Project 2

Knut Magnus Aasrud Anna Stray Rongve
Amund Midtgard Raniseth

1 Oktober 2019

1. Abstract

This paper first solves a buckling beam problem as a classical wave function problem in one dimension. Thereafter we extend the problem to quantum mechanics where electrons move in a three dimensional harmonic oscillator potential. We solve these problems as an eigenvalue problem with three different methods; the Jacobi method, the bisection method and Armadillos eigenvalue solver.

Our algorithm for Jacobi is rather inefficient when dealing with large matrices, though having an increasing precision when increasing the size(seen in 4.2 Quantum mechanics eigenvalue calculations). The bisection method is a simple and efficient method with a set precision. Compared to the Jacobi method it was found to be about 530 times faster(0.14 seconds for Bisection and 86 seconds for Jacobi) for a 200 x 200 matrix.

Scaling the equations in order to make them dimensionless is an important part of this project. One reason is to reduce numerical error(round off) when e.g. adding or subtracting small numbers many times. It also makes the behavior of the system more general.

2. Introduction

This project aims to look at different numerical methods for solving eigenvalue problems, which is relevant in many areas of physics, especially when solving differential equations. In this project we will also explore the eigenvalue solver's value specifically, with a classical case - the buckling beam problem - and a quantum case - electrons as quantum dots.

We will solve the following equation:

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

where ρ and λ are the scaled values from a given differential equation representing a physical system.

The eigenvalue algorithm mainly explored in this paper is the Jacobi eigenvalue algorithm first proposed by Carl Gustav Jacob Jacobi. He proposed this algorithm already in 1846 (Jacobi 1846), but it only became widely used with the rise of the computer in the 1950s.

In addition to the Jacobi method for eigenvalues, we will also compare this to a somewhat simpler method. This utilizes bisection to find the roots corresponding to the eigenvalues, which should be more effective.

3. Theory and technicalities

3.1 The problem

In this project, we are considering two wave function problems in one dimension. Generally, the differential equation we are to solve can be written like this:

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

This equation can be applied to both problems by making ρ and λ appropriate scaled values for the system in question.

Buckling beam

For the buckling beam, we are solving this differential equation:

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

where

- the length of the beam is L.
- $x \in [0, L]$ denotes the distance along the beam.
- $u(x)$ is the vertical displacement in y -direction.

γ, F and L are known variables. We say that the scaled value $\rho = \frac{x}{L}$,making the variable ρ defined in $[0, 1]$. The boundary conditions for the scaled function $u(\rho)$ are now $u(0) = u(1) = 0$ - nice and general.

We can now rewrite (2) as:

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

Quantum case

In the quantum case of the differential equation, we have one or two electrons as quantum dots in a 3-dimensional space, both stuck in a harmonic oscillator potential. They repel eachother by the static Coloumb interaction and we assume spherical symmetry. Their dynamics are represented by the radial part of the Schrödinger equation:

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

where $V(r)$ is the harmonic oscillator potential $\frac{1}{2}kr^2$ with $k = m\omega^2$ and E is the energy of the harmonic oscillator. The quantum number l is the orbital momentum of the electron, the oscillator frequency is ω and its energies are:

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

The radial position has boundary conditions $u(0) = u(\infty) = 0$. Since this is already transformed to spherical coordinates, we have $r \in [0, \infty)$. If we substitute $R(r) = \frac{1}{r}u(r)$ we get

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

Moving on, we introduce a dimensionless variable ρ which contains a variable α which we can define later.

If $\rho = \frac{1}{\alpha}r$ the equation reads:

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

Multiplying both sides by $2m\alpha^2/\hbar^2$ we get

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

We can now fix the constant α to eliminate all the constants

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

If we now define

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

the Schroedinger's equation can be rewritten as

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

which is solvable by an eigenvalue solver. The difference this time, however, is the added potential $\rho^2 u(\rho)$.

3.2 Unitary transformations

A unitary matrix $\mathbf{Q}$ has this property:

$$\mathbf{Q}^\dagger \mathbf{Q} = \mathbb{I} \Rightarrow \mathbf{Q}^\dagger = \mathbf{Q}^{-1}$$

where $\mathbb{I}$ is the identity.

