Draft version September 30, 2019
Preprint typeset using LaTeX style emulateapj v. 12/16/11

EVALUATING THE JACOBI METHOD, EFFICIENCY AND APPLICATIONS

Bruce Chappell and Markus Bjørklund

Draft version September 30, 2019

ABSTRACT

In this letter we investigate numerical solutions to Eigenvalue problems, specifically the buckling beam two point boundary value problem and the Schrödinger equation in one dimension, applying the Jacobi rotation algorithm. The results show that while Jacobi's algorithm is very elegant and produces correct results, it is very slow and too inconvenient for problems requiring better resolution or bigger matrices. One advantage of the Jacobi method, and other algorithms which use the concept of similarity transformations, is however that they conserve the orthogonality of the Eigenvectors.

Subject headings: Jacobi method — Linear algebra: Eigensolvers — methods: computational

1. INTRODUCTION

Many problems in the natural sciences can be discretized and approximated into an Eigenvalue problem. It is therefore of high importance to have efficient algorithms for these types of problems. In this project we will solve the general wave equation for the buckling beam problem and the Schrödinger equation for interacting and non-interacting electrons in a potential well. We will solve these systems by formulating them as eigenvalue problems. On the numerical side, we will study Jacobi's method for rotating matrices to find their eigenvalues and eigenvectors. We have presented theory for the relevant equations and algorithm in the theory section and discussed the numerical approach in the method section. The next section presents our results, with a focus on CPU runtime and numerical output. Closing thoughts are given in the conclusions and further research sections.

2. THEORY

2.1. *The Schrödinger equation*

Note: Some steps will be skipped here. A full derivation can be found in (2).

The radial part of the Schrödinger equation is given by

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

We consider the case of the harmonic oscillator potential, which has $V(r)=(1/2)kr^2$ with $k=m\omega^2$, where E is the energy and ω is the oscillator frequency.

With the substitutaion $R(r)=(1/r)u(r)$ we obtain

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

Now in spherical coordinates we have $r\in[0,\infty)$ and Dirichlet boundary conditions $u(0)=0$ and $u(\infty)=0$.

For the case of a repulsive Coulomb interaction, we get the result

$$\left(-\frac{\hbar^2}{m}\frac{d^2}{dr^2}+\frac{1}{4}kr^2+\frac{\beta e^2}{r}\right)\psi(r)=E_r\psi(r). \tag{3}$$

2.2. *Scaling the equations*

To simplify the problem, we scale the equations. We introduce the variable $\rho=\frac{r}{\alpha}$, where α is a constant. Introducing this into the equations, and requiring that

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

as well as defining

$$\lambda=\frac{2m\alpha^2}{\hbar^2}E, \tag{5}$$

we arrive at

$$-\frac{d^2}{d\rho^2}u(\rho)+\rho^2u(\rho)=\lambda u(\rho), \tag{6}$$

for the case of one electron. For the case of two electrons with a repulsive Coulomb interaction, we also define a frequency

$$\omega_r^2=\frac{1}{4}\frac{mk}{\hbar^2}\alpha^4, \tag{7}$$

fix the constant α by

$$\frac{m\alpha\beta e^2}{\hbar^2}=1 \tag{8}$$

and introducing

$$\lambda=\frac{m\alpha^2}{\hbar^2}E, \tag{9}$$

we arrive at

$$-\frac{d^2}{d\rho^2}\psi(\rho)+\omega_r^2\rho^2\psi(\rho)+\frac{1}{\rho}=\lambda\psi(\rho). \tag{10}$$

