Final version October 3, 2018

Preprint typeset using LaTeX style emulateapj v. 12/16/11

# TWO-ELECTRON HARMONIC OSCILLATOR POTENTIAL SOLVED AS AN EIGENVALUE PROBLEM THROUGH JACOBI'S METHOD
# PROJECT 2 - FYS4150 / FYS3150

Jakob Borg, Julie Thingwall & Wanja Paulsen

*Final version October 3, 2018*

## ABSTRACT

In this project our goal is to develop algorithms to solve different eigenvalue problems using Jacobi's method. We will look at problems that can be described using a so-called tridiagonal Toeplitz matrix to represent the second derivative in a differential equation. We want to use the same algorithm to solve three different problems, so we introduce scaled equations to be able to rewrite three different physical systems into similar differential equations. Firstly we will look at the classical two point boundary problem of a buckling beam, and find the unexpected results of the three first eigenvalues to be $\lambda \in [-19988.3, -19959.3, -19911]$. This does not agree with our analytical results used to test the algorithm. Then we look at the quantum mechanical problem of particles in a spherical harmonic potential. First for the case of a single electron, which also has analytic solutions we can test against, and finds the correct eigenvalues $\lambda \in [2.998, 6.990, 10.976]$. Then for two interacting electrons we test with 4 different frequencies for the harmonic potential, which yields unexpected results. For low frequencies, $\omega = 0.01$, we find the expected eigenvalues equal to those found from Armadillos eigenvalue solver, $\lambda \in [12.036, 41.7421, 90.475]$. For higher frequencies we seem to get the same kind of error we got for the buckling beam. For all $\omega \in [0.5, 1.0, 5.0]$, all the eigenvalues we get are approximately equal to $\lambda \approx 20000$.

*Subject headings:* ODE — Quantum mechanics — Harmonic oscillator — Jacobis algorithm — Numerical errors

## 1. INTRODUCTION

The goal of a physicist is always to understand and explain the world around us. One powerful tool to do so is differential equations. They describe the rate of change in systems, and let us understand the past and predict the future. However, one caveat of differential equations is that they can be quite tricky to solve analytically. Luckily, today we can use computers and numerical methods to approximate solutions to a wide set of differential equations which are near, or even completely impossible to solve exact and analytically.

One such set of differential equations are eigenvalue problems. Eigenvalue problems are a sub-type of boundary value problems, equations where we know the solution at each end point. Such problems can be found all over nature, both in the macroscopic and microscopic world, and this is where the power of differential equations really shines through. From a mathematical perspective, all that changes are constants, scaling and units, the fundamental equations stays the same. This means if we know how to solve this type of equations for one situation, we need only rescale it and we can solve a completely different situation.

In this project we wish to solve a set of eigenvalue problems by the use of linear algebra and numerical methods. To show the versatility of a boundary value problem we will both look at a buckling beam fastened in both ends, and electrons in a harmonic oscillator potential. To do this we will implement an algorithm called Jacobi's rotation algorithm, which is an algorithm using unitary transformations to find eigenvalues. Mainly the programming will be done in C++ using the linear algebra package Armadillo (1), and Python will be used for visualization of results.

julie.thingwall@astro.uio.no
wanja.paulsen@fys.uio.no
jakobbor@student.matnat.uio.no
[1] Department of Physics, University of Oslo, P.O. Box 1048 Blindern, N-0316 Oslo, Norway

## 2. THEORETICAL BACKGROUND

### 2.1. *Unitary transformation*

An extremely important factor of unitary transformations is that they preserve orthogonality (4). Consider an orthogonal basis set of vectors $\mathbf{v_i}$, such that $\hat{\mathbf{v}}_\mathbf{j}^T \hat{\mathbf{v}}_\mathbf{i} = \delta_{ij}$, and a unitary matrix $\hat{\mathbf{U}}$, such that $\hat{\mathbf{U}}^T\hat{\mathbf{U}} = \hat{\mathbf{I}}$. A unitary transformation can be expressed as

$$\hat{\mathbf{w}}_\mathbf{i} = \hat{\mathbf{U}}\hat{\mathbf{v}}_\mathbf{i}, \tag{1}$$

where

$$\hat{\mathbf{w}}_\mathbf{j}{}^T\hat{\mathbf{w}}_\mathbf{i} = \left(\hat{\mathbf{U}}\hat{\mathbf{v}}_\mathbf{j}\right)^\mathbf{T}\hat{\mathbf{U}}\hat{\mathbf{v}}_\mathbf{i}$$
$$\hat{\mathbf{w}}_\mathbf{j}{}^T\hat{\mathbf{w}}_\mathbf{i} = \vec{\mathbf{v}}_\mathbf{j}{}^T\vec{\mathbf{U}}^T\vec{\mathbf{U}}\vec{\mathbf{v}}_\mathbf{i}$$
$$\hat{\mathbf{w}}_\mathbf{j}{}^T\hat{\mathbf{w}}_\mathbf{i} = \delta_{ij}.$$

Here one can see that the unitary transformation indeed preserves orthogonality. The reason this is so important is that a unitary transformation also preserves the eigenvalues. This means one can manipulate a matrix to fit given needs but still get the desired answers. To see this, consider the matrix equation

$$\hat{\mathbf{A}}\vec{\mathbf{x}} = \lambda\hat{\mathbf{x}} \tag{2}$$

and the unitary, or similarity transformation

$$\hat{\mathbf{B}} = \hat{\mathbf{U}}^T\hat{\mathbf{A}}\hat{\mathbf{U}}$$

If we now act on eq. (2) with $\hat{\mathbf{U}}^T$ we get

$$\hat{\mathbf{U}}^T\hat{\mathbf{A}}\hat{\mathbf{x}} = \lambda\hat{\mathbf{U}}^T\hat{\mathbf{x}}$$
$$\hat{\mathbf{U}}^T\hat{\mathbf{A}}\hat{\mathbf{U}}\hat{\mathbf{U}}^T\hat{\mathbf{x}} = \lambda\hat{\mathbf{U}}^T\hat{\mathbf{x}}$$
$$\hat{\mathbf{B}}(\hat{\mathbf{U}}^T\hat{\mathbf{x}}) = \lambda(\hat{\mathbf{U}}^T\hat{\mathbf{x}})$$

