FYS4150 – Project 2

Joachim Falck Brodin, Fredrik Jaibeer Mahal Nordeng,
and Endrias Getachew Asgedom

September 30, 2020

Abstract

Three physical problems, the buckling beam and quantum dots with one and two electrons, are reformulated as eigenvalue-problems and solved numerically with the Jacobi rotation algorithm. The method and results are compared in terms of computation time and precision with Armadillo's built in *eig_sym* function, as a function of different matrix sizes. In each case analytical solutions are available fro comparison. We find the method to perform very well in terms of precision for the buckling beam problem (relative error $\sim 10^{-11}$), however in the case of the quantum dots we have a loss of precision (giving us relative error $\sim 10^{-4}$) due to complication resulting from the way we represent discretely a boundary that in reality approaches infinity. We find the computational time for the Jacobi algorithm goes as n^2, where n is the number of grid points (or the matrix size). Compared with Armadillo's inbuilt method Jacobi's algorithm is computationally faster for $n < 10$, but for larger matrices Armadillo's function is much faster. The eigenvalues and eigenvectors from the Jacobi algorithm match the corresponding results from Armadillo's function with a high degree of precision.

Contents

GitHub repository at `https://github.com/endrias34/FYS4150`

1 Introduction

In this project we investigate three different physics problems and reformulate them as matrix-vector equations, that in turn can be cast as eigenvalue-problems. We encounter eigenvalue-problems in many fields of physical sciences and engineering. Here, we will first look at a classical-mechanics problem of a buckling beam, see Fig. 1. Initially, the two end points of the beam is held fixed and a force is applied at one end of the beam in the direction towards the origin while the other end is held fixed. The differential equation describing the motion of the beam is transformed into an eigenvalue problem after discretizing and approximating the differentiation by second order Taylor expansion. Consequently, the resulting eigenvectors represent the vertical displacement along the beam, while the eigenvalues correspond to the energy, here given by the length of the beam, the material properties, and the applied force.

In the second and third part of the assignment we consider a quantum dot in a three dimensional harmonic oscillator potential with one and two electrons. As the potential is spherically symmetric the Schrodinger equation can be separated into angular and radial contributions. Here, we consider the radial contribution and treat the problem as one-dimensional, as functions of a scaled distances from the origin for a single electron case and scaled distance between the center of the two, for the two electrons. For this problem, the eigenvectors represent the corresponding wavefunctions, when squared it gives the probability density functions of the scaled distances, and the eigenvalues correspond to the energy levels.

We will use the Jacobi rotation algorithm to solve the three different eigenvalue problems specified above. The accuracy of the Jacobi algorithm is assessed by comparing it with the analytical solution and also with Armadillo's built-in eigenvalue problem solver. The computational efficiency of the Jacobi algorithm is analysed in terms of the size of the matrix and the computation time it takes. We also benchmark the run time of Jacobi's rotation algorithm against Armadillo's implementation.

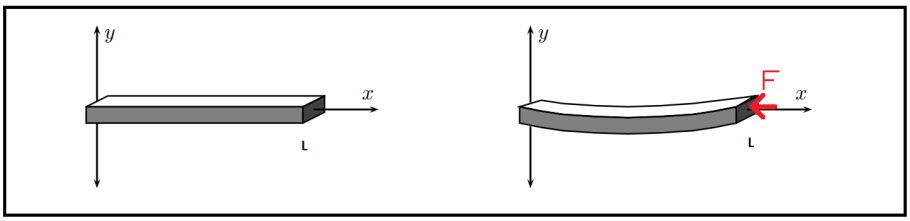

Figure 1: Illustration of the buckling of a beam

2 Theory

In this section we present the problem formulation for the buckling of a beam and quantum dots in a three dimensional harmonic oscillator potential with one and two electrons [1].

2.1 The buckling beam as an eigenvalue problem

A horizontally oriented beam of length L is fixed at both endpoints, as seen in Fig. 1. The beam can oscillate in the vertical direction, with the vertical displacement along the horizontal direction, x, denoted as $u(x)$, expressed by the differential equation

$$\gamma \frac{d^2u(x)}{dx^2} = -Fu(x), \tag{1}$$

where γ is a material constant and F is an applied force. We assume Dirichlet boundary conditions, $u(0) = u(L) = 0$. We define the dimensionless variable

$$\rho = \frac{x}{L}, \quad \rho \in [0, 1] \tag{2}$$

We rewrite Eq. (1) as

$$\frac{d^2u(\rho)}{d\rho^2} = -\lambda u(\rho), \tag{3}$$

where $\lambda = FL^2/\gamma$. We now wish to discretize our equation, and do so along the lines of what we did in Project 1,

$$-\frac{u_{i+1} + u_{i-1} - 2u_i}{h^2} = \lambda u_i, \tag{4}$$

with step length, $h = \frac{1-0}{N}$, $i \in \{0, 1, 2, ..., N\}$. Again, we can rewrite this as a set of linear equations,

$$\begin{bmatrix} d & e & 0 & 0 & 0 & \dots & 0 \\ e & d & e & 0 & 0 & \dots & 0 \\ 0 & e & d & e & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & \dots & \dots & e & d & e \\ 0 & \dots & \dots & \dots & \dots & e & d \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ \vdots \\ u_{N-2} \\ u_{N-1} \end{bmatrix} = \lambda \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ \vdots \\ u_{N-2} \\ u_{N-1} \end{bmatrix}, \tag{5}$$

where $d = 2/h^2$, $e = -1/h^2$ and the endpoints are left out. The tridiagonal Toeplitz matrix has analytical eigenpairs, with eigenvalues

$$\lambda_j = d + 2e\cos\left(\frac{j\pi}{N}\right), \quad j = 1, 2, ..., N-1. \tag{6}$$