2.3. *Numerical model*

We will discretize the problem to reduce it to an Eigenvalue problem. We divide our variable ρ into N points, and define a step length h as

markus.bjorklund@astro.uio.no
[1] Institute of Theoretical Astrophysics, University of Oslo, P.O. Box 1029 Blindern, N-0315 Oslo, Norway

$$h = \frac{\rho_N - \rho_0}{N}. \tag{11}$$

We then get a discretized variable rho_i, given by

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

Using the approximation

$$\frac{d^2}{d\rho^2}u(\rho) = \frac{u(\rho+h) - 2u(\rho) + u(\rho-h)}{h^2} + O(h^2), \tag{12}$$

for the second derivative, and using shorthand notation, we can write the Schrödinger equation as

$$-\frac{u_{i+1} - 2u_i + u_{i-1}}{h^2} + V_i u_i = \lambda u_i \tag{13}$$

We define the diagonal elements d as

$$d_i = \frac{2}{h^2} + V_i, \tag{14}$$

where we have $V_i = \rho^2$ for the case of one electron, and $V_i = \omega_r^2\rho^2 + \frac{1}{\rho}$ for the case of the repulsive Coulomb interaction. The non-diagonal matrix elements given as

$$e_i = -\frac{1}{h^2}. \tag{15}$$

This simplifies the equation further, and it reduces to

$$d_i u_i + e_i u_{i-1} + e_i u_{i+1} = \lambda u_i, \tag{16}$$

We get the Eigenvalue equation $\mathbf{A}\vec{u} = \lambda\vec{u}$, with $\mathbf{A}$ given as

$$\mathbf{A} = \begin{bmatrix} d_1 & e_1 & 0 & 0 & \ldots & 0 & 0 \\ e_1 & d_2 & e_2 & 0 & \ldots & 0 & 0 \\ 0 & e_2 & d_3 & e_3 & 0 & \ldots & 0 \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ 0 & \ldots & \ldots & \ldots & \ldots e_{N-3} & d_{N-2} & e_{N-2} \\ 0 & \ldots & \ldots & \ldots & \ldots & e_{N-2} & d_{N-1} \end{bmatrix} \tag{17}$$

2.4. *Unitary transformations*

Unitary or orthogonal transformations have the property of conserving the dot product and orthogonality. We will show this for orthogonal matrices, but the argument extends to unitary transformations in the complex domain.

Suppose we have two vectors $\mathbf{v}_i$ and $\mathbf{v}_j$. Now, if we consider a orthogonal matrix $\mathbf{U}$, it has the property that $\mathbf{U}^T = \mathbf{U}^{-1}$ such that $\mathbf{U}^T\mathbf{U} = \mathbf{U}\mathbf{U}^T = \mathbf{I}$. Applying the transformation, we get new vectors $\mathbf{w}_i = \mathbf{U}\mathbf{v}_i$ and $\mathbf{w}_j = \mathbf{U}\mathbf{v}_j$. Now let's consider the dot product between these new vectors:

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

which finally gives us

$$\mathbf{w}_i \cdot \mathbf{w}_j = \mathbf{v}_i \cdot \mathbf{v}_j. \tag{18}$$

This shows the dot product is conserved. If we also assume that $\mathbf{v}_i$ and $\mathbf{v}_j$ is part of an orthogonal basis, such that $\mathbf{v}_i \cdot \mathbf{v}_j = \delta_{ij}$, then we from equation 18 that $\mathbf{w}_i \cdot \mathbf{w}_j = \delta_{ij}$ is also true.

2.5. *Jacobi's rotation algorithm*

We are now ready to implement our solver. Using the above result, we know that similarity transformations will conserve the Eigenpairs of the equation. The idea is to apply multiple rotation matrices

$$\mathbf{S}_n^T\mathbf{S}_{n-1}^T ... \mathbf{S}_1^T\mathbf{A}\mathbf{S}_1\mathbf{S}_2 ... \mathbf{S}_n$$

such that A eventually becomes diagonalized. To find these rotation matrices, we use the equations

$$t = -\tau \pm \sqrt{1+\tau^2}, \tag{19}$$

$$c = \frac{1}{\sqrt{1+t^2}}, \tag{20}$$

and

$$s = tc, \tag{21}$$

where we have defined $\tan\theta = t = s/c$, with $s = \sin\theta$ and $c = \cos\theta$ and

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

It can be shown, if you choose to zero out the largest off-diagonal element, that the norm of the new off-diagonal elements after rotation is smaller than the reduction in the off-diagonal element we're trying to zero out. Therefore, Jacobi's algorithm will always tend to all off-diagonal elements eventually being zero.

