Eigenvalue problems, from the equations of a buckling beam to Schroedinger's equation for two electrons

All code can be found at:
https://github.uio.no/shafaqns/FYS4150/tree/master/project2

Shafaq Naz Sheikh
26th September 2020

ABSTRACT In this report, we look at ordinary differential equations (ODE's) that when discretized, become eigenvalue problems. The ODE's we look at are the buckling beam problem, and quantum dots, with one electron and then with two electrons. These equations represent different physical phenomena, but with some simple changes in the code for one of them, we can get an idea of the solutions of all of them. This is quite fascinating, as once the code is written for the buckling beam equation, it can be reused with some changes, to solve the other two equations. The algorithm we look at is the Jacobi method, or the Jacobi rotation method, to find the eigenvalues of the system. We then compare this method with armadillo's function in c++, eig_sym, which finds the eigenvalues and eigenvectors of the system. The results show that armadillo's function is faster than the Jacobi rotation method.

INTRODUCTION AND THEORY

The buckling beam problem

Starting off with the first equation, the buckling beam problem, which is a classical wave function problem in one dimension. This ODE is stated as

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

Here $u(x)$ is the vertical displacement of the beam in the y direction. The beam has length L, $x \in [0, L]$, and F is a force applied at (L,0) in the direction towards the origin. The parameter γ is a constant defined by properties like the rigidity of the beam. We apply the dirichlet boundary conditions and set u(0) = u(L) = 0. This is a two point boundary value problem, and since this represents some physical phenomenon, it has some units applied to it. However we would like to make this problem dimensionless so it is easier to deal with. To do this we define a dimensional variable, $\rho = x/L$, meaning that we have $\rho \in [0,1]$. By reordering the equation, we get;

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

Now this equation is dimensionless as u is a function of ρ. The equation can be discretized as

$$u'' = \frac{u(\rho + h) - 2u(\rho) + u(\rho + h)}{h^2} + O(h^2)$$

Where h is our step, next we define the minimum and maximum values for ρ, $\rho_{min} = 0\ and\ \rho_{max} = 1$. We can then define h as,

$$h = \frac{\rho_{max} - \rho_{min}}{N}$$

where ρ is defined as,

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

We can rewrite the differential equation in a more compact way,as

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

This is quite similar to the discretization we had in the first project, similar to that discretization, we can rewrite this equation in matrix, which gives the eigenvalue problem.

$$\begin{bmatrix} d & a & \dots & 0 \\ a & d & a & 0 \\ 0 & a & \ddots & a \\ 0 & \dots & a & d \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ \dots \\ u_{N-1} \end{bmatrix} = \lambda \begin{bmatrix} u_1 \\ u_2 \\ \dots \\ u_{N-1} \end{bmatrix} \qquad (1)$$

$$A\boldsymbol{u} = \lambda\boldsymbol{u}$$

We have not included the endpoints, u_0 and u_N as the solution is zero at the endpoints. The elements in the matrix are given by,

$$d = \frac{2}{h^2}\ diagonal\ elements\ , a = -\frac{1}{h^2}\ nondiagonal\ elements$$

Quantum dots, with one electron

Extending this concept to quantum mechanics, the next ODE we will look at is the quantum dots, with one electron. Here we will assume that these electrons move in a three-dimensional harmonic oscillator potential and repel each other via the static coulomb interaction. We assume spherical symmetry. We are first interested in the solution of the radial part of Schroedinger's equation for one electron. This equation reads;

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

- $V(r)$ is the harmonic oscillator potential, $\left(\frac{1}{2}\right)kr^2\ with\ k = m\omega^2$
- E is the energy of the harmonic oscillator in three dimensions
- ω is the oscillator frequency

The energies are

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

- l is the quantum number, it is the orbital momentum of the electron

We make a transformation to spherical coordinates, so $r \in [0, \infty)$. Then we substitute $R(r) = \left(\frac{1}{r}\right)u(r)$ and obtain,

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

The boundary conditions are $u(0) = 0$ and $u(\infty) = 0$, we want to make this dimensionless, we do this by introducing the dimensionless variable $\rho = \left(\frac{1}{\alpha}\right)r$ where α is a constant with dimension length. We will set $l = 0$ in this project. We get

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

Inserting $V(\rho) = \left(\frac{1}{2}\right)k\alpha^2\rho^2$ and after a few manipulations, we can rewrite Schroedinger's equation as