The eigenvectors are

$$\mathbf{u}_j = [\sin(\frac{j\pi}{N}), \sin(\frac{2j\pi}{N}), ..., \sin(\frac{(N-1)j\pi}{N})]^T, \quad j = 1, 2, \dots N-1.$$

2.2 Quantum dots in three dimensions: one electron

We consider a three-dimensional harmonic oscillator potential, $V(r) = \frac{1}{2}kr^2 = \frac{1}{2}m\omega^2 r^2$. We perform a transformation to spherical coordinates and the Schrödinger

equation is separable into $\psi(r,\theta,\phi) = R(r)Y_l^m(\theta,\phi)$. The radial part of the Schrödinger equation reads

$$-\frac{\hbar^2}{2m}\left(\frac{1}{r^2}\frac{d}{dr}r^2\frac{d}{dr} - \frac{l(l+1)}{r^2}\right)R(r) + V(r)R(r) = ER(r), \quad r \in [0,\infty] \tag{7}$$

The energies are given by

$$E_{nl} = \hbar\omega\left(2n + l + \frac{3}{2}\right), \quad n = 0,1,2,\ldots \text{ and } \quad l = 0,1,2,\ldots \tag{8}$$

We perform the substitution $R(r) = (1/r)u(r)$ in Eq. (7),

$$-\frac{\hbar^2}{2m}\frac{d^2}{dr^2}u(r) + \left(V(r) + \frac{l(l+1)}{r^2}\frac{\hbar^2}{2m}\right)u(r) = Eu(r),\ u(0) = 0 \text{ and } u(\infty) = 0. \tag{9}$$

To make our equation dimensionless we substitute $\rho = (1/\alpha)r$, α is a constant with dimension length.

$$-\frac{\hbar^2}{2m\alpha^2}\frac{d^2}{d\rho^2}u(\rho) + \left(V(\rho) + \frac{l(l+1)}{\rho^2}\frac{\hbar^2}{2m\alpha^2}\right)u(\rho) = Eu(\rho). \tag{10}$$

In this case we set $l = 0$ and insert $V(\rho) = (1/2)k\alpha^2\rho^2$.

$$-\frac{\hbar^2}{2m\alpha^2}\frac{d^2}{d\rho^2}u(\rho) + \frac{k}{2}\alpha^2\rho^2u(\rho) = Eu(\rho). \tag{11}$$

We multiply with $2m\alpha^2/\hbar^2$ on both sides.

$$-\frac{d^2}{d\rho^2}u(\rho) + \frac{mk}{\hbar^2}\alpha^4\rho^2u(\rho) = \frac{2m\alpha^2}{\hbar^2}Eu(\rho). \tag{12}$$

The constant α can now be fixed so that

$$\frac{mk}{\hbar^2}\alpha^4 = 1 \quad \rightarrow \quad \alpha = \left(\frac{\hbar^2}{mk}\right)^{1/4}. \tag{13}$$

We introduce

$$\lambda = \frac{2m\alpha^2}{\hbar^2}E. \tag{14}$$

Eq. (13) then becomes

$$-\frac{d^2}{d\rho^2}u(\rho) + \rho^2u(\rho) = \lambda u(\rho), \tag{15}$$

and our problem is restated as an eigenvalue-problem. Again we need to discretize the equation, along the lines of what we did in Project 1, the discretized equation becomes

$$-\frac{u_{i+1}-2u_i+u_{i-1}}{h^2}+\rho_i^2u_i=\lambda u_i. \tag{16}$$

If we make the following substitutions,

$$d_i=\frac{2}{h^2}+\rho_i^2 \quad \text{and} \quad e_i=-\frac{1}{h^2}, \tag{17}$$

the Shroedingers equation takes the form

$$e_iu_{i-1}+d_iu_i+e_iu_{i+1}=\lambda u_i. \tag{18}$$

We have analytical eigenvalues from by Eqs. (8) and (13), which together give $\lambda=3,7,11,15,\dots$.

We need to define a maximum value of ρ, as $\rho=\infty$ is not applicable numerically.

We rewrite Eq. (18) as a matrix eigenvalue problem,

$$\begin{bmatrix} d_1 & e_1 & 0 & 0 & \dots & 0 & 0 \\ e_1 & d_2 & e_2 & 0 & \dots & 0 & 0 \\ 0 & e_2 & d_3 & e_3 & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & \dots & \dots & \dots e_{N-3} & d_{N-2} & e_{N-2} \\ 0 & \dots & \dots & \dots & \dots & e_{N-2} & d_{N-1} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ \dots \\ \dots \\ \dots \\ u_{N-1} \end{bmatrix} = \lambda \begin{bmatrix} u_1 \\ u_2 \\ \dots \\ \dots \\ \dots \\ u_{N-1} \end{bmatrix}, \tag{19}$$

again excluding the endpoints. We can solve Eq. (19) with Jacobi's method to obtain both the eigenvalues and eigenstates of one electron in a harmonic oscillator potential, which we again can compare with the analytical values.