3. METHODS

This section will describe the methods we utilized to acquire our results.

3.1. *Code*

The code used to aquire the results in this report comes from the program *main.py*. There is also a working c++ implementation, consisting of the program *Jacobi.cpp* and the plotting program *plot.py*, however it is slightly unfinished, so we chose to use the python implementation.

3.2. *Algorithm*

The program used to solve these eigenvalue problems uses a class to create an object that has methods to build the matrix in Eq 17 and implement Jacobi's Method. This class allows the user to select which potential to add along the diagonal: 0 for the buckling beam, $\omega^2\rho^2$ for the non interacting system of two electrons, and $\omega^2\rho^2 + \frac{1}{\rho}$ for the interacting system of two electrons. The methods programmed to perform the rotation are heavily based off of code found in [1]. After obtaining the eigeinvalues and eigenvectors for a given potential, number of iterations, and ρ_{max}, we plotted the first eigenvectors and repeated

for different parameters. We also implemented a function to compare the computing time for Jacobi's Method and the Numpy linalg.eig function. This function compared the two times for the buckling beam problem for matrices of varying size. Another class was also used to create a fixed $4x4$ matrix which could then be used in our unit tests.

3.3. *Parameter Selection*

As specified in in section 2.1.1, the Schrödinger equation has Dirichlet boundary conditions of $u(0) = 0$ and $u(\infty) = 0$. This requires us to approximate ∞ with a given ρ_{max}. From Eq 11, step size is dependent on both ρ_{max} and N, requiring and intelligent choice for both parameters when solving for potentials with different values of ω. Through a process of trial and error we found the following ρ_{max} values and $N = 100$ to offer a compromise between speed and resolution:

ω	0.01	0.5	1	5
ρ_{max}	45	5	4	2

3.4. *Unit Tests*

To ensure or code was functioning properly, we implemented two unit tests at the beginning of the main program. The first unit test applied the function *offdiagmax* on a hard coded matrix with a max element at [0,3] and checked to make sure the function returned the right index for the maximum element. This is shown in the psuedo code bellow:

```
def diagonal_unit_test():
    A = Test_object()
    A.build_test
    max_index = A.offdiagmax
    if (max_index[0] != 0 and max_index[1] != 3):
        print('Max index finder not function, test failed')
        exit()
    else:
        print('Max index finder working')
```

In addition, we wrote a unit test to ensure Jacobi's method was working properly by testing the orthogonality of the resulting eigenvectors and comparing the eigenvalues against those produced by Numpy's linalg.eig function.

4. RESULTS

When solving a matrix of size $100x100$ for the buckling beam problem, we found Jacobi's method to be reasonably accurate compared to the analytical solution for the first three eigenvalues.

$\lambda_{analytical}$	3	7	11
λ_{Jacobi}	2.9992	6.9962	10.9908

This result can be improved by increasing the size of the matrix but at just $N = 100$ the Jacobi method was already inconveniently slow with a runtime of 111.2 seconds. When compared to Numpy's eigenvalue solver for the buckling beam problem, the Jacobi method's shortcomings become obvious.

