\psi} \tag{5}$$

Here $d_i = \frac{2}{h^2} + V_i$, where we have defined $V_i = p_i^2$, and $e_i = -\frac{1}{h^2}$. We can see that this matrix equation only differ by the diagonal elements in comparison with the buckling beam problem. Many of our assumption has made the problem a lot simpler to handle numerically and analytically.

We define minimum and maximum values for the variable ρ, $\rho_{\min} = 0$ and $\rho_{\max}$, respectively. Results of the energies have to be checked for different values of $\rho_{\max}$, since it cannot set $\rho_{\max} = \infty$.

With a given number of mesh points, N, we define the step length h as, with $\rho_{\min} = \rho_0$ and $\rho_{\max} = \rho_N$,

$$h = \frac{\rho_N - \rho_0}{N}.$$

The value of ρ at a point i is then

$$\rho_i = \rho_0 + ih \qquad i = 1, 2, \ldots, N.$$

A similar scaling can be done in the two-electron case. The radial part of the Schrödinger equation for two electrons with no Coulomb interaction shows:

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

where $E^{(2)}$ stands for the total energy for the system with two electrons.

By defining $\mathbf{r} = \mathbf{r_1} - \mathbf{r_2}$ and $\mathbf{R} = \frac{1}{2}(\mathbf{r_1} + \mathbf{r_2})$, the equation can be separated, via the ansatz for the wavefunction $u(r, R) = \psi(r)\theta(R)$ and the energy is given by the sum of the relative energy E_r and the center-of-mass energy E_R, that is $E^{(2)} = E_r + E_R$.

$$\left(-\frac{\hbar^2}{m}\frac{d^2}{dr^2} - \frac{\hbar^2}{4m}\frac{d^2}{dR^2} + \frac{1}{4}kr^2 + kR^2\right) u(r, R) = E^{(2)}u(r, R).$$

We then add the repulsive Coulomb interaction between two electrons, $V(r) = \frac{\beta e^2}{r}$ and introduce yet again a dimensionless variable $\rho = \frac{r}{\alpha}$, where $\rho \in [0, \infty]$

$$-\frac{d^2}{d\rho^2}\psi(\rho) + \frac{1}{4}\frac{mk}{\hbar^2}\alpha^4\rho^2\psi(\rho) + \frac{m\alpha\beta e^2}{\rho\hbar^2}\psi(\rho) = \frac{m\alpha^2}{\hbar^2}E_r\psi(\rho)$$

By fixing $\alpha = \frac{\hbar^2}{m\beta e}$ and defining $\omega_r^2 = \frac{mk}{4\hbar^2}\alpha^4$ this gives us λ to be:

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

we then transform the equation to:

$$-\frac{d^2}{d\rho^2}\psi(\rho) + \omega_r^2\rho^2\psi(\rho) + \frac{1}{\rho}\psi(\rho) = \lambda\psi(\rho) \tag{6}$$

We will again convert this into a discretized form which leads into a set of linear equation on matrix form, where it consist of eigenvectors with corresponding eigenvalues where the eigenvectors is operated on with a tridiagonal matrix:

$$\begin{bmatrix} d_1 & e_1 & 0 & 0 & \dots & 0 \\ e_1 & d_2 & e_2 & 0 & \dots & 0 \\ 0 & e_2 & d_3 & e_3 & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & \dots & e_{N-3} & d_{N-2} & e_{N-2} \\ 0 & \dots & \dots & \dots & e_{N-2} & d_{N-1} \end{bmatrix} \begin{bmatrix} \psi_1 \\ \psi_2 \\ \dots \\ \dots \\ \dots \\ \psi_{N-1} \end{bmatrix} = \lambda \begin{bmatrix} \psi_1 \\ \psi_2 \\ \dots \\ \dots \\ \dots \\ \psi_{N-1} \end{bmatrix}.$$

where $e_i = -\frac{1}{h^2}$ and $d_i = \frac{2}{h^2} + \omega_r^2 \rho_i^2 + \frac{1}{\rho_i}$ for the two electron case.

