Numerical solution to eigenvalue problem for classical and quantum mechanical applications

Project 2, FYS4150
Autumn 2019

Jowita Borowska

ABSTRACT

In the following project we introduce three physical cases: the buckling beam problem, one electron in the three-dimensional quantum harmonic oscillator potential and two electrons interacting via Coulomb repulsion in the three-dimensional harmonic oscillator well. All three problems can be represented mathematically as a set of linear equations (eigenvalue problems), involving a tridiagonal symmetric matrix. We apply a series of similarity transformations - Jacobi rotations - in order to diagonalize the matrix and update the orthogonal basis of eigenvectors. Then, the set of eigenvalues can be found as entries on the main diagonal of the resulting matrix. It turns out that Jacobi algorithm is quite accurate (in case of the buckling beam the difference between numerical and analytical solution is of order 10^{-10} for 100×100 matrix). However, it takes many iterations of the algorithm in order to diagonalize the matrix (to a given approximation of off-diagonal elements being zero). We show that the convergence is quadratic, i.e. we need around $1.8N^2$ transformations in order to fully diagonalize N-dimensional matrix. This transfers to a long CPU time (88 s for N = 250), especially in comparison with an eigenvalue solver from Armadillo library (21 s for N = 2500). Moreover, we study the impact of varying oscillator potential strengths, ω_r, on the ground state of two-electron system, showing that the eigenvalues, hence energy, increases for higher ω_r.

1 Introduction

Many physical problems, both in classical and quantum mechanics, mathematically come down to finding the eigenvalues associated with a particular set of linear equations. The process of solving such a problem requires then, first and foremost, a proper scaling of a differential equation, as well as rewriting it in the form of matrix equation. Thereafter, the matrix has to be diagonalized through a number of similarity transformations, so that the

eigenvalues can be found as entries on the main diagonal of the resulting matrix. The method of diagonalization that we will employ in the following project is called Jacobi algorithm (and is thoroughly explained in the Method section).

We will tackle three different examples of eigenvalue problems - the buckling beam, one electron in the three-dimensional quantum harmonic oscillator potential and, finally, two electrons in the three-dimensional harmonic oscillator well, interacting via Coulomb repulsion. All of them can be mathematically represented as systems of linear equations, involving a tridiagonal Toeplitz matrix. This matrix has analytical eigenpairs, making it possible to compare our analytical solution to the numerical one (see Results and discussion).

Additionally, some unit tests will be developed and implemented into the program in order to verify the mathematical bahaviour of the algorithm. Moreover, we will discuss the efficiency of Jacobi's method in juxtapostion to the eigenvalue solver from Armadillo library (precisely eig_sym function).

The main program (project2_main.cpp) is written in C++ and together with the Python code containing all data and generating the plots in this document (project2_plots.py), available in the following Github repository: github.com/jowborowska/CompPhysics

2 Methods

2.1 Mathematical reformulation of equations to solve

In the following subsection we will focus on introducing the particular physical cases (classical buckling beam and quantum harmonic oscillator) with underlying mathematical aspect. Thereafter, the equations will be scaled, discretized and rewritten in the form of linear set of equations, involving a tridiagonal Toeplitz matrix.

The buckling beam problem

First, we will consider a classical wave function problem in one dimension - the buckling beam problem, associated with the following differential equation

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

where γ is a constant dependent on the rigidity of the beam, $u(x)$ is the vertical displacement of the beam and F is a force applied at the end of the beam in the direction towards the origin. Since the beam has length L, we have $x \in [0, L]$. We apply Dirichlet boundary conditions, so that at the ends of the beam the vertical displacement in the y-direction is equal zero, $u(0) = u(L) = 0$.

Thereafter, we scale the equation by introducing a dimensional variable $\rho = x/L$ (then $\rho \in [0, 1]$),

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

where we set $\lambda = FL^2/\gamma$. Moreover, we implement the approximation to the second derivative resulting in

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

where h is a stepsize.