Where we have used the fact that, since $\hat{\mathbf{U}}$ is unitary, then $\hat{\mathbf{U}}\hat{\mathbf{U}}^T = \hat{\mathbf{I}}$. Here we can see that the eigenvalues never change, only the eigenfunctions, which are now $(\hat{\mathbf{U}}^T\hat{\mathbf{x}})$. This preservation of orthogonality, and thus eigenvalues is the basis for Jacobis method.

### 2.2. *Frobenius norm*

The Frobenius norm is a very useful tool in numerical linear algebra because it is preserved in orthogonal transformations, i.e it is invariant under rotations in our case (3). It can be defined such that the Frobenius norm of a matrix $\hat{\mathbf{A}}$ is

$$\left\|\hat{\mathbf{A}}\right\|_\mathbf{F} = \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{n}|a_{ij}|^2}. \tag{3}$$

### 2.3. *Jacobi algorithm for eigenvalues*

Jacobi eigenvalue algorithm is an algorithm for determining the solution of a system of linear equations. It is an iterative method based upon unitary transformations for reducing a matrix to diagonal form (3). This process is most commonly done in two steps where you first reduce the problem to a tridiagonal form, then to a diagonal form. If we want to find the eigenvalues of an arbitrary matrix $\hat{\mathbf{A}}$ by the use of the Jacobi algorithm, we perform a similarity transformation such that

$$\hat{\mathbf{B}} = \hat{\mathbf{S}}^T\hat{\mathbf{A}}\hat{\mathbf{S}}. \tag{4}$$

The matrix $\hat{\mathbf{S}}$ is an orthogonal $(n \times n)$ transformation matrix such that $\hat{\mathbf{S}}^T = \hat{\mathbf{S}}^{-1}$. It is on the form

$$\begin{pmatrix} 1 & 0 & \dots & 0 & 0 & \dots & 0 & 0 \\ 0 & 1 & \dots & 0 & 0 & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & 0 & \dots \\ 0 & 0 & \dots & \cos\theta & 0 & \dots & 0 & \sin\theta \\ 0 & 0 & \dots & 0 & 1 & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & 0 & \dots \\ 0 & 0 & \dots & 0 & 0 & \dots & 1 & 0 \\ 0 & 0 & \dots & -\sin\theta & \dots & \dots & 0 & \cos\theta \end{pmatrix}. \tag{5}$$

This matrix performs a plane rotation around an angle $\theta$ in Euclidean $n$-dimensional space (3). The angle $\theta$ in this matrix is arbitrary, to do the similarity transformation this angle should be chosen so that all off-diagonal elements in $\hat{\mathbf{B}}$ is zero. This is done by first calculating the Frobenius norm subsection 2.2 of the matrix $\hat{\mathbf{A}}$, then it can be shown that the off-diagonal elements of $\hat{\mathbf{B}}$ is (3)

$$off(\hat{\mathbf{B}}^2) = \left\|\hat{\mathbf{B}}\right\|_\mathbf{F}^2 - \sum_{i=1}^{n} b_{ii}^2 = off(\hat{\mathbf{A}})^2 - 2a_{kl}^2. \tag{6}$$

By this equation we see that the matrix we want to solve moves closer to diagonal form for each similarity transformation.

### 2.4. *Manipulation of Schroedingers equation with Harmonic Oscillator potential and couloumb interaction*

#### 2.4.1. *One electron in Harmonic Oscillator potential*

Common knowledge of quantum mechanics has taught us that the Schroedinger's equation for spherically symmetric potential $V(r)$ in three dimensions can be separated into a radial part and a part representing the colatitude and azimuthal angle when written in spherical coordinates, such that (2)

$$\psi(r,\theta,\phi) = R(r)\Theta(\theta)\Phi(\phi) = R(r)Y_{ml}(\theta,\phi) \tag{7}$$

where the angles can be grouped into the spherical harmonics, $Y_{ml}(\theta,\phi)$ (2). The spherical harmonics can be rewritten such that Schroedinger's equation is on the form

$$-\frac{\hbar}{2m}\left(\frac{1}{r^2}\frac{\mathrm{d}}{\mathrm{d}r}r^2\frac{\mathrm{d}}{\mathrm{d}r} - \frac{l(l+1)}{r^2}\right)R(r) + V(r)R(r) = ER(r) \tag{8}$$

where $r \in [0,\infty]$. The potential $V(r)$ can for example be the quantum harmonic oscillator. In that case we have that the potential is on the form $V(r) = \frac{1}{2}m\omega^2r^2$ where $m$ is the mass of the electron and $\omega$ is the oscillator frequency. The energies will then be on the form

$$E_{nl} = \hbar\omega(2n + l + \frac{2}{3}) \qquad n = 0,1,2.. \quad l = 0,1,2.. \quad . \tag{9}$$

Here $n$ is the number of the state, where $n = 0$ is the ground state and $l$ is the orbital momentum of the electron.

Lastly we substitute $R(r) = \frac{1}{r}u(r)$ in eq. (8) with the new limits $u(0) = 0$ and $u(\infty) = 0$ so that the equation

reads

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

If we know choose that $u(r)$ is the wave-function of an electron, we have defined an equation describing the movement of an electron in a harmonic oscillator potential.

