PHYS 905 - Project 2

Terri Poxon-Pearson

February 27, 2017

In this report we implement Jacobi's Algorithm to solve Shroedinger's equation for electrons in a harmonic oscillator. We first explore the case of a single electron and are able to reproduce the lowest eigenvalues to high accuracy. Next we explore the case of two interacting electrons in the oscillator. For an oscillator frequency of $\omega = 0.25$ we are able to reproduce the lowest eigenvalue of 1.2499 which can be found analytically. Jacobi's method also provides easy access to eigenvectors. Studying these wave functions gives physical insight into the two-electron system.

Contents

1 Introduction

In this project we will be solving Schroedinger's equation for the case of electrons in a 3D harmonic oscillator. First, we will first solve the simpler, non-interacting case with just one electron in the harmonic oscillator potential using Jacobi's method. Next, we will add a second electron with interacts via a repulsive, Coulomb potential. This will involve a change into center of mass coordinates. Both problems will be solved for l=0 case. Additionally, we will explore our calculation's sensitivity to parameters such as the number of mesh points and maximum integration radius.

This report will begin with a brief introduction to the physical system we are studying in this project, as well as a description of how these equations are discretized. Next we will describe the method used to solve this eigenvalue problem. We will discuss the implementation of this algorithm, including a discussion of sensitivity to various integration variables and tests used to check the code's validity. Finally, I will present the results of my calculations, followed by some conclusions.

2 Methods

2.1 Physics of Single Electron in 3D Oscillator

If we assume spherical symmetry, the radial part of Shroedinger's equation for a single electron in a harmonic oscillator can be writen as

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

where $V(r)$ is the harmonic oscillator potential. The energies E for the 3D harmonic oscilator are well know and given by

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

To simplify the expression, we can make the common substitution $R(r) = (1/r)u(r)$, leaving us with

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

We have two boundary conditions to constrain this differential equations. First, we want the wave function to disappear at the origin, so $u(0) = 0$. Second, the wave function should be localized to the oscillator well, so we want $u(\infty) = 0$. Here, we will only consider $l = 0$ states, which eliminates one term of the equation. Finally, we can indroduce a dimensionless variable $\rho = (1/\alpha)r$ and insert $V(\rho) = (1/2)k\alpha^2\rho^2$, leaving us with

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

Multiplying through my the leading factors leaves us with

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

Now we can choose that

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

and define

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

That leaves us with the final expression

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

This is the equation that we must solve numerically. There are analytic solutions in the case of $l=0$ and the three lowest eigen values are $\lambda_0=3,\lambda_1=7,\lambda_2=11$. We can compare the accuracy of our results to these exact solutions.

2.2 Discretization of Schroedinger's Equation

As in Project 1, we will discretize the differential equation and use the three point formula to express the second derivative. This expression is given by

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

A full derivation of this expression can be found in [1]. Here, h is the step size where

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

and the value of ρ at a point i is then

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

N defines the number of grid points. Because this is a radial integral, the define minimum value for ρ is 0 and $\rho_{\max}=\rho_N$,

Using the compact notation from project one, we can rewrite the Schroedinger equation as

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

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

This expression can also be cast as a matrix equation. The left hand side takes the form of a modified tridiagonal matrix. Instead of just having 2 in the diagonal and -1 in the off diagonal, non-zero elements, we now have

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

and

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

Instead of having some known function f on the right hand side, we instead have $\lambda \cdot u$. Now, for each point on the wavefuntion u_i, we can define the Schroedinger equation takes the following form