We want to discretize the equation above in order to solve it numerically. We have seen that $\rho \in [0,1]$, so that $\rho_{min} = \rho_0 = 0$ and $\rho_{mx} = \rho_N = 0$, where N is a number of mesh points used. Then, the value of ρ at point i is $\rho_i = \rho_0 + ih$, for $i = 1, 2, ..., N$, whereas $u(\rho_i) = u_i$, $u(\rho_i + h) = u_{i+1}$ and $u(\rho_i - h) = u_{i-1}$. The discretized version of equation is therefore

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

with the stepsize defined as

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

Furthermore, this can be represented in the form of matrix equation, as

$$\begin{bmatrix} \frac{2}{h^2} & -\frac{1}{h^2} & 0 & 0 & \dots & 0 & 0 \\ -\frac{1}{h^2} & \frac{2}{h^2} & -\frac{1}{h^2} & 0 & \dots & 0 & 0 \\ 0 & -\frac{1}{h^2} & \frac{2}{h^2} & -\frac{1}{h^2} & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & \dots & \dots & -\frac{1}{h^2} & \frac{2}{h^2} & -\frac{1}{h^2} \\ 0 & \dots & \dots & \dots & \dots & -\frac{1}{h^2} & \frac{2}{h^2} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ \dots \\ u_{N-2} \\ u_{N-1} \end{bmatrix} = \lambda \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ \dots \\ u_{N-2} \\ u_{N-1} \end{bmatrix},$$

where the matrix is a symmetric tridiagonal matrix, having $2/h^2$ as entries on the main diagonal, $-1/h^2$ as all elements on the diagnals above and below the main one and zeros elsewhere.

One electron in a 3D harmonic oscillator potential

The second case is an electron moving in a three-dimensional harmonic oscillator potential. Assuming spherical symmetry, the radial part of Schroedinger's equation reads

$$-\frac{\hbar^2}{2m}\frac{d^2}{dr^2}u(r) + \left(V(r) + \frac{l(l+1)}{r^2}\frac{\hbar^2}{2m}\right)u(r) = Eu(r),$$

where we have already substituted $R(r) = u(r)/r$. Here $V(r) = (1/2)m\omega^2 r^2$ is the harmonic oscillator potential, l is the angular momentum quantum number and E is the energy of the harmonic oscillator in three dimensions. We have spherical coordinates, so that $r \in [0, \infty)$ and the boundary conditions again become $u(0) = u(\infty) = 0$.

Next, we can introduce a variable $\rho = r/\alpha$ (where α is a constant with dimension length, so that ρ is dimensionless) and obtain

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

where $V(\rho) = (1/2)m\omega^2\alpha^2\rho^2$. Setting $l = 0$, multiplying both sides with $2m\alpha^2/\hbar^2$ and fixing the introduced constant as $\alpha = \sqrt{\hbar/m\omega}$ gives

$$-\frac{d^2}{d\rho^2}u(\rho) + \rho^2 u(\rho) = \frac{2m\alpha^2}{\hbar^2}Eu(\rho) = \lambda u(\rho),$$

where $\lambda = 2m\alpha^2 E/\hbar^2 = 2E/(\hbar\omega)$.

The discretization is performed in the exact same way as in the case of a buckling beam

(definitions of ρ_i, $u(\rho_i)$, h, etc.). Now, however, we have $\rho \in [0, \infty)$, so that $\rho_{min} = \rho_0 = 0$ and $\rho_{max} = \rho_N = \infty$, which clearly has to be represented numerically by some big number. We use again the same approximation to the second derivative and get the following dicretized version of the Schroedinger equation

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

where $V_i = \rho_i^2$ is the harmonic oscillator potential. In the form of matrix equation we have

$$\begin{bmatrix} \frac{2}{h^2} + V_1 & -\frac{1}{h^2} & 0 & 0 & \dots & 0 & 0 \\ -\frac{1}{h^2} & \frac{2}{h^2} + V_2 & -\frac{1}{h^2} & 0 & \dots & 0 & 0 \\ 0 & -\frac{1}{h^2} & \frac{2}{h^2} + V_3 & -\frac{1}{h^2} & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & \dots & \dots & -\frac{1}{h^2} & \frac{2}{h^2} + V_{N-2} & -\frac{1}{h^2} \\ 0 & \dots & \dots & \dots & \dots & -\frac{1}{h^2} & \frac{2}{h^2} + V_{N-1} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ \dots \\ u_{N-2} \\ u_{N-1} \end{bmatrix} = \lambda \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ \dots \\ u_{N-2} \\ u_{N-1} \end{bmatrix},$$

which again involves a tridiagonal symmetric matrix, with $2/h^2 + V_i = 2/h^2 + \rho_i^2$ ($i = 1, 2, .., N-1$) as elements on the main diagonal, $-1/h^2$ on the diagonals closest to the main one and zeros otherwise.