III. Method

i. Jacobi method

To be able to solve the eigenvalue problem presented, the jacobi method is a nice way to go about. The idea of this method is to transform our tridiagonal matrix **A** from equation 3 as well as the matrices from the quantum dot problem for one and two electrons. The goal is to transform them such that all non-diagonal elements essentially becomes zero, while the diagonal elements gives out the eigenvalues, λ. For doing so, there is indeed an orthogonal transformation matrix needed. We define a orthogonal transformation matrix **S**:

$$\mathbf{S} = \begin{bmatrix} 1 & 0 & \dots & 0 & 0 & \dots & 0 & 0 \\ 0 & 1 & \dots & 0 & 0 & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & \cos\theta & 0 & \dots & 0 & \sin\theta \\ 0 & 0 & \dots & 0 & 1 & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & 0 & 0 & \dots & 1 & 0 \\ 0 & 0 & \dots & -\sin\theta & 0 & \dots & 0 & cos\theta \end{bmatrix} \tag{7}$$

This orthogonal transformation matrix is used to transform the original matrix into a diagonalized matrix. We then end up with a new matrix after some matrix operations:

$$\mathbf{B} = \mathbf{S}^T \mathbf{A} \mathbf{S} \tag{8}$$

If we again look at our original matrix equation, and try to transform it using the orthogonal transformation matrix. By multiplying $\mathbf{S}^T$ and $\mathbf{S}^T\mathbf{S} = \mathbf{I}$ in equation 8

$$(\mathbf{S^T A S})(\mathbf{S^T u}) = \lambda \mathbf{S^T u}$$

By our definition we then end up with:

$$\mathbf{B}(\mathbf{S^T u}) = \lambda(\mathbf{S^T u}) \tag{9}$$

Where we have preserved the eigenvalues, but changed the eigenvectors $\mathbf{u}$.

One important property of our orthogonal transformation matrix is; $\mathbf{S^T} = \mathbf{S^{-1}}$. Our matrix $\mathbf{S}^{N\times N}$, can be difficult to keep track on, so by defining some indexes to keep track of the non-zero elements is a good way to start:

$$s_{kk} = s_{ll} = \cos\theta, \quad s_{kl} = -s_{lk} = -\sin\theta, \quad s_{ii} = -s_{ii} = 1 \quad i \neq k \quad i \neq l$$

Indexing is only necessary in the matrix where the elements differs from zero. The general expressions for the elements on the transformed matrix $\mathbf{B}$ is as following:

$$b_{ii} = a_{ii} \qquad i \neq k, i \neq l \tag{10}$$
$$b_{ik} = a_{ik}c - a_{il}s \qquad i \neq k, i \neq l \tag{11}$$
$$b_{il} = a_{ij}c + a_{ik}s \qquad i \neq k, i \neq l \tag{12}$$
$$b_{kk} = a_{kk}c^2 - 2a_{kl}cs + a_{ll}s^2 \tag{13}$$
$$b_{ll} = a_{ll}c^2 + 2a_{kl}cs + a_{kk}s^2 \tag{14}$$
$$b_{kl} = (a_{kk} - a_{ll})cs + a_{kl}(c^2 - s^2) \tag{15}$$

Where we have defined $s = \sin\theta$ and $c = \cos\theta$. The angle θ is arbitrary, and we choose the angle to give out zero elements on the non-diagonals. We need to find a criteria for which the non-diagonal elements in $\mathbf{B}$ is essential zero. We see by equationn 15 that the non-diagonal elements is expressed as:

$$b_{kl} = (a_{kk} - a_{ll})cs + a_{kl}(c^2 - s^2) = 0$$

Where we are forcing this to be equal to zero. By doing some reordering we can define a τ:

$$a_{kl}(c^2 - s^2) = (a_{ll} - a_{kk})cs$$

$$\frac{(c^2 - s^2)}{cs} = \frac{(a_{ll} - a_{kk})}{a_{kl}}$$

Multplying this with $\frac{1}{2}$, we end up with:

$$\frac{(c^2 - s^2)}{2cs} = \frac{(a_{ll} - a_{kk})}{2a_{kl}}$$

$$\tau = \frac{(a_{ll} - a_{kk})}{2a_{kl}} \tag{16}$$

Further on we define we know that tan $= \frac{\sin}{\cos}$, we then call tan for t. from this we have to expression for τ. We know need to find the values for t, for which the problem is satisfied:

$$\tau = \frac{(c^2 - s^2)}{2cs}$$

by reorganizing this we end up with:

$$2\tau = \frac{c^2}{cs} - \frac{s^2}{cs} = \frac{c}{s} - \frac{s}{c}$$

Multplying both sides with $\frac{s}{c}$ we end up with:

$$t^2 + 2\tau t - 1 = 0$$

The solution for t is then given by:

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

In our numerical analysis we need to rewrite this into a more solid equation, which tolerates high τ-values. The reason for this is to avoid numerical loss of precision in our calculations.

$$t = \frac{1}{\tau \pm \sqrt{1 + \tau^2}} \tag{17}$$

By now we can write the expressions for *sin* and *cos* as:

$$c = \frac{1}{\sqrt{1 + t^2}} \qquad\qquad s = ct \tag{18}$$

The jacobi method is an inefficient method when it comes to large matrices, especially in our case where we already have a tridiagonoal matrix.

IV. Implementation

Programs used in this project can be found on https://github.com/patrykpk/FYS4150/tree/master/Project_2 and the "README.md" explains how to run the scripts. All calculations are done in C++, while the plotting is done in Python. Our C++ program is based on the codes found in the course's github repository https://github.com/CompPhysics/ComputationalPhysics/tree/master/doc/Projects/2019/Project2 [4].

i. Overview: Implementation of code

This subsection will include how we implemented our code and include some of the code:

- Constructing the tridiagonal Toeplitz matrix, with the potentials added on the diagonal elements corresponding to each of the cases we are studying.

- Now we choose the matrix elements a_{kl} so that we have those with largest value, that is $|a_{kl}| = \max_{i \neq j} |a_{ij}|$.

Listing 1: C++ Finding Maximum value of non-diagonal elements

```
double offdiag(mat A, int *k, int *l, int N){
    double max;
    for (int i = 0; i < N; i++){
        for (int j = i+1; j < N; j++){
            double aij = fabs(A(i,j));
            if (aij > max){
                 max = aij; *k = i; *l = j;
             }
        }
    }
    return max;
}
```

The simplicity of calculating the maximum values of the non-diagonal elements furthered by knowing that we have a tridiagonal matrix which is symmetric. Thereby our non-diagonal elements is equal on the upper and lower diagonal, that is $\mathbf{A}^{\mathbf{T}} = \mathbf{A}$.

- Implementing Jacobi's method:
  - Computing thereafter $\tau = \frac{a_{ll} - a_{kk}}{2a_{kl}}$, $\tan\theta, \cos\theta$ and $\sin\theta$.
  - Computing thereafter the similarity transformation for this set of values (k, l), obtaining the new matrix $\mathbf{B} = \mathbf{S}^{\mathbf{T}}(k, l, \theta)\mathbf{AS}(k, l, \theta)$.

- Setup a while test where one compares the norm of the newly computed off-diagonal matrix elements

$$max_{\text{off}(\boldsymbol{A})} = |a_{kl}| > \epsilon$$

Listing 2: C++ While-loop

```
start = clock();
while (max_off > tolerance && iterations <= max_iterations){
    max_off = offdiag(V, &k, &l, N);
    B = Jacobi(A, S, k, l, N);
    iterations++;
}
finish = clock();
double Jacobi_Time = (double) (finish - start)/(CLOCKS_PER_SEC);
```

We have defined a tridiagonal matrix which we check for maximum values along the non-diagonals. These non-diagonal elements are called by reference, which means that the values k and l always store the maximum element in a matrix. The Jacobi method is implemented throughout the while-loop, until either the criteria for $\epsilon = 10^{-10}$ or max iterations N^3 is reached. By this our new and final matrix **B** is determined when the tolerance of ϵ or the maximum number of iterations is reached.

V. Numerical results

i. Numerical analysis of the Buckling beam problem

The numerical analysis of the buckling beam problem takes into account several things. The execution time between the Jacobi method and Armadillo has been checked for N = 10 to N = 300. Number of iterations needed to achieve the satisfied diagonal matrix ***B*** has also been included.