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

We define the minimum and maximum values for ρ , we define $\rho_{min} = 0$ and $\rho_{max} = \infty$. In practice we should try different values for ρ_{max} since we cannot set it to infinity. The discretization is similar to the one in the buckling beam equation. We define h and ρ,

$$h = \frac{\rho_{max} - \rho_{min}}{N}$$

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

The equation can be rewritten as

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

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

This leads to the same matrix eigenvalue problem as it is shown in (1), however the elements in the matrix are slightly different. In this case the matrix elements are defined as;

$$d = \frac{2}{h^2} + V_i \; diagonal \; elements \; , a = \; -\frac{1}{h^2} \; nondiagonal \; elements$$

Quantum dots, with two electrons

We will now look study two electrons in a harmonic oscillator well which also interact via a repulsive Coulomb interaction. Let us start with the single-electron equation written as

$$-\frac{\hbar^2}{2m}\frac{d^2}{dr^2}u(r) + \frac{1}{2}kr^2u(r) = E^{(1)}u(r)$$

Where $E^{(1)}$ stands for the energy with one electron only, for two electrons with no repulsive Coulomb interaction, we have the following Schroedinger's equation

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

Note that we deal with a two electron wave function $u(r_1, r_2)$ and two-electron energy $E^{(2)}$. With no interaction this can be written out as the product of two single-electron wave functions, that is we have a solution on closed form. We introduce a relative coordinate $\boldsymbol{r} = \boldsymbol{r_1} - \boldsymbol{r_2}$ and the center-of-mass coordinate $\boldsymbol{R} = 1/2(\boldsymbol{r_1} - \boldsymbol{r_2})$**.** With these new coordinates, the radial Schroedinger equation reads

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

The equation for r and R can be separated by separation of variables. We make the ansatz for the wave function $u(r, R) = \psi(r)\phi(R)$ and the energy is given by the sum of the relative energy E_r and the center-of-mass energy E_R that is

$$E^{(2)} = E_r + E_R$$

We add then a repulsive Coulomb interaction between two electrons, namely a term

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

with βe^2= 1.44 eVnm. Adding this term, the r-dependent Schroedinger equation becomes

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

This equation is not dimensionless, so to make it dimensionless we introduce the a dimensionless variable $\rho = r/\alpha$.

We arrive at the dimensionless Schroedinger's equation which is given by,

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

We treat ω_r as a parameter which reflects the strength of the oscillator potential. We will study the cases $\omega_r = 0.01, 0.5, 1\ and\ 5$ for the ground state only, that is, the lowest-lying state. This leads to the eigenvalue problem just like in (1). In this case the matrix elements are defined by,

$$d = \frac{2}{h^2} + \omega_r^2\rho^2 + \frac{1}{\rho}\ diagonal\ elements\ , a = -\frac{1}{h^2}\ nondiagonal\ elements$$

ALGORITHMS

In order to solve the two point boundary value problem, we have to find the eigenvalues of the system shown in (1). To find the eigenvalues and eigenvectors we use the Jacobi rotation method. Firstly, we assume that our matrix is real and symmetric, that is $A \in \mathbb{R}^{n\times n}$, then A has n eigenvalues, $\lambda_1, \lambda_2, \ldots \lambda_n$. If A is real and symmetric then there exists a real orthogonal matrix S, such that

$$S^T AS = diag(\lambda_1, \lambda_2, \ldots \lambda_n)$$

The strategy of the Jacobi rotation method is to perform a series of similarity transformations on the original matrix A, in order to reduce it to a diagonal form. The importance of a similarity transformation lies in the fact that the resulting matrix has the same eigenvalues, but different eigenvectors. The rotation matrix , S, is an orthogonal matrix.

$$S = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & \ldots & \ldots & \ldots & 0 \\ \ldots & 0 & 1 & \ldots & \ldots & 0 \\ 0 & \ldots & cos\theta & 1 & -sin\theta & 0 \\ 0 & \ldots & 0 & 0 & \ddots & \ldots \\ 0 & 0 & sin\theta & \ldots & cos\theta & 1 \end{bmatrix}$$

Let us call the elements of S s_{ij}.The general algorithm would be ;

