Solving electrons' motion in three dimensional harmonic oscillator well with Jacobi's method

Timofey Golubev, Hao Lin, Xingze Mao

March 6, 2017

Abstract

A code is developed to solve the time-independent Schrödinger equations for two systems, using Jacobi's method. In the first system, an electron is subject to a harmonic oscillator potential. The obtained three lowest energy levels are found to match the analytic results. The second system concerns the motion of two interacting electrons in a range of harmonic oscillator potentials. Ground state energies for different potentials are analyzed and the corresponding wave functions are plotted. In addition, unit tests are employed to ensure the correctness of the code.

1 Introduction

Jacobi's method is an iterative method for solving the eigenvalue problem of real symmetric matrices. Solving time-independent Schödinger equations can typically be casted as an eigenvalue problem, and thus provides a good playground to test the Jacobi's method. In this project, we aim to study the behavior of two different electron systems confined in a 3D harmonic oscillator well. To this end, we implement the Jacobi's method and apply it to solve the corresponding Schödinger equations.

This report is organized as follows: Section 2 discusses the physics background of the two systems of interest. It is then followed by a series of descriptions of methods we employ in the project in Section 3, including discretization of the Schrödinger equations, Jacobi's algorithm, how it is implemented, and the unit tests we include in the code. Results and plots are displayed and discussed in Section 4. Finally, we draw our conclusions in Section 5.

2 Physics background

2.1 System I: Single electron in 3-dimensional Harmonic Oscillator well

The time-independent Schrödinger equation for an electron moving in a 3-dimensional harmonic oscillator well is

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

The first part is the kinetic energy and l is the angular momentum quantum number. The harmonic oscillator potential is $V(r)=\frac{1}{2}kr^2=\frac{1}{2}m\omega^2r^2$, with ω being the oscillator frequency. The analytic results of the eigenenergies are $E=\hbar\omega(2n+l+3/2)$ where $n,l=0,1,2,3\dots$. By setting $R(r)=\frac{u(r)}{r}$, we can express equation 1 as

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

Since the wavefunction must be square-integrable, the boundary conditions are $u(0)=0$ and $u(\infty)=0$. In this project, we are interested in the case of $l=0$. With the substitutions

$$\alpha=(\frac{\hbar^2}{mk})^{1/4}, \qquad \rho=\frac{r}{\alpha}, \qquad \text{and} \qquad \lambda=\frac{2m\alpha^2}{\hbar^2}E$$

the Schrödinger equation for $l=0$ can be written as

$$-\frac{d^2}{d\rho^2}u(\rho)+\rho^2u(\rho)=\lambda u(\rho). \tag{3}$$

The eigenvalues are $\lambda=4n+3$ and $V(r)=\rho^2$ is the potential in terms of the new variables [1].

2.2 System II: Two electrons in 3-dimensional Harmonic Oscillator well

If there is no interaction between two electrons moving in the same well, they will just move independently. The calculation for each electron can be carried out separately and is the same as the single electron situation we discussed earlier. The wavefunction for the system will be the product of wavefunctions for each electron and the total energy will just be the sum of two electrons' energies.

If we include the Coulomb repulsion between two electrons, there will be one additional term in the Schrödinger equation

$$\left(-\frac{\hbar^2}{2m}\frac{d^2}{dr_1^2}-\frac{\hbar^2}{2m}\frac{d^2}{dr_2^2}+\frac{1}{2}kr_1^2+\frac{1}{2}kr_2^2+\frac{\beta e^2}{|\vec{r}_1-\vec{r}_2|}\right)u(r_1,r_2)=Eu(r_1,r_2). \tag{4}$$

The presence of the coupling term poses difficulty to the problem. For this new problem, we need to choose a new coordinate system comprised of the center-of-mass coordinate $\vec{R} = \frac{1}{2}(\vec{r}_1 + \vec{r}_2)$ and the relative coordinate $\vec{r} = \vec{r}_1 - \vec{r}_2$. In this new coordinate system, the Schrödinger equation reads

$$\left(-\frac{\hbar^2}{m}\frac{d^2}{dr^2} - \frac{\hbar^2}{4m}\frac{d^2}{dR^2} + \frac{1}{4}kr^2 + kR^2 + \frac{\beta e^2}{r} \right) u(r, R) = Eu(r, R). \tag{5}$$

