Numerically Solving the Schrödinger Equation for the 3 Dimensional Harmonic Oscillator Potential

Noah Green
Michigan State University

March 4, 2016

Abstract

Solving the quantum harmonic oscillator analytically is generally only possible in the simplest of cases. It is often then necessary to use numerical methods to find and approximate solution. Here, it is shown that Jacobi's eigenvalue algorithm can be used to solve the radial equation of the 3 dimensional harmonic oscillator for two non-interacting electrons and two Coulomb-interacting electrons. It was found that this algorithm is exceedingly slow in its calculations, increasing its computation time quartically as the dimension of its matrix. It was also found that the accuracy of the calculation monotonically increases as the number of grid points is increased, but this accuracy is highly dependent on the maximum range of the discretization of the Schrödinger equation.

1 Introduction

The quantum harmonic oscillator is ubiquitous in many areas of physics. It can be used to model many types of systems in which the particles are confined by a force that is approximately proportional to the distance to its source, like quantum dots or molecular bonds. Here, it is shown that the Schrödinger equation for the one and two electron harmonic oscillator can be solved numerically using Jacobi's eigenvalue algorithm. In section 2, it is shown that the Schrödinger equation for these two cases can be discretized and solved by finding the eigenvalues and eigenvectors of the resulting matrix. In section 3, it is shown that eigensystems can be solved by repeated application of rotation matrices. This method is known as Jacobi's rotation algorithm [1]. We used this method to solve the previously mentioned cases of the harmonic oscillator, and the results are shown in section 4.

2 The Quantum Harmonic Oscillator

We will be solving the three dimensional harmonic oscillator. The harmonic oscillator potential is spherically symmetric, so we will primarily be concerned with the radial part of the Schrödinger equation, that is

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

where m is the mass of the oscillator, ω is the oscillator frequency, and l is the orbital momentum quantum number. We will be looking only at cases where $l = 0$. The details of solving equation (1) analytically can be found in any standard quantum

mechanics textbook. Following reference [1], we can rewrite equation (1) to be

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

where we have defined the following,

$$\begin{aligned} u(r) &= rR(r) \\ \alpha &= \left(\frac{\hbar}{m\omega}\right)^{1/2} \\ \rho &= \frac{r}{\alpha} \\ \lambda &= \frac{2m\alpha}{\hbar^2}E. \end{aligned}$$

Note that by solving for $u(r)$, we force the Dirichlet boundary conditions of $u(0) = 0$ (since we multiply by r) and $u(\infty) = 0$ (by conservation of probability). If we let $m = m_e$, the mass of the electron, then solving equation (2) will give us the solution to the single-electron harmonic oscillator potential. If we let $m = 2m_e$ and r be the relative distance between 2 electrons, then solving equation (2) will give us the solution to the non-interacting double-electron harmonic oscillator potential.

For the case of 2 electrons that interact via the Coulomb force, it is shown in reference [1] that the radial Schrödinger equation can be represented as

$$-\frac{d^2}{d\rho^2}u(\rho) + \omega_r^2\rho^2 u(\rho) + \frac{1}{\rho} = \lambda u(\rho), \tag{3}$$

where $\rho = r/\alpha$ is the dimensionless relative distance ($\mathbf{r} = \mathbf{r}_1 - \mathbf{r}_2$) between the

electrons and we have the following definitions

$$\omega_r = \frac{1}{2}\frac{m\omega}{\hbar}\alpha^2$$
$$\alpha = \frac{\hbar^2}{m\beta e^2}$$
$$\lambda = \frac{m\alpha^2}{\hbar^2}E$$

with β as the proportionality constant in Coulomb's law. Let $V^{(1)} = \rho^2$ and $V^{(2)} = \omega_r\rho^2 + \frac{1}{\rho}$. Then the general equation we want to solve takes the following form,

$$-\frac{d^2}{d\rho^2}u(\rho) + V(\rho)u(\rho) = \lambda u(\rho). \tag{4}$$

To solve equation (4) numerically, it must be "discretized" first. This is done by dividing the domain of u into n steps, where each step has length

$$h = \frac{\rho_{\max} - \rho_{\min}}{n}.$$

In our case, $\rho_{\min} = 0$, and $\rho_{\max}$ is chosen to be much larger than the length scale of the problem. Equation (4) can then be written as [4]

$$-\frac{u_{i+1} - 2u_i + u_{i-1}}{h^2} + V_i u_i = \lambda u_i, \tag{5}$$

where u_i and V_i are their respective functions evaluated at $\rho_i = \rho_{\min} + ih$, $i = 1, 2, ..., n$. However, this can be rewritten in terms of a linear algebra problem by letting each u_i be the i^{th} term of an $n-1$ dimensional vector ($n-1$ since the endpoints

