Project 1

Eric Aboud[1] and Parker Brue[1]

[1] *Department of Physics and Astronomy, Michigan State University, East Lansing, MI 48823*

We present an algorithm that allows us to solve the Schrödinger equation for electron-electron interactions. Using Jacobi rotation methods, we are able to solve for the eigenvalues for a known potential equation. Using inherent eignenvalue solvers, we are able to verify the eigenvalues obtained through the Jacobi rotation methods. Comparisons between potential strengths provide insight on electron-electron interactions.

I. INTRODUCTION

One of the simplest cases of the Schrödinger equation, an essential equation in the understanding of quantum mechanics, comes from electron-electron interactions within a three-dimensional harmonic oscillator. To further simplify the problem, we can first solve for a case with no Coulomb interactions. Using multiple eigenvalue solvers (Jacobi rotations and built-in solvers) we are able to determine the eigenvalues resulting from a given Schrödinger equation. Analysis of the results provide insight on the effectiveness of the eigenvalue solvers.

II. THEORY, ALGORITHMS AND METHODS

A. Schrödinger's Equation for two electrons in a three-dimensional harmonic oscillator well, noninteracting

The radial part of Schroedinger's equation for one electron 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 $(1/2)kr^2$ with $k=m\omega^2$ and E is the energy of the harmonic oscillator in three dimensions. The oscillator frequency is ω and the energies are

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

with $n=0,1,2,\dots$ and $l=0,1,2,\dots$.

Due to the transformation to spherical coordinates $r\in[0,\infty)$. The quantum number l is the orbital momentum of the electron. Substituting $R(r)=(1/r)u(r)$:

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

boundary conditions are $u(0)=0$ and $u(\infty)=0$.

We introduced a dimensionless variable $\rho=(1/\alpha)r$ where α is a constant length:

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

In stipulations of the project, $l=0$. We inserted $V(\rho)=(1/2)k\alpha^2\rho^2$:

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

And multiplied by $2m\alpha^2/\hbar^2$ on both sides:

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

The constant α was fixed after some algebra so that

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

We defined

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

so we were able to rewrite Schrödinger equation as

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

And evaluate it by considering the canon form of second derivative of a function u

$$u''=\frac{u(\rho+h)-2u(\rho)+u(\rho-h)}{h^2}+O(h^2),\qquad(1)$$

where h is defined as our step length. Minimum and maximum values for the variable ρ are $\rho_{\min}=0$ and $\rho_{\max}$ (In our case, we set $\rho_{\max}$ to 7.0).

With a specified number of mesh points, N, h can be defined as, with $\rho_{\min}=\rho_0$ and $\rho_{\max}=\rho_N$,

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

The value of ρ at a point i is then

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

Rewriting the Schrödinger equation for ρ_i as

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

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

The diagonal matrix element is:

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

and the non-diagonal matrix element is:

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

All non-diagonal matrix elements are equal, a constant. The Schrödinger equation is now:

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

u_i is unknown. we rewrote this so we could solve for the matrix eigenvalues.

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

The values of u at the two endpoints are known through the boundary conditions, allowing us to skip the involved rows and columns. we specified the matrix with our values for d_i and e_i

$$\begin{bmatrix} \frac{2}{h^2} + V_1 & -\frac{1}{h^2} & 0 \\ -\frac{1}{h^2} & \frac{2}{h^2} + V_2 & -\frac{1}{h^2} \\ 0 & -\frac{1}{h^2} & \frac{2}{h^2} + V_{N-1} \end{bmatrix}$$

This is the matrix we performed calculations on for the non interacting case.

B. Schrödinger's Equation for two electrons in a three-dimensional harmonic oscillator well, interacting

We took the case of two electrons in a harmonic oscillator well, but introduced an interaction via the repulsive Coulomb interaction. The single-electron equation:

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

$E^{(1)}$ is the energy within one electron. For two electrons with no repulsive Coulomb interaction, the Schrödinger equation evolves to

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

$$E^{(2)}u(r_1, r_2).$$

To evaluate this further, we introduced the relative coordinate $\mathbf{r} = \mathbf{r}_1 - \mathbf{r}_2$ and the center-of-mass coordinate $\mathbf{R} = 1/2(\mathbf{r}_1 + \mathbf{r}_2)$. The radial Schrödinger equation evolved to:

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

By using an ansatz, $u(r, R) = \psi(r)\phi(R)$, the equations for r and R were separated, giving the energy through the sum of the relative energy E_r and the center-of-mass energy E_R:

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

After taking care of this, we added then the repulsive Coulomb interaction between two electrons,

$$V(r_1, r_2) = \frac{\beta e^2}{|\mathbf{r}_1 - \mathbf{r}_2|} = \frac{\beta e^2}{r},$$

with $\beta e^2 = 1.44$ eVnm.