This equation is separable with respect to R and r. The eigenfunction for the system can be written as product of two separate functions $u(r, R) = \phi(r)\psi(R)$ and the energy of the system is the sum of the relative energy E_r and center-of-mass energy E_R. Instead of solving one complicated equation, we now have two simpler equations:

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

The first equation is just the same as the equation for a single electron in a harmonic oscillator well. Here we will only focus on the second equation which gives us the relative energy for the system. With the substitutions

$$\alpha = \frac{\hbar^2}{m\beta e^2}, \qquad \rho = r/\alpha, \qquad \omega_r^2 = \frac{1}{4}\frac{mk}{\hbar^2}\alpha^2, \qquad \text{and} \qquad \lambda = \frac{m\alpha^2}{\hbar^2}E$$

the equation is simplified as:

$$-\frac{d^2}{d\rho^2}\phi(\rho) + \left(\omega_r^2 \rho^2 + \frac{1}{\rho} \right) \phi(\rho) = \lambda \phi(\rho). \tag{6}$$

The potential is now $V(r) = \omega_r^2 \rho^2 + \frac{1}{\rho}$. Aided by the finite difference method, this equation can be expressed by a set of linear equations which can be written as a tridiagonal matrix. By diagonalizing the matrix, we can find the eigenvalues and eigenfunctions for this system.

3 Methods

3.1 Discretization

To solve equation 3 numerically, we use the finite difference method to replace the second order differential term.

$$-\frac{d^2}{d\rho^2}u(\rho) = \frac{2u(\rho) - u(\rho + h) - u(\rho - h)}{h^2} + O(h^2), \tag{7}$$

where h is the discretization step size. After defining $d(\rho) = \frac{2}{h^2} + \rho^2$ and $e = -\frac{1}{h^2}$, the differential equation becomes

$$eu(\rho - h) + d(\rho)u(\rho) + eu(\rho + h) = \lambda u(\rho). \tag{8}$$

If we divide the interval $[0, \rho_{max}]$ evenly into $N+1$ points and spacing $h = \frac{\rho_{max}}{N}$, the differential equation will become a set of linear equations. Since we know the solutions at the endpoints from the boundary conditions $u(0) = u(\rho_{max}) = 0$, we have left a system of $(N-1)$ equations to solve:

$$\begin{aligned}
d(h)u(h) + eu(2h) &= \lambda u(h), \\
eu(h) + d(2h)u(2h) + eu(3h) &= \lambda u(2h), \\
eu(2h) + d(3h)u(3h) + eu(4h) &= \lambda u(3h), \\
&... \\
eu((N-3)h) + d((N-2)h)u((N-2)h) + eu((N-1)h) &= \lambda u((N-2)h), \\
eu((N-2)h) + d((N-1)h)u((N-1)h) &= \lambda u((N-1)h).
\end{aligned}$$

Alternatively, we can express these equations in the matrix form

$$\begin{bmatrix} d(h) & e & 0 & 0 & 0 & \dots & 0 \\ e & d(2h) & e & 0 & 0 & \dots & 0 \\ 0 & e & d(3h) & e & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & 0 & 0 & e & d((N-2)h) & e \\ 0 & \dots & 0 & 0 & 0 & e & d((N-1)h) \end{bmatrix} \begin{bmatrix} u(h) \\ u(2h) \\ u(3h) \\ \dots \\ u((N-2)h) \\ u((N-1)h) \end{bmatrix}$$

$$= \lambda \begin{bmatrix} u(h) \\ u(2h) \\ u(3h) \\ \dots \\ u((N-2)h) \\ u((N-1)h) \end{bmatrix}.$$

The above equation can also be written as

$$Au = \lambda u, \tag{9}$$

where A is the tridiagonal matrix and u is the eigenvector. Thus the continuous Schrödinger equation has been transformed to an eigenvalue problem.