#### 2.4.2. *Two electrons in Harmonic Oscillator potential and couloumb interaction*

For two electrons the problem in eq. (8) will be on the form

$$\Big(-\frac{\hbar^2}{2m}\Big(\frac{\mathrm{d}^2}{\mathrm{d}r_1^2}-\frac{\mathrm{d}^2}{\mathrm{d}r_2^2}\Big)+\frac{1}{2}k\big(r_1^2+r_2^2\big)\Big)u(r_1,r_2) = E^{(2)}u(r_1,r_2), \tag{11}$$

where $u(r_1,r_2)$ is the two-electron wave function without the couloumb interaction and $E^{(2)}$ is the two-electron energy. Without the couloumb interaction this can be written out as just a product of two single-electron wave functions. (3)

Instead of having separate coordinates for the two electrons we should introduce a relative coordinate $\mathbf{r}=\mathbf{r_1}-\mathbf{r_2}$ and a center-of-mass coordinate $\mathbf{R}=\frac{(\mathbf{r_1}-\mathbf{r_2})}{2}$. The Schroedinger equation will then be

$$\Big(-\frac{\hbar^2}{2m}\frac{\mathrm{d}^2}{\mathrm{d}r^2}-\frac{\hbar^2}{4m}\frac{\mathrm{d}^2}{\mathrm{d}R^2}+\frac{1}{4}kr^2+kR^2\Big)u(r,R) = E^{(2)}u(r,R), \tag{12}$$

where the new wave function is on the separable form $u(r,R)=\psi(r)\phi(R)$ and the new energy is $E^{(2)}=E_r+E_R$.

We can now introduce Coloumb interaction into our harmonic oscillator potential. The interaction is on the form

$$V(r_1,r_2)=\frac{\beta e^2}{|\mathbf{r_1}-\mathbf{r_2}|}=\frac{\beta e^2}{r}, \tag{13}$$

where $\beta e^2=1.44$ eVnm is a constant describing the interaction between the electrons. If we now omit the contribution from the center-of-mass wave-function $\phi(R)$ in the wave-function $u(r,R)$ and look at the $r$-dependent Schroedinger equation we have

$$\left(-\frac{\hbar^2}{m}\frac{\mathrm{d}^2}{\mathrm{d}r^2}+\frac{1}{4}kr^2+\frac{\beta e^2}{r}\right)\psi(r)=E_r\psi(r). \tag{14}$$

Because we are interested in the interaction between the two electrons in the harmonic oscillator potential, this is the form of the Schroedinger equation for two electrons we are interested in solving numerically.

## 3. METHOD

### 3.1. *Discretizing the functions*

#### 3.1.1. *A buckling beam*

The first eigenvalue problem we model and solve numerically is a buckling beam fastened in both ends with Dirichlect boundary conditions **??**[REF SECTION]. The fastened buckling beam can be represented by the differential equation

$$\gamma\frac{\mathrm{d}^2u(x)}{\mathrm{d}x^2}=-Fu(x) \tag{15}$$

where $u(x)$ is the displacement of the beam in $y$-direction and $\gamma$ is the a constant that defines the rigidity of the beam among other properties. The length is given as $x\in[0,L]$ and the force $F$ applied to the beam at $(0,L)$ is in the direction towards the mid-point of the beam.

To solve this eigenvalue problem numerically we made eq. (15) dimensionless before discretizing it. Starting by defining a dimensionless variable $\rho=\frac{x}{L}$ which $\rho\in[0,1]$ we can reorder our equation as

$$\frac{\mathrm{d}^2u(\rho)}{\mathrm{d}\rho^2}=\frac{-FL^2}{R}u(\rho)=-\lambda u(\rho) \tag{16}$$

where $R$ is a unit-fixing constant and $\lambda=\frac{FL^2}{R}$.

Moving on to discretizing the problem, we first recognize that a second derivative can be discretized by the use of Taylor expansions. To do this for a function $u(\rho)$ one would get

$$u(\rho\pm h)=u(\rho)\pm h\frac{\mathrm{d}u}{\mathrm{d}\rho}+\frac{h^2}{2!}\frac{\mathrm{d}^2u}{\mathrm{d}\rho^2}\pm\frac{h^3}{3!}\frac{\mathrm{d}^3u}{\mathrm{d}\rho^3}+\mathcal{O}\big(h^4\big), \tag{17}$$

where $h$ is the step size. In our case this this can then be rearranged for the double derivative, which yields

$$\frac{\mathrm{d}^2u}{\mathrm{d}\rho^2}=\frac{u(\rho+h)+u(\rho-h)-2u(\rho)}{h^2}+\mathcal{O}\big(h^2\big). \tag{18}$$

Defining minimum and maximum values for $\rho$ such that $\rho_{min}=0$ and $\rho_{max}=1$ to represent the step size $h$ gives us

$$h=\frac{\rho_N-\rho_0}{N}, \tag{19}$$

where $N$ is the number of mesh points. This means that the value of $\rho$ at a given point $i$ is

$$\rho_i=\rho_0+ih \qquad i=1,2,...,N. \tag{20}$$

From this equation we can then rewrite eq. (18) as

$$-\frac{u(\rho_i+h)-2u(\rho_i)+u(\rho_i-h)}{h^2}=\lambda u(\rho_i) \tag{21}$$

or in a shorter notation as

$$-\frac{u_{i+1}-2u_i+u_{i-1}}{h^2}=\lambda u_i. \tag{22}$$

The problem of a buckling beam can now be represented as an eigenvalue problem such that

$$\begin{bmatrix} d & a & 0 & 0 & ... & 0 & 0\\ a & d & a & 0 & ... & 0 & 0\\ 0 & a & d & a & 0 & ... & 0\\ ... & ... & ... & ... & ... & ... & ...\\ 0 & ... & ... & ... & a & d & a\\ 0 & ... & ... & ... & ... & a & d \end{bmatrix}\begin{bmatrix} u_1\\ u_2\\ u_3\\ ...\\ u_{N-2}\\ u_{N-1}\end{bmatrix}=\lambda\begin{bmatrix} u_1\\ u_2\\ u_3\\ ...\\ u_{N-2}\\ u_{N-1}\end{bmatrix}. \tag{23}$$

where $d = \frac{2}{h^2}$ and $a = -\frac{1}{h^2}$.