Two electrons with Coulomb interaction in a 3D harmonic oscillator potential

The last case involves two electrons in a three-dimensional harmonic oscillator well. Firstly, with no repulsive Coulomb interaction between the electrons, the radial Schroedinger equation reads

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

where we have introduced the relative coordinate, $\mathbf{r} = \mathbf{r_1} - \mathbf{r_2}$ and the coordinate of the center of mass, $\mathbf{R} = (\mathbf{r_1} + \mathbf{r_2})/2$, as well as $E^{(2)} = E_r + E_R$, denoting two-electron energy (sum of the relative energy and center-of-mass energy). We can further separate the equations for r and R via the ansatz for the wave function $u(r, R) = \psi(r) + \phi(R)$, which gives

$$\left(-\frac{\hbar^2}{m}\frac{d^2}{dr^2} + \frac{1}{4}m\omega^2 r^2\right) \psi(r) = E_r \psi(r)$$

and

$$\left(m\omega^2 R^2 - \frac{\hbar^2}{4m}\frac{d^2}{dR^2}\right) \phi(R) = E_R \phi(R).$$

Thereafter, we add the repulsive Coulomb interaction between two electrons to the r-dependent Schroedinger equation and get

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

where $\beta e^2 = 1.44\,\text{eVnm}$.

As in the single-electron case, we introduce a dimensionless variable $\rho = r/\alpha$ and repeat the same approach, resulting in

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

Moreover, we define a new *frequency* as $\omega_r = m\omega\alpha^2/(2\hbar)$ and require constant α to be equal $\alpha = \hbar^2/(m\beta e^2)$, so that the equation takes the following form

$$-\frac{d^2}{d\rho^2}\psi(\rho) + \omega_r^2\rho^2\psi(\rho) + \frac{1}{\rho} = \frac{m\alpha^2}{\hbar^2}E_r\psi(\rho) = \lambda\psi(\rho),$$

with $\lambda = m\alpha^2 E/\hbar^2 = \hbar^2 E/(m\beta^2 e^4)$.

Using the same method of discretizetion we obtain

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

which in the matrix-equation form is

$$\begin{bmatrix} \frac{2}{h^2} + W_1 & -\frac{1}{h^2} & 0 & 0 & \dots & 0 & 0 \\ -\frac{1}{h^2} & \frac{2}{h^2} + W_2 & -\frac{1}{h^2} & 0 & \dots & 0 & 0 \\ 0 & -\frac{1}{h^2} & \frac{2}{h^2} + W_3 & -\frac{1}{h^2} & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & \dots & \dots & -\frac{1}{h^2} & \frac{2}{h^2} + W_{N-2} & -\frac{1}{h^2} \\ 0 & \dots & \dots & \dots & \dots & -\frac{1}{h^2} & \frac{2}{h^2} + W_{N-1} \end{bmatrix} \begin{bmatrix} \psi_1 \\ \psi_2 \\ \psi_3 \\ \dots \\ \psi_{N-2} \\ \psi_{N-1} \end{bmatrix} = \lambda \begin{bmatrix} \psi_1 \\ \psi_2 \\ \psi_3 \\ \dots \\ \psi_{N-2} \\ \psi_{N-1} \end{bmatrix},$$

where $W_i = \omega_r^2\rho_i^2 + 1/\rho_i$ $(i = 1, 2, ..., N-1)$. Again, the matrix takes the same form (just entries on the main diagonal are modified).

2.2 Eigenvalue solver based on Jacobi's method

All the three described cases portray eigenvalue problems and are on the form

$$\mathbf{A}\mathbf{v} = \lambda\mathbf{v},$$

where λ are the eigenvalues, $\mathbf{v}$ are corresponding eigenvectors and $\mathbf{A}$ is a tridiagonal symmetric matrix of dimensionality $N-1$ (since we fix the endpoints with indices 0 and N, applying boundary conditions). In order to find the spectrum of the matrix, $\lambda(\mathbf{A})$, we need to perform a number of similarity transformations to diagonalize the original matrix $\mathbf{A}$. Thereafter, the set of eigenvalues can be found as the entries on the main diagonal of resulting matrix,

$$\mathbf{A_{diag}} = \begin{bmatrix} \lambda_1 & 0 & 0 & 0 & \dots & 0 & 0 \\ 0 & \lambda_2 & 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & \lambda_3 & 0 & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & \dots & \dots & 0 & \lambda_{N-2} & 0 \\ 0 & \dots & \dots & \dots & \dots & 0 & \lambda_{N-1} \end{bmatrix}.$$

Resulting matrix has the same eigenvalues as the original one, but the eigenvectors are in general different, so that they also need to be transformed throughout the process.