3.2 Jacobi's method

In this section, we will focus on real matrices and real vectors unless specified otherwise. Results for complex matrices and vectors can be obtained analogously with unitary matrices in place of orthogonal matrices and conjugate transposition in place of transposition.

Before we dive into the Jacobi's method to diagonalize our matrix, we first show that a real orthogonal transformation preserves the inner product and thus orthonormality. Suppose $\vec{v_i}$ is a basis vector and the basis is orthonormal, in other words, $\vec{v_j}^T\vec{v_i} = \delta_{ij}$. A new set of basis $\vec{w_i}$ is obtained by applying an orthogonal transformation U to $\vec{v_i}$, $\vec{w_i} = U\vec{v_i}$, and $U^TU = \mathbb{1}$, where $\mathbb{1}$ is identity matrix.

$$\vec{w_j}^T\vec{w_i} = (U\vec{v_j})^T U\vec{v_i} = (\vec{v_j})^T(U^TU)\vec{v_i} = (\vec{v_j})^T\mathbb{1}\vec{v_i} = \delta_{ij}. \tag{10}$$

Thus orthogonal transformation preserves the inner product and orthonormality.

To solve the equation of the form $Au = \lambda u$, where A is a square matrix, and λ and u are the eigenvalue and eigenvector respectively, we can apply a orthogonal transformation U^T to both sides of the equation and use that UU^T gives the identity matrix between A and u:

$$U^TAUU^Tu = \lambda U^Tu. \tag{11}$$

Defining $B = U^TAU$, it is clear that λ is also the eigenvalue of B and corresponds to the eigenvector u. If after the transformation, B happens to be a diagonal matrix, all of its diagonal values will be its eigenvalues and also the eigenvalues of matrix A. So we only need to find the right transformation matrix U for A. We can rearrange the above equation as $AU = UB$ and express U and B explicitly

$$A\left[U_1, U_2, \ldots, U_n\right] = \left[U_1, U_2, \ldots, U_n\right]\begin{bmatrix}\lambda_1 & 0 & \ldots & 0\\ 0 & \lambda_2 & \ldots & 0\\ \ldots & \ldots & \ldots & \ldots\\ 0 & 0 & \ldots & \lambda_n\end{bmatrix}$$

$$= \left[\lambda_1U_1, \lambda_2U_2, \ldots, \lambda_nU_n\right].$$

where U_i is the i^{th} column of the matrix U. From the above equation, we can see that U_i is also the eigenvector of matrix A corresponding to eigenvalue λ_i. So it is clear now that if we can find the right orthogonal matrix U to make $B = U^TAU$ diagonal, all of A's eigenvalues and eigenvectors can be found.

In this project, we employ the Jacobi's method to build orthogonal matrices for our matrix A. To illustrate this algorithm, we start with a 2×2 symmetric matrix $\left(\begin{smallmatrix}a_{ll} & a_{lk}\\ a_{kl} & a_{kk}\end{smallmatrix}\right)$. The orthogonal matrix U is chosen to be of the form of $\left(\begin{smallmatrix}cos\theta & sin\theta\\ -sin\theta & cos\theta\end{smallmatrix}\right)$, where θ is to be determined [1]. To simplify the notation, we use s for sin and c for cos in the following. Then the new matrix after applying the transformation to matrix A is

$$U^TAU = \begin{bmatrix}c & -s\\ s & c\end{bmatrix}\begin{bmatrix}a_{ll} & a_{lk}\\ a_{kl} & a_{kk}\end{bmatrix}\begin{bmatrix}c & s\\ -s & c\end{bmatrix}$$

$$= \begin{bmatrix}c^2a_{ll} - 2sca_{kl} + s^2a_{kl} & sc(a_{ll} - a_{kk}) + (c^2 - s^2)a_{kl}\\ sc(a_{ll} - a_{kk}) + (c^2 - s^2)a_{kl} & s^2a_{ll} + 2sca_{kl} + c^2a_{kk}\end{bmatrix}.$$

We expect to end up with a diagonal matrix after the transformation, so the off-diagonal elements should be 0.

$$sc(a_{ll} - a_{kk}) + (c^2 - s^2)a_{kl} = 0. \tag{12}$$

By making the substitution $\tau = \frac{a_{kk}-a_{ll}}{2a_{kl}}$ and using t for $tan\theta$, we have the following equation

$$t^2 + 2\tau t - 1 = 0. \tag{13}$$

Solutions to the above equation are $t = -\tau \pm \sqrt{1+\tau^2}$. Then we can obtain c and s by the relations $c = \frac{1}{\sqrt{1+t^2}}$ and $s = tc$.