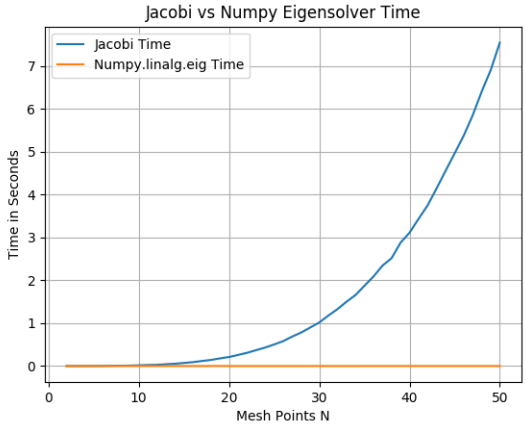

FIG. 1.— Solving time for the two methods plotted as function of matrix mesh points

The slow speed of Jacobi's method stems from its necessity to store the entire matrix and the number of iterations the method needs to converge to a diagonal matrix. This relationship is shown below.

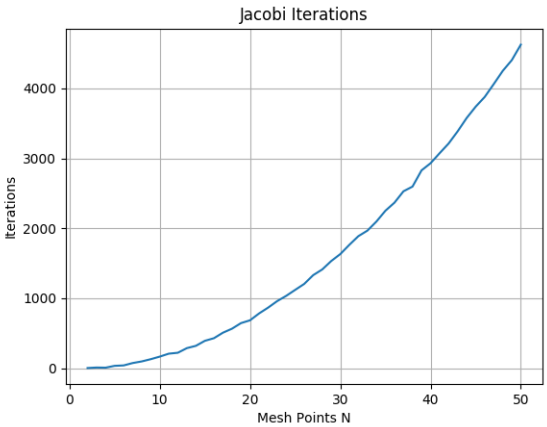

FIG. 2.— Here the number of iterations required to converge to a solution are plotted against the number of matrix mesh points

The ground state eigenvectors were then found for potentials of various angular frequency ω. We examined potentials for both two non-interacting electrons and two electrons subject to the coulomb potential $\frac{1}{\rho}$. This is shown in the following Figures 3-6.

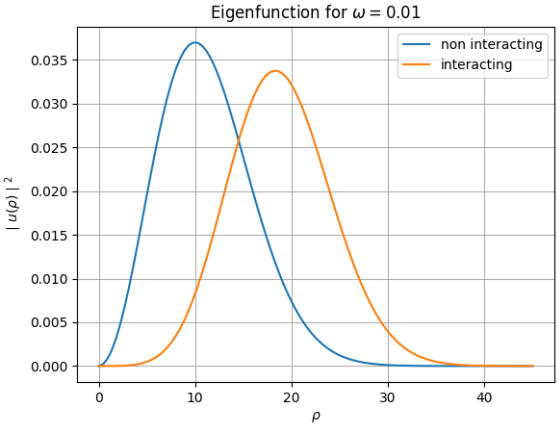

FIG. 3.— Ground state eigenvectors for interacting and non-interacting particles in a potential well with angular frequency $\omega = 0.01$.

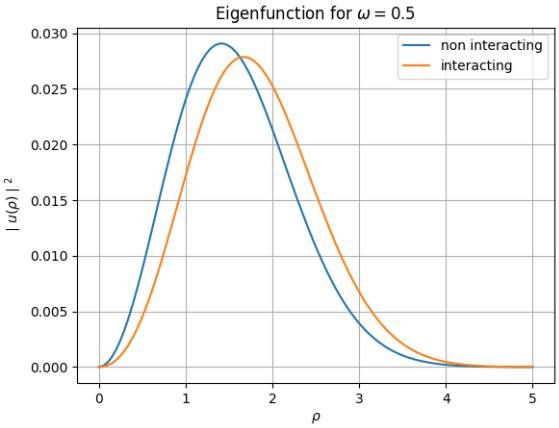

FIG. 4.— Ground state eigenvectors for interacting and non-interacting particles in a potential well with angular frequency $\omega = 0.5$.