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

This is, of course, the form of an eigenvalue problem and the entire matrix can be written as

$$\begin{bmatrix} \frac{2}{h^2}+V_1 & -\frac{1}{h^2} & 0 & 0 & \dots & 0 & 0 \\ -\frac{1}{h^2} & \frac{2}{h^2}+V_2 & -\frac{1}{h^2} & 0 & \dots & 0 & 0 \\ 0 & -\frac{1}{h^2} & \frac{2}{h^2}+V_3 & -\frac{1}{h^2} & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & \dots & \dots & -\frac{1}{h^2} & \frac{2}{h^2}+V_{N-2} & -\frac{1}{h^2} \\ 0 & \dots & \dots & \dots & \dots & -\frac{1}{h^2} & \frac{2}{h^2}+V_{N-1} \end{bmatrix} \begin{bmatrix} u_0 \\ u_1 \\ \dots \\ \dots \\ \dots \\ u_N \end{bmatrix} = \lambda \begin{bmatrix} u_0 \\ u_1 \\ \dots \\ \dots \\ \dots \\ u_N \end{bmatrix}. \tag{1}$$

Since the values of u at the two endpoints are known via the boundary conditions, we can skip the rows and columns that involve these values.

2.3 Jacobi's Rotation Algorithm

In this project we implement Jacobi's method to solve the eigenvalue problem. I will briefly explain the method here, but a full explanation of the method can be found in [1]. The method is based on a series of rotations performed on the matrix which eliminates the off diagonal matrix elements. These rotations are allowable because unitary transformations preserve orthogonality. For a proof of this property, see 6.1.

Because we want to eliminate off diagonal elements of the array, we search the matrix for the largest off-diagonal element. Once I identified, we need to determine the rotation angle that will bring this element to 0. For the identified largest element, a_{kl}, we can find this angle through the relation

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

We can then define the angle θ by using the relation $\cot 2\theta = 1/2(\cot\theta - \tan\theta)$. Solving for $tan(\theta)$ (t) we get

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

Finally, t can be easily related to $cos(\theta)(c)$ and $sin(\theta)(s)$ by

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

and

$$s = tc$$

Once the rotation angle is determined, we apply the transformation. After the transformation, the matrix elements are changed so that

$$\begin{aligned}
b_{ii} &= a_{ii}(i \neq k, j \neq l) \\
b_{ik} &= a_{ik}c - a_{il}s(i \neq k, j \neq l) \\
b_{il} &= a_{il}c + a_{ik}s(i \neq k, j \neq l) \\
b_{kk} &= a_{kk}c^2 - 2a_{kl}cs + a_{ll}s^2 \\
b_{kk} &= a_{ll}c^2 + 2a_{kl}cs + a_{kk}s^2 \\
b_{kl} &= 0.
\end{aligned}$$

This process can then be repeated until all off diagonal elements are effectively 0 (less than some defined tolerance). When the process is complete, you will have a diagonal matrix with eigenvalues along the diagonal. Finally, the corresponding eigenvectors are easy to obtain. Originally, we define an NXN unit matrix, r, which is the same size as the matrix being diagonalized. Then, after each transformation, this matrix can be transformed using the relation

$$\begin{aligned}
r_{ik} &= r_{ik}c - r_{il}s \\
r_{il} &= r_{il}c + r_{ik}s.
\end{aligned}$$

2.4 Physics of Interacting Electrons in 3D Oscillator

If we now add a second electron to the harmonic oscillator, they will interact through the Coulomb force. If there was no interaction, we would just have the sum of two single particle Schroedinger equations, which can be written as

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

where $u(r_1, r_2)$ is a two electron wave function and $E^{(2)}$ is the two electron energy.

To simplify this problem, we can perform a change of variables. We will use the relative coordinate $\mathbf{r} = \mathbf{r}_1 - \mathbf{r}_2$ and the center-of-mass coordinate $\mathbf{R} = 1/2(\mathbf{r}_1 + \mathbf{r}_2)$. This gives us

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

As shown in [2], the equations for r and R can be separated via the ansatz for the wave function $u(r,R) = \psi(r)\phi(R)$ and the energy is given by

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

Now if we add the Coulomb term,

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

with $\beta e^2 = 1.44$ eVnm, we get

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

If we preform a similar process of multiplying through by leading coefficients and reintroduce the dimensionless variable $\rho = r/\alpha$, we get

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

Now wi will massage this equation to create an analogue to the non-interacting case. To do this, we define a new 'frequency'

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

and fix α by

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

and define

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

Now, we can rewrite Schroedinger's equation as

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

In this form, ω_r reflects the strength of the oscillator potential.