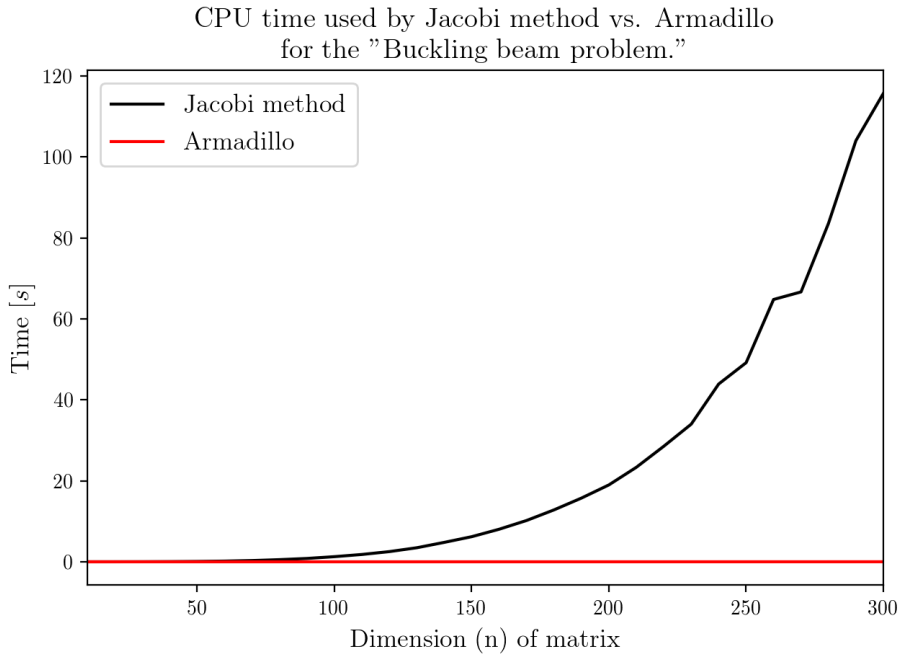


Figure 1: Comparison of the CPU time for Jacobi's method and Armadillo's [1] [2] built-in solver for eigenvalue problems *eig_sym* to diagonalize the Toeplitz matrix to a diagonal matrix, for the "Buckling beam problem".

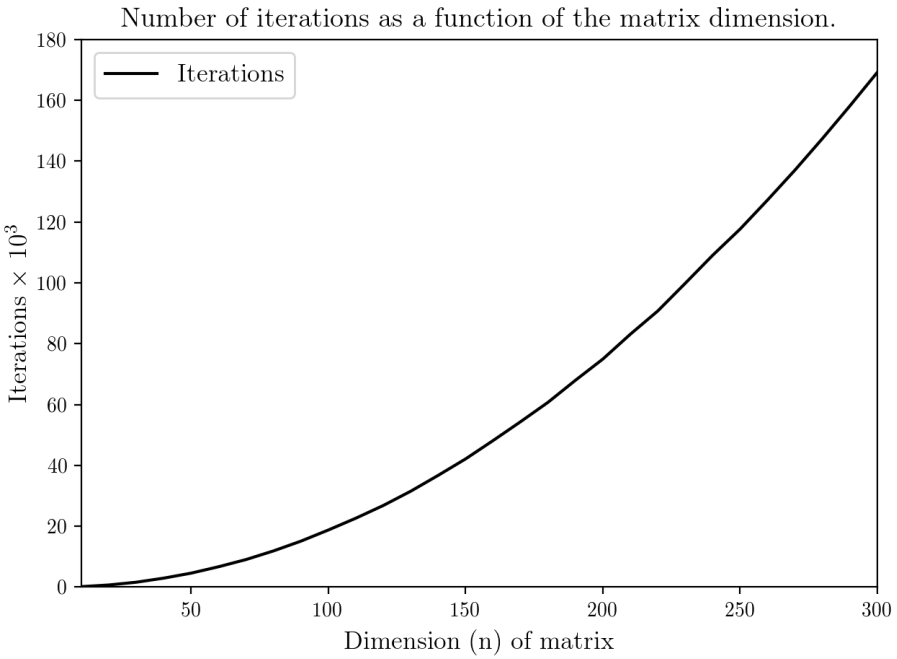


Figure 2: Amount of iterations needed to diagonalize the Toeplitz matrix in "Buckling beam problem" as a function of dimension (n) of the matrix with the tolerance for non-diagonal elements set to $\varepsilon = 10^{-10}$.