For a matrix with dimension $n > 2$, a sequence of orthogonal transformations U can be applied to the original matrix A sequentially to annihilate each pair of non-diagonal elements. These orthogonal transformations will be of the form

$$U = \begin{bmatrix} 1 & 0 & \dots & 0 & \dots & 0 & \dots & 0 & 0 \\ 0 & 1 & \dots & 0 & \dots & 0 & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & cos\theta & \dots & sin\theta & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & -sin\theta & \dots & cos\theta & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & 0 & \dots & 0 & \dots & 1 & 0 \\ 0 & 0 & \dots & 0 & \dots & 0 & \dots & 0 & 1 \end{bmatrix}.$$

Each time we apply such a transformation, the difference of the matrix's norm [1] is

$$\|\mathbf{B} - \mathbf{A}\|_F^2 = 4(1-c) \sum_{i=1, i\neq k,l}^{n} (a_{ik}^2 + a_{il}^2) + \frac{2a_{kl}^2}{c^2}. \tag{14}$$

The second term is the contribution from the pair of off-diagonal elements we want to annihilate; the first term is the contribution from other off-diagonal elements. In each transformation, we want the target pair of off-diagonal elements to become zero while keeping the change of other off-diagonal elements small. Otherwise, the later transformations may introduce large off-diagonal elements. Therefore, we want to keep the first term in the equation as small as possible so that the second term dominates. If c is close to 1, the first term almost vanishes and the second term has the dominate role. If c approaches 0, the second term blows up and will still dominate, but the first term will also have some non-negligible contribution. Thus we want the solution which corresponds to larger c and $|\theta| \leq \pi/4$.

3.3 Implementation

3.3.1 Jacobi's algorithm

The goal is to make all off-diagonal elements smaller than some small tolerance value with as few transformations as possible. Therefore, each transformation will aim at annihilating the largest off-diagonal elements. Each transformation is performed as one iteration of the `while` loop in the code below. In each iteration, we use the `maxnondiag` function to search for the diagonal element with largest absolute value. Then the function `Jacobi_rotate` applies a single transformation to matrix A.

```
//Perform Jacobi rotations always acting on the largest
//(abs value) non-diag element
while ( maxnondiag > tolerance && iterations <= maxiter)
{
   int p, q;
   maxnondiag  = offdiag(A, p, q, n);
   Jacobi_rotate(A, R, p, q, n);
   iterations++;
}
```

3.3.2 Boundary Conditions

The problem we want to study has boundary conditions $u(0) = u(\infty) = 0$. When solving the problem numerically, we cannot discretize ρ to infinity so we must select a finite value ρ_{max} while ensuring accurate results. We select ρ_{max} to be large enough to be in a region where the wavefunctions are nearly zero. At the same time, we do not want to select ρ_{max} to be too large, since that will result in a larger step size in our discretization, making the results less accurate. To achieve the same step size as with a smaller ρ_{max}, we would need to increase the number of mesh points which increases computational cost.

3.3.3 Optimizing the discretization

In the two electron case, we study the behavior with varying oscillator frequencies ω_r. For different values of ω_r, the range of the wavefunctions varies significantly. Thus, for optimal discretization which gives accurate results with minimal computational cost, it is necessary to select ρ_{max} carefully. Thus, we have implemented an automatic search for a suitable ρ_{max} based on first performing a rough estimate of the groundstate wavevector by using a large ρ_{max} which we know is large enough for the entire ω_r range of interest (i.e. 0.01 to 5). Then we search for the ρ when the right side of the wavefunction falls below a small tolerance value. The ρ where this occurs is set to equal ρ_{max} and the solution is re-computed for improved accuracy.

```
//Find radius at which ground state eigvector (wavefnc)
//becomes small (rho_max)
for(int i=n-1; i>0; i--){
    if(groundstate_eigvector(i) > 1.0E-6){
        rho_max = rho(i);
        cout <<"rho_max = " << rho_max << endl;
        break; //once find rho_max, break out of the loop
    }
}
```

3.4 Unit Tests

We have implemented several unit tests in the *Debug* version of our *single* electron case code in order to verify that the algorithm works properly.