Here we studied the cases $\omega_r = 0.01$, $\omega_r = 0.5$, $\omega_r = 1$, and $\omega_r = 5$ for the ground state. Again, we only study $l = 0$ states and we omit the center-of-mass energy.

3 Code and Implementation

All of the programs, results, and benchmarks for this work can be found in my GIT repository (https://github.com/poxonpea/PHYS905). All codes for this project were written in FORTRAN.

3.1 Implementing Single and Interacting Electron Cases

The code containing the heart of the Jacobi algorithm is shown below

```
subroutine rotate(testnum,testmat,testr,maxj,maxi)
  implicit none
  integer::i,j,maxi,maxj,testnum,q
  real(8), dimension(testnum,testnum)::testmat,testr
  real(8)::tau,t,s,c,tempii,tempjj,tempqj,tempqi,temprqi,temprqj
! calculate tau and tangent, sine, and cosine
  if (testmat(maxj,maxi) /=0) then
     tau = (testmat(maxi,maxi) - testmat(maxj,maxj))/(2 * testmat(maxj,maxi))
     if (tau .gt. 0.d0) then
        t = 1.d0/(tau + sqrt(1.d0+tau**2.d0))
     else
        t = -1.d0/(-tau + sqrt(1.d0+tau**2.d0))
     end if
     c = 1.d0/(sqrt(1.d0 + t**2.d0))
     s = t * c
  else
     c=1.d0
     s=0.d0
  end if

  ! redefine matrix elements using temperary matrix elements
  tempii = testmat(maxi,maxi)
  tempjj = testmat(maxj,maxj)

  testmat(maxj,maxj) = tempjj*(c**2.d0) - 2.d0*testmat(maxj,maxi)*c*s + tempii*(s**2.d0)
  testmat(maxi,maxi) = tempjj*(s**2.d0) + 2.d0*testmat(maxj,maxi)*c*s + tempii*(c**2.d0)
  testmat(maxj,maxi) = 0.d0
  testmat(maxi,maxj) = 0.d0

  do q=1,testnum
     if (q /= maxj .and. q /=maxi) then
       tempqj=testmat(q,maxj)
       tempqi=testmat(q,maxi)
       testmat(q,maxj)= tempqj*c - tempqi*s
       testmat(maxj,q)= testmat(q,maxj)
       testmat(q,maxi)= tempqi*c + tempqj*s
       testmat(maxi,q)= testmat(q,maxi)
    end if
    temprqj=testr(q,maxj)
    temprqi=testr(q,maxi)
    testr(q,maxj)=c*temprqj - s*temprqi
    testr(q,maxi)=c*temprqi + s*temprqj
  end do

  return
end subroutine rotate
```

Figure 1: Code from Jacobi.f90 which implements Jacobi's Algorithm

This subroutine is implementing after a separate subroutine searches the matrix to find the largest off diagonal element. When running Jacobi.f90, the user decides whether to run for 1 or 2 electrons. The only difference is the potential used. That can be seen in the code below.

```
function Potential(x,NumElectrons,omega)
  implicit none
  real(8)::Potential,x,omega
  integer:: NumElectrons

  if (NumElectrons == 1) then
     Potential = x*x
  else if (NumElectrons ==2) then
     Potential = omega*omega*x*x + 1.d0/x
  end if

  return

end function Potential
```

Figure 2: The potential implemented in Jacobi.f90

Depending on the user choice, the code will either select the simpler, one electron potential, or the potential for the interacting case.

3.2 Tests of Code

This code implements a number of tests to ensue the calculation is working properly. The first test is doing a test calculation on a 4 by 4 matrix. The user can choose this option at runtime and Jacobi.f90 will fill a symmetric 4 by 4 matrix with the values:

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 2 & 6 & 7 & 8 \\ 3 & 7 & 11 & 12 \\ 4 & 8 & 12 & 16 \end{bmatrix} \tag{2}$$

I used mathematic to produce the exact eigenvalues and eigenfunctions. After the run, it will print the exact eigenvalues, as well as the computed values.