2.3 Quantum dots in three dimensions: two electrons

To expand our problem to two electrons with no repulsive Coulomb interaction, the Schroedinger equation reads,

$$\left(-\frac{\hbar^2}{2m}\frac{d^2}{dr_1^2}-\frac{\hbar^2}{2m}\frac{d^2}{dr_2^2}+\frac{1}{2}kr_1^2+\frac{1}{2}kr_2^2\right)u(r_1,r_2)=E^{(2)}u(r_1,r_2), \tag{20}$$

with a two-electron wave function $u(r_1,r_2)$ and two-electron energy $E^{(2)}$.

If we assume that the electrons are not interacting, we can write $u(r_1,r_2)=u(r_1)u(r_2)$. The Schrödinger equation can be reordered in terms of the center-of-mass coordinate $\vec{R}=(\vec{r}_1+\vec{r}_2)/2$ and the relative coordinate $\vec{r}=\vec{r}_1-\vec{r}_2$, and reads

$$\left(-\frac{\hbar^2}{m}\frac{d^2}{dr^2}-\frac{\hbar^2}{4m}\frac{d^2}{dR^2}+\frac{1}{4}kr^2+kR^2\right)u(r,R)=E^{(2)}u(r,R). \tag{21}$$

We separate into one R-dependent equation and one r-dependent equation. We can now include the repulsive Coulomb interaction between the two electrons, by adding a term to the potential energy

$$V(r_1, r_2) = \frac{\beta e^2}{|\mathbf{r}_1 - \mathbf{r}_2|} = \frac{\beta e^2}{r}, \tag{22}$$

where $\beta e^2 = 1.44$nmeV.

If we perform a separation of variables, the r-dependent Schrödinger equation with Coulomb interaction reads

$$\left(-\frac{\hbar^2}{m}\frac{d^2}{dr^2} + \frac{1}{4}kr^2 + \frac{\beta e^2}{r}\right)\psi(r) = E_r\psi(r). \tag{23}$$

The energy of the system is given by $E^{(2)} = E_r + E_R$. We define $\rho = r/\alpha$ and divide by $\hbar^2/m$, and insert into Eq. (23).

$$-\frac{d^2}{d\rho^2}\psi(\rho) + \frac{1}{4}\frac{mk}{\hbar^2}\alpha^4\rho^2\psi(\rho) + \frac{m\alpha\beta e^2}{\rho\hbar^2}\psi(\rho) = \frac{m\alpha^2}{\hbar^2}E_r\psi(\rho). \tag{24}$$

We define

$$\omega_r^2 = \frac{1}{4}\frac{mk}{\hbar^2}\alpha^4, \tag{25}$$

and fix α by requiring

$$\frac{m\alpha\beta e^2}{\hbar^2} = 1. \tag{26}$$

We insert into Eq. (21),

$$-\frac{d^2}{d\rho^2}\psi(\rho) + \omega_r^2\rho^2\psi(\rho) + \frac{1}{\rho}\psi(\rho) = \lambda\psi(\rho), \tag{27}$$

with

$$\lambda = \frac{m\alpha^2}{\hbar^2}E_r. \tag{28}$$

Discretized, Eq. (27) reads,

$$-\frac{\psi_{i+1} - 2\psi_i + \psi_{i-1}}{h^2} + \omega_r^2\rho_i^2\psi_i + \frac{1}{\rho_i}\psi_i = \lambda\psi_i. \tag{29}$$

We define

$$d_i = \frac{2}{h^2} + \omega_r^2\rho_i^2 + \frac{1}{\rho_i} \quad \text{and} \quad e_i = -\frac{1}{h^2}. \tag{30}$$

We again have an eigenvalue problem that can be solved by Jacobi's method.

3 Method

Here, we first describe the similarity transformation that the Jacobi method is using for iterating towards the true eigenvalues. Then, we present how to implement the Jacobi rotation algorithm. Finally, we introduce unit tests as a way to check that the implementation is done correctly.

3.1 The Jacobi rotation algorithm

We consider an invertible matrix, $\boldsymbol{G}$, such that $\boldsymbol{G^{-1}AG} = \boldsymbol{B}$. Beacuse $\boldsymbol{G}$ is invertible we see, by first left multiplying by $\boldsymbol{G}$ on both sides, then right multiplying, by $\boldsymbol{G^{-1}}$, that $\boldsymbol{A} = \boldsymbol{GBG^{-1}}$, and that $\boldsymbol{A}$ and $\boldsymbol{B}$ are similar. Similar matrices have the property that their characteristic polynomial [2] is the same, meaning, among other things, that their eigenvalues are equal (see proof in Appendix B).

The Jacobi algorithm systematically reduces the norm of the off-diagonal elements, implying that if we repeat our procedure enough times, our matrix becomes diagonal, or close to it, depending on number of iteration or other thresholds set, leaving us with $\boldsymbol{B} \simeq \boldsymbol{A}$. However, the method is slow. Typically we need between $3n^2$ - $5n^2$ rotations, as when done by the Given rotation matrix [3] (we have actually used $\boldsymbol{G^T}$):

$$G^T(i,j,\theta) = \begin{bmatrix} 1 & \dots & 0 & \dots & 0 & \dots & 0 \\ \vdots & \ddots & \vdots & & \vdots & & \vdots \\ 0 & \dots & c & \dots & s & \dots & 0 \\ \vdots & & \vdots & \ddots & \vdots & & \vdots \\ 0 & \dots & -s & \dots & c & \dots & 0 \\ \vdots & & \vdots & & \vdots & \ddots & \vdots \\ 0 & \dots & 0 & \dots & 0 & \dots & 1 \end{bmatrix}, \tag{31}$$