Unitary transformations are key to the method we are implementing, so we need to make sure that they preserve the orthogonality of the eigenvectors we apply it on. Our starting basis being orthogonal is more formally written like this:

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

Applying our unitary matrix on these, transforms them into a new basis

$$\mathbf{w}_i = \mathbf{Q}\mathbf{v}_i.$$

Multiplying the above equation from the left with $\mathbf{w}_j^T = (\mathbf{Q}\mathbf{v}_j)^T$, we get

$$\mathbf{w}_j^T \mathbf{w_i} = (\mathbf{Q}\mathbf{v}_j)^T \mathbf{Q}\mathbf{v}_i$$

$$\mathbf{w}_j^T \mathbf{w_i} = \mathbf{v}_j^T \mathbf{Q}^T \mathbf{Q}\mathbf{v}_i$$

Since $\mathbf{Q^T Q} = \mathbb{I}$ (unitary matrices are orthogonal as well),

$$\mathbf{w}_j^T \mathbf{w_i} = \mathbf{v}_j^T \mathbf{v_i} = \delta_{ij}.$$

This means our unitary transformation preserves the orthogonality of our basis.

3.3 Jacobi's method

The Jacobi eigenvalue method is an iterative method for finding eigenvalues. It is based on the idea of doing a number of unitary basis transformations on the matrix in question, with the goal of diagonalizing it.

$$\mathbf{Q}_n^\dagger \mathbf{Q}_{n-1}^\dagger ... \mathbf{Q}_1^\dagger \mathbf{A} \mathbf{Q}_1 ... \mathbf{Q}_{n-1} \mathbf{Q}_n = \mathbf{D},$$

where $\mathbf{A}$ is the starting matrix, $\mathbf{Q}_i$ are unitary matrices and $\mathbf{D}$ is diagonal, containing the eigenvalues.

Quickly jumping back to our original differential equation (1), we first discretize it:

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

where h is the step size. Or more compactly,

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

The set of all equations from u_1 to u_{N-1} can be written as the eigenvalue equation (the endpoints, u_0 and u_N, are not included):

$$\mathbf{A}\mathbf{u} = \lambda \mathbf{u}. \tag{3}$$

Here $\mathbf{A}$ is a tridiagonal matrix and $\mathbf{u}$ contains all the function values.

$$\mathbf{A} = \begin{bmatrix} d & a & 0 & 0 & \dots & 0 & 0 \\ a & d & a & 0 & \dots & 0 & 0 \\ 0 & a & d & a & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & \dots & \dots & a & d & a \\ 0 & \dots & \dots & \dots & \dots & a & d \end{bmatrix} \quad \text{and} \quad \mathbf{u} = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ \dots \\ u_{N-2} \\ u_{N-1} \end{bmatrix} \tag{4}$$

The diagonal $d = \frac{2}{h^2}$ and the non-diagonal $a = -\frac{1}{h^2}$.

Now we can solve our equation numerically using Jacobi's method and compare with the analytical eigenvalues:

$$\lambda_i = d + 2a\cos(\frac{j\pi}{N+1}) \qquad j = 1, 2, \ldots, N. \tag{5}$$

In order to solve equation (3) we implement Jacobi's rotation algorithm. In our Jacobi method we define the following:

$$\tan\theta = t = s/c, s = \sin\theta, c = cos\theta,\ \cot 2\theta = \tau = \frac{a_{ll} - a_{kk}}{2a_{kl}}.$$

We define θ so all non diagonal elements of the transformed matrix become zero. Since

$$\cot 2\theta = \frac{1}{2}(\cot\theta - \tan\theta)$$

We can rewrite as..

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

giving

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

Then

$$c = \frac{1}{\sqrt{1+t^2}} \quad \text{and} \quad s = tc$$

At this point we have our sines, cosines, tangens and cotangens, but we still do not have a rotation matrix. The next step is to search our matrix **A** for the largest element and save its indices. This is done in `/Code/Jacobi_method/off.cpp`.

The next step is to use these indices to create our orthogonal transformation matrix **Q**. Then we can multiply our original matrix **A** from the right with **Q** and from the left with its transposed $\mathbf{Q}^\dagger$. This is done in `/Code/Jacobi_method/Jacobi_rotate.cpp`. We then repeat this process until the largest off-diagonal element is below a set tolerance.

3.4 Our method applied

To solve the buckling beam problem, the approach is quite straight forward. Our solver finds the eigenvalues λ, which gives the values of interest.