The algorithm that we will employ in order to diagonalize the matrix $\mathbf{A}$ and arrive on the form $\mathbf{A_{diag}}$ is called Jacobi's method. We will be following Golub and Van Loan in the subsequent description of the algorithm (supporting some parts with the *Lecture Notes*).

The idea behind Jacobi's method is to iteratively reduce the Forbenius norm of the off-diagonal elements,

$$\text{off}(A) = \sqrt{\sum_{i=1}^{N-1}\sum_{\substack{j=1\\ j\neq i}}^{N-1} a_{ij}^2},$$

to the point when the off-diagonal elements are smaller than some tolerance value, δ, and can be approximated as zero (so that we are left with non-zero elements only on the main diagonal - eigenvalues). In order to do that, we will apply Jacobi rotations on the form

$$\mathbf{J}(p,q,\theta) = \begin{bmatrix} 1 & \dots & 0 & \dots & 0 & \dots & 0 \\ \vdots & \ddots & \vdots & & \vdots & & \vdots \\ 0 & \dots & cos(\theta) & \dots & sin(\theta) & \dots & 0 \\ \vdots & & \vdots & \ddots & \vdots & & \vdots \\ 0 & \dots & -sin(\theta) & \dots & cos(\theta) & \dots & 0 \\ \vdots & & \vdots & & \vdots & \ddots & \vdots \\ 0 & \dots & 0 & \dots & 0 & \dots & 1 \end{bmatrix},$$

where p and q correspond to indices of the matrix, so that $J_{pp} = J_{qq} = cos(\theta)$, $J_{pq} = sin(\theta)$ and $J_{qp} = -sin(\theta)$. The first step in the Jacobi method is then choosing an index pair (p,q) that satisfies $1 \leq p < q \leq N-1$. In order to maximize the reduction of off(A), we want to choose (p,q), so that a_{pq}^2 is maximal. Thereafter, we have to compute $cos(\theta)$ and $sin(\theta)$, such that

$$\begin{bmatrix} a'_{pp} & 0 \\ 0 & a'_{qq} \end{bmatrix} = \begin{bmatrix} cos(\theta) & sin(\theta) \\ -sin(\theta) & cos(\theta) \end{bmatrix}^T \begin{bmatrix} a_{pp} & a_{pq} \\ a_{qp} & a_{qq} \end{bmatrix} \begin{bmatrix} cos(\theta) & sin(\theta) \\ -sin(\theta) & cos(\theta) \end{bmatrix},$$

so that we can subsequently overwrite matrix $\mathbf{A}$ with (closer to being diagonalized) matrix $\mathbf{A}'$, where $\mathbf{A}' = \mathbf{J}(p,q,\theta)^T\mathbf{A}\mathbf{J}(p,q,\theta)$. From the matrix-equation above we obtain

$$\begin{aligned} a'_{pp} &= cos^2(\theta)a_{pp} - cos(\theta)sin(\theta)a_{qp} - cos(\theta)sin(\theta)a_{pq} + sin^2(\theta)a_{qq} \\ 0 = a'_{pq} &= cos^2(\theta)a_{pq} + cos(\theta)sin(\theta)a_{pp} - cos(\theta)sin(\theta)a_{qq} - sin^2(\theta)a_{qp} \\ 0 = a'_{qp} &= cos^2(\theta)a_{qp} + cos(\theta)sin(\theta)a_{pp} - cos(\theta)sin(\theta)a_{qp} - sin^2(\theta)a_{pp} \\ a'_{qq} &= sin^2(\theta)a_{pp} + cos(\theta)sin(\theta)a_{qp} + cos(\theta)sin(\theta)a_{pq} + cos^2(\theta)a_{qq} \end{aligned}.$$

Taking the second equation and using the fact that $\mathbf{A}$ matrix is symmetric, so that $a_{pq} = a_{qp}$ results in

$$0 = a'_{pq} = a_{pq}\left[cos^2(\theta) - sin^2(\theta)\right] + (a_{pp} - a_{qq})cos(\theta)sin(\theta),$$

$$\Rightarrow cot(2\theta) = \frac{cot(\theta) - tan(\theta)}{2} = \tau = \frac{a_{qq} - a_{pp}}{2a_{pq}} \quad \text{and} \quad tan(\theta) = t = \frac{sin(\theta)}{cos(\theta)}.$$

We conclude that $tan(\theta) = t$ solves the quadratic equation

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

$$\Rightarrow t_{min} = \begin{cases} 1/(\tau + \sqrt{1+\tau^2}) \text{ if } \tau \geq 0, \\ 1/(\tau - \sqrt{1+\tau^2}) \text{ if } \tau < 0. \end{cases}$$