are already known). The Schrödinger equation then takes the form

$$\mathbf{A}\mathbf{u} = \lambda\mathbf{u} \tag{6}$$

where

$$\mathbf{A} = \begin{pmatrix} \frac{2}{h^2} + V_1 & -\frac{1}{h^2} & 0 & 0 & \dots & 0 & 0 \\ -\frac{1}{h^2} & \frac{2}{h^2} + V_2 & -\frac{1}{h^2} & 0 & \dots & 0 & 0 \\ 0 & -\frac{1}{h^2} & \frac{2}{h^2} + V_3 & -\frac{1}{h^2} & 0 & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & \dots & \dots & \dots & \frac{2}{h^2} + V_{n-2} & -\frac{1}{h^2} \\ 0 & \dots & \dots & \dots & \dots & -\frac{1}{h^2} & \frac{2}{h^2} + V_{n-1} \end{pmatrix}. \tag{7}$$

Solving this matrix for its eigenvalues λ_i and eigenvectors $\mathbf{u}_i$ will give us an approximation to the energy levels and wave functions respectively for equation (4).

3 Jacobi's Eigenvalue Algorithm

Solving equation (7) is a non-trivial task for large values of n. In mathematics classes, the canonical way to find a solution is to solve the characteristic polynomial of the matrix to get its eigenvalues. However, by the Abel-Ruffini impossibility theory, there is not a general way to do this for $n \geq 5$ [1].

The Jacobi eigenvalue algorithm takes a different approach. It uses the fact that in the vector space spanned by the eigenvectors of a matrix, the matrix will be diagonal with its eigenvalues along the diagonal. Jacobi's algorithm works for a symmetric matrix by applying a series of rotations to the matrix until its only non-zero elements

are its eigenvalues along its main diagonal. The eigenvectors can then be recovered from the column space of the product of rotation matrices that minimized the non-diagonal elements.

In the following code, the matrix we want to solve is named "SolveMe." Note that the GNU Scientific Library(GSL)[2] was used to define vectors and matrices. The Jacobi algorithm was implemented as follows:

1. **Find the maximum non-diagonal element** by iterating over the non-diagonal elements in the upper triangle of the symmetric matrix. Return the coordinates and value of the absolute maximum element:

```
double Solver::MaxVal( int &i, int &j ){
double max = 0.;
// Symmetric matrix. Only have to iterate over upper triangle.
for( int row = 0; row < N; row++ ){
  for( int col = row; col < N; col++){
    if( row != col && abs(max)
< abs( gsl_matrix_get( SolveMe, row, col ) ) ){
max = gsl_matrix_get( SolveMe, row, col );
i = row;
j = col;
    }
  }
}
return max;
}
```

2. **Apply rotation matrix to "rotate away" maximum non-diagonal element.**

   Note that rotation matrices take the form

$$\mathbf{R} = \begin{pmatrix} 1 & 0 & \dots & 0 & 0 & \dots & 0 & 0 \\ 0 & 1 & \dots & 0 & 0 & \dots & 0 & 0 \\ \vdots & \dots & \ddots & \dots & \dots & \dots & 0 & \vdots \\ 0 & 0 & \dots & \cos\theta & 0 & \dots & 0 & \sin\theta \\ 0 & 0 & \dots & 0 & 1 & \dots & 0 & 0 \\ \vdots & \dots & \dots & \dots & \dots & \ddots & 1 & \vdots \\ 0 & 0 & \dots & -\sin\theta & 0 & \dots & 0 & \cos\theta \end{pmatrix}.$$

In the previous step, we found the coordinates of the largest off-diagonal matrix element. We construct our rotation matrix so that the upper right $\sin\theta$ has the same coordinates. For a matrix $\mathbf{A}$, we can then apply the rotation to get a new matrix $\mathbf{B}$ via

$$\mathbf{B} = \mathbf{R}^T\mathbf{A}\mathbf{R},$$

This results in a set of equations involving $\sin\theta$, $\cos\theta$, and the matrix elements of $\mathbf{A}$ and $\mathbf{B}$,

$$\begin{aligned}
B_{ik} =& A_{ik}\cos\theta - A_{il}\sin\theta, i \neq k, i \neq l \\
B_{il} =& A_{il}\cos\theta + A_{ik}\sin\theta, i \neq k, i \neq l \\
B_{kk} =& A_{kk}\cos^2\theta - 2A_{kl}\cos\theta\sin\theta + A_{ll}\sin^2\theta \\
B_{ll} =& A_{ll}\cos^2\theta + 2A_{kl}\cos\theta sin\theta + A_{kk}\sin^2\theta \\
B_{kl} =& (A_{kk} - A_{ll})\cos\theta\sin\theta + A_{kl}(\cos^2\theta - \sin^2\theta)
\end{aligned}$$