For the quantum dots, however, some modification is necessary. The compact discretized Schroedinger equation will be

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

where $V_i = \rho_i^2$ and h is the steplength. From this it is clear that on tridiagonal matrix form it is written

$$\begin{pmatrix} d_i & e_i & 0 & \dots & \dots & \dots & 0 \\ e_i & d_i & e_i & 0 & \dots & \dots & 0 \\ 0 & e_i & d_i & e_i & 0 & \dots & 0 \\ \vdots & 0 & \ddots & \ddots & \ddots & & \vdots \\ \vdots & \vdots & & 0 & e_i & d_i & e_i \\ 0 & \dots & \dots & \dots & 0 & e_i & d_i \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ \vdots \\ u_{N-2} \\ u_{N-1} \end{pmatrix} = \lambda \begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ \vdots \\ u_{N-2} \\ u_{N-1} \end{pmatrix}$$

with the diagonal elements $d_i = \frac{2}{h^2} + V_i$ and the non-diagonal elements $e_i = -\frac{1}{h^2}$.

It is now clear that the eigenvalue solver we made will be able to find these eigenvalues. However it will have to be tweaked by finding a sufficient number of integration points, N and an approximation of ρ_{max} to infinity that yields eigenvalues close enough to the analytical ones of which the first four are $\lambda = 3, 7, 11, 15$.

With these eigenvalues it is possible to calculate the energy and position of the electron, but we will not demonstrate that here. We will instead investigate what number of integration points, N, and what approximation of infinity we can use to get sufficiently precise eigenvalues.

To do this we fix $\rho_{max} = 10$ and find the average deviation of our calculated eigenvalues from the analytical eigenvalues, for $N = 100, 200, 300, 400$. We thereafter plot the error and time versus the number of integration points. This can be found in the project repository in `/Code/Quantum-case/main.cpp`

Then we fix the number of integration points to $N = 200$ and calculate the average error for the approximations $\rho_{max} = 4, 5, 6, 7, 8, 9, 10, 11$ and plot the error and time versus the approximation of ρ_{max}. This can be found in the project repository in `/Code/Quantum-case/main_rho.cpp`

3.5 Using bisection

Bisection is a method of finding the roots of a polynomial. As an alternative to the Jacobi method, we generate the characteristic polynomial P_A of the tridiagonal matrix in question and find the roots of it. These are the eigenvalues we seek. With the tridiagonal matrix defined in eq. (4), $P_{A,n}$ is the characteristic polynomial of a matrix of size n.

$$P_{A,n}(\lambda) = (d-\lambda)P_{A,n-1}(\lambda) - aP_{A,n-2}(\lambda), \qquad P_{A,0}(\lambda) = 1, \qquad P_{A,1}(\lambda) = d - \lambda$$

Our approach to finding roots of this polynomial involves testing over several sub-domains $[a, b]$. In every sub-domain we do bisection. This is simply defining a midpoint c and checking which of the domains $[a, c]$ and $[c, b]$ contain a root (if any). If, for example $f(a) * f(c) < 0$, we conclude that a root is in $[a, c]$ ($f(a)$ and $f(c)$ have different signs). We obviously also check if c is a root. This procedure is done until we are sufficiently close to the root we are seeking (Hjorth-Jensen (2010)).

4. Results

4.1 Buckling beam problem

Our program `/Code/Buckling_beam/` gives these eigenvalues for a matrix of size $N = 10$:

```
0.0810
0.3175
0.6903
1.1692
1.7154
2.2846
2.8308
3.3097
3.6825
3.9190
```

These correspond both with the eigenvalues from armadillos diagonalizer and with the analytical ones. The max non-diagonal element after the complete Jacobi method is 9.65129×10^{-8}. Time spent on the different solvers and the number of rotations can be found by running the program `main.exe`.

4.2 Quantum mechanics eigenvalue calculations

The investigation of a sufficient amount of integration points N, with the program in `/Code/Quantum-case/`, gave us the plot shown in figure 1 on the next page.

The investigation of the best approximation to infinity gave us the plot shown in figure 2 on page 10.

It is also worth noting that all these graphs are reproducible, except the time graph of the approximation of ρ_{max} which were very different every time.