We align $\boldsymbol{G's}$ sine (the s in the matrix) such that it is multiplied by our maximum off-diagonal element (also adjusting -sine, so that a square of c, s, -s, c is preserved inside of $\boldsymbol{G}$). As an example let us consider a case where our max off diagonal element is at $\boldsymbol{A_{in}}$:

$$G^T(i,j,\theta) = \begin{bmatrix} 1 & \dots & 0 & \dots & 0 & \dots & 0 \\ \vdots & \ddots & \vdots & & \vdots & & \vdots \\ 0 & \dots & c & \dots & 0 & \dots & s \\ \vdots & & \vdots & \ddots & \vdots & & \vdots \\ 0 & \dots & 0 & \dots & 1 & \dots & 0 \\ \vdots & & \vdots & & \vdots & \ddots & \vdots \\ 0 & \dots & -s & \dots & 0 & \dots & c \end{bmatrix}, \tag{32}$$

In the algorithm we will take advantage of the fact that we mostly only use the 4 element from the Given rotation matrix. Still, since each rotations takes $4n$ operations, this yields approximately between $12n^3$ - $20n^3$ total operations. In our case, this results in approximatly a quadratic increase in time relative to matrix size/grid points.

Algorithm (3.1) shows the pseudo-code fro the Jacobi rotation algorithm.

Algorithm 1 Jacobi Rotation Algorithm

Input: $\mathbf{A}, \mathbf{R}, \mathbf{k}, \mathbf{l}$

// $\boldsymbol{A}$ is our matrix, $\boldsymbol{R}$ our eigenvectors (initialized as the Identity matrix),
// and k and l are the indices of the largest off diagonal element in $\boldsymbol{A}$
// updated before every iteration of the while loop

while $\boldsymbol{A}$'s $\boldsymbol{argmax}$ off diagonal $> \varepsilon$ **do** // Or a max iteration of choice

  $tau = (\boldsymbol{A}[l,l] - \boldsymbol{A}[k,k])\ /\ (2 * \boldsymbol{A}[k,l])$

  if $tau >= 0$ **then**
    $t = 1/(tau + \sqrt{1 + tau^2})$ // We do this to avoid roundoff errors
  else
    $t = -1/(-tau + \sqrt{1 + tau^2})$
  end if

  $c = 1/\sqrt{1 + t^2}$
  $s = c * t$
  $a_{kk} = \boldsymbol{A}[k,k]$
  $a_{ll} = \boldsymbol{A}[l,l]$

  for $i = 1$ to $i = n - 1$ **do**
    if i != k and i != l **then**

      $a_{ik} = \boldsymbol{A}[i,k]$
      $a_{il} = \boldsymbol{A}[i,l]$
      $\boldsymbol{A}[i,k] = c * a_{ik} - s * a_{il}$
      $\boldsymbol{A}[k,i] = \boldsymbol{A}[i,k]$
      $\boldsymbol{A}[i,l] = c * a_{il} + s * a_{ik}$
      $\boldsymbol{A}[l,i] = \boldsymbol{A}[i,l]$

    end if // Now we set the new eigenvectors

    $b_{ik} = \boldsymbol{R}[i,k]$
    $b_{il} = \boldsymbol{R}[i,l]$
    $\boldsymbol{R}[i,k] = c * b_{ik} - s * b_{il}$
    $\boldsymbol{R}[i,l] = c * b_{ik} + s * b_{ik}$

  end for

  $\boldsymbol{A}[k,k] = c^2 * a_{kk} - 2 * c * s * \boldsymbol{A}[k,l] + s^2 * a_{ll}$
  $\boldsymbol{A}[l,l] = s^2 * a_{kk} + 2 * c * s * \boldsymbol{A}[k,l] + c^2 * a_{ll}$
  $\boldsymbol{A}[k,l] = 0$
  $\boldsymbol{A}[l,k] = 0$

end while

Output: $\mathbf{A}, \mathbf{B}$

3.2 Unit tests

When implementing our algorithm, we need to stay aware of the possibility of errors in the coding. This leads us to implement three different unit test in our program. We choose to see if we have a working algorithm for getting the largest off-diagonal elements from our matrix, that the Jacobi rotation is returning the correct eigenvalues, and that our eigenvectors are orthogonal.

This is implemented by using a matrix where we already know the largest off-diagonal element and eigenvalues, and we check orthogonality using the dot product of the eigenvectors.

4 Results

We start our analysis of the Jacobi's method for solving eigenvalue-problems by performing unit tests. All the tests were successful for our implementation of the Jacobi rotation algorithm and hence it allowed us to use it for solving problems like; the buckling of a beam, and quantum dots in three dimension with one and two electrons.

4.1 The buckling beam problem

The buckling of a beam presented in section 2.1 is analysed here for solving the problem using Jacobi rotation algorithm and Armadillo [4]. The Jacobi rotation algorithm is a computationally heavy algorithm for solving larger size eigenvalue problems. To see the dependence of the number of rotations (or similarity transformations) needed to solve a given size eigenvalue problem, we computed the eigenvalues and eigenvectors of the buckling beam problem for different matrix sizes, see Fig. 2. After fitting our data with a second order polynomial function, we observed that the number of similarity transformations increases as a square of the matrix sizes. To analyse the efficiency of the Jacobi algorithm, we compared the CPU time consumption of the Jacobi algorithm with that of using Armadillo (cf. Fig. 3). From Fig. 3 it possible to see that the Jacobi algorithm is faster than Armadillo for a matrix size of 10×10 or smaller, but it quickly become highly inefficient for matrix sizes larger then 10×10.