Table 1: Comparing execution time & number of iterations for the buckling beam problem with altering dimension N. Values from this table are extracted from GitHub under the folder "Results/Buckling_beam".

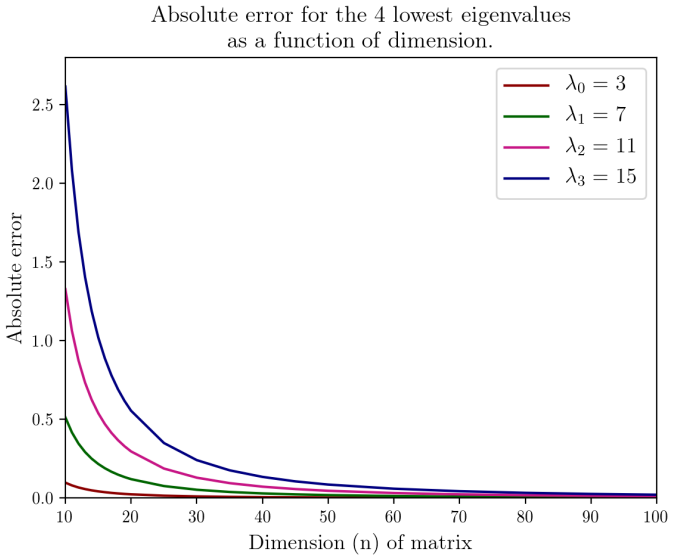
Execution time* & *Number of iteration				
N	*Jacobi method [s]*	*Armadillo [s]*	Number of iterations	Iterations in N^2
50	0.088551	0.000213	4589	$1.84N^2$
100	1.270201	0.001131	18773	$1.88N^2$
150	6.1998370	0.002375	42140	$1.87N^2$
200	18.991274	0.004293	74944	$1.87N^2$
250	49.172810	0.007059	117655	$1.88N^2$
300	115.68532	0.010477	169180	$1.88N^2$

Information seen in Table 1 represents values that have been visualized by Figure 1 and Figure 2 and shows some interesting numbers related to the buckling beam. Table 1 is only ran over a few N, from 50 to 300N's with a spacing of 50N. Values seen from this table are done with tolerance for the off-diagonal elements set to $\varepsilon = 10^{-10}$. The vast time-consuming difference in between the Jacobi method and the solver from Armadillo is discussed in the upcoming section. The number of iterations as a function of matrix dimension N is also included in table 1, in addition

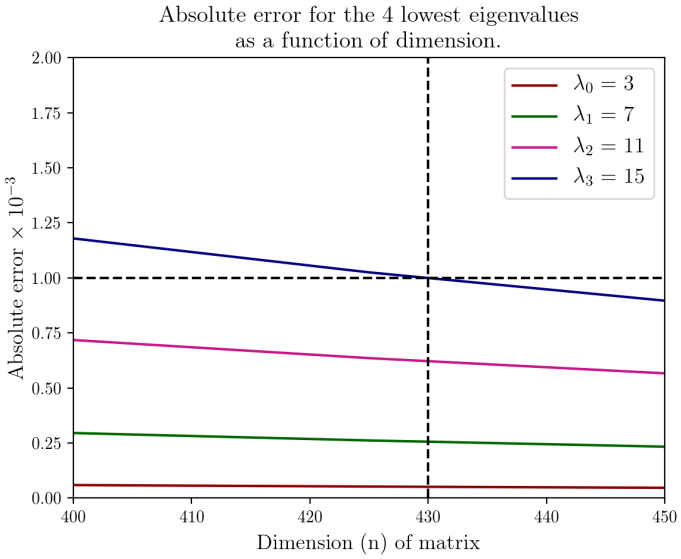
the dimensionality was extracted from the iterations to represent the behaviour in a clearer form.

ii. Numerical analysis of Quantum dots for one and two electrons

Adding the harmonic oscillator potential required finding an appropriate approximation to infinity for ρ_{max}. This was done by trial and error, and the results given in this section include that one value of $\rho_{max} = 5.5$ that was found.