Choosing the smaller of the roots implies that the rotation angle satisfies $|\theta| \leq \pi/4$, which minimizes $cos(\theta)$ and the difference between $\mathbf{A}$ and updated matrix $\mathbf{A}'$ (see more about that in Golub & Van Loan). We finally obtain

$$cos(\theta) = 1/\sqrt{1 + t_{min}^2},$$
$$sin(\theta) = t_{min}\, cos(\theta).$$

Now, we can apply the Jacobi rotation to the matrix $\mathbf{A}$, as well as to the matrix containing eigenvectors associated with the set of eigenvalues (starting off with identity matrix of dimension $N-1$), $\mathbf{V}$,

$$\mathbf{A}' = \mathbf{J}(p,q,\theta)^T \mathbf{A}\mathbf{J}(p,q,\theta),$$
$$\mathbf{V}' = \mathbf{V}\mathbf{J}(p,q,\theta).$$

We repeat the above process until we reach the approximated convergence to zero, i.e. until the absolute value of the largest off-diagonal element of matrix $\mathbf{A}$ will be smaller than some chosen tolerance-value, $|a_{pq}| < \delta$ (in this project we have chosen $\delta = 10^{-8}$) and we (approximately) get the matrix $\mathbf{A_{diag}}$.

In summary:
while $|\mathbf{a_{pq}}| > \delta$
- choose (p,q), so that $|a_{pq}| = \max_{i \neq j} |a_{ij}|$,
- compute $cos(\theta)$ and $sin(\theta)$,
- apply Jacobi rotation to $\mathbf{A}$ matrix and eigenvectors, $\mathbf{V}$.

2.3 Numerical implementation

The main program is written in C++ and consists of several functions. Firstly, we have a function that initializes necessery vectors and matrices, creating:
- a vector of evenly spaced ρ values, from ρ_1 to ρ_{N-1} (without the endpoints, as these are fixed by boundary conditions),
- a square matrix of eigenvectors with dimension $N-1$ (this is an identity matrix, so that each row corrseponds to one eigenvector and they all are orthogonal),
- tridiagonal symmetric matrix $\mathbf{A}$, with elements on the main diagonal dependent on the physical case we consider (buckling beam, one- or two electrons in the harmonic oscillator potential).

Furthermore, there are functions performing Jacobi algorithm, extracting eigenpairs (eigenvalues and corresponding eigenvectors) after diagonalization and writing data to an output file (that can be subsequently used for further analysis and generating plots with a Python program). We have also implemented a function based on the eigenvalue solver from Armadillo library (`eig_sym`) in order to compare its computational efficiency with our method.

Jacobi rotations

Most of the steps in Jacobi's method seem quite straightforward to implement numerically. However, the similarity transformation with Jacobi rotation matrix, $\mathbf{A}' = \mathbf{J}(p,q,\theta)^T \mathbf{A}\mathbf{J}(p,q,\theta)$, and updating eigenvectors, $\mathbf{V}' = \mathbf{V}\mathbf{J}(p,q,\theta)$, requires some deeper analysis. We will look at

a three-dimensional example of eigenvectors rotation,

$$\begin{bmatrix} v_{11} & v_{12} & v_{13} \\ v_{21} & v_{22} & v_{23} \\ v_{31} & v_{32} & v_{33} \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\theta) & sin(\theta) \\ 0 & -sin(\theta) & cos(\theta) \end{bmatrix} = \begin{bmatrix} v_{11} & cos(\theta)v_{12} - sin(\theta)v_{13} & sin(\theta)v_{12} + cos(\theta)v_{13} \\ v_{21} & cos(\theta)v_{22} - sin(\theta)v_{23} & sin(\theta)v_{22} + cos(\theta)v_{23} \\ v_{31} & cos(\theta)v_{32} - sin(\theta)v_{33} & sin(\theta)v_{32} + cos(\theta)v_{33} \end{bmatrix}.$$

Here we have $p = 2$ and $q = 3$, so that in general we update the elements as

$$\begin{aligned} v_{ip} &= cos(\theta)v_{ip} - sin(\theta)v_{iq} \\ v_{iq} &= sin(\theta)v_{ip} + cos(\theta)v_{iq} \end{aligned}.$$