To "rotate away" the largest matrix element in $\mathbf{A}$, we set B_{kl} to zero and solve for $s = \sin\theta$ and $c = \cos\theta$. With some trigonometric identities, we find that $t^2 + 2\tau t - 1 = 0$, where $t = \tan\theta$ and

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

Using the quadratic equation, we find that $t = -\tau \pm \sqrt{1+\tau^2}$. Note that since the term under the square root only differs from τ^2 by 1, so one of the roots has an absolute value of less than 1. Choosing this root gives a rotation of magnitude $|\theta| \leq \pi/4$, due to the range of the tangent function. We can then get s and c by

$$c = \frac{1}{sqrt1 + t^2}$$

and $s = tc$.

In the code for this algorithm, the rotations are also allowed to accumulate on an identity matrix for the effect of getting the eigenvectors of the matrix once it is fully diagonalized. In C++ code, this part of the algorithm is:

```
// Get the row and column of the max off-diagonal element
int k,l;
double Akl, Akk, All;

Akl = MaxVal(k,l);
Akk = gsl_matrix_get( SolveMe, k, k );
```

```
      All = gsl_matrix_get( SolveMe, l, l );

      double sn, cs;

// If max off-diagonal element is zero, then rotation is
//     the identity matrix
if( Akl != 0.0 ){

  // Solve for values of sine and cosine that will
  //     rotate max off-diagonal matrix element to zero
  double tn, tau;
  tau = ( All - Akk )/(2.*Akl);

  // Choose minimum root for rotation <= pi/4
  if( tau >= 0 ){
    tn = 1./( tau + sqrt(1. + tau*tau) );
  }
  else{
    tn = -1./( -tau + sqrt(1. + tau*tau) );
  }

  cs = 1./sqrt(1+tn*tn);
  sn = tn*cs;
}
else{
  // Set rotation to identity if
  //     max off-diagonal element is zero
cs = 1.;
sn = 0.;
      }

      // Apply rotation to the matrix A
      //     (represented by 'SolveMe'): A' = (R^T)(A)(R)
      double Aik, Ail, Rik, Ril;

      gsl_matrix_set( SolveMe, k, k, cs*cs*Akk
      - 2.*cs*sn*Akl + sn*sn*All );
      gsl_matrix_set( SolveMe, l, l, sn*sn*Akk
      + 2.*cs*sn*Akl + cs*cs*All );
      gsl_matrix_set( SolveMe, k, l, 0. );
```

```
      gsl_matrix_set( SolveMe, l, k, 0. );
      for( int i = 0; i < N; i++ ){
if( i != k && i != l ){
  Aik = gsl_matrix_get( SolveMe, i, k );
  Ail = gsl_matrix_get( SolveMe, i, l );
  gsl_matrix_set( SolveMe, i, k, cs*Aik - sn*Ail );
  gsl_matrix_set( SolveMe, k, i, gsl_matrix_get( SolveMe, i, k ) );
  gsl_matrix_set( SolveMe, i, l, cs*Ail + sn*Aik );
  gsl_matrix_set( SolveMe, l, i, gsl_matrix_get( SolveMe, i, l ) );
}

Rik = gsl_matrix_get( Evec, i, k );
Ril = gsl_matrix_get( Evec, i, l );

gsl_matrix_set( Evec, i, k, cs*Rik - sn*Ril );
gsl_matrix_set( Evec, i, l, cs*Ril + sn*Rik );
      }
```

3. **Calculate quadrature sum of non-diagonal elements.** For a matrix $\mathbf{A}$, calculate

$$\text{off}(\mathbf{A}) = \sqrt{\sum_{i=1}^{n} \sum_{j=1, j \neq i}^{n} A_{ij}^2}.$$

In C++ code, this is:

```
 // Returns quadrature sum of off-diagonal elements
double Solver::SumOffDiag(){
  double sum = 0;
  for( int i = 0; i < N; i++){
    for(int j = i; j < N; j++){
      if( i != j ){
sum += gsl_matrix_get( SolveMe, i, j )
  *gsl_matrix_get( SolveMe, i, j );
      }
    }
  }
  return sqrt(2*sum);
}
```

4. **Repeat until quadrature sum of non-diagonal elements is below desired tolerance.** This was achieved by placing step 2 in a *while* loop. We used a tolerance of 10^{-10}.

