Project 2: Solving Schrödinger's equation for two electrons in a 3D harmonic oscillator well

Joshua Bradt
(Dated: March 4, 2016)

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

I. INTRODUCTION

A commonly studied system in physics is that of the electron confined in a potential well of some sort. This arises in, for example, the study of quantum dots, quantum computing, and other areas of solid-state physics.

A. One electron case

Assuming spherical symmetry, we can write the radial part of Schrödinger's equation as follows for the confined electron in three dimensions:

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

Here, $R(r)$ is the radial wave function of the electron, l is the angular momentum, $V(r)$ is the confining potential, and E is the energy. If we take the potential to be the three-dimensional harmonic oscillator potential,

$$V(r)=\frac{1}{2}m\omega^2r^2, \tag{2}$$

then the energies are known to be

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

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

Next, make the substitution $R(r)=(1/r)u(r)$ in (1) to find

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

If we set $l=0$ and thereby neglect the centrifugal barrier term – which is, after all, ultimately just an extra term in the potential – we can write this as

$$-\frac{\hbar^2}{2m}\frac{d^2}{dr^2}u(r)+\frac{1}{2}m\omega^2r^2u(r)=Eu(r)$$

where $V(r)$ has been replaced with the expression for the harmonic oscillator potential introduced above. This can then be simplified by introducing the dimensionless variable $\rho=r/\alpha$ where α is some constant with the dimensions of length:

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

This can be rearranged to find

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

which suggests that we define $\alpha=\sqrt{\hbar/m\omega}$ to get

$$-\frac{d^2u}{d\rho^2}+\rho^2u(\rho)=\frac{2}{\hbar\omega}Eu(\rho)=\lambda u(\rho),\quad \lambda\equiv\frac{2E}{\hbar\omega}. \tag{4}$$

B. Two-electron case

The theory of the two-electron case is very similar to that laid out above in Section I A. Begin with Schrödinger's equation for the two electrons in a potential V:

$$\left(-\frac{\hbar^2}{2m}\frac{\partial^2}{\partial r_1^2}-\frac{\hbar^2}{2m}\frac{\partial^2}{\partial r_2^2}+V(r_1,r_2)\right)u(r_1,r_2)=Eu(r_1,r_2). \tag{5}$$

Here, V must include the interaction of each electron with the well and with the other electron:

$$V(r_1,r_2)=\frac{1}{2}m\omega^2r_1^2+\frac{1}{2}m\omega^2r_2^2+\frac{ke^2}{|\mathbf{r}_1-\mathbf{r}_2|}.$$

Equation 5 can be simplified by introducing the relative coordinate $r=|\mathbf{r}_1-\mathbf{r}_2|$ and center-of-mass coordinate $R=(1/2)(\mathbf{r}_1+\mathbf{r}_2)$. This substitution gives

$$\left(-\frac{\hbar^2}{m}\frac{\partial^2}{\partial r^2}-\frac{\hbar^2}{4m}\frac{\partial^2}{\partial R^2}+\frac{1}{4}m\omega^2r^2+m\omega^2R^2+\frac{ke^2}{r}\right)u(r,R)=(E_r+E_R)u(r,R).$$

For simplicity, we'll neglect the center-of-mass motion and write

$$\left(-\frac{\hbar^2}{m}\frac{d^2}{dr^2}+\frac{1}{4}m\omega^2r^2+\frac{ke^2}{r}\right)u(r)=E_ru(r).$$

Like in the one-electron case, we can define a unitless variable $\rho = r/\alpha$ and write

$$\left(-\frac{\hbar^2}{m\alpha^2}\frac{d^2}{d\rho^2} + \frac{1}{4}m\omega^2\alpha^2\rho^2 + \frac{ke^2}{\alpha\rho}\right)u(\rho) = E_r u(\rho),$$

which can be rearranged to get the equation

$$\left(-\frac{d^2}{d\rho^2} + \frac{m^2\omega^4\alpha^4}{4\hbar^2}\rho^2 + \frac{ke^2m\alpha}{\hbar^2\rho}\right)u(\rho) = \frac{m\alpha^2 E_r}{\hbar^2}u(\rho).$$

Define $\alpha = \hbar^2/mke^2$, $\omega_r^2 = m^2\omega^4\alpha^4/4\hbar^2$, and $\lambda = m\alpha^2 E_r/\hbar^2$ to find

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

Equation (6) is nearly identical to (4); the only difference is the replacement of the potential $V(\rho) = \rho^2$ from the one-electron case with the new two-electron potential $V(\rho) = \omega_r^2\rho^2 + 1/\rho$. This implies that we can develop a numerical solution to the general equation

$$-\frac{d^2u}{d\rho^2} + V(\rho)u(\rho) = \lambda u(\rho). \tag{7}$$

and then plug in the two different potentials to solve the one- and two-electron cases.