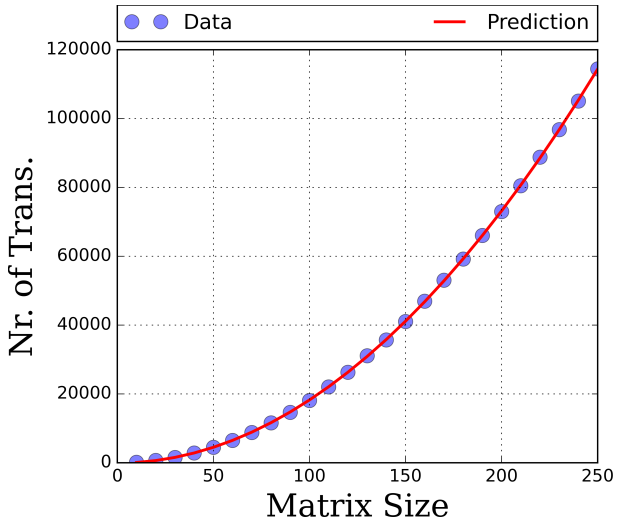

Figure 2: Number of similarity transformations needed for Jacobi rotation algorithm to diagonalize a given matrix size.

The first three eigenvalues and the corresponding eigenvectors for the buckling beam problem are shown in Fig. 4 and 5, respectively. In Fig. 4 we also show the relative error in logarithmic scale. In general, the accuracy of both Jacobi and Armadillo method measured in terms of relative error, increases when the matrix size increases. Nonetheless, all the relative errors are smaller than 10^{-11}. The relative errors for the eigenvalues and eigenvectors are of the same order and hence we only presented here the eigenvalues relative errors.

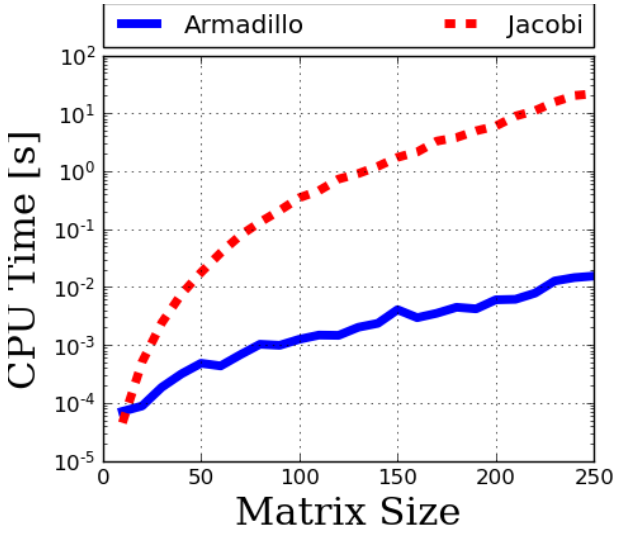

Figure 3: CPU time in seconds for solving the buckling beam eigenvalue problem for different matrix sizes using Jacobi rotation and Armadillo.

The three first eigenvectors of the buckling beam problem are shown in Fig. 5. Here, the number of discretization points (or matrix size) are fixed to be 250 and the solution from the Jacobi rotation method, Armadillo, and its corresponding analytical solution are presented. It is pertinent to notice the match between the analytical and the numerical solution. Physically, the eigenvectors of the buckling beam problem illustrate the different modes of vibration (i.e., standing waves) of the beam.

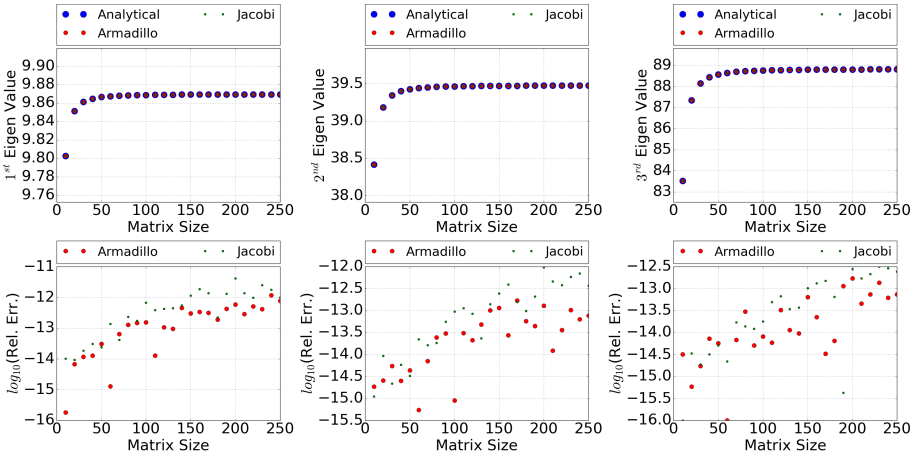

Figure 4: The first three eigenvalues of the buckling beam problem computed using Jacobi rotation, Armadillo and its corresponding analytical solution. The relative error of the Jacobi and Armadillo method is shown in the bottom row.