3.4.1 Largest non-diagonal element

As discussed in Methods, it is essential that the Jacobi Rotation algorithm properly finds the largest non-diagonal element in the matrix being transformed. We test the `maxnondiag` function on a randomly generated 5×5 matrix.

```
int p, q;
double maxnondiag;
arma_rng::set_seed_random();
mat test = randu(5,5);
cout << "test matrix = " <<endl;
cout << test << endl;
maxnondiag = offdiag(test, p, q, 5);
cout << "maxnondiag element = " << maxnondiag << endl;
```

3.4.2 Orthonormality

We test that the transformations preserve orthonormality, by checking two randomly selected eigenvectors from the eigenvector matrix R every 10 transformations. To choose the eigenvectors randomly, we generate two random indices and we make sure that the indices are not the same. Then the eigenvectors with these indices are extracted from matrix R. We verify that the dot product equals 0 and the norm equals 1, with the nuance that we must allow for a tolerance (10^{-15}) for deviation from 0 and 1 due to the limited precision of storing numbers in a computer.

```
while ( maxnondiag > tolerance && iterations <= maxiter)
{
   maxnondiag  = offdiag(A, p, q, n);
   Jacobi_rotate(A, R, p, q, n);
   if (iterations \% 10 ==0){
      random_index1 = rand()\%n;
      random_index2 = rand()\%n;
      while (random_index1 == random_index2)
      {
        random_index2 = rand()\%n;
      }
      random_eigvector1 = R.col(random_index1);
      random_eigvector2 = R.col(random_index2);
      norm_test = norm(random_eigvector1);

      if (abs(norm_test-1.0) > 10E-15){
          cout << "Normality test failed: " << "norm = "
          << norm_test << endl;
      }

      dot_product = dot(random_eigvector1,
                              random_eigvector2);
      if (dot_product > 10E-15){
          cout << "Orthagonality test failed: "
          << "dot_product = " << dot_product << endl;
      }
  }
  iterations++;
}
cout << "Orthonormality tests passed" << endl<< endl;
```

3.4.3 Comparison with Armadillo eigenvalue solver

We compare the results for lowest eigenvalues from the Jacobi Rotation algorithm to the results from the Armadillo eigenvalue solver `eig_sym`. We found that for an $n \times n$ tridiagonal matrix A, on order of n^2 iterations of the Jacobi Rotation are needed to give matching results (to four digits after the decimal point) with Armadillo's solver for the lowest eigenvalues. Thus we set `maxiter` to n^2. Incidentally, the number of off-diagonal elements is also $O(n^2)$. Loosely speaking, every off-diagonal element on average gets operated on once.The matrix is thus expected to converge to a diagonal one. Also we found that Armadillo's solver is drastically faster than the Jacobi Rotation algorithm. For example, for the same 200×200 matrix, Armadillo's solver took $0.0018s$ while Jacobi Rotation took $1.24s$.

```
vec eigval = eig_sym(A);
vec lowest_eigval(num_eig);
for (int i=0;i<num_eig;i++){
    lowest_eigval(i) = eigval(i);
}
cout<< lowest_eigval<<endl;
```