Additionally, I include a test to check the orthogonality of the transformation. After every 10 transformation I take the dot product of the two vectors in the eigenvector matrix and check that value is below some tolerance (the value is typically around 10^{-17}). If the absolute value of the dot product were to exceed my tolerance, the program would terminate and an error message will print to screen. The user can choose at runtime whether or not to print these

orthogonality checks during the calculation. The printing will significantly increase the calculation time.

4 Results and Discussion

4.1 Exploring Dependence on Parameters

The results of the calculation will be sensitive to the choice of ρ_{max} and N. Both of these parameters can be defined by the user while running the program. I explored the sensitivity to these parameters for the case of the non-interacting electron. In three dimension, we know the eigenvalues for the three lowest lying, $l = 0$ states should be $\lambda_0 = 3$, $\lambda_1 = 7$, and $\lambda_2 = 11$. Below is a table of convergence for these results for different choices of ρ_{max} and N.

As you can see from the results, $\rho_{max} = 6$ and $N = 200$ is sufficient for convergence and I used this combination for my runs.

4.2 Convergence

	Testing convergence for λ=3						
ρ_{max}	N						
	25	50	100	150	200	250	300
2	3.5260	3.5287	3.5294	3.5295	3.5296	3.5296	3.5296
4	2.9920	2.9980	2.9995	2.9998	2.9999	2.9999	3.0000
6	2.9819	2.9955	2.9989	2.9995	2.9997	2.9998	2.9999
8	2.7829	2.9918	2.9980	2.9991	2.9995	2.9997	2.9998
10	2.9491	2.9874	2.9969	2.9986	2.9992	2.9995	2.9997

	Testing convergence for λ=7						
ρ_{max}	N						
	25	50	100	150	200	250	300
4	6.9631	6.9934	7.0009	7.0023	7.0028	7.0030	7.0003
6	6.9088	6.9774	6.9944	6.9975	6.9986	6.9991	6.9994
8	5.9037	6.9598	6.9900	6.9956	6.9975	6.9984	6.9989
10	6.7399	6.9369	6.9843	6.9930	6.9960	6.6675	6.9983

	Testing convergence for λ=11						
ρ_{max}	N						
	25	50	100	150	200	250	300
2	11.1170	11.1561	11.1660	11.1678	11.1684	11.1687	11.1689
4	10.9747	11.0529	11.0724	11.0760	11.0772	11.0778	11.0781
6	10.7757	10.9448	10.9863	10.9939	10.9966	10.9978	10.9985
8	9.1020	10.9015	10.9755	10.9891	10.9939	10.9961	10.9973
10	10.3499	10.8453	10.9617	10.9830	10.9905	10.9934	10.9958

4.3 Number of Iterations

Jacobi's algorithm does not prescribe the necessary number of iterations to zero all off diagonal elements. To get a sense for this relationship, I ran my code for the case of a single electron for many different matrix sizes, N. The plot below shows the number of iterations necessary for all off diagonal elements to call below 10^{-8}.