The r-dependent Schrödinger equation evolved to

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

Through a similar process in the noninteracting case, we were able to introduce a dimensionless variable $\rho = r/\alpha$, and reduce the Schroedinger's equation to:

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

Going further, We defined a new 'frequency'

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

and fixed the constant α to

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

resulting in

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

This further reduced Schrödinger's equation to

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

In our case, we treated ω_r as a parameter reflecting the strength of the oscillator potential, and used the values $\omega_r = 0.01$, $\omega_r = 0.5$, $\omega_r = 1$, and $\omega_r = 5$ in our evaluations.

C. Implementing the Jacobi Rotation Algorithm

Before we implemented the Jacobi algorithm, we first defined some basic nomenclature. We defined the quantities $\tan\theta = t = s/c$, with $s = \sin\theta$ and $c = \cos\theta$ and

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

Through manipulation of the angle, θ, we got non-diagonal matrix elements of the transformed matrix a_{kl} to become non-zero and consequently, the quadratic equation (using $\cot 2\theta = 1/2(\cot\theta - \tan\theta)$

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

resulting in

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

and c and s are easily obtained via

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

and $s = tc$.

We started with an (n x n) orthogonal transformation matrix, with the stipulation that $\mathbf{S^T} = \mathbf{S^{-1}}$ Nonzero matrix elements are (for clarification, in this case s_{index} doesn't refer to $sin\theta$):

$s_{kk} = s_{ll} = c, s_{kl} = -s_{lk} = -s, s_{ii} = -s_{ii} = 1, i \neq k, l$

So, a similarity transformation for a symmetric matrix $\mathbf{B} = \mathbf{S^T AS}$, or in an example 3x3 matrix form

$$\mathbf{B} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c & -s \\ 0 & s & c \end{bmatrix} \begin{bmatrix} a_{ii} & a_{ik} & a_{il} \\ a_{ki} & a_{kk} & a_{kl} \\ a_{li} & a_{lk} & a_{ll} \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & c & s \\ 0 & -s & c \end{bmatrix}$$

results in

$b_{ii} = a_{ii}, i \neq k, l$
$b_{ik} = a_{ik}c - a_{il}s, i \neq k, l$
$b_{il} = a_{il}c + a_{ik}s, i \neq k, l$
$b_{kk} = a_{kk}c^2 - 2a_{kl}cs + a_{ll}s^2$
$b_{ll} = a_{ll}c^2 + 2a_{kl}cs + a_{kk}s^2$
$b_{kl} = (a_{kk} - a_{ll})cs + a_{kl}(c^2 - s^2)$

with θ reducing b_{kl}, the matrix element with the largest value, to zero.

$$\mathbf{B} = \begin{bmatrix} b_{ii} & b_{ik} & b_{il} \\ b_{ki} & b_{kk} & 0 \\ b_{li} & 0 & b_{ll} \end{bmatrix}$$

The algorithm comes into play when this process is reapplied to the newly transformed matrix $\mathbf{B}$, attacking the now largest value of b_{ik} for the resultant matrix, $\mathbf{C}$.

$$\mathbf{C} = \begin{bmatrix} c & -s & 0 \\ s & c & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} b_{ii} & b_{ik} & b_{il} \\ b_{ki} & b_{kk} & 0 \\ b_{li} & 0 & b_{ll} \end{bmatrix} \begin{bmatrix} c & s & 0 \\ -s & c & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

In essence, the algorithm iterates through this transformation until all of the non diagonal matrix elements are reduced to zero, or are smaller than an established threshold tolerance (We used a tolerance of 10^{-10}).