4 Results and discussion

4.1 Single Electron

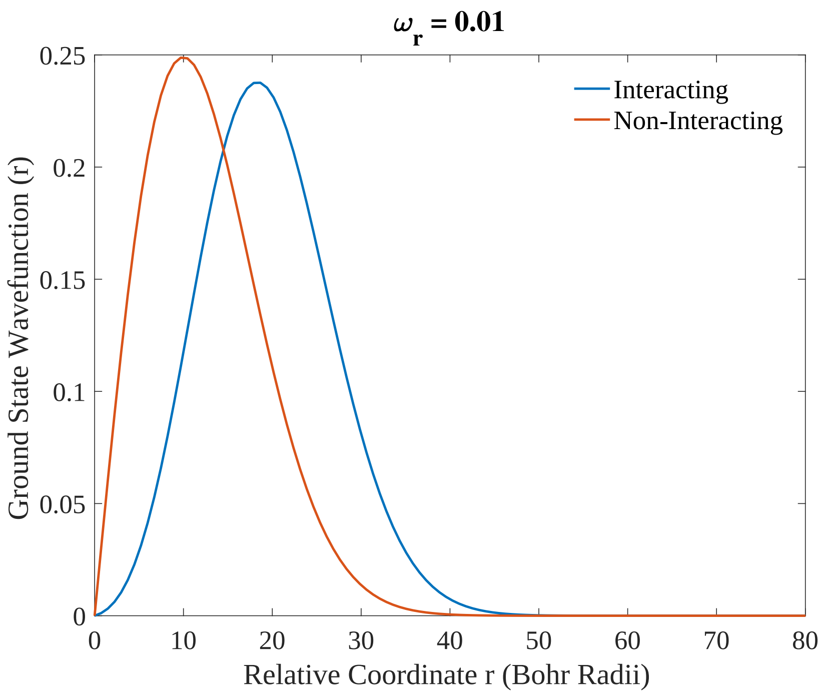
Table 1 shows the calculated lowest eigenvalues for a single electron in a simple harmonic oscillator potential. The N=200 mesh points calculation was done with both Jacobi Rotation algorithm and Armadillo's `eig_sym`. The difference in the results between the two algorithms for the lowest 5 eigenvalues is at most 0.0001. The 1500 mesh points was done only in Armadillo's solver since the Jacobi Rotation algorithm for such a large matrix is too computationally intensive. The 1500 mesh points calculation is exact to the analytic results to 4 decimal point precision for the lowest 3 eigenvalues. Armadillo's overestimation of the eigenvalues for the 4th and 5th lowest values is due to the ρ_{max} being finite.

Jacobi's Method N=200	Armadillo for N=1500	Analytical results
2.9999	3.0000	3
6.9989	7.0000	7
10.9972	11.0000	11
14.9949	15.0001	15
18.9957	19.0040	19

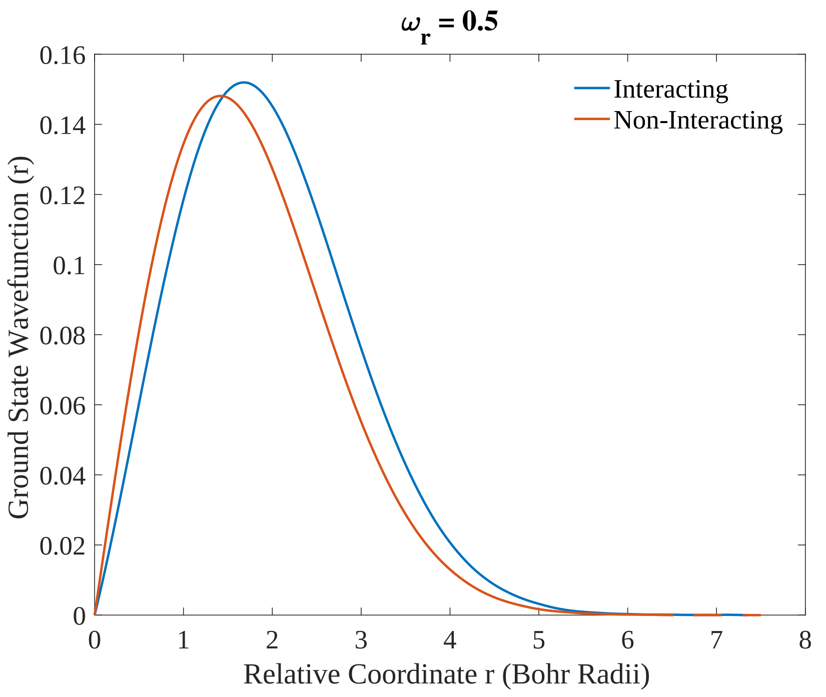
Table 1: Results for lowest eigenvalues for single electron in harmonic oscillator potential calculated with different numbers of mesh points with $\rho_{max} = 5.5a_0$ where a_0 is the Bohr radius.