Once the algorithm finishes, the eigenvalues can then be recovered from the diagonal elements of the matrix **A**, and the eigenvectors can be recovered as the columns of the matrix that was accumulating the rotations.

4 Results of Calculations

In all cases, the Jacobi algorithm was run using a tolerance of 10^{-10}, and a cap of 10^5 on the maximum number of rotations to find a solution. It can be seen in table (1) that the Jacobi eigenvalue algorithm generated values are identical to those given by GSL's QR algorithm.

Algorithm	**Grid Points**	$\rho_{\max}$	**E1**	**E2**	**E3**
GSL	10	10	2.72256	5.89458	9.95979
Jacobi	10	10	2.72256	5.89458	9.95979
GSL	50	10	2.98793	6.93939	10.8514
Jacobi	50	10	2.98793	6.93939	10.8514
GSL	100	5	2.99923	6.99617	10.9908
Jacobi	100	5	2.99923	6.99617	10.9908
GSL	100	10	2.99693	6.98465	10.9625
Jacobi	100	10	2.99693	6.98465	10.9625
GSL	100	20	2.98769	6.93817	10.8484
Jacobi	100	20	2.98769	6.93817	10.8484
GSL	100	50	2.92112	6.59186	9.9604
Jacobi	100	50	2.92112	6.59186	9.9604
GSL	250	4.61	2.99989	6.99953	11.0016
Jacobi	250	4.61	2.99989	6.99953	11.0016

Table 1: Comparison of eigenvalues given by the Jacobi algorithm and GSL's QR algorithm for various grid point and $\rho_{\max}$ values.

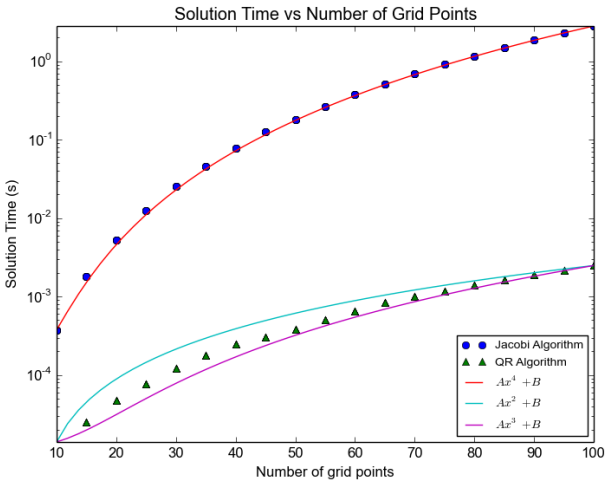

Figure 1: Plot of the time to converge to a solution versus the number of grid points. Used $\rho_{\max} = 4.61$ with the non-interacting harmonic potential.

However, Jacobi's algorithm remains inferior to QR in terms of computation time. In figure (1), we can see that Jacobi's algorithm increases like n^4, while QR increases somewhere between n^2 and n^3.

The quartic increase in computation time for Jacobi's algorithm is expected. Each rotation requires n^2 operations. Additionally, as we can see in figure (2) that the number of rotations increases as the square of the number of grid points. This is because the number of matrix elements that need to be "rotated away" also increases as the square of the number of grid points. Using these facts, we get a computation time that increases like $(n^2)^2 = n^4$.

To help evaluate the accuracy of the Jacobi eigenvalue algorithm, a χ^2 test was used

$$\chi^2 = \sum_{\text{Eigenvalues}} \left(\frac{E_{\text{ana}} - E_{\text{calc}}}{\sigma} \right)^2 \tag{8}$$

where E_{ana} and E_{calc} are the analytical and calculated eigenvalues respectively, and σ is the maximum error allowed for an accuracy of 4 significant digits. If most of the eigenvalues in the χ^2 function are accurate to 4 significant digits, then the χ^2

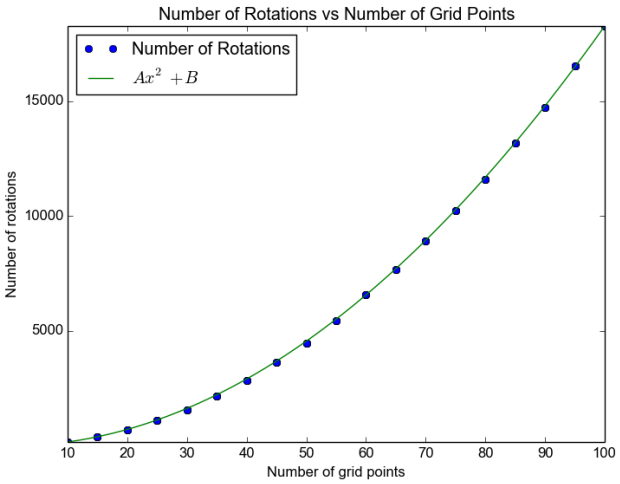

Figure 2: Plot of the number of rotations required for all non-diagonal elements to be zero versus the number of grid points used. Used $\rho_{\max} = 4.61$ with the non-interacting harmonic potential.