Similar analysis for $\mathbf{J}(p,q,\theta)^T\mathbf{A}\mathbf{J}(p,q,\theta)$ has been performed by Hjorth-Jensen (2015), p.217-218. Based on that, the generalized algorithm for updating a-elements becomes

$$\begin{aligned} &\text{If} \quad i \neq p \quad \text{and} \quad i \neq q \\ &\qquad a_{ip} = a_{pi} = a_{ip}cos(\theta) - a_{iq}sin(\theta) \\ &\qquad a_{iq} = a_{qi} = a_{iq}cos(\theta) + a_{ip}sin(\theta) \\ &\quad \text{If} \quad i = p \quad \text{or} \quad i = q \\ &\qquad\qquad a_{pp} = a_{pp}cos^2(\theta) - 2a_{pq}cos(\theta)sin(\theta) + a_{qq}sin^2(\theta) \\ &\qquad\qquad a_{qq} = a_{pp}sin^2(\theta) + 2a_{pq}cos(\theta)sin(\theta) + a_{qq}cos^2(\theta) \\ &\qquad\qquad a_{pq} = 0 \\ &\qquad\qquad a_{qp} = 0 \end{aligned}.$$

Unit tests

In order to test some mathematical properties of the algorithm, we can implement unit tests. For instance, we know that an orthogonal transformation preserves the orthogonality of the obtained eigenvectors (Jacobi rotation is a real othogonal matrix). We consider a basis of vectors $\mathbf{v_i}$,

$$\mathbf{v}_i = \begin{bmatrix} v_{i1} \\ \cdots \\ \cdots \\ v_{in} \end{bmatrix},$$

and assume that the basis is orthogonal, that is

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

The eigenvector after an orthogonal transformation becomes $\mathbf{w_i} = \mathbf{J}\mathbf{v_i}$. Moreover, a transpose of an orthogonal operator is its own inverse, $J^TJ = I \Rightarrow J^T = J^{-1}$, so that

$$\mathbf{w}_j^T\mathbf{w}_i = (\mathbf{J}\mathbf{v}_j)^T(\mathbf{J}\mathbf{v}_i) = \mathbf{v}_j^T\mathbf{J}^T\mathbf{J}\mathbf{v}_i = \mathbf{v}_j^T\mathbf{J}^{-1}\mathbf{J}\mathbf{v}_i = \mathbf{v}_j^T\mathbf{v}_i = \delta_{ij}.$$

We can then devise a test that checks if the orthogonality is preserved after a number of Jacobi rotations.

Another test, that can be implemented is creating a small, simple matrix and checking if the function searching for the largest off-diagonal entry returns the correct element and pair of indices (p, q). Moreover, f.ex. in case of the buckling beam problem, we can compare three first eigenvalues obtained through Jacobi's method with their analytical form.

Two latter tests have been implemented into the program (using Catch, C++ Automated Test Cases in a Header).

3 Results and discussion

All the implemented tests are passed by the program, so we can focus on the analysis of results. The two following subsections, discussing the convergence of Jacobi's algorithm and its efficiency, are based on solving the buckling beam problem. Thereafter, we will show the results for each case of interest separately, comparing some solutions to their analytical form.

3.1 Convergence of Jacobi's method

The program counts number of similiarity transformations (Jacobi rotations) needed in order to diagonalize the matrix (to the point when absolute values of all off-diagonal elements are smaller than $\delta = 10^{-8}$). We compile and run the program for matrices with various dimensions in order to inspect the dependence between these two quantities, which is shown on Figure 1. Here, dots indicate the exact values, read off from the Terminal window, while the curve is a best-fit approximation, giving $1.8N^2$ Jacobi rotations for a square matrix with dimension N (more precisely, $N - 1$). This behaviour is in agreement with Golub & Van Loan's analysis, as they have predicted quadratic convergence.

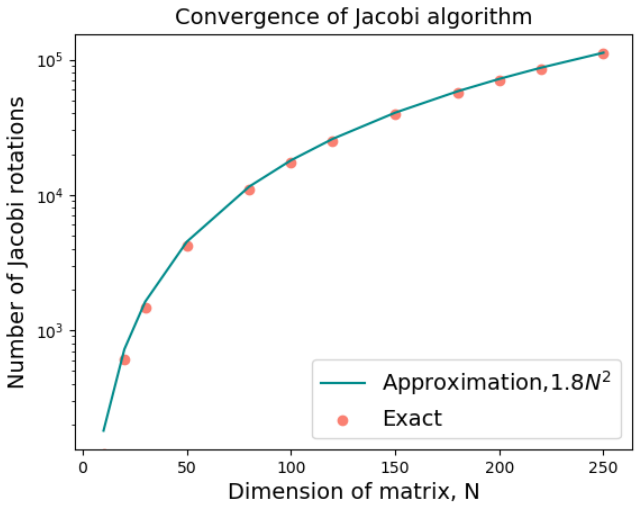

Figure 1: The number of Jacobi rotations (in logarithmic scale) needed in order to diagonalize the matrix of dimension N. Dots indicate the exact number of similarity transformations performed on the matrix with corresponding dimensionality, while a solid line is a best-fit curve, that approximates the dependence between these two quantities, $1.8N^2$.