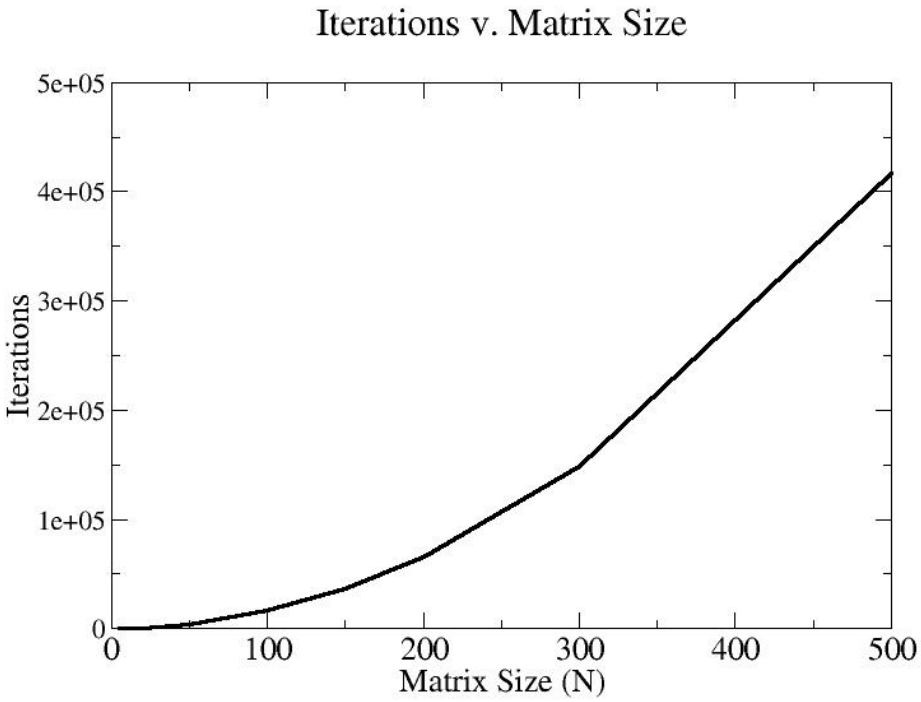

Figure 3: Demonstrating the behavior of Iterations v. Matrix Size

There does not appear to be a simple relationship between the necessary number of iterations and size of the matrix. The growth is non-linear, but slower than exponential growth. A very rough fit in excel found a relatively good fit (by eye) with the second order polynomial $y = 1.7x^2 - 11.6x + 116$. Running for N=1000 took too long for me to obtain a solution, which is unsurprising because, according to this formula, it would require almost 1.7 million iterations.

4.4 Computational Speeds

The efficiency of the Jacobi algorithm can be compared to other eigenvalue solvers. I performed the same calculation using the Linear Algebra PACKage (LAPACK). Specifically I used the DSYEV function which produces eigenvalues and eigenvectors for real, symmetric, matrices. For the case of a non-interacting electron I used both my solver and DSYEV with $\rho_{max} = 6$ and $N = 200$ and timed both calculations using the cpu_time function. DSYEV completed the calculation in 2.18 seconds, while my implementation of Jacobi's algorithm took 7.68 seconds to obtain similar accuracy.

4.5 Comparing Results to Analytic Results

For the simple case of two electrons in a harmonic oscillator with l=0, there are some analytic results available. For this report, I compared my results with M. Taut, Phys. Rev. A, V.48 (1993) [**?**]. Their expressions differ from our development by a factor of 1/2 so in order to compare, their eigenvalues must be multiplied by two. For $\omega = 0.25$, they obtained a lowest eigenvalue of 0.6250, which should be multiplied by 2 to obtain 1.25. For $\rho_{max} = 10$ and $N = 350$, my lowest eigenvalue is 1.2499. This corresponds to a relative error of $1.28E^{-5}$

In addition, I varied the value of ω and studied the effect on the wave funtions and eigenvalues. For these calculations, I ran with $\rho_{max} = 10$ and $N = 150$. The results for the eigenvalues are shown in the table below and the eigenfunctions are show in the figure.