4.2 Two electrons

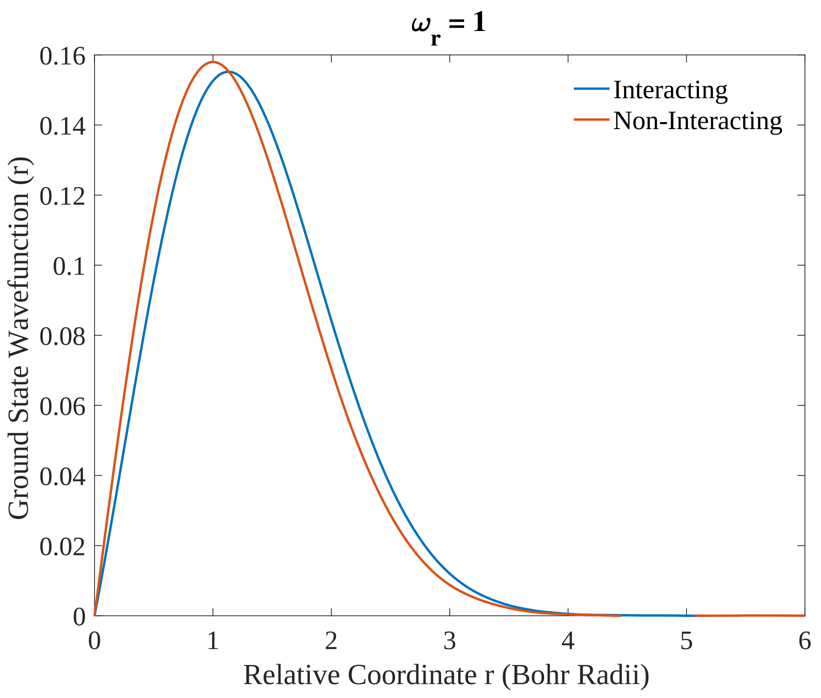
In the figures below, we plot the ground state wavefunctions for two electrons in a simple harmonic oscillator potential with and without Coulomb interaction. Each of the figures corresponds to the result for a different oscillator frequency ω_r. The horizontal axis is the relative coordinate between two electrons in the unit of Bohr radius $a_0 \simeq 0.529\text{Å}$ and the vertical axis is the wavefunction. The blue and red lines show the interacting and non-interacting cases respectively. Red lines show better localization than blue lines for all four frequencies studied. Physically, this means that two electrons prefer to stay away from each other if a repulsive interaction exists. With larger oscillator frequency, the difference between the red and blue lines is smaller and smaller. This is because the increase of oscillator frequency means the harmonic oscillator potential is more strongly constraining the electrons to the origin of the potential and overpowers the Coulomb repulsion which is separating the two electrons. We can also understand this phenomenon by looking at the formula of potential: $V(r) = \omega_r^2\rho^2 + 1/\rho$, the first term corresponds to harmonic oscillator potential and the second one is for Coulomb repulsion. With small ω_r, the $1/\rho$ Coulomb term dominates for small $\rho < 1/\omega_r^{\frac{2}{3}}$, which results in a big difference between the wavefunctions for the interacting and non-interacting cases. With large ω_r, the harmonic potential dominates which will minimize this difference.



$\omega_r = 0.01$
Ground State Wavefunction (r)
Relative Coordinate r (Bohr Radii)
Interacting
Non-Interacting
0
0.05
0.1
0.15
0.2
0.25
0
10
20
30
40
50
60
70
80



$\omega_r = 0.5$
Ground State Wavefunction (r)
Relative Coordinate r (Bohr Radii)
Interacting
Non-Interacting
0
0.02
0.04
0.06
0.08
0.1
0.12
0.14
0.16
0
1
2
3
4
5
6
7
8


$\omega_r = 1$
Interacting
Non-Interacting
Ground State Wavefunction (r)
Relative Coordinate r (Bohr Radii)
0.16
0.14
0.12
0.1
0.08
0.06
0.04
0.02
0
0
1
2
3
4
5
6