1. Choose a tolerance ϵ, typically a small number like 10^{-8}.
2. Setup a while loop, and run it as long as $(abs(\max\_nondiagonal) > tolerance)$
3. Now find the non-diagonal element with the maximum absolute value, that is, the element a_{kl}
4. Compute $\tau = \frac{(a_{ll}-a_{kk})}{2a_{kl}}, tan\theta, sin\theta\ and\ cos\theta$
5. Compute the similarity transformations at the index (k,l), and obtaining the new matrix $B = S^T AS$

The quantities to be computed are $\tau, t, s\ and\ c$. these can be computed as,

$$\tau = \frac{a_{ll} - a_{kk}}{2a_{kl}}$$

Then t can be computed as $t = -\tau \pm \sqrt{1+\tau^2}$, to prevent loss of numerical precision, choose t to be the smallest solution. Then cosine and sine are easily computed as,

$$cosine = \frac{1}{\sqrt{1+t^2}}, and\ sine = t \times cosine$$

This is an iterative method, at each iteration the element $a_{kl} = a_{lk}$ will be eliminated. This is because we know the matrix is symmetric so $a_{kl} = a_{lk}$. Note however at every iteration only the kth and lth

rows and columns are affected, so we can update these column alone to save computational cost. The new matrix B can simply be computed by these equations;

$$b_{kk} = a_{kk}cos\theta^2 - 2a_{kl}cos\theta sin\theta + a_{ll}sin\theta^2$$
$$b_{ll} = a_{ll}cos\theta^2 + 2a_{kl}cos\theta sin\theta + a_{kk}sin\theta^2$$
$$b_{ik} = a_{ik}cos\theta - a_{il}sin\theta, i \neq k, i \neq l$$
$$b_{il} = a_{il}cos\theta + a_{ik}sin\theta\ , i \neq k, i \neq l$$
$$b_{kl} = 0$$

The algorithm can be written as;

Initialize matrix R as identity matrix,
Choose tolerance ϵ

While (max_nondiag > tolerance){
find the maximum nondiagonal element $|a_{kl}|$, store the indices k, l as well
if $a_{kl} \neq 0$:
compute τ,
if $\tau > 0$:
t = $1/(\tau + \sqrt{1+\tau^2})$
if $\tau < 0$:
t = $-1/(-\tau + \sqrt{1+\tau^2})$
c = $1/(\sqrt{1+t^2})$
s = $t\ c$
else:
c = 1.0
s = 0.0
$b_{kk} = a_{kk}c^2 - 2a_{kl}cs + a_{ll}s^2$
$b_{ll} = a_{ll}c^2 + 2a_{kl}cs + a_{kk}s^2$
$b_{kl} = 0$
b_{lk} = 0
For i = 1 to n:
if $i \neq k\ and\ i \neq l$:
$b_{ik} = a_{ik}c - a_{il}s$
$b_{ki} = b_{ik}$
$b_{il} = c\ a_{il} + s\ a_{ik}$
$b_{li} = b_{il}$
}

MATHEMATICAL PROPERTIES

Let us take a closer look at the matrix in (1). The matrix A is a tridiagonal Toeplitz matrix, it has the same structure as the matrix in the first project. The matrix is also symmetric.

A **symmetric** matrix, is a matrix B such that $B^T = B$ (the T denotes the transpose). Such a matrix is necessarily square. Its main diagonal entries are arbitrary, but its other entries occur in pairs, on opposite sides of the main diagonal. From the spectral theorem for symmetric matrices, we know that an $n \times n$ symmetric matrix B has n real eigenvalues. Therefore we can expect our matrix A to have n real eigenvalues.

The matrix S is a unitary matrix, this means that its transpose is its inverse. This means that $S^{-1} = S^T$, which then means that,

$$SS^{-1} = S^{-1}S = SS^T = S^TS = I$$

A unitary transformation preserves the orthogonality of the obtained eigenvectors. To see this first assume that we have a basis of orthogonal vectors v_i, that is

$$\boldsymbol{v_j^T v_i} = \delta_{ij}$$

If we have a unitary transform, $\boldsymbol{w_i} = \boldsymbol{U v_i}$ then ;

$$< \boldsymbol{w_i}, \boldsymbol{w_j} >=< \boldsymbol{U v_i}, \boldsymbol{U v_j} > = (\boldsymbol{U v_i})^T (\boldsymbol{U v_j}) = \boldsymbol{v_i^T U^T U v_j} = \boldsymbol{v_i^T} (\boldsymbol{U^T U}) \boldsymbol{v_j} = \boldsymbol{v_i^T v_j} = \delta_{ij}$$

So we can see that the orthogonality is preserved, therefore multiplying the matrix A with a unitary matrix will not change its eigenvalues.