4.3 Jacobi v. bisection

The program in `/Code/Bisection/` compares the speed of the Jacobi method against finding the eigenvalues as roots of the characteristic polynomial with

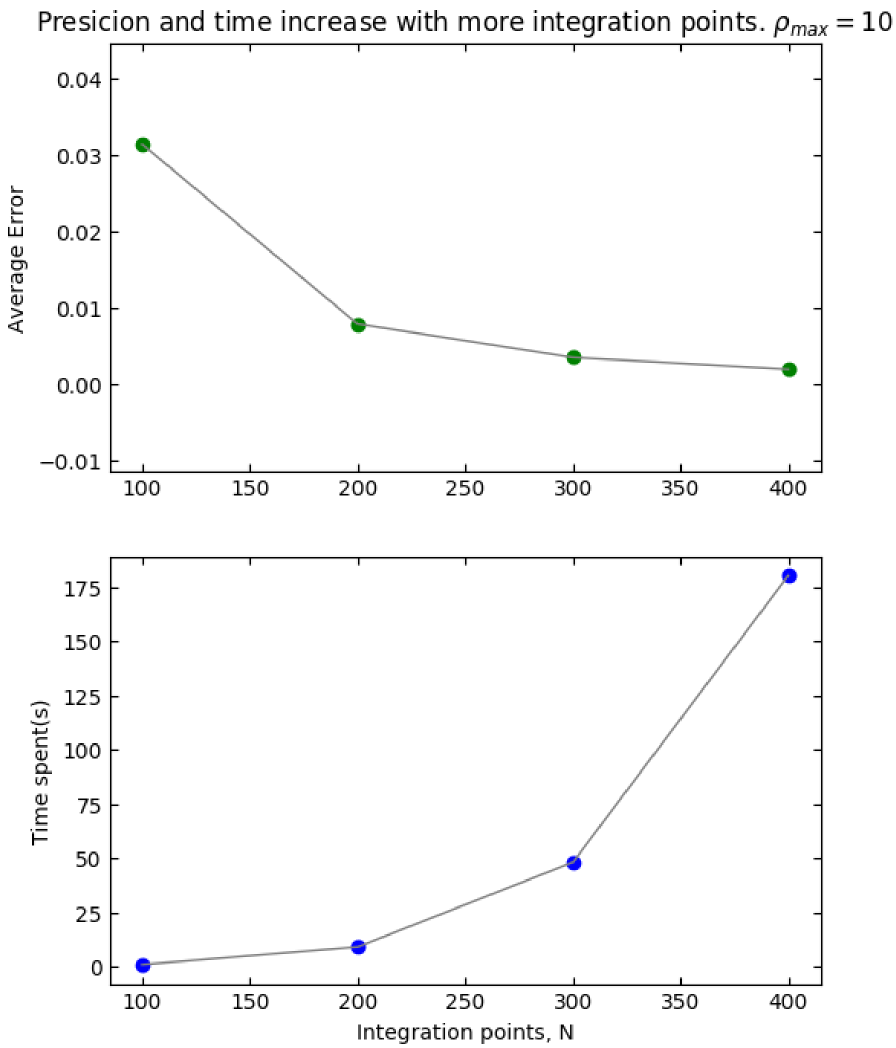

Figure 1: Shows time spent and average error vs number of integration points, N

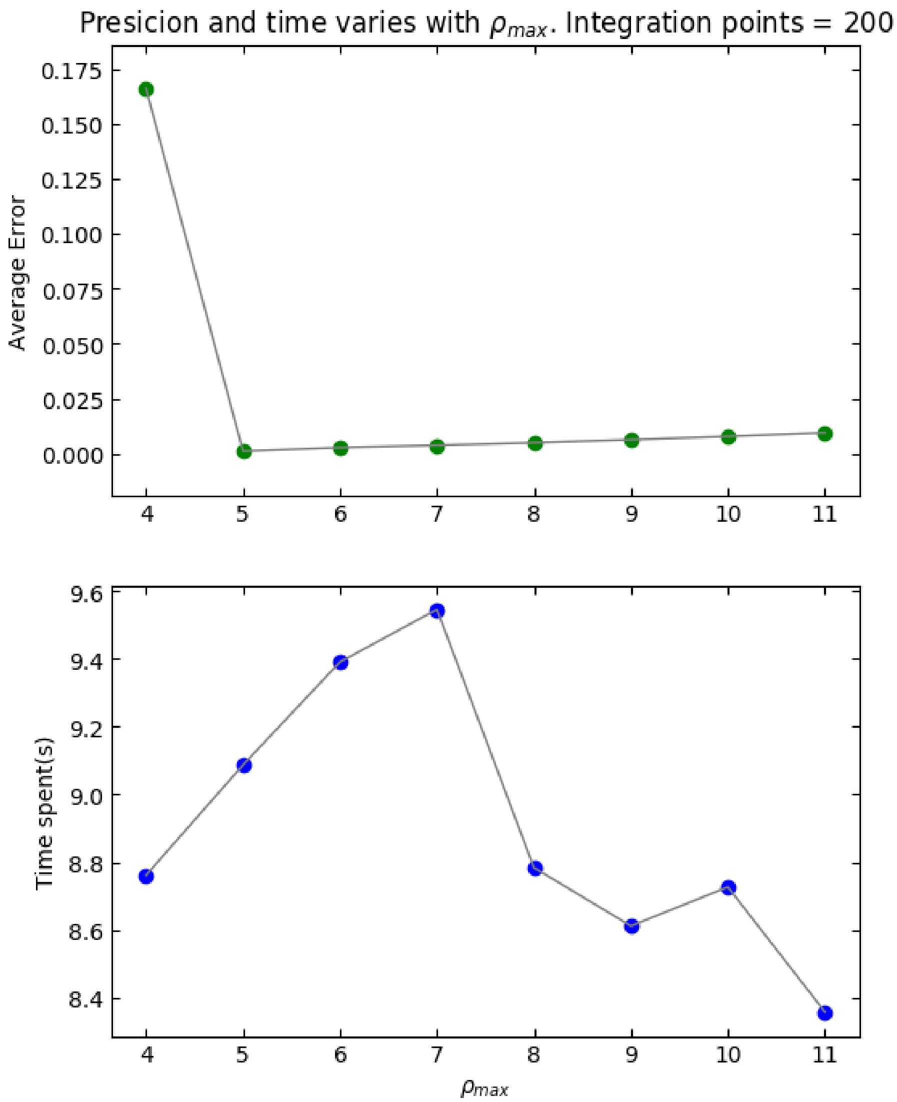

Figure 2: Shows time spent and average error vs approximation of infinity

bisection. With $N = 200$, we get these quite staggering results:

```
Time spent on bisection method: 0.140625 s
Time spent on Jacobi method: 86.875 s
```

Even though $N = 200$ is quite a realistic number of steps (maybe even a bit few, in cases), the bisection method blazes ahead.

5. Discussion

5.1 Buckling beam

After running our Jacobi solver, the max non-diagonal element of our approximately diagonalized matrix is so adequately small, which fits with the fact that our eigenvalues on the buckling beam problem greatly correspond with both the alternative armadillo-solver as well as the analytical solutions. These results can be scaled back to useful values (something we will not go into here). However, running our solver with more integration points significantly increases the calculation time - to the point where this approach does not seem viable for this type of application.

5.2 Quantum mechanics eigenvalue calculations

From the figures (1 and 2) presented in the results we see that while a higher number of integration points yields better results, though also rapidly increasing time.