### 3.1.2. *Electron in three dimensional harmonic oscillator potential*

The second eigenvalue problem is from quantum mechanics. We will study an electron moving in a three-dimensional harmonic oscillator potential.

We again introduce a dimensionless variable $\rho = r/\alpha$ where $\alpha$ is constant and has dimension length, our harmonic oscillator potential then reads $V(\rho) = \frac{1}{2}k\alpha^2\rho^2$. We also choose $l = 0$, thus our equation has become

$$-\frac{\hbar^2}{2m\alpha^2}\frac{\mathrm{d}^2}{\mathrm{d}\rho^2}u(\rho) + \frac{k}{2}\alpha^2\rho^2 u(\rho) = Eu(\rho). \tag{24}$$

Multiplying with $2m\alpha^2/\hbar^2$ on both sides give us

$$-\frac{\mathrm{d}^2}{\mathrm{d}\rho^2}u(\rho) + \frac{mk}{\hbar^2}\alpha^4\rho^2 u(\rho) = \frac{2m\alpha^2}{\hbar^2}Eu(\rho), \tag{25}$$

fixing the constants so that $\frac{mk}{\hbar^2}\alpha^4 = 1$ or $\alpha = \left(\frac{\hbar^2}{mk}\right)^{1/4}$ means that we can define the right hand side such that the eigenvalue of the problem is

$$\lambda = \frac{2m\alpha^2}{\hbar^2}E. \tag{26}$$

The Schroedinger equation we are left with then is on the form

$$-\frac{\mathrm{d}^2}{\mathrm{d}\rho^2}u(\rho) + \rho^2 u(\rho) = \lambda u(\rho). \tag{27}$$

The first thing to notice is that this equation is, as in our first problem of a buckled beam, on the form of a second derivative. This means that we can use the same Taylor expansion approach in eq. (17), but this time we define $\rho_{min} = 0$ and $\rho_{max} = 8.0$, this means that the value of $\rho$ at a point $i$ is

$$\rho_i = \rho_0 + ih \qquad i = 1, 2, ..., N. \tag{28}$$

The step length will again be defined as $h = \frac{\rho_{max}-\rho_{min}}{N}$ where $N$ is the number of mesh points. Thus eq. (27) can be written as

$$-\frac{u(\rho_i + h) - 2u(\rho_i) + u(\rho_i - h)}{h^2} + \rho_i^2 u(\rho_i) = \lambda u(\rho_i) \tag{29}$$

or in a more compact way as

$$-\frac{u_{i+1} - 2u_i + u_{i-1}}{h^2} + \rho_i^2 u_i = \lambda u_i \tag{30}$$

where the harmonic oscillator potential is $V_i = \rho_i^2$.

We can again write this in the form of an eigenvalue problem as for the buckled beam, where the diagonal elements will be changing such that

$$d_i = \frac{2}{h^2} + V_i \tag{31}$$

and the non-diagonal elements will be constant

$$e_i = -\frac{1}{h^2}. \tag{32}$$

This means that we can rewrite our Schroedinger equation so that

$$d_i u_i + e_{i-1}u_{i-1} + e_{i+1}u_{i+1} = \lambda u_i \tag{33}$$

where $u_i$ is the unknown variable. The matrix problem of an electron in a three dimensional harmonic oscillator is then

$$\begin{bmatrix} d_0 & e_0 & 0 & 0 & ... & 0 & 0 \\ e_1 & d_1 & e_1 & 0 & ... & 0 & 0 \\ 0 & e_2 & d_2 & e_2 & 0 & ... & 0 \\ ... & ... & ... & ... & ... & ... & ... \\ 0 & ... & ... & ... & e_{N-1} & d_{N-1} & e_{N-1} \\ 0 & ... & ... & ... & ... & e_N & d_N \end{bmatrix} \begin{bmatrix} u_0 \\ u_1 \\ u_2 \\ ... \\ u_{N-1} \\ u_N \end{bmatrix} = \lambda \begin{bmatrix} u_0 \\ u_1 \\ u_2 \\ ... \\ u_{N-1} \\ u_N \end{bmatrix}. \tag{34}$$

### 3.1.3. *Two electrons in a three dimensional harmonic oscillator potential*

Our third and final problem is two electrons in a three dimensional harmonic oscillator potential with Coloumb interaction. What we are interested in solving numerically is the interaction and relative coordinate $r$ between the two electrons in the potential, so we will only discretize and solve eq. (14).