(a) Part of the results showing the absolute error between the four lowest analytical and computed eigenvalues from N=10 to N=100.



(b) Part of the results showing the dimension of the matrix where the four lowest energy eigenvalues are below the threshold for deviation of 10^{-3}.

Figure 3: Absolute error for the four lowest energy eigenvalues as a function of dimension for two different sections with $\rho_{max} = 5.5$ (a) between N=10 to N=100 and (b) between N = 400 and N=450.

Figure 3 shows how the absolute error between the analytical eigenvalues and computed eigenvalues changes as a function of the dimensionality of the matrix. Acquiring high precision with three leading digits accuracy to the analytical results is given by the horizontal line. The value of N where all the eigenvalues have the desired precision can be seen from Figure 3 (b) .

In Table 2 we have found for which N the absolute error is less than 10^{-3}. In comparison with Figure 3, we can see that in order to achieve the desired precision of 10^{-3} for all eigenvalues, we must increase our matrix dimension to N = 430. The necessary dimension for eigenvalues λ_0, λ_1 and λ_2 are also included in Table 2.

Table 2: Dimension of the matrix where the four lowest energy eigenvalues have acquired the desired precision of 10^{-3} from the analytical results. Analytic and numerical results are added to the table as well as absolute error with $\rho_{max} = 5.5$. Values from this table are extracted from GitHub under the folder "Results/Qdot1".

Three leading digits deviation from the analytical results				
N	*Eigenvalue*	*Analytical*	*Numerical*	*Absolute Error*
98	λ_0	3	2.9990154	9.846×10^{-4}
218	λ_1	7	6.9990053	9.947×10^{-4}
340	λ_2	11	10.999005	9.948×10^{-4}
430	λ_3	15	14.999002	9.976×10^{-4}

For the two-electron system, we implemented a coulomb potential as well as ω_r. Results represented for this system is obtained with various ω_r:

Table 3: Table showing the different groundstate eigenvalues for different ω_r. Parameter N and ρ_{max} set to 500 and 30 respectively. Groundstate eigenvalues with corresponding ω_r. Values from this table are extracted from GitHub under the folder "Results/Qdot2".

Groundstate eigenvalue as a function of ω_r.	
ω_r	*Eigenvalue*
0.01	0.10861300
0.50	2.2298316
1.00	4.0566975
2.00	7.5184786
3.00	10.870268
4.00	14.163794
5.00	17.419189

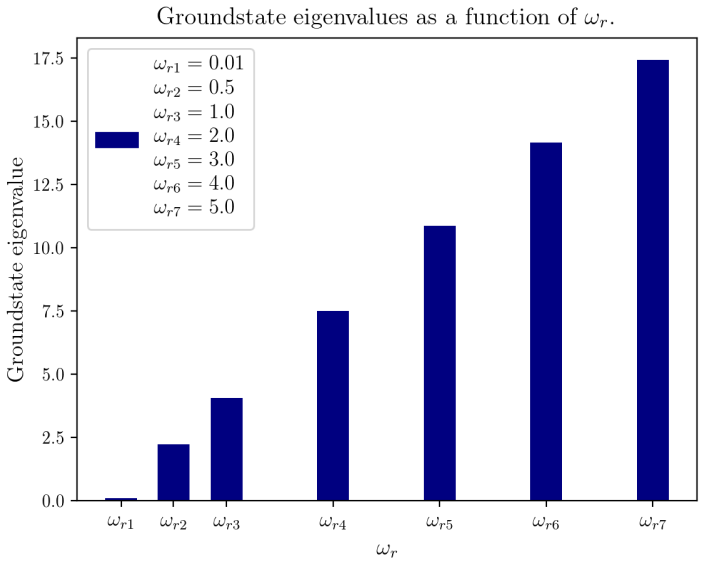


Figure 4: Plot of the data from table 3.

Figure 4 shows how the ground state eigenvalue varies with increasing ω_r. We can clearly see that by increasing the diagonal elements[1] in our tridiagonal Toeplitz matrix we get higher values of the ground state eigenvalues.