We may implement our quantities into a full C++ algorithm that provides a rotation to a given matrix (Figure 1).

```
void Jacobi_rotate ( mat &A, mat &R, int k, int
    l, int n )
{
double s, c;
if ( A(k,l) != 0.0 ) {
double t, tau;
tau = (A(l,l) - A(k,k))/(2*A(k,l));

if ( tau >= 0 ) {
t = 1.0/(tau + sqrt(1.0 + tau*tau));
} else {
t = -1.0/(-tau +sqrt(1.0 + tau*tau));
}

c = 1/sqrt(1+t*t);
s = c*t;
} else {
c = 1.0;
s = 0.0;
}
double a_kk, a_ll, a_ik, a_il, r_ik, r_il;
a_kk = A(k,k);
a_ll = A(l,l);
A(k,k) = c*c*a_kk - 2.0*c*s*A(k,l) + s*s*a_ll;
A(l,l) = s*s*a_kk + 2.0*c*s*A(k,l) + c*c*a_ll;
A(k,l) = 0.0;
A(l,k) = 0.0;
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) = c*a_ik - s*a_il;
A(k,i) = A(i,k);
A(i,l) = c*a_il + s*a_ik;
A(l,i) = A(i,l);
}
// The new eigenvectors
r_ik = R(i,k);
r_il = R(i,l);

R(i,k) = c*r_ik - s*r_il;
R(i,l) = c*r_il + s*r_ik;


}
return;
}
```

FIG. 1: Jacobi Rotation algorithm that produces a diagonal matrix revealing the eigenvalues of the original tridiagonal matrix.

D. Unitary Transformation

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

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

By assuming that the basis is orthogonal:

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

A unitary transformation

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

will preserve dot product and orthogonality.

$\mathbf{v}_j^T \mathbf{v}_i \not= 0$ when j=i, otherwise, $\mathbf{v}_j^T \mathbf{v}_i = 0$.

looking at the preservation of inner product, and therefore preservation of dot product and orthogonality,

$$(\boldsymbol{v_i}, \boldsymbol{v_j}) = (\boldsymbol{U}\boldsymbol{v_i}, \boldsymbol{U}\boldsymbol{v_j}) = (\boldsymbol{U}|\boldsymbol{v_i}\rangle)^\dagger \boldsymbol{U}|\boldsymbol{v_j}\rangle = \langle \boldsymbol{v_i}|\boldsymbol{U}^\dagger \boldsymbol{U}|\boldsymbol{v_j}\rangle$$

Since $\boldsymbol{U}^\dagger \boldsymbol{U} = 1$ as a consequence of being unitary,

$$\langle \boldsymbol{v_i}|\boldsymbol{U}^\dagger \boldsymbol{U}|\boldsymbol{v_j}\rangle = \langle \boldsymbol{v_i}|\boldsymbol{v_j}\rangle$$

where $\langle \boldsymbol{v_i}|\boldsymbol{v_j}\rangle = \mathbf{v}_j^T \mathbf{v}_i = \delta_{ij}$.

III. ANALYSIS

A. Unit Tests

In order to verify the accuracy of the eigenvalue solvers, multiple tests were used. The first test that was used was an orthogonality preservation test. By taking two different eigenvectors of the diagonal matrix formed via the eigenvalue solvers, we would expect them to be orthogonal. Similarly, if we took identical eigenvectors we would expect a nonzero solution. These results are also portrayed by column vectors of a tri-diagonal matrix, including the one that was used to demonstrate the potential of the three-dimensional Schrödinger equation.
We set up a test that took two column vectors of the potential tridiagonal matrix and solved for the dot product. By calculating this, we verified that similar column vectors of the potential matrix produced a nonzero value (8.88 for a $N = 5$ matrix) and that non-similar column vectors produced zero.
We used this test after solving for the non-interacting case. We found that similar eigenvectors produced a nonzero value (8.62 for a $N = 5$ matrix) and non-similar eigenvectors produced zero.
We then used this test after solving for the interacting case. We found that non-similar eigenvectors produced zero while similar eigenvalues produced a non-zero solution (5.22 for a $N = 5$ matrix, while the potential matrix for this case yielded 3.03).
A second test was performed to check the accuracy of the Jacobi rotation eigenvalue solver. This test involved the use of small N matrices to compare to the c++ software package Armadillo [1].

B. Non-interacting Case

We began our analysis with a simple non-interacting form of the Schrödinger equation. By setting a simple potential, we are able to easily calculate and verify the eigenvalues that are found via the Jacobi rotation method [**??**]. By setting a step length,

$$h = \frac{\rho_{max} - \rho_0}{N} \tag{2}$$

where ρ_{max} was set to 7.0, ρ_0 set to 0.0, and N the number of inputted mesh points, we should expect to find the same eigenvalues for a given N.