II. NUMERICAL SOLUTION

To solve (7) numerically, begin by expressing the second derivative using the finite difference equation

$$u''(\rho) = \frac{u(\rho+h) + u(\rho-h) - 2u(\rho)}{h^2} + O(h^2) \tag{8}$$

with step size h. Discretizing (7) between $\rho = 0$ and some arbitrary $\rho_{\max}$ then yields

$$-\frac{u_{i+1} + u_{i-1} - 2u_i}{h^2} + V_i u_i = \lambda u_i \tag{9}$$

for $i = 0, 1, \ldots, N$, $\rho_i = ih$, and $h = \rho_{\max}/N$. Rearranging this equation gives

$$-\frac{1}{h^2}u_{i+1} - \frac{1}{h^2}u_{i-1} + \left(\frac{2}{h^2} + V_i\right)u_i = \lambda u_i, \tag{10}$$

which suggests treating the problem as a system of linear equations

$$\mathbf{A}\mathbf{u} = \lambda\mathbf{u}. \tag{11}$$

Here, the matrix $\mathbf{A}$ is defined as

$$\mathbf{A} = \begin{pmatrix} \frac{2}{h^2} + V_1 & -\frac{1}{h^2} & & & & \\ -\frac{1}{h^2} & \frac{2}{h^2} + V_2 & -\frac{1}{h^2} & & & \\ & -\frac{1}{h^2} & \ddots & \ddots & & \\ & & \ddots & \ddots & -\frac{1}{h^2} & \\ & & & -\frac{1}{h^2} & \frac{2}{h^2} + V_{N-1} \end{pmatrix}, \tag{12}$$

a tridiagonal matrix. The vector $\mathbf{u}$ is simply $\mathbf{u} = \begin{pmatrix} u_1 & u_2 & \ldots & u_{N-1} \end{pmatrix}^T$.

A. Jacobi's rotation algorithm

One way to solve the system of linear equations we've found is by using Jacobi's rotation algorithm. This method iteratively applies similarity transformations to the matrix $\mathbf{A}$ until it becomes diagonal. That is, for orthogonal matrices $\mathbf{S}_i$, the matrix is transformed as

$$\mathbf{S}_m^T\mathbf{S}_{m-1}^T \ldots \mathbf{S}_1^T\mathbf{A}\mathbf{S}_1\mathbf{S}_2 \ldots \mathbf{S}_m = \mathbf{D}$$

where the elements of the final matrix are $D_{ij} = \delta_{ij}$. Naturally, identical transformations must be applied to the eigenvector and eigenvalue to maintain equality, but since the constant eigenvalue can be factored out of all of the matrix multiplications, these similarity transformations do not change the eigenvalues of the matrix.

Jacobi's rotation algorithm, in particular, chooses these orthogonal matrices to be

$$\mathbf{S} = \begin{pmatrix} 1 & 0 & \ldots & 0 & \ldots & 0 & \ldots & 0 \\ 0 & 1 & \ldots & 0 & \ldots & 0 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots & & \vdots & & \vdots \\ 0 & 0 & \ldots & \cos\theta & \ldots & \sin\theta & \ldots & 0 \\ \vdots & \vdots & & \vdots & & \vdots & & \vdots \\ 0 & 0 & \ldots & -\sin\theta & \ldots & \cos\theta & \ldots & 0 \\ \vdots & \vdots & & \vdots & & \vdots & & \vdots \\ 0 & 0 & \ldots & 0 & \ldots & 0 & \ldots & 1 \end{pmatrix}, \tag{13}$$

a generalized rotation matrix with elements

$$S_{ij} = \begin{cases} 1 & i = j, \quad i, j \notin \{p, q\} \\ \cos\theta & (ij = pp) \cup (ij = qq) \\ \sin\theta & ij = pq \\ -\sin\theta & ij = qp \\ 0 & \text{elsewhere.} \end{cases}$$

Note that the $\cos\theta$ elements are not necessarily along the diagonal of the matrix. The values of $\cos\theta$ and $\sin\theta$ can be found by looking at the only elements affected by the transformation:

$$\begin{pmatrix} b_{pp} & b_{pq} \\ b_{qp} & b_{qq} \end{pmatrix} = \begin{pmatrix} c & -s \\ s & c \end{pmatrix}\begin{pmatrix} a_{pp} & a_{pq} \\ a_{qp} & a_{qq} \end{pmatrix}\begin{pmatrix} c & s \\ -s & c \end{pmatrix}.$$