As before we introduce the dimensionless variable $\rho = r/\alpha$. Rewriting the harmonic oscillator potential $V(\rho) = \frac{1}{2}k\alpha^2\rho^2$ and

$$\left(-\frac{\mathrm{d}^2}{\mathrm{d}\rho^2} + \frac{1}{4}\frac{mk}{\hbar^2}\alpha^4\rho^2 + \frac{m\alpha\beta e^2}{\rho\hbar^2}\right)\psi(\rho) = \frac{m\alpha^2}{\hbar^2}E_r\psi(\rho). \tag{35}$$

We want this equation to end up being similar to eq. (21), this means that we need to rewrite our equation more. We introduce a new constant $\omega_r$ which will act as a oscillator frequency and a parameter describing the oscillator potential. We define it so that

$$\omega_r^2 = \frac{1}{4}\frac{mk}{\hbar^4}\alpha^4 \tag{36}$$

and fix the constant $\alpha$ such that

$$\alpha = \frac{\hbar 2}{m\beta e^2}. \tag{37}$$

We can then write the Schroedinger equation for the two-electron interaction such that

$$-\frac{\mathrm{d}^2}{\mathrm{d}\rho^2}\psi(\rho) + \omega_r^2\rho^2\psi(\rho) + \frac{1}{\rho} = \lambda\psi(\rho) \tag{38}$$

where $\lambda = m\alpha^2 E/\hbar^2$ will be the eigenvalues of the problem. We will study the problem for oscillator frequencies $\omega_r = 0.01$, $\omega_r = 0.5$, $\omega_r = 1.0$ and $\omega_r = 5.0$ for the ground state.

## 3.2. *Numerical algorithms*

Now that the problem in eq. (15) is discretized to eq. (23) we can develop an algorithm to solve it numerically. Our goal is to do a number of similarity transformations by using the Jacobi algorithm subsection 2.3 such that the matrix given in eq. (23) is a diagonal matrix containing only the eigenvalues. Firstly, the non-zero elements of eq. (5) can be written in compact form as

$$\begin{aligned} s_{kk} &= s_{ll} = \cos\theta \\ s_{kl} &= -s_{kl} = \sin\theta \\ s_{ii} &= -s_{ii} = 1 \quad i \neq k \quad i \neq l. \end{aligned}$$

But instead of implementing this to create a matrix $\hat{\mathbf{S}}$ in our algorithm we can rewrite the similarity transformation in eq. (4) as

$$\begin{aligned} b_{ii} &= a_{ii}, \quad i \neq k, \quad i \neq l \\ b_{ik} &= a_{ik}\cos\theta - a_{il}\sin\theta, \quad i \neq k, \quad i \neq l \\ b_{il} &= a_{il}\cos\theta + a_{ik}\sin\theta, \quad i \neq k, \quad i \neq l \\ b_{kk} &= a_{kk}\cos^2\theta - 2a_{kl}\cos\theta\sin\theta + a_{ll}\sin^2\theta \\ b_{ll} &= a_{ll}\cos^2\theta + 2a_{kl}\cos\theta\sin\theta + a_{kk}\sin^2\theta \\ b_{kl} &= (a_{kk} - a_{ll})\cos\theta\sin\theta + a_{kl}(\cos^2\theta - \sin^2\theta). \end{aligned}$$

This means that what we need to do is to choose the angle $\theta$ so that each non-diagonal element $b_{kl}$ in $\hat{\mathbf{B}}$ becomes zero. It will as mentioned require several similarity transformations to make all elements $b_{kl} = 0$, so using an algorithm to streamline this is essential, especially for large matrices.

One way of streamlining this process of going through all non-diagonal elements is to implement an algorithm for choosing the largest element in each similarity transformation, and eliminating this. This will reduce the number of required transformations. We also only need to go through the upper triangular part because $\hat{\mathbf{A}}$ is a symmetric matrix, this will also speed up the process. This algorithm is implemented in our program as

```
void largest_offdiagonal(uword N, mat &A,
    int &k, int &l){
    //Finds largest off-diagonal element
        value in a matrix A
    double max = 0;
    for (int i = 0; i < N; i++){
        for (int j = i+1; j < N; j++){
            double a_kl = fabs(A(i,j));
            if (a_kl > max){
            max = fabs(a_kl); k = i; l = j;
            }
        }
    }
}
```

Now that we have an algorithm for finding the largest off-diagonal elements, we need to find the ideal angle $\theta$. We start by defining that $s = \sin\theta$, $c = \cos\theta$ and thus $\tan\theta = t = s/c$. We can then define

$$\cot 2\theta = \tau = \frac{a_{ll} - a_{kk}}{2a_{kl}} \tag{39}$$

The problem here is that we have elements $a_{kl} = 0$, which means that we need to consider this in our algorithm to avoid division by zero. If these values are zero we do not need to rotate this element as it is the desired value, which means that $\theta = 0$. Thus for $a_{kl} = 0$ we have that $\cos\theta = 1$ and $\sin\theta = 0$. For all other values we start by using the relation $\cot 2\theta = \frac{1}{2}(\cot\theta - \tan\theta)$ so we can obtain a quadratic equation such that

$$t^2 + 2\tau t - 1 = 0 \quad \rightarrow \quad t = -\tau \pm \sqrt{1+\tau^2}. \tag{40}$$

From this we can obtain that $c = 1/\sqrt{1+t^2}$ and $s = tc$. From this we also see that the angle must be in the interval $|\theta| \leq \pi/4$.