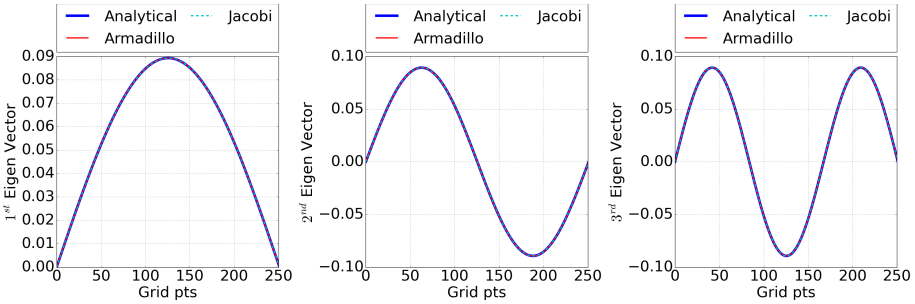

Figure 5: The first three eigen vectors of the buckling beam problem computed using Jacobi rotation, Armadillo and its corresponding analytical solution.

4.2 Single electron in a harmonic oscillator potential

Numerically solving the problem of a single electron in a harmonic oscillator potential requires an optimization for finding a suitable ρ_{max} value. In addition, since we have an analytical solutions for the eigenvalues, we can check the number of integration points (discretization points) required to reach the analytical solutions with an acceptable relative error. After searching ρ_{max} within the range between 1 and 10, and the number of discretization points (N) between

10 and 250, we found the smallest relative error when $\rho_{max} = 5$ and $N = 250$. The first three eigenvalues as a function of the discretization points is shown in Fig. 6 (top row) for the three different ways of solving the problem. The relative error for the Jacobi method and Armadillo are shown in Fig. 6 (bottom row). The relative error for the first eigenvalue when using $N = 250$ is $\sim 10^{-4.5}$, for the second $\sim 10^{-4}$, and the third $\sim 10^{-4}$. Compared to the buckling beam problem, solving the single electron problem using numerical eigenvalue problem solvers is less precise.

We calculated the radial probability density of the electron for the first three eigenvalues (cf. Fig. 7). As the eigenvalues gets larger (i.e., higher energy levels) the probability of finding the electron also shifts away from the origin.

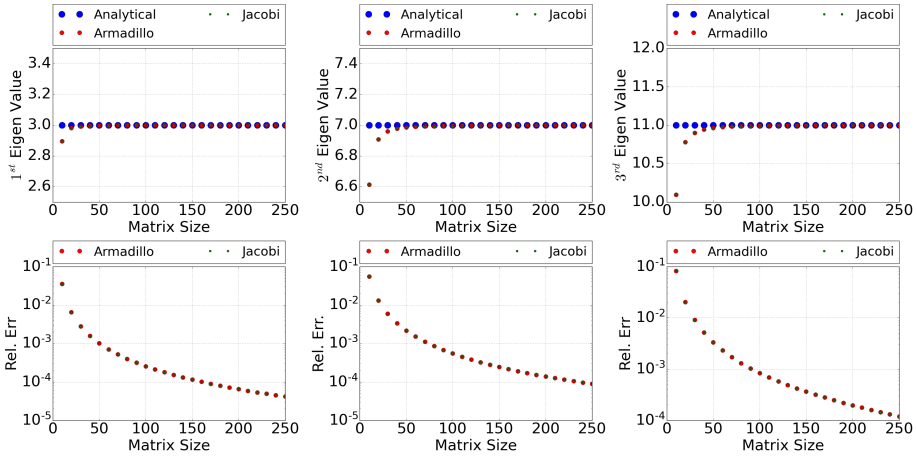

Figure 6: The first three eigenvalues of the single electron problem computed using Jacobi rotation, Armadillo and its corresponding analytical solution. The relative error of the Jacobi and Armadillo method is shown in the bottom row. Notice we have used $\rho_{max} = 5$

.

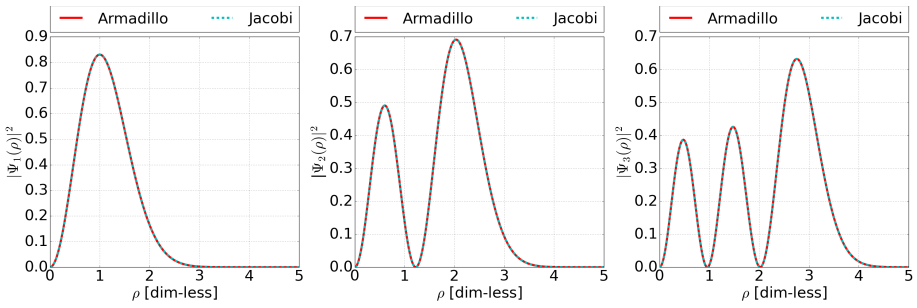

Figure 7: The first three radial probability densities of the single electron problem computed using Jacobi rotation and Armadillo.

4.3 Two electrons in a harmonic oscillator potential

We now consider two electrons in a harmonic potential with and without Coulomb interactions. For the two electrons with Coulomb intraction case the eigenvalue problem presented in section 2.3 has analytical solution [5] for some specific values of ω_r. We compared our Jacobi algorithm solution with that of the analytical and computed relative errors (cf. Table 1). All the numerical calculations were performed using 250 integration points. Here, it is pertinent to notice that as the oscillator parameter ω_r decreases the value of ρ_{max} needed to be increased to mach the analytical solution.

Table 1: The ground state eigenvalues for the two interacting electrons computed using analytical and Jacobi rotation methods. different values of ω_r.

ω_r	ρ_{max}	Analytical	Jacobi	Relative Error
0.25	10	1.25	1.2499	1×10^{-04}
0.05	20	0.35	0.3499	2.6×10^{-05}
0.01827	30	0.1644	0.16444	2.99×10^{-04}

The probability density for the ground state of the two interacting electrons system is shown in Fig. 8 for the oscillator parameters $\omega_r = 0.01, 0.5, 1$, and 5. Here, we can see the peak of the probability density function move closer to zero when ω_r gets larger. This is due to the fact that the oscillator potential gets wider when ω_r gets smaller and this allows the two electrons to move further away from each other.