function should give a result on the order of the number of eigenvalues used in its sum. It was found that such an accuracy is achieved when the number of grid points is about 250. Increasing the number of grid points only increases the resolution of the wave function, so it is expected to see a monotonic decrease in χ^2 with this increase. This is can be seen in figure (3).

It is interesting to note that the accuracy of the calculation for a given number of grid points has a heavy dependence on the maximum range $\rho_{\max}$. If $\rho_{\max}$ is too small, then it is too close to the length scale of the problem and is not a good approximation for infinity. If $\rho_{\max}$ is too large, then all of the interesting features of the wave function are restricted to the beginning of the range of ρ, which reduces the number of grid points that contribute to the lower energy eigenfunctions. This dependence can be seen in figure (4).

It was found that the optimum value for the maximum range with $N = 250$ is $\rho_{\max} = 4.61$. Using these values, the lowest three energy levels for the single electron harmonic oscillator were able to be calculated to 4 significant digits. Their values can

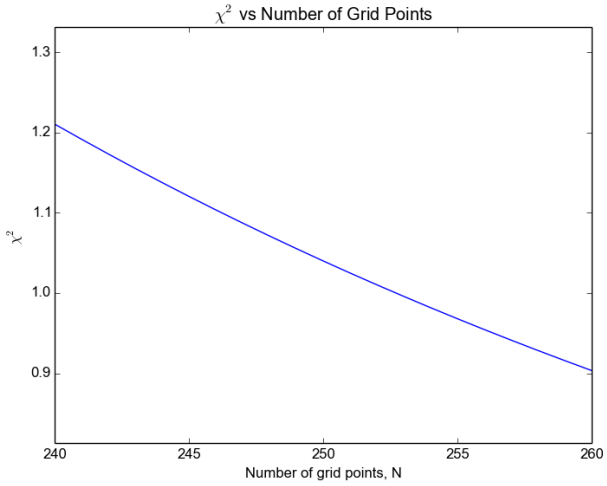

Figure 3: Plot of the χ^2 function vs the number of grid points in the region where eigenvalues reach an accuracy of 4 significant digits. Used $\rho_{\max} = 4.61$ with the non-interacting harmonic potential.

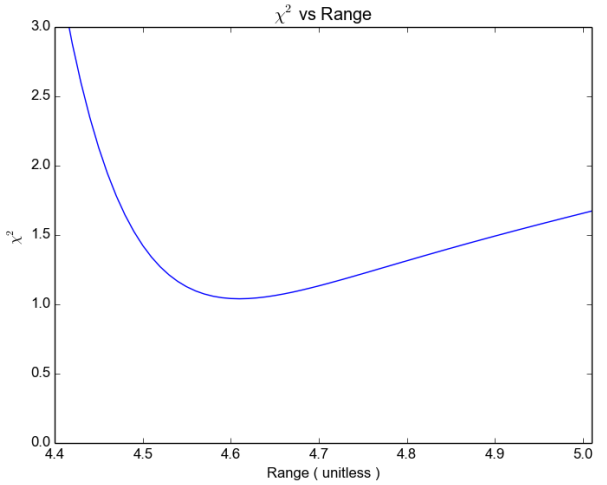

Figure 4: Plot of the χ^2 function vs $\rho_{\max}$ in the region where eigenvalues reach an accuracy of 4 significant digits. Used $N = 250$.

be found in table (2) under the non-interacting case with the ω_r value of 1, where the expected results are $\lambda = 3, 7, 11$.

$\omega_\mathbf{r}$	**E1**	**E2**	**E3**
Non Interacting			
0.01	0.523415	1.92596	4.249
0.5	2.12132	4.95436	7.87206
1	2.99989	6.99953	11.0016
5	6.70768	15.6498	24.5903
Interacting			
0.01	0.598781	0.968214	1.34651
0.5	3.00078	5.70915	8.46232
1	4.05769	7.90882	11.8171
5	8.32214	17.0252	25.8243

Table 2: Table of the first three energy levels of the (non)interacting double electron harmonic oscillator.