Before implementing this into our algorithm we need to consider the situation where the quadratic eq. (40) contains a large value for $\tau$. A computer will then try to solve the problem $t \sim -\tau \pm \tau$ which can cause round-off errors and thus be a source of numerical uncertainty. By rewriting this equation by its' conjugate we can obtain

$$t = \frac{(-\tau \pm \sqrt{1+\tau^2})(\pm\tau \pm \sqrt{1+\tau^2})}{\pm\tau \pm \sqrt{1+\tau^2}}.$$

This is implemented as an if-else test in our algorithm so that

```
    if ( A(k,l) != 0.0){
        double t, tau;
        tau = (A(l, l)-A(k, k))/(2*A(k,l));
            //Calculate tau with max element
            indices

        if (tau >= 0.){ // t = tan(theta)
            t = 1./(tau + sqrt(1+tau*tau));
        }
        else{
            t = -1./(-tau + sqrt(1+tau*tau));
        }
        c = 1./sqrt(1 + t*t); // c = cos(
            theta)
        s = c*t; // s = sin(theta)
    }
    else {
        c = 1.0;
        s = 0.0;
    }
```

As for the symmetry transformation we do not save the values of $\hat{\mathbf{B}}$ in a new matrix, instead we update the values of $\hat{\mathbf{A}}$ to avoid unnecessary memory space. To save time we also require that $a_{kl} = 0$ in each symmetry transformation. Thus our implementation of the Jacobi algorithm for eigenvalues are as follows

```
void Jakobi_rotate(uword N, mat &A, int &l,
    int &k){
    double c, s;

    //Location of calculation of angle and
        if-else test in real program

    double a_kk, a_ll;
    a_kk = A(k,k); a_ll = A(l,l);

    A(k,k) = a_kk*c*c - 2*A(k,l)*c*s + a_ll*
        s*s;
    A(l,l) = a_ll*c*c + 2*A(k,l)*c*s + a_kk*
        s*s;
    A(k,l) = 0; A(l,k)= 0; //By definition

    double a_ik, a_il;
    for (int i = 0; i < N; i++){
        if (i != k && i != l){
         a_ik = A(i,k); a_il = A(i,l);

         A(i,k) = a_ik*c - a_il*s;
         A(i,l) = a_il*c + a_ik*s;
         A(k,i) = A(i,k);
         A(l,i) = A(i,l);
        }
    }
}
```

Everything so far will be implemented in a while-loop which will check whether or not the largest off-diagonal element is smaller than a specific value and that the number of symmetry transformations don't exceed a limit of $5N^2$.

### 3.3. *Testing our algorithms*

Before we can trust the results from our numerical algorithms we have to control that the algorithms performs as we expect. This is done through a couple of unit test functions, where we use our algorithms on small matrices where we easily control the expected results. We don't want to perform all the tests on every run of the program. Therefor we add an optional command line argument that may be used to include the call to the test functions in the main, and also to exit the main function before all the computation of our problems begin if this optional argument is exactly equal 1.

In this project we have to check our implementation of the Jacobi rotation method, as well as our algorithm for finding the largest off-diagonal element. For the Jacobi method we implement two tests, one to control the eigenvalues and one to make sure the orthogonality of the eigenvectors is maintained through the symmetric transformation.

If our tests are passed, we can continue on analyzing the results from our algorithms. We also do a test-run of the test-functions, where we on purpose implement an error in the iterations. This way we know that the test-functions picks up on the correct errors.

#### 3.3.1. *Largest off-diagonal test*

For this test we construct a $10 \times 10$ test matrix $\hat{\mathbf{T}}$ with random numbers between zero and one. We then set a specific element in the upper half of the matrix to a high value, $T_{k,l} = 200$. If our implementation of the algorithm is correct it should be able to determine the indices $k$ and $l$ of the largest element.

#### 3.3.2. *Eigenvalues from Jacobi*

In this test we control the values of the eigenvalues calculated by our algorithm against the values found using Armadillo's eigenvalue solver. The solver from Armadillo is trustworthy but slow for huge matrices, so we construct a small test matrix $\hat{\mathbf{T}} \in \mathbb{R}^{4\times4}$. First we find the expected eigenvalues using Armadillo, and then let our symmetry transformation update $\hat{\mathbf{T}}$ until the eigenvalues can be read along the main diagonal. We define the test to be successful if the absolute difference between all the eigenvalues calculated by Armadillo's solver and our algorithm is less than a set tolerance of $tol = 1 \cdot 10^{-5}$.

#### 3.3.3. *Orthogonality maintenance through symmetric rotation*

This test is done in a similar way as in subsection 3.3.2, but for an even smaller $3 \times 3$ test matrix. As the maintenance of orthogonality is a strict criteria it will generally hold for all the columns under the same rotation. We therefor only need to test a couple of vectors through a lot of rotations. We set a fixed number of rotations to be done, $iterations = 5000$, and then perform a test for orthogonality for every hundredth iteration. The test is done by requiring both that the inner product of the first two columns with them self is less than the tolerance $tol = 1 \cdot 10^{-5}$ away from exactly 1, while the inner product of the first and second column and the second and third column is less than the same tolerance. This way we control that the columns are normalized and orthogonal on each other.

## 4. RESULTS

### 4.1. *Buckling beam*

For the buckling beam problem, we know that the analytic eigenvalues should follow

$$\lambda_j = d + 2a\cos\left(\frac{j\pi}{N+1}\right), j = 1, 2...(N-1). \quad (41)$$

When we calculate the eigenvalues using our implementation of the jacobi algorithm and Armadillos eigenvalue solver, we get the results shown in table 1. These values where calculated for a $100 \times 100$-matrix. All results are run with a matrix of this size. When trying to run for a $1000 \times 1000$-matrix, we find that the program simply runs forever.

| | $\lambda_0$ | $\lambda_1$ | $\lambda_0$ |
|---|---|---|---|
| Analytical value | 0.0 | 9.67 | 38.7 |
| Jacobi's algorithm | -19988.3 | -19959.3 | -19911 |
| Armadillo solver | -19988.3 | -19959.3 | -19911 |

TABLE 1
TABLE COMPARING THE THREE FIRST EIGENVALUES FOR THE BUCKLING BEAM PROBLEM. THE FIRST ROW CONTAINS THE ANALYTICAL SOLUTION TO THE EIGEN VALUES. THE SECOND AND THIRD ROW ARE THE NUMERICAL RESULTS, WHERE THE SECOND ROW IS FROM THE JACOBI ALGORITHM AND THE THIRD ROW IS FROM THE ARMADILLO SOLVER. THERE ARE *some* DIFFERENCES BETWEEN THE ANALYTICAL AND NUMERICAL RESULTS.