RESULTS

The aim of this exercise was to find the eigenvalues. All the three problems present themselves as an eigenvalue problem, where the matrix A, has a similar structure, namely a tridiagonal Toeplitz matrix. The only difference being the matrix elements in each case. The matrix is a result of the discretization of the second derivative. After I have written the code for the bucking beam, I have reused it to solve the other problems, by redefining the diagonal and the non-diagonal matrix elements.

I have used two methods to find the eigenvalues. The first method being the Jacobi rotation method, and the second method is the armadillo function eig_sym(). In addition, for the matrix in the buckling beam problem, we even have analytic eigenvalues and eigenvectors. In order to know if my Jacobi rotation method works as it should, I have compared my results with the eigenvalues I get by using the armadillo function. I have also implemented two test functions to test my Jacobi method, before computed the eigenvalues for the matrices, I first run my test functions. Since the test functions pass, it means that my Jacobi method is working fine, and therefore I can trust my results. Of course the eigenvalues are correct up to a given tolerance, the tolerance I have used is 10^{-8}.

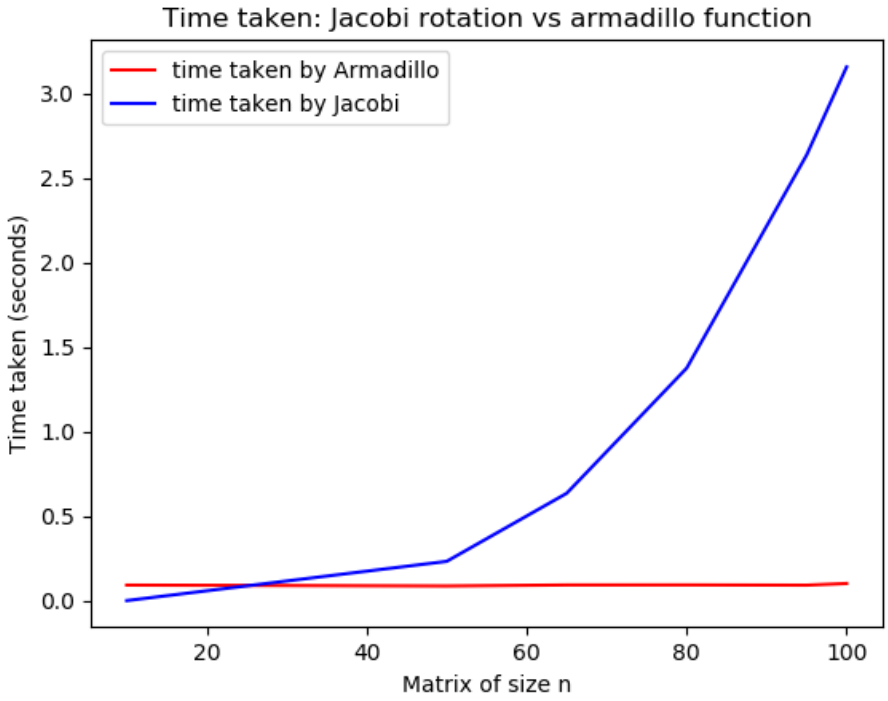

FIGURE 1: A plot of the time taken to compute the eigenvalues using the Jacobi method and the armadillo function, eig_sys()