The eigenvalue solutions for the interacting case (equation (3)) have also been included in table (2). Exact solutions in this case can be found for some select values of ω_r in reference [5]. The results in table (2) at least appear to be of the same order of magnitude as these exact solutions. Note that in the non-interacting case, $N = 250$ grid points were used with a $\rho_{\max} = 4.61$. These values were only optimal for $\omega_r = 1$, but they were unchanged for the different values of $omega_r$ to see how the wave function is affected for non-optimal values. For the interacting case, $N = 250$ grid points were still used, but the range was changed to $\rho_{\max} = 20, 8, 6, 4.61$ for $\omega_r = 0.01, 0.5, 1, 5$ respectively. These values are not necessarily optimal (an optimization routine is beyond the scope of this project), but by looking at the interacting case's wave function plots, we can see they are close to optimal.

To allow for easier comparison, the wave function plots for the interacting and

non-interacting cases can be found in appendix (A). The wave functions in both cases act as expected, with $n-1$ nodes for the n^{th} energy level. We can see that the optimized case in figure (A.3) that the interesting features of the wave functions take up the entire range of the plot and smoothly go to zero at $\rho_{\max}$. This can be compared to the non-optimized cases. In figure (A.1) that for $\omega_r = 0.01$, the longer wavelength wave functions are abruptly cut off at $\rho_{\max}$ without smoothly going to zero (i.e. $\psi'(\rho_{\max} \neq 0$). Conversely, for the short wavelength $\omega_r = 5.0$ in figure (A.1), we can see that the wave functions all have a magnitude of about zero at $\rho = 3.5$. Hence, the grid points between $\rho = 3.5$ and $\rho_{\max} = 4.61$ are essentially wasted for these lower energy levels. This reduces the number of effective grid points used in calculating the eigensystem, which ultimately causes the accuracy of the energy level calculations to suffer.

Comparing the interacting and non-interacting cases, it was expected that the repulsion between the electrons would cause an upward shift in the energy levels of the system. Looking at table (2), this is exactly what is seen in most cases. The $\omega_r = 0.01$ case is the exception but, as discussed above, the accuracy in this case is hindered by $\rho_{\max}$ being too small. Little difference is seen in comparing the wave functions of the two cases, but since the energy shift is relatively small – on the same order as the energy levels themselves – this is not unexpected. There is a phase difference in the ground state of the interacting case in figures (A.5) and (A.7), but since the probability distribution is given as the square of the magnitude of the wave function, this will have no effect on further calculations.

5 Conclusion

We have shown that Jacobi's eigenvalue algorithm can be used with the discretized Schrödinger equation to calculate the solutions to the non-interacting and interacting double electron harmonic oscillator problems to an accuracy that is dependent on the number of grid points used. For a given number of grid points, the accuracy of the calculation has a heavy dependence on the maximum range $\rho_{\max}$. If $\rho_{\max}$ is too small, then interesting features of the wave functions get cut off and are not included in the eigenvalue calculation. If $\rho_{\max}$ is too large, then the wave functions go to zero too soon so the number of effective grid points is reduced. If the actual energy levels are known analytically or experimentally, this could be remedied by using optimization software that minimizes the χ^2 function with respect to $\rho_{\max}$ for a given number of grid points. If the actual energy levels are unknown, we could instead look at the wave functions for a given number of grid points and set $\rho_{\max}$ to the minimum value of ρ where they smoothly go to zero.

Additionally, in comparing Jacobi's algorithm with the QR algorithm used by GSL it was found that the calculational results were identical, but Jacobi's algorithm had a much longer computation time. Its computation time increased quartically as a function of the number of grid points. GSL's QR algorithm was shown to be much faster, with computation time increasing quadratically or cubically with the number of grid points. As a result, it is not recommended to use Jacobi's eigenvalue algorithm in time intensive calculations.

Appendix A

Wave Function Plots

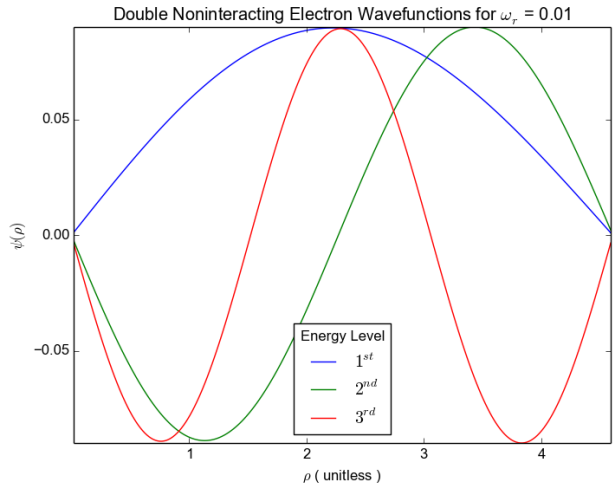

Figure A.1: Plot of the first three non-interacting double electron harmonic oscillator wave functions. $N = 250$, $\rho_{\max} = 4.61$, $\omega_r = 0.01$.