Here we see that the results are a bit off, to say the least. It would seem that there might be something wrong with how we generate the matrix A initially, as this is the only part of our numerical implementation that differs from the two next quantum mechanical systems.

Another interesting result to note is that to solve this system using Jacobis method, the code runs for $t \approx 1.387$ s, while using Armadillos solver only takes $t \approx 0.045s$. Our implementation of the algorithm is thus not the most efficient. It is worth mentioning that we are always solving a tridiagonal, symmetric matrix, which one would expect to be faster than for an arbitrary matrix with all elements being non-zero. This means the Jacobi algorithm is probably not the most efficient to use for more complex matrices.

### 4.2. *One electron in 3-dimensional harmonic oscillator potential*

For one electron in a 3-dimensional harmonic oscillator potential, we can see that our solver is significantly more accurate than in the buckling beam problem. Here we get eigenvalues close to what we would expect from the analytical restuls, as seen in table 2. Again we observe

| | $\lambda_0$ | $\lambda_1$ | $\lambda_0$ |
|---|---|---|---|
| Analytical value | 3 | 7 | 11 |
| Jacobi's algorithm | 2.998 | 6.990 | 10.976 |
| Armadillo solver | 2.998 | 6.990 | 10.976 |

TABLE 2
TABLE COMPARING THE THREE FIRST EIGENVALUES FOR AN ELECTRON IN A HARMONIC OSCILLATOR POTENTIAL. THE FIRST ROW CONTAINS THE ANALYTIC SOLUTION TO THE EIGENVALUES. THE SECOND AND THIRD ROW ARE THE NUMERICAL RESULTS, WHERE THE SECOND ROW IS FROM THE JACOBI ALGORITHM AND THE THIRD ROW IS FROM THE ARMADILLO SOLVER. HERE WE CAN SEE THAT THE CALCULATED AND ANALYTIC VALUES ARE CLOSE.

that Jacobi's algorithm is slower than Armadillos solver. Armadillos solver uses $t \approx 0.021$s, while the algorithm uses $t \approx 1.222$s. It would have been interesting to compare these results to the simpler bucking beam problem, as we're now looking at a more complex tridiagonal matrix where the elements along the diagonal are no longer the same, but as the results we get from the first situation are suboptimal, to say the least, this comparision would be obsolete in our case.

### 4.3. *Two non-interacting electrons in Harmonic Oscillator potential*

| $\omega = 0.01$ | $\lambda_0$ | $\lambda_1$ | $\lambda_2$ |
|---|---|---|---|
| Jacobi's algorithm | 12.036 | 41.7421 | 90.475 |
| Armadillo solver | 12.036 | 41.7421 | 90.475 |

| $\omega = 0.5$ | $\lambda_0$ | $\lambda_1$ | $\lambda_2$ |
|---|---|---|---|
| Jacobi's algorithm | 20001.3 | 20001.3 | 20001.3 |
| Armadillo solver | 12.1114 | 41.8235 | 90.5582 |

| $\omega = 1.0$ | $\lambda_0$ | $\lambda_1$ | $\lambda_2$ |
|---|---|---|---|
| Jacobi's algorithm | 20001.9 | 20001.9 | 20001.9 |
| Armadillo solver | 12.3366 | 42.0682 | 90.8079 |

| $\omega = 5.0$ | $\lambda_0$ | $\lambda_1$ | $\lambda_2$ |
|---|---|---|---|
| Jacobi's algorithm | 20005.5 | 20005.5 | 20005.5 |
| Armadillo solver | 18.8906 | 50.0466 | 98.9188 |

TABLE 3
TABLES SHOWING THE CALCULATE EIGENVALUES FOR FOUR DIFFERENT OSCILLATOR FREQUENCIES. WE CAN SEE THAT IN THE FIRST TABLE, JACOBIS METHOD AND ARMADILLOS SOLVER YIELDS THE SAME RESULTS, BUT FOR THE OTHER THREE FREQUENCIES, JACOBIS METHOD DIVERGES.

Finally we look at how adding another electron changes the situation. In this scenario we are not only interested in the eigenvalues, but also the eigenvectors or eigenfunctions, as each eigenfunction represents a wave function for a given energy level. We also change up the potential a bit. Until now we've only looked at a potential with oscillator frequency $\omega = 1$, now we look at $\omega \in [0.01, 0.5, 1, 5]$. Here, we do not have any analytical solutions to compare with so we trust Armadillos solver, as the answers seems reasonable compared with the case of only one electron. The different eigenvalues for the different oscialltor potential can be seen in table 3

Here we see that compared to Armadillos solution, or results are way off. This seems to be a different problem than with the buckling beam, as only our solutions are wrong, and not armadillos. It would seem that somewhere in our algorithm for this specific case, a numerical operation skyrockets and overshoots the answer by several orders of magnitude. Sadly, we have not been able to identify the source of this problem.

Because of this we choose to use Armadillos solver to find the eigenfunctions for visiualization purposes. Had we written our own function with our own calculated eigenvalues we would probably not have gotten anything close to wave functions. The plots of all the scaled wave functions can be seen in fig. 1

From table 3 we see that adding another electron to the potential raises all energy levels. This manifests itself in the wavefunctions where we see that for higher eigenvalues, meaning higher energy levels, we get an increasing amount of nodes.