[1]The diagonal element for the two electron case is expressed as $d_i = \frac{2}{h^2} + \omega_r^2 \rho_i^2 + \frac{1}{\rho_i}$

VI. Discussion

From Table 1, we clearly see that Armadillo's *eig_sym*-function ([1] [2]) outperforms our implementation of Jacobi's rotation algorithm with respect to CPU time. For $n = 50$ the difference in CPU time between Armadillo and Jacobi is over two orders of magnitude. And for $n \geq 100$ the difference in CPU time becomes over three orders of magnitude.

This is no surprise, considering the poor convergence rate of the Jacobi method, where one needs typically $3n^2$ - $5n^2$ rotations and each rotation requires $4n$ operations, resulting in $12n^3$ - $20n^3$ operations in order to zero out non-diagonal matrix elements [4]. The consideration of number of flops is vital when analysing CPU-time, another way of looking at this is by understanding for which matrix the Jacobi method is designed for. Evaluating the Armadillo *eig_sym* with Jacobi's method we see that *eig_sym* is a better way to go about. We need to keep in mind that we are working with a trigional Toeplitz matrix, which has special properties. Then Armadillo's *eig_sym*-function is probably a more efficient algorithm for finding eigenvalues of tridiagonal matrices.

Table 2 gives us the numerical eigenvalues for the case of quantum dot with one electron. The numerical results are close to the analytical ones ($\lambda = 3,\ 7,\ 11,\ 15,\ \ldots$), but reproducing the analytical results with four leading digits after the decimal point proofs to be difficult using Jacobi's rotation algorithm. As we can see from the results, the CPU-time exhibits an exponential behaviour as a function of dimension N. By this it becomes fairly difficult to obtain a lower absolute error than 10^{-3}.

In the quantum dot case with two electrons, we see that the eigenvalues for the ground state increases with increasing ω_r and constant ρ_{max}. This makes sense, since a stronger harmonic oscillator potential should lead to higher energies. The value for ρ_{max} was determined by comparing our results with the results of M. Taut [3], where a $\rho_{max} \approx 30$ was needed for good precision for $\omega_r \approx 0.01$. For such a high ρ_{max}, the dimension N of the matrix had to be sufficiently high to get sufficient accuracy for higher ω_r, which led to running Jacobi's method with $N = 500$ & $\rho_{max} = 30$ for the different values of ω_r.

VII. Conclusion

To conclude the Jacobi's method is a reliable method which offer a neat way to solve eigenvalue problems, however the method is slow for high values of dimension N. The results showed clearly that the time exhibited an exponential behaviour as a function of N. The special properties of a tridiagonal Toeplitz matrix did not fully take advantage of the Jacobi method. By simplifying our algorithm to only operate on the upper diagonals, we reduced the number of flops by a factor of two.

Evaluating the eigenvalue problem for the one electron case showed us that the eigenvalues $[\lambda_0, \lambda_1, \lambda_2, \lambda_3]$, was possible to calculate analytically with a high precision set to 10^{-3}. Higher

precision is possible for higher N. By trial and error we found out that the value $\rho_{max} = 5.5$, gave the best analytical results. Setting the dimension $N = 430$ we where able to find the eigenvalue of the first five states with an absolute error of 10^{-3}.

Implementing two electrons into our model we found that the eigenvalues of the ground state increased with increasing ω_r. As we mentioned in the discussion, a stronger harmonic potential leads to higher energies, as one should expect. Our ρ_{max} where altered in comparison with M.Taut [3]. Increasing ρ_{max} led us to increase the matrix dimension to keep the desired precision for the given ω_r.

References

[1] Conrad Sanderson and Ryan Curtin. Practical Sparse Matrices in C++ with Hybrid Storage and Template-Based Expression Optimisation. *Mathematical and Computational Applications*, 24(3):70, 2019.

[2] Conrad Sanderson and Ryan Curtin. Armadillo: a template-based C++ library for linear algebra. *Journal of Open Source Software*, 1(3):26, 2016.

[3] M. Taut. Two electrons in an external oscillator potential: Particular analytic solutions of a coulomb correlation problem. *Physical Review A*, 48(5):3561, 1993.

[4] M. Hjorth-Jensen. Computational Physics. *University of Oslo*, `https://github.com/CompPhysics/ComputationalPhysics`, 2013.