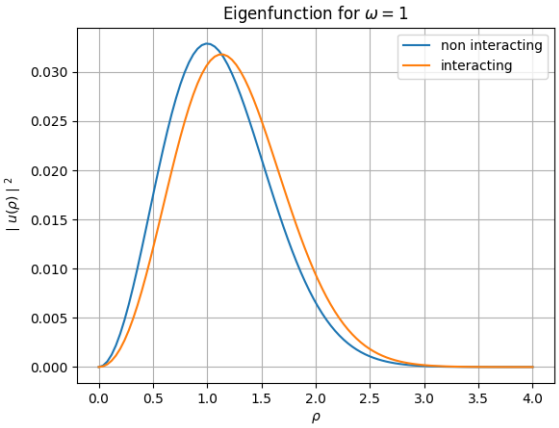

FIG. 5.— Ground state eigenvectors for interacting and non-interacting particles in a potential well with angular frequency $\omega = 1$.

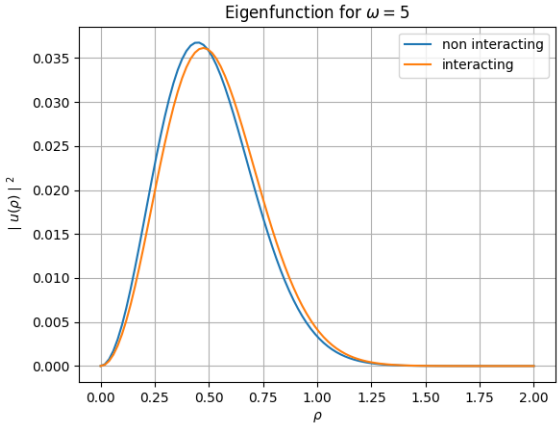

FIG. 6.— Ground state eigenvectors for interacting and non-interacting particles in a potential well with angular frequency $\omega = 5$.

5. DISCUSSION

The speed of the Jacobi Method is very dependant on the size of the matrix that is to be rotated. As seen in Fig 1, Numpy's eigenvalue solver is much more efficient as the size of the matrix is increased. The inefficiency of Jacobi's method is reasonable considering how the required iterations to converge to a solution scale with the size of the matrix. While it does give us accurate results when compared with the Numpy solver, speed is a problem. for this reason we used the Numpy function to solve for the ground state eigenvectors in Figures 3-6. With low angular frequency and a wider potential well, the ground state functions spread out. As the frequency is increased, the wave functions become stepper and less spread because the potential increases and does a better job of confining the particles. It is also shown that the wave function for the interacting electrons is always wider than that of the non interacting particles. This is a result of the coloumb potential always trying to keep the particles away from each other, thus spreading out the wave function.

6. CONCLUSIONS

We have studied the efficiency of the Jacobi method and examined solutions to the Schrödinger equation for a two electron system. The Jacobi method gave us sensible answers, but it was extremely inefficient in terms of run time. There is also the problem of memory usage, as you need to store several full matrices for applying the rotational matrices. We used the Numpy eigenvalue solver to study how the Schrödinger equation for a two electron system in a harmonic oscillator potential. As we increase and decrease ω, the electrons move closer together and farther apart respectively.

6.1. *Further research*

As a continuation of this project, we could examine wave function behavior for types of potentials other than the harmonic oscillator. We could also compare our result with the analytical solution found in [3].

REFERENCES

[1][(Hjorth-Jensen, 2018)]MHJ Hjorth-Jensen, Morten Sep 15 2017, "Computational Physics Lectures:Eigenvalue problems" , http://compphysics.github.io/ComputationalPhysics/doc/pub/eigvalues/pdf/eigvalues-print.pdf

[2][(Hjorth-Jensen, 2018)]MHJ Hjorth-Jensen, Morten Sep 6 2018, "Project 2" http://compphysics.github.io/ComputationalPhysics/doc/Projects/2019/Project2/pdf/Project2.pdf

[3][(Taut, 1993)] Taut, M. (1993). Two electrons in an external oscillator potential: Particular analytic solutions of a Coulomb correlation problem. Phys Rev., A 48,3561

7. APPENDIX

All source code, data and figures can be found at the github repository: https://github.com/marbjo/FYS4150/tree/master/Project2