### 4.4. *Unit tests*

Our test runs completes successfully. The indices of the largest off-diagonal are found, the orthogonality is maintained through 5000 iterations and the eigenvalues are found to perfectly match to our desired precision $\mathcal{O}(10^{-5})$.

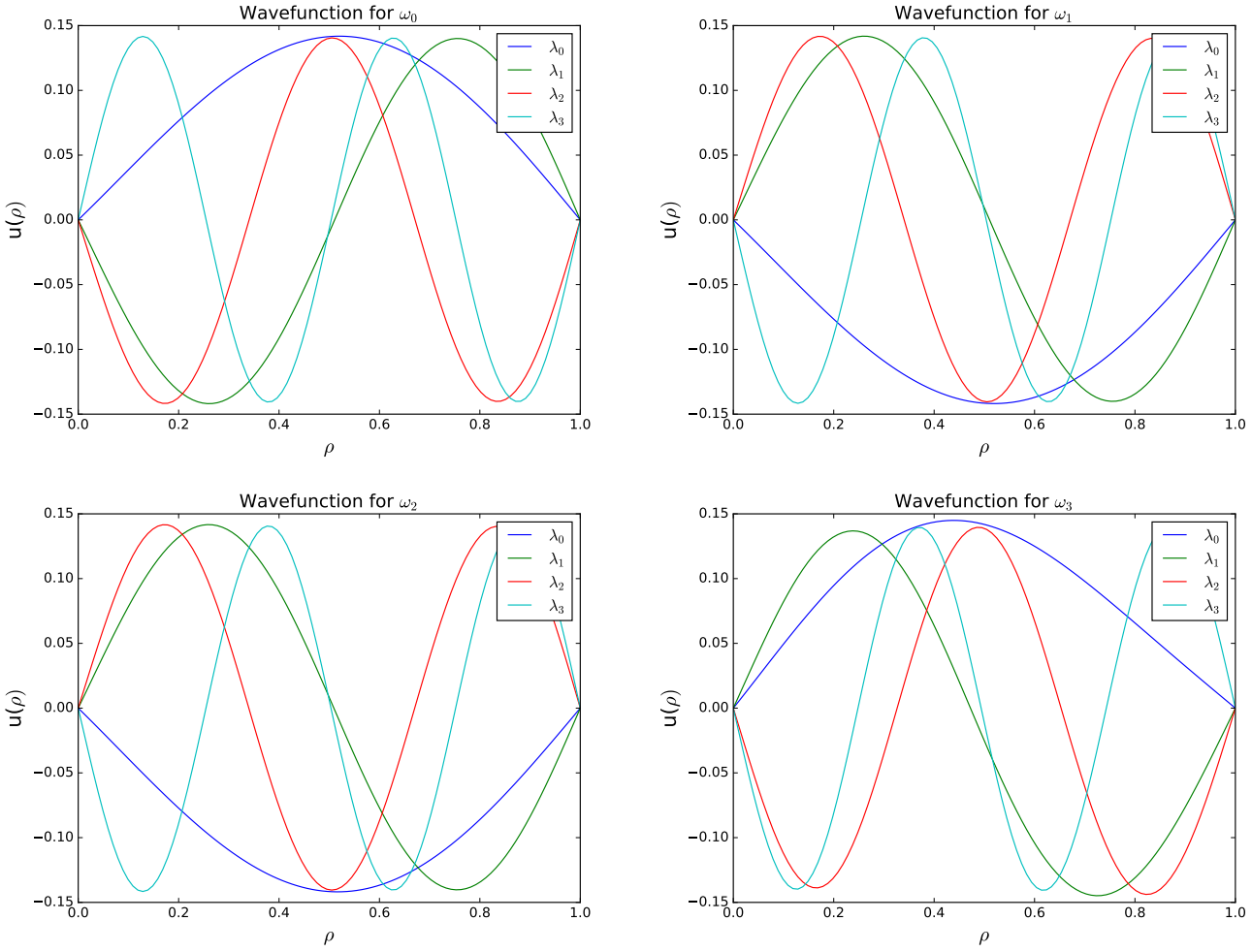


FIG. 1.— Plots showing the wave functions for the first four eigenvalues in four different harmonic oscillator potentials.

## 5. PERSPECTIVES FOR FUTURE IMPROVEMENTS

### 5.1. *Using unit test libraries*

Our implementation of test functions are functional, but not ideal. To streamline the process, and reducing the cluttering in our `main.cpp` file, we could for example use the CATCH library. CATCH may be implemented through a header file only, and contains a framework for implementing unit tests. This way we could make a separate file for all the testing through different test cases, and import the the algorithms to be tested through include statements from `main.cpp`.

## 6. CONCLUSIONS

Overall we observe that the Jacobi algorithm is not the most effective way to solve eigenvalue problems. Even for our simple initial tridiagonal matrix the implemented algorithm is slower than Armadillos eigenvaule solver. It is of course important to note that, given some of our more flimsy results, our implementation of the algorithm might not be the most effective and thus it is challenging to draw a definite conclusion on whether it is a slow algorithm or if our implementation is lacking. Overall it is definitively not possible to draw a meaningful conclusion to these results because of the several unknown errors in our algorithms.

## 7. APPENDIX: LINK TO ALL PROGRAMS

Link to all programs in Github

## REFERENCES

[1]C. SanSan and R. Curtin. Armadillo: a template based c++ library for linear algebra. Journal of Open Source Software, Vol 1., pp.26, 2016. [Online; accessed 10-September-2018].

[2]D. J. Griffiths. *Introduction to quantum mechanics*. Cambridge University Press, 2016.

[3]H.J. Morten. Computational physics - lecture notes fall 2015, 2015.

[4]D. C. Lay. *Linear algebra and its applications*. Pearson Education, 2015.