3.2 Efficiency compared to eigenvalue solver from Armadillo

The program measures also CPU time needed in order to perform algorithms. We have compiled and run the program (for various dimensions, N, of matrix to diagonalize) using an eigenvalue solver from Armadillo library, as well as our solver, based on Jacobi's method. CPU times associated with computations are plotted against N, as shown on Figure 2. We have tested Jacobi algorithm for various sizes of matrices, $N \in [10, 250]$. The CPU time varies from 0.002 s to 88 s in that range (however, mark that very small values of CPU time are always associated with relatively big uncertainties and change from run to run of the

program). For Armadillo's eig_sym function, we have used $N \in [10, 2500]$. In this case, the CPU time varies from very small values (of order 10^{-4} s) to approximately 21 s for $N = 2500$. The Jacobi algorithm is then much slower and much less efficient than the eigenvalue solver from Armadillo.

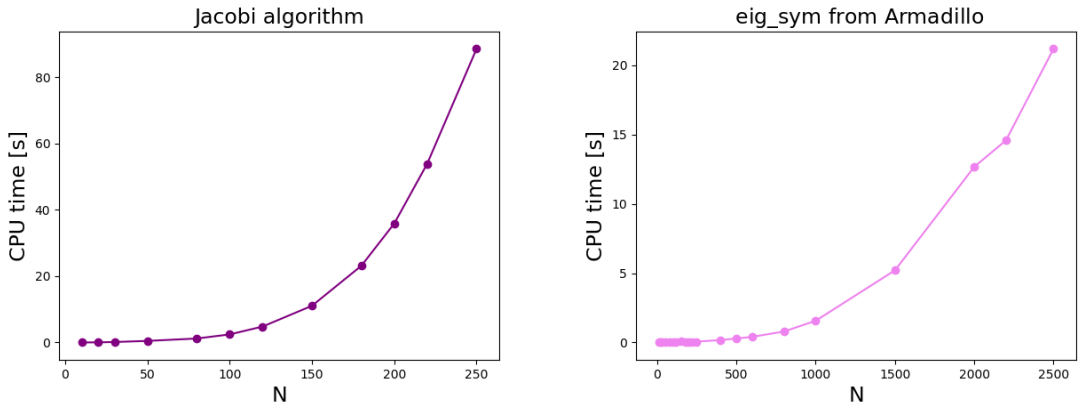

Figure 2: CPU time (in seconds) needed in order to diagonalize the matrix of dimension N with Jacobi's method (left plot) and CPU time needed to perform eigen decomposition of N-dimensional matrix, using eig_sym function from Armadillo library (right-hand side). Mark that we use $N \in [10, 250]$ for Jacobi's method and $N \in [10, 2500]$ for Armadillo function, which still is much faster.

3.3 Comparison with analytical results

For the buckling beam problem, results produced by Jacobi algorithm are in a very good agreement with the analytical results,

$$\lambda_j = \frac{2}{h^2}\left[1 - cos\left(\frac{j\pi}{N+1}\right)\right],$$

having an error (absolute difference between an analytical and numerical eigenvalue) of order 10^{-14} for matrix with $N = 20$. The error increases for matrices of larger dimensions (for $N = 100$ it is of order 10^{-10}), but still remains very small, so the algorithm is very accurate.

In the case of one electron in a 3D harmonic oscillator potential, we have defined (in the Method section) that $\lambda = 2E/(\hbar\omega)$. The energy, E, depends on the principal quantum number, n, and angular momentum quantum number, l, so that $E_{nl} = \hbar\omega(2n + l + 3/2)$. We have $n = 0, 1, 2, ..$ and in this project l is set to zero, $l = 0$. Then, the exact eigenvalues are $\lambda_n = 4n + 3$, so that $\lambda_0 = 3$, $\lambda_1 = 7$, $\lambda_2 = 11$ and so on. We want to reproduce these values numerically, choosing a correct number of mesh points, N, and approximation to ρ_{max}. Different combinations of these parameters, together with resulting eigenvalue-approximations, can be found in Table 1. It is clear that all the approximations are in quite good agreement, but $\rho_{max} = 7$, $N = 300$ seems to be the best one (giving $\lambda_0 = 2.9998$, $\lambda_1 = 6.9992$ and $\lambda_2 = 10.9979$). Even better accuracy could probably be achieved by decreasing the tolerance value, δ.