Here, we've defined $c \equiv \cos\theta$ and $s \equiv \sin\theta$. To make the matrix $\mathbf{A}$ diagonal, we want to make $b_{pq} = b_{qp} = 0$. Therefore, we carry out the matrix multiplication and find

$$b_{qp} = a_{pq}(c^2 - s^2) + (a_{pp} - a_{qq})cs = 0. \tag{14}$$

If we define

$$\tau = \frac{a_{qq} - a_{pp}}{2a_{pq}} \tag{15}$$

$$t = \tan\theta = \frac{s}{c} \tag{16}$$

then (14) can be rearranged and written as

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

which has roots

$$t = -\tau \pm \sqrt{\tau^2 + 1} = \frac{1}{\tau \pm \sqrt{\tau^2 + 1}}. \tag{17}$$

To help prevent t from diverging due to loss of numerical precision when $|\tau| \ll 1$, we choose the smaller of the two roots. Thus,

$$t = \begin{cases} \dfrac{1}{\tau + \sqrt{\tau^2 + 1}} & \tau \geq 0 \\ \dfrac{1}{\tau - \sqrt{\tau^2 + 1}} = \dfrac{-1}{-\tau + \sqrt{\tau^2 + 1}} & \tau < 0. \end{cases} \tag{18}$$

Knowing t then gives values for s and c via trigonometric identities

$$c = \frac{1}{\sqrt{1 + t^2}} \quad \text{and} \quad s = tc, \tag{19}$$

and thereby defines the elements of the rotation matrix $\mathbf{S}$. Thus, the elements of the transformed matrix $\mathbf{B} = \mathbf{S}^T\mathbf{A}\mathbf{S}$ are

$$b_{ip} = ca_{ip} - sa_{iq}, \quad i \notin \{p, q\} \tag{20}$$
$$b_{iq} = ca_{iq} + sa_{ip}, \quad i \notin \{p, q\} \tag{21}$$
$$b_{pp} = c^2 a_{pp} - 2csa_{pq} + s^2 a_{qq} \tag{22}$$
$$b_{qq} = c^2 a_{qq} + 2csa_{pq} + s^2 a_{pp} \tag{23}$$
$$b_{pq} = a_{pq}(c^2 - s^2) + (a_{pp} - a_{qq})cs = 0 \tag{24}$$

with equivalent transformations for the elements opposite the diagonal to maintain orthogonality. These equations are the basis of the Jacobi rotation algorithm.

The algorithm itself consists of a few basic steps:

1. Find the largest off-diagonal element in $\mathbf{A}$. The indices of this element are p and q.
2. Calculate τ using (15), $\tan\theta$ using (18), and $\cos\theta$ and $\sin\theta$ using (19).
3. Calculate the new matrix elements b_{ij} using (20–24).
4. Check to see if the norm of the off-diagonal elements is less than some tolerance ϵ. If not, repeat the process until it is.

An implementation of this algorithm in C can be found online in the repository at [1] in the file `project2/src/jacobi.c`.

III. RESULTS

The algorithm described above in Section II A was run with one- and two-electron potentials.

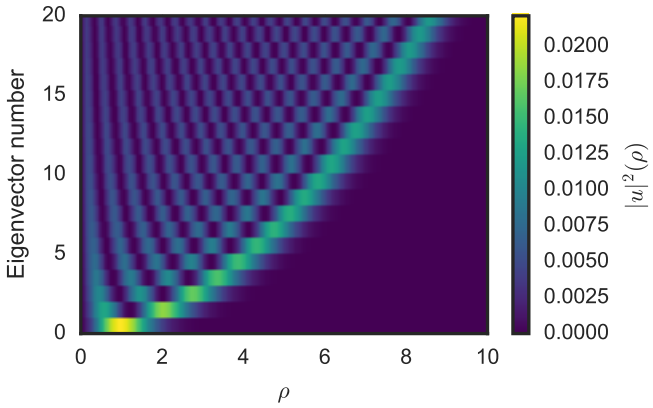

FIG. 1. The probability distributions of the first 20 eigenfunctions of the one-electron system. The distributions are plotted as a function of ρ from left to right, and distributions with increasing energy are plotted from bottom to top.

A. One-electron case

For the one-electron case, the code was run with a maximum ρ of 10 and $N = 375$ grid points. This caused it to converge to the first three expected eigenvalues of 3, 7, and 11 to four significant figures after 228 887 similarity transformations. The probability distributions $|u(\rho)|^2$ corresponding to the first 20 eigenvalues are shown in Figure 1.

To check whether this program is correct, the eigenvalues and eigenvectors were compared to those found by a divide-and-conquer eigenvalue algorithm provided by the Armadillo C++ linear algebra library [2]. The first 20 eigenvalues differ by 10^{-7} at most, which is within the limits of floating-point precision. The probability distributions produced by the algorithm from Armadillo differ from those produced by the Jacobi algorithm by 10^{-10} at most, which implies a difference on the order of 10^{-5} in the magnitude of the eigenvectors. Assuming the results from the Armadillo library are correct, this confirms that the Jacobi algorithm is implemented correctly in this code.