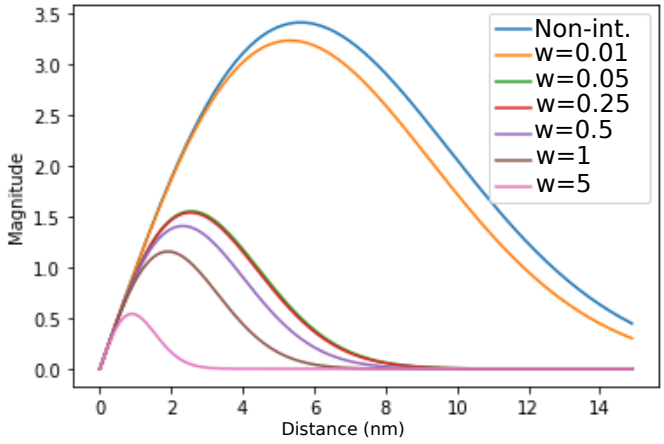

FIG. 2: A plot of the how the strength constant (ω_r) affects the wave function. The plot is the magnitude of the wave function versus the radial distance between the electrons.

Mesh Points (N)	Time$_{Jacobi}$ (sec)	Time$_{Armadillo}$ (sec)
2	0.00033	0.000044
5	0.00057	0.000087
8†	0.00107	0.000103
10	0.00136	0.000120
100	0.04778	0.001862

TABLE I: A time comparison between the Jacobi rotation and armadillo [1] eigenvalue solvers. The † represents the number of mesh points used to find the lowest three eigenvalues for the non-interacting case.

Using a N value of eight, we are able to find the first three lowest eigenvalues: 2.7405, 5.8654, and 9.6309. Using the Jacobi rotation method, we find that the matrix goes through N^2 similarity transformations to get a matrix with zero[1] non-diagonal matrix elements.
We were able to use Armadillo [1], to find the eigenvalues of the non-interacting case for the Schrödinger equation to verify our results from the Jacobi rotation method. Both methods provided identical eigenvalues. Using both methods, we were able to calculate the amount of time required to carry out the calculations (Table III B). It may be observed that the Jacobi method produces results in a similar amount of time as the Armadillo equivalent for a low number of mesh points. However, as the number of mesh points increases, the Jacobi method becomes less efficient.

C. Interacting Case

We then progressed to the Coulomb interaction case for the Schrödinger equation. We now find that we can solve for the same Schrödinger equation with a modified potential to account for the Coulomb interactions. The interaction potential was found to be directly proporational to a coefficient, ω_r, that dictates strength of the oscillator potential. Using the found energy-eigenvalue relationship,

$$E = \frac{\hbar^2}{m\alpha^2}\lambda \tag{3}$$

where alpha is a fixed constant, we were able to determine the ground state energy for various ω_r (Table III C).

ω_r	$E^1_{GS}(J)$	$E^2_{GS}(J)$ [2]
0.01	0.11063	–
*0.05	0.12310	0.1750
*0.25	0.53431	0.6250
0.5	0.54408	–
1	0.96438	–
5	4.38549	–

TABLE II: Comparison between the ground state energy and ω_r. $E^1_{GS}(J)$ are ground state energies found in the present work and $E^2_{GS}(J)$ are accepted values found in Ref. [2]. The * indicates ω_r values that overlap with the accepted values in Ref. [2]. In order to get a good approximation to the accepted value, the ground state energy for $\omega_r = 0.25$ was found using the second lowest eigenvalue.

Using the general solution of the three dimensional Schrödinger equation from [2], we may produce a plot that describes the relationship between the wavefunction, u(r), and the radial distance, r (Figure 2). As the constant (ω_r) increases, the magnitude of the wave function decreases.

IV. CONCLUSION

We were able to construct an algorithm that produces the eigenvalues of a tridiagonal matrix that describes the potential of a three dimensional electron-electron interacting Schrödinger equation. Using units tests and Armadillo functions [1], we were able to verify the results of the algorithm. By timing the algorithm, we were able to see that the Jacobi method produces results in a similar time as the Armadillo equivalent for a small number of mesh points. As the number of mesh points increased, the Jacobi method became much less efficient than the Armadillo equivalent.
The Jacobi rotation algorithm provides one way of finding the eigenvalues for a differential equation. While it may not be as efficient as other methods, it provides accurate results.

[1] C. Sanderson and R. Curtin, Journal of Open Source Software **1**, 26 (2016).
[2] M. Taut, Phys. Rev. A **48**, 3561 (1993).

[1] Zero to a certain approximation (i.e. 10^{-15} was set to zero).