With the changing of ρ_{max} we see that first the error decreases, but after $\rho_{max} = 5$ we actually start to see an increase in error again. This might be because a higher ρ_{max} gives a bigger step-size which again gives lower numbers on the off-diagonal elements, which in turn yields fewer Jacobi rotations before the off-diagonal elements are below the tolerance for being called zero.

The time spent on the calculations seem pretty random and that is probably because the changing of ρ_{max} doesn't make the computer do any more or less work, it simply changes the numbers. The fluctuations might therefore just be that the computer has different background tasks running at different times.

5.3 Jacobi v. bisection

The bisection method is more specialized to our specific matrix and therefore outraces the Jacobi method once the number of integration points get larger. However, the Jacobi method might do better with a denser matrix, since the bisection method depends on first finding the characteristic polynomial.

6. Conclusion

We found that using Jacobi's method to find eigenvalues was very doable. While the bisection is much faster, it does not give us the opportunity to find eigenvectors.

It is also pretty easy to modify an algorithm like this to work for many different scenarios. We demonstrated both a buckling beam and a quantum particle here, which both turned out very good.

As a method of diagonalization however, there are more effective methods than the Jacobi method. It is too slow and works best as a pedagogic tool.

Appendix

`GitHub Repository` - The source code and all the executables are in the folder `/Code/`.

References

{#refs}

Hjorth-Jensen, Morten. 2010. "Computational Physics.Pdf."

Jacobi, C.G.J. 1846. "Über ein leichtes Verfahren, die in der Theorie der Säkularstörungen vorkommenden Gleichungen numerisch aufzulösen." doi:https://doi.org/10.1515%2Fcrll.1846.30.51.