ω	λ
0.01	0.31164
0.5	2.23006
1.0	4.05761
5.0	17.44201

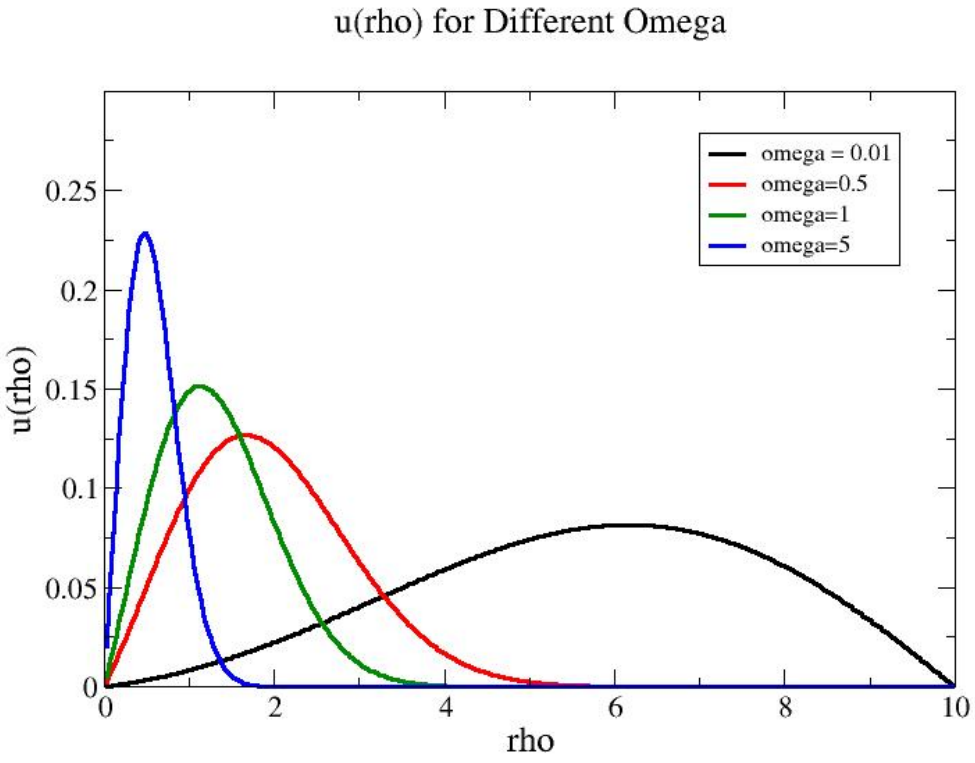

Figure 4: A zoomed in view of the convergence to the exact solution

As you can see, as ω increases, the reduced wave function gets pushed in towards the origin. This can be interpreted as larger values of ω corresponding to a stronger oscillator well. In this case, the Coulomb repulsion is overcome by the potential strength, and the electrons being to move into the center of the well.

5 Conclusions

In this project we implemented the Jacobi algorithm to solve an eigenvalue problem that describes interacting and non-interacting electrons in a harmonic oscillator. We tested our code through the implementation of unit tests an comparing results to known values. For the non interacting case, we were able to reproduce the lowest eigenvalues to a high level of accuracy. For the interacting case, we were able to reproduce the lowest eigenvalue which can be found analytically. By studying the eigenfunction we are able to obtain a

physical interpretation of ω where larger values correspond to a tighter oscillator, while larger values correspond to a wider, weaker oscillator.

We also explored the efficiency of the Jacobi algorithm. Time comparisons to eigenvalue solvers showed that Jacobi's method is not particularly efficient. This is not surprising because there is not prescribed number iterations necessary to complete the transformation. Additionally, this algorithm does not explicitly take advantage of the tridiagonal nature of the matrix.

6 Appendices

6.1 Appendix A

Consider a basis of orthogonal basis vectors $\mathbf{v}_i$,

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

Orthogonality requires that

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

We can apply an orthogonal or unitary transformation such that

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

Unitarity enforces that the product of a matrix with its conjugate transpose is the identity matrix. Orthogonal matrices are a subset of real, unitary matrices. This condition implies that the product of a matrix with its transpose is the identity matrix. These two conditions can be expressed as

$$\mathbf{U}^*\mathbf{U} = \mathbf{U}\mathbf{U}^* = \mathbb{I}$$

$$\mathbf{U}^T\mathbf{U} = \mathbf{U}\mathbf{U}^T = \mathbb{I}.$$

If we now look at the product of our transformed matrix with its transpose, we find

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

Therefore, the dot product is preserved.

References

[1] Hjorth-Jensen, Mortehn. Computational Physics, Lecture Notes Fall 2015. August 2015.

[2] Broida, J. PHYS 130B, Quantum Mechanics II Class Notes. Fall 2009. http://www.physics.ucsd.edu/students/courses/fall2009/physics130b/IdentParts.pdf

[3] Taut, M. Physical Review A, Volume 48, Number 5 November 1993.