ρ_{max}	N	$\lambda_0, \lambda_1, \lambda_2$
5	100	2.9992, 6.9961, 10.9907
5	200	2.9998, 6.9990, 10.9978
7	100	2.9985, 6.9923, 10.9813
7	200	2.9996, 6.9981, 10.9953
7	300	2.9998, 6.9992, 10.9979
9	200	2.9994, 6.9968, 10.9923
9	300	2.9997, 6.9986, 10.9966
10	200	2.9992, 6.9961, 10.9905
10	300	2.9997, 6.9983, 10.9958
10	400	2.9998, 6.9990, 10.9976

Table 1: Table representing the numerical approximation to the first three eigenvalues, $\lambda_0, \lambda_1, \lambda_2$, for different matrix-dimensionalities, N, and values of ρ_{max}. The analytical solutions are $\lambda_0 = 3$, $\lambda_1 = 7$, $\lambda_2 = 11$, so that we can see that all the approximations are in quite good agreement, but $\rho_{max} = 7$, $N = 300$ seems to be the best one.

3.4 Two electrons in a 3D harmonic oscillator potential - ω_r depenence

In the case of two interacting electrons in a 3D harmonic oscillator well, we want to inspect how eigenvalues depend on the introduced *frequency*, ω_r. We will study the ground state, $n = 0$, with $l = 0$, using $\rho_{max} = 10$ and $N = 100$ (as we have seen that accuracy is similar, regardless of any small changes of these parameters). As can be seen on Figure 3, λ_0 increases for bigger values of ω_r. Since $\lambda_0 = E_0\hbar^2/(m\beta^2e^4)$, we can conclude that energy of the ground state, E_0, increases for higher ω_r (other factors in the expression are constant here).

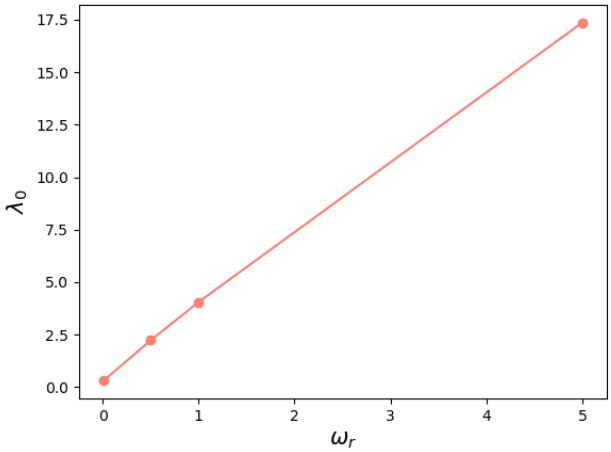

Figure 3: Eigenvalues associated with the ground state, $\lambda_0 = E_0\hbar^2/(m\beta^2e^4)$, for four different values of *frequency*, $\omega_r = 0.01$, $\omega_r = 0.5$, $\omega_r = 1$ and $\omega_r = 5$. Parameter ω_r reflects the strength of oscillator potential.

4 Conclusions

As we have seen, many different physical problems, both in classical and qunatum mechanics, come down to finding eigenvalues of a properly scaled and discretized set of linear equations. In some cases, the equations have analytical solutions, making it possible to verify the accuracy of a numerical algorithm we use. However, for more complicated systems, with interactions between particles and other additional terms, numerical solution may be the only obtainable one (so it is certainly worth to develop numerical solvers).

Jacobi's method turned out to be quite accurate, even for matrices with smaller dimensions. Here, the choice of convergence-tolearnce parameter, δ, have influenced the accuracy of approximation to a fully diagonalized matrix. Nevertheless, we have clearly seen that Jacobi algorithm is not computationally efficient and runs very slow compared to the eigenvalue solver from Armadillo library. We may then gain some control by implementing our own algorithm, but loose the efficiency, important especially in the case of large dimensionalities and such iterative algorithms.

BIBLIOGRAPHY

Golub, G. H. & Van Loan C. F. (2013) *Matrix Computations.* Baltimore: The Johns Hopkins University Press

Hjorth-Jensen, M. (2015) *Computational Physics - Lecture Notes.* Retrieved from: www.github.com/CompPhysics/ComputationalPhysics/blob/master/doc/Lectures/lectures2015.pdf