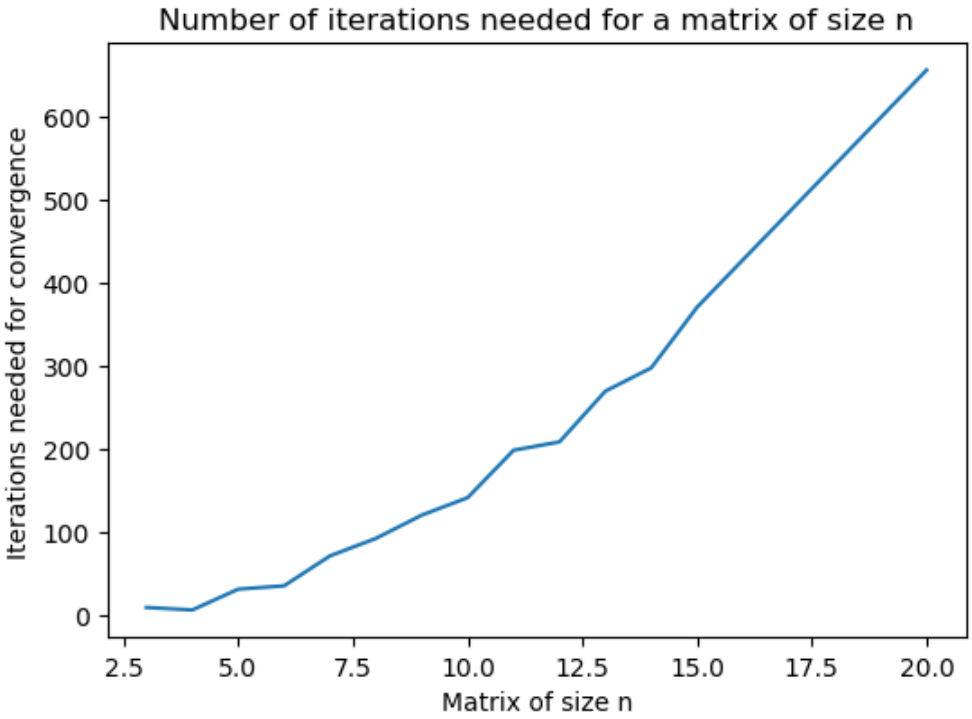

FIGURE 2: A plot of the number of iterations needed to find the eigenvalues, for different matrix sizes.

Another thing of interest is the number of iterations required before the matrix is transformed into a diagonal matrix, that is, how many similarity transforms are required before the matrix A has the eigenvalues on its diagonal. Figure 2 shows this as a plot of the matrix size and the number of iterations needed. As it can be seen, as n increases, we need more iterations to find the eigenvalues. This is what one would expect, as larger matrix means more off-diagonal elements that have to be zeroed out. In this case, one transform can only zero out two elements, so I would expect more iterations as n increases.

I have also looked at the timing of the computations when using the Jacobi method, and the built in armadillo function eig_sys. I have found the average time after running my program six times.It is clear from the plot that the armadillo function is generally faster than the Jacobi method. However for a matrix of size less than 25, the Jacobi method does work faster. This could be due to the fact that the Jacobi method is specialized for the tridiagonal matrix that we have in this problem. But for larger matrices, more iterations are needed, then armadillo would be preferred.

When it comes to computing eigenvalues and eigenvectors, I am sure that my Jacobi rotation returns the correct eigenvalues, this is because I have compared my numerical results to the analytical results, at least for a matrix of small size, like n=3, and n=4.

DISCUSSION

The Jacobi rotation method works well for the matrix in this case. The matrix in this problem is symmetric, and tridiagonal, it is a sparse matrix. This reduces the computational complexity since most elements are zero. The algorithm is easy to implement, and in this case I can check the numerical results with the analytical eigenvalues. The Jacobi method is an iterative method, which means that it will keep going until we set a condition to stop. For small matrices the results may be close to the exact value, but for larger matrices, we would require a lot of rotations, and the method might not converge. The computed eigenvalues are correct up to a tolerance, which we set our self. The armadillo function however, works much faster even for larger matrices.

EVALUATION

In order to improve my results, I could compute the relative error to see how good is my

approximation to the eigenvalues. For small n, I have just checked by myself, but for larger n, it would be impossible to see the computational error. I have implemented two test functions for the buckling beam equation, however to improve the program, I could add more unit tests to make sure the functions are working as they should. In addition, when it comes to the quantum dots with one and two electrons, I was unable to find the optimum rho max, therefore I was unable to find the exact eigenvalues for that equation.

CONCLUSION

The Jacobi rotation method successfully finds the eigenvalues and eigenvectors of the matrices that we have in this problem. For the buckling beam problem, my results are reliable up to a tolerance, my test functions pass, and I get a good approximation to the exact value. However, the Jacobi method has a slow convergence rate, so it would be better to use other methods to find the eigenvalues. The armadillo function computes the eigenvalues much faster than the Jacobi method, so it is preferable for larger matrices. In order to get a better approximation, I should calculate the relative error to check how reliable my results really are.

REFERENCES

1. Morten Hjorth-Jensen. Computational Physics, Lecture notes, Fall 2015, chapter 7.3 and 7.4, page 214. 2015
2. Lay, D, 2015. Linear Algebra And Its Applications. Pearson, pp.413-415.