To investigate the effect of the Coulomb interaction on the two electrons, we computed the probability density function for the interacting and non-interacting two electrons with $\omega_r = 1$ and 0.5 (see Fig. 9). Both the interacting and non-interacting cases show the probability density functions peak shift to larger distances between the two electrons when ω_r gets smaller. Moreover, we also notice a counter intuitive behaviour that the probability density function for interacting electrons with smaller ω_r appear further away from each other than its corresponding non-interacting case. This behaviour is due to the infinite range of the Coulomb interaction and the fact that the harmonic oscillator energy reduces to zeros as ω_r approaches to zero.

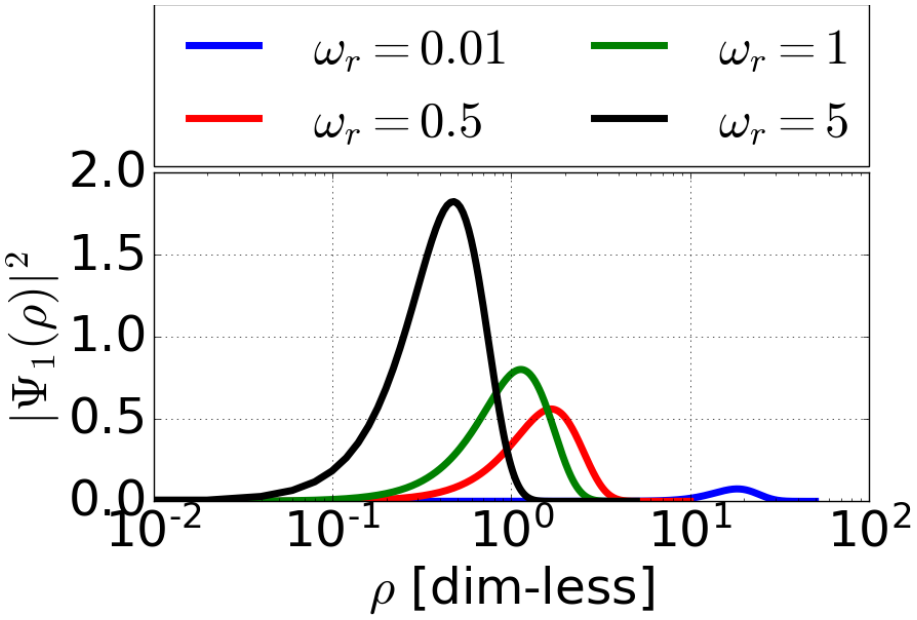

Figure 8: The radial probability densities of the ground state of the two interacting electrons in a harmonic oscillator potential for different values of ω_r. Notice ρ is the scaled relative distance between the two electrons.

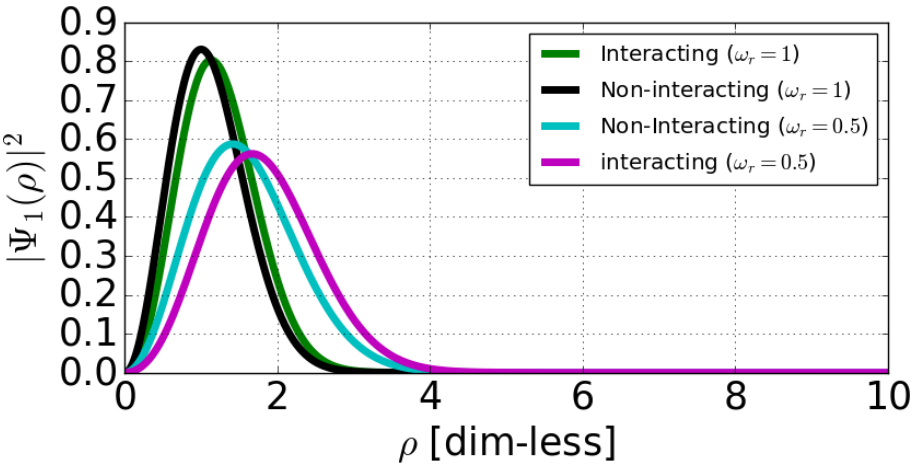

Figure 9: The radial probability densities of the ground state of the two interacting and non-interacting electrons in a harmonic oscillator potential for $\omega_r = 1$ and $\omega_r = 0.5$. We see that when the potential of the well gets larger (steered by setting ω_r smaller) the Columb interaction, $1/\rho$, effectivly is forcing the electrons further away from the center compared to the non-interacting case.

5 Discussion

Jacobi rotation algorithm for solving larger size eigenvalue problems is inefficient. From Fig. 2 we have seen that the number of similarity transformations performed by the Jacobi algorithm depends on the number of gird points (or size of the matrix n). This dependence was shown to be quadratic (n^2) in our case. We have also seen the number of FLOPs per iteration go as n^2, therefore the total number of FLOPs for the Jacobi algorithm is $\mathcal{O}(n^4)$.

The three eigenvalue problems we consider in this project results in a tridiagonal matrix eigenvalue problem. However, we used the Jacobi rotation algorithm, which is designed for solving small size dense matrices. Therefore, the comparison of Jacobi's algorithm with that of Armadillo's implementation (which identifies the matrix type and use the most efficient method) shows how inefficient Jacobi's algorithm is for larger size matrices.