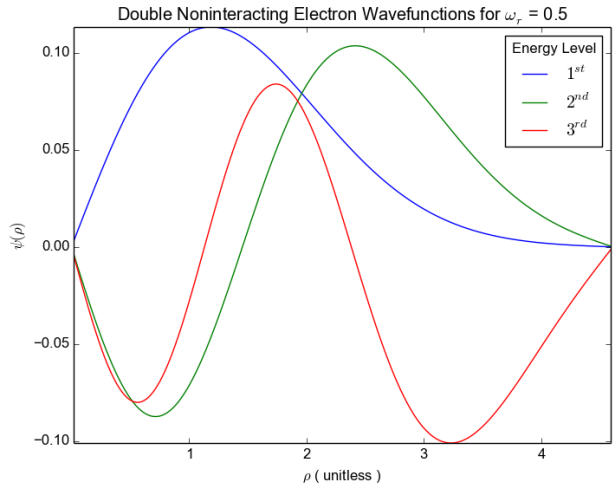

Figure A.2: Plot of the first three non-interacting double electron harmonic oscillator wave functions. $N = 250$, $\rho_{\max} = 4.61$, $\omega_r = 0.5$.

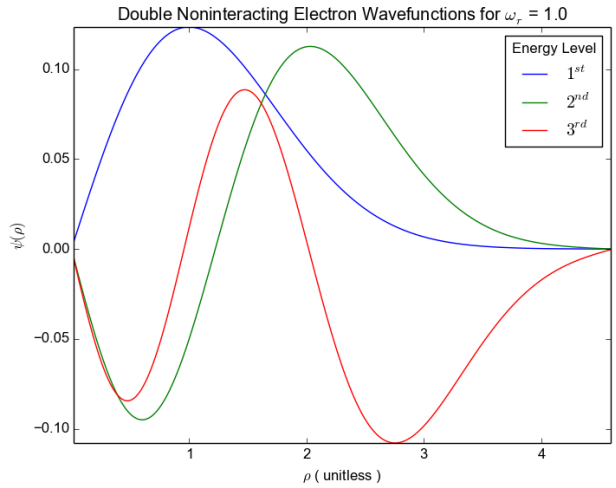

Figure A.3: Plot of the first three non-interacting double electron harmonic oscillator wave functions. $N = 250$, $\rho_{\max} = 4.61$, $\omega_r = 1.0$.

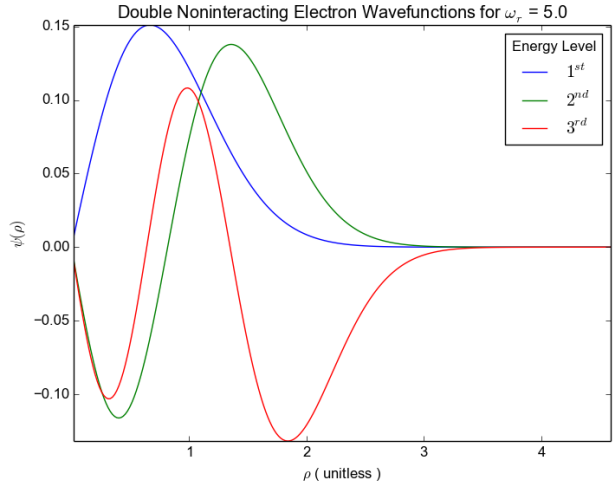

Figure A.4: Plot of the first three non-interacting double electron harmonic oscillator wave functions. $N = 250$, $\rho_{\max} = 4.61$, $\omega_r = 5.0$.

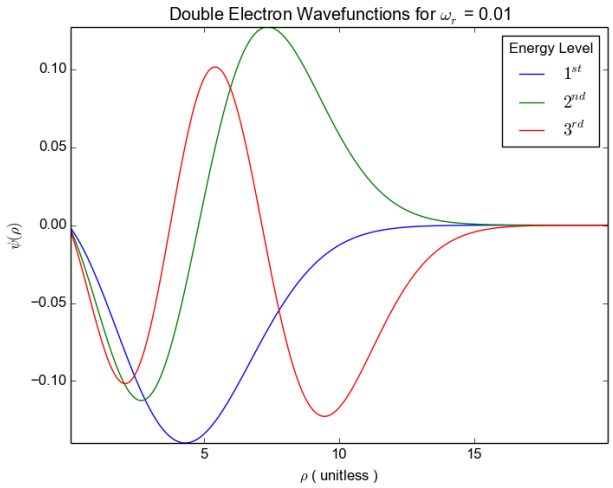

Figure A.5: Plot of the first three interacting double electron harmonic oscillator wave functions. $N = 250$, $\rho_{\max} = 20.0$, $\omega_r = 0.01$.