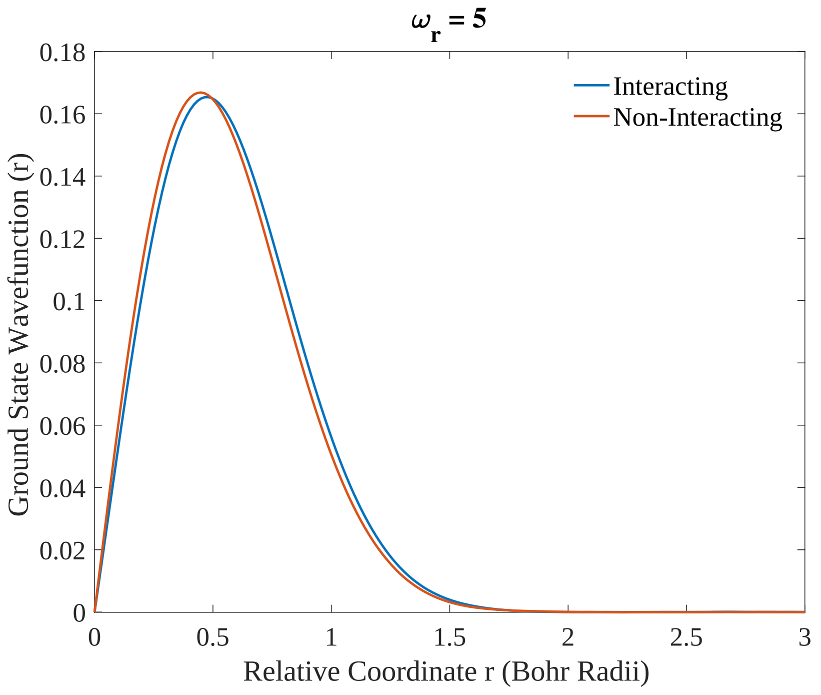

$\omega_r = 5$
Interacting
Non-Interacting
Ground State Wavefunction (r)
Relative Coordinate r (Bohr Radii)
0.18
0.16
0.14
0.12
0.1
0.08
0.06
0.04
0.02
0
0
0.5
1
1.5
2
2.5
3

Table 2, shows the ground state eigenvalues for several ω_r values for which analytical solutions can be obtained. We calculated the eigenvalues using Jacobi's method with 200 meshpoints and ρ_{max} was found automatically as described in Section 3.3.3. Our results are consistent with the analytically calculated eigenvalues reported by Taut [2]. It should be noted that potential V used in Taut, differs by a factor of 1/2 from our potential. We incorporated this factor of two into our eigenvalues λ, when rearranging the Schrödinger equation, thus we multiply the reported eigenvalues by two to correct for this difference.

ω_r	Calculated eigenvalue	Analytic eigenvalue
0.25	1.2500	1.25
0.05	0.3499	0.35
0.01827	0.1644	0.1644
0.008673	0.0954	0.0954

Table 2: Results for lowest eigenvalue for two interacting electrons in harmonic oscillator potential calculated for different values of ω_r.

5 Conclusion

A code based on Jacobi's method is developed to solve the eigenvalue problem. The code is then applied to a system of one electron moving in a harmonic oscillator potential. By applying this code to two electrons in the same potential, we confirm that Coulomb repulsion and harmonic potential have opposite effects on constraining the relative wavefunction of the two electrons, where the former pushes the electrons away from each other and the latter one is trying to localize both electrons at the center of the potential. With proper choice of ρ_{max}, the code can get results which are close to the analytical results [2].

Supplementary Material

All programs and benchmarks calculations can be found in the GIT repository: `https://github.com/tgolubev/PHY905MSU/tree/master/Project2`

References

[1] Morten Hjorth-Jensen. Diagonalization and eigenvalue problems. `https://compphysics.github.io/ComputationalPhysicsMSU/doc/web/course`, 2017.

[2] M. Taut. Two electrons in an external oscillator potential: Particular analytic solutions of a coulomb correlation problem. *Phys. Rev. A*, 48:3561–3566, Nov 1993.