6 Conclusion

The Jacobi method, despite being inefficient compared to Armadillo's implementation, managed to generate the correct eigenvalues and eigenvectors for the buckling beam and quantum dots problems considered in this project.

The accuracy of the Jacobi method is evaluated by comparing the resulting eigenvalues and eigenvectors with the corresponding exact analytical solutions. For a matrix size of 250×250, the relative error for the buckling beam problem is in the order of $\sim 10^{-11}$. However, for the quantum dot problems with one and two electrons, the relative error is in the order of $\sim 10^{-4}$. The loss of accuracy in the later case is due to the fact that we have approximated $\rho_{max} = \infty$ with a finite value.

The eigenvalues and eigenvectors of the buckling beam provide physical insight about the energy and mode of vibration, respectively. For the quantum dot examples, the eigenvalues are related to the different energy states of the electron(s) while the eigenvectors represent the wavefunctions, which can be squared to provide the probability densities.

References

[1] M. Hjorth-Jensen, "Computational physics, lecture notes fall 2015," *Department of Physics, University of Oslo*, p. 217, 2015.

[2] "Characteristic polynomial." [Online]. Available: https://en.wikipedia.org/wiki/Characteristic_polynomial

[3] W. Givens, "Given rotation matrix." 1950s. [Online]. Available: https://en.wikipedia.org/wiki/Givens_rotation

[4] C. Sanderson and R. Curtin, *Armadillo: a template-based C++ library for linear algebra.*, Journal of Open Source Software, http://dx.doi.org/10.21105/joss.00026, 2016.

[5] M. Taut, "Two electrons in an external oscillator potential: Particular analytic solutions of a coulomb correlation problem," *Phys. Rev. A*, vol. 48, pp. 3561–3566, Nov 1993. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevA.48.3561

A Orthogonality Preservation

Here, we proof the preservation of orthogonality and dot product after similarity transformations. Similarity transformations are unitary transformations. Now, we consider an orthogonal basis vector $\mathbf{v}_i \in \mathbf{R}^n$ and an unitary transformation matrix $\mathbf{U} \in \mathbf{R}^{n\times n}$. Defining $\mathbf{w}_i = \mathbf{U}\mathbf{v}_i$, the dot product can be computed as

$$\begin{aligned}\mathbf{w}_j^T \cdot \mathbf{w}_i &= (\mathbf{U}\mathbf{v}_j)^T \cdot (\mathbf{U}\mathbf{v}_i) = \mathbf{U}^T\mathbf{v}_j^T \cdot \mathbf{U}\mathbf{v}_i = \mathbf{U}^T\mathbf{U}\mathbf{v}_j^T \cdot \mathbf{v}_i \\ &= \mathbf{U}^{-1}\mathbf{U}\mathbf{v}_j^T \cdot \mathbf{v}_i = \mathbf{I}\mathbf{v}_j^T \cdot \mathbf{v}_i = \delta_{ij}\end{aligned}$$

where $\mathbf{I} \in \mathbb{R}^{n\times n}$ is the identity matrix and the third last equality holds due to $\mathbf{U}^T = \mathbf{U}^{-1}$ for orthogonal matrices.

B Similarity transformations

If $n \times n$ matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ are similar, then they have the same characteristic polynombial and hence the same eigenvalues.

Proof. :

$$\begin{aligned}\boldsymbol{B} &= \boldsymbol{G^{-1}AG} \\ \boldsymbol{B} - \lambda\boldsymbol{I} &= \boldsymbol{G^{-1}AG} - \lambda\boldsymbol{G^{-1}G}, && \text{since } \boldsymbol{I} = \boldsymbol{G^{-1}G}\text{, also note } \lambda \text{ is a scalar} \\ &= \boldsymbol{G^{-1}AG} - \lambda\boldsymbol{G^{-1}G} \\ &= \boldsymbol{G^{-1}}\ \boldsymbol{AG} - \boldsymbol{G^{-1}}\ \lambda\boldsymbol{G} \\ &= \boldsymbol{G^{-1}}\ (\boldsymbol{AG} - \lambda\boldsymbol{G}) \\ \boldsymbol{B}-\lambda\boldsymbol{I} &= \boldsymbol{G^{-1}}(\boldsymbol{A}-\lambda\boldsymbol{I})\boldsymbol{G}\end{aligned}$$

Now we use a rule for determinates which is stating that:

$$\det(\boldsymbol{AB}) = \det(\boldsymbol{A})*\det(\boldsymbol{B})$$

We then have

$$\begin{aligned}\det(\boldsymbol{B}-\lambda\boldsymbol{I}) &= \det(\boldsymbol{G^{-1}}[\boldsymbol{A}-\lambda\boldsymbol{I}]\boldsymbol{G}) \\ &= \det(\boldsymbol{G^{-1}})*\det(\boldsymbol{A}-\lambda\boldsymbol{I})*\det(\boldsymbol{G}) \\ &= \det(\boldsymbol{G^{-1}})\ *\det(\boldsymbol{G})*\det(\boldsymbol{A}-\lambda) \\ &= \det(\boldsymbol{G^{-1}G})*\det(\boldsymbol{A}-\lambda), && \text{where } \det(\boldsymbol{G^{-1}G}) = \det(\boldsymbol{I}) \\ \det(\boldsymbol{B}-\lambda\boldsymbol{I}) &= \det(\boldsymbol{A}-\lambda\boldsymbol{I})\end{aligned}$$

□