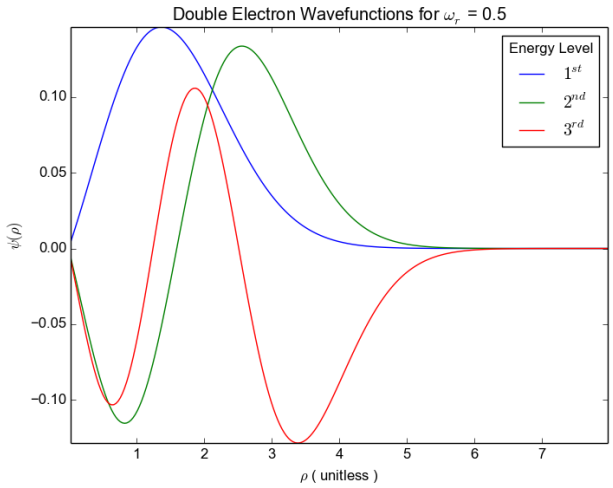

Figure A.6: Plot of the first three interacting double electron harmonic oscillator wave functions. $N = 250$, $\rho_{\max} = 8.0$, $\omega_r = 0.5$.

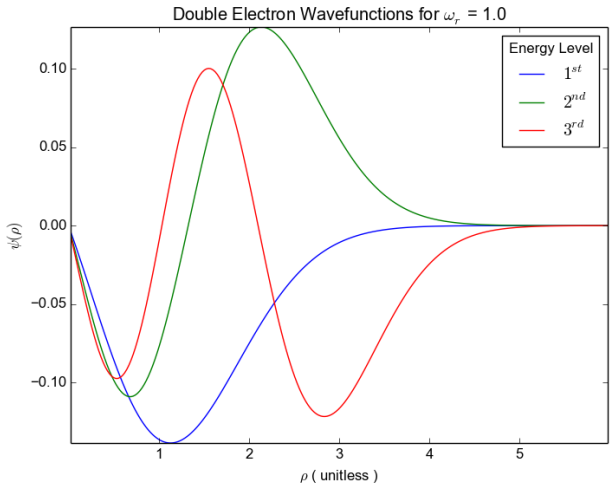

Figure A.7: Plot of the first three interacting double electron harmonic oscillator wave functions. $N = 250$, $\rho_{\max} = 6.0$, $\omega_r = 1.0$.

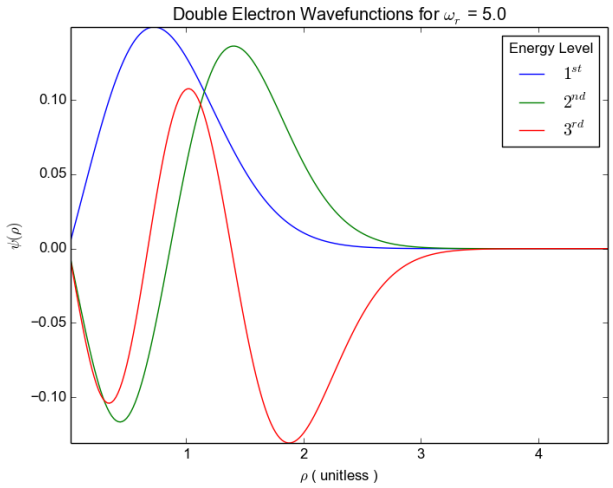

Figure A.8: Plot of the first three interacting double electron harmonic oscillator wave functions. $N = 250$, $\rho_{\max} = 4.61$, $\omega_r = 5.0$.

Bibliography

[1] Phil Duxbury and Morten Hjorth-Jensen. Msu computational physics (phy480/905) course material. 2016.

[2] M. Galassi and et al. *GNU Scientific Library Reference Manual*. 3 edition, 2015.

[3] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing In Science & Engineering*, 9(3):90–95, 2007.

[4] Rubin Landau, Manuel Páez, and Cristian Bordeianu. *A Survey of Computational Physics*. Princeton University Press, Princeton, NJ, 2008.

[5] M. Taut. Two electrons in an external oscillator potential: Particular analytical solutions of a coulomb correlation problem. *Phys. Rev. A*, 1993.