B. Two-electron case

The Jacobi algorithm code was run with the two-electron potential $\omega_r^2\rho^2 + (1/\rho)$ for four values of ω_r: 0.01, 0.5, 1.0, and 5.0. Once again, the maximum value of ρ was set to 10, and the number of grid points was 375. The probability distributions corresponding to the lowest three eigenvalues produced by the code are shown in Figure 4 with the corresponding one-electron distributions for comparison. The one- and two-electron distributions are very similar when $\omega_r = 1.0$ since the two potentials only differ by the addition of the $1/\rho$ term in that case. Otherwise, we can see that the electron distri-

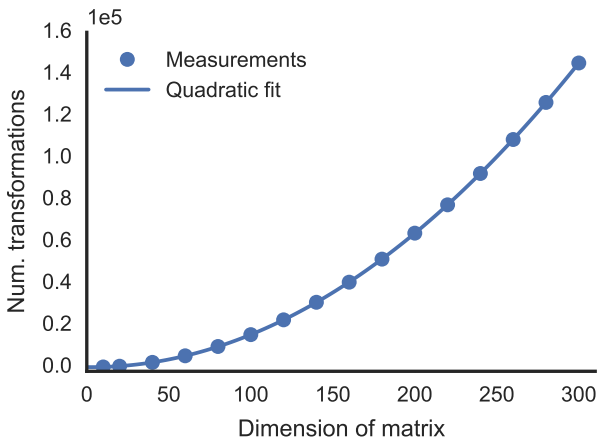

FIG. 2. Scaling of the number of similarity transformations required as a function of the dimension of the matrix. The fit shown is the quadratic function $y = 1.647x^2 - 9.496x + 7.899$.

bution is pushed to higher values of ρ as ω_r decreases. This is because a small value of ω_r allows the $1/\rho$ term to dominate the potential. On the other hand, a strong oscillator potential (large ω_r) will dominate the Coulomb repulsion to drive the distribution toward smaller values of the separation ρ.

C. Performance

The Jacobi code was run for increasing numbers of grid points to measure the code's performance. The number of similarity transformations required to diagonalize the matrix grew as $O(1.647N^2)$ to leading order. A quadratic fit to this data is shown in Figure 2.

We also compared the time taken by the Jacobi code to that taken by the function from Armadillo. The time taken by each program was measured using the Unix `time` utility on a 2012 Mac Mini desktop computer with a 2.3 GHz Intel Core i7 processor. The results are shown in Figure 3 with power-law fits. The Jacobi algorithm was found to scale as $O(N^{3.814})$, while the algorithm from Armadillo scaled as $O(N^{1.817})$. The quality of this measurement is, however, limited by the fact that the `time` utility measures the execution time of the entire program, including the overhead for allocating and filling the matrices. The utility also only measures to a granularity of 1 ms, so it cannot effectively measure the execution time of either algorithm for small N.

[1] J. Bradt, "Comp-phys repository," `https://github.com/jbradt/comp-phys` (2016).
[2] C. Sanderson, *NICTA*, Tech. Rep. (2010).

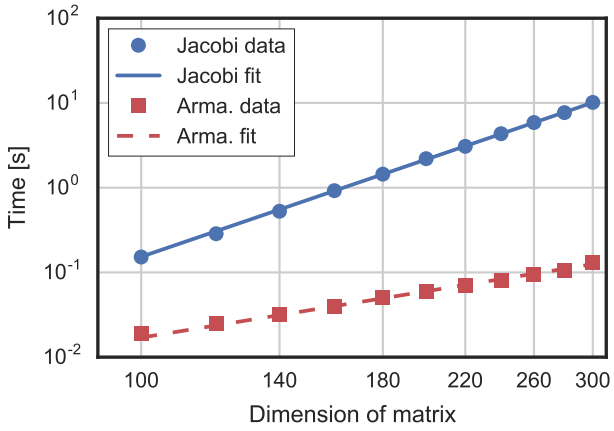

FIG. 3. Time taken by each solver as a function of the dimension of the matrix, shown on a log-log scale. Each set of data was fit with a power law function $y = ax^b$. The fit shown for the Jacobi solver data is the function $t = (3.60 \times 10^{-9})N^{3.814}$, while the fit to the Armadillo data is $t = (3.96 \times 10^{-6})N^{1.817}$. Data for $N < 100$ was not used since the times measured for these small values of N were dominated by overhead and limited by the low precision of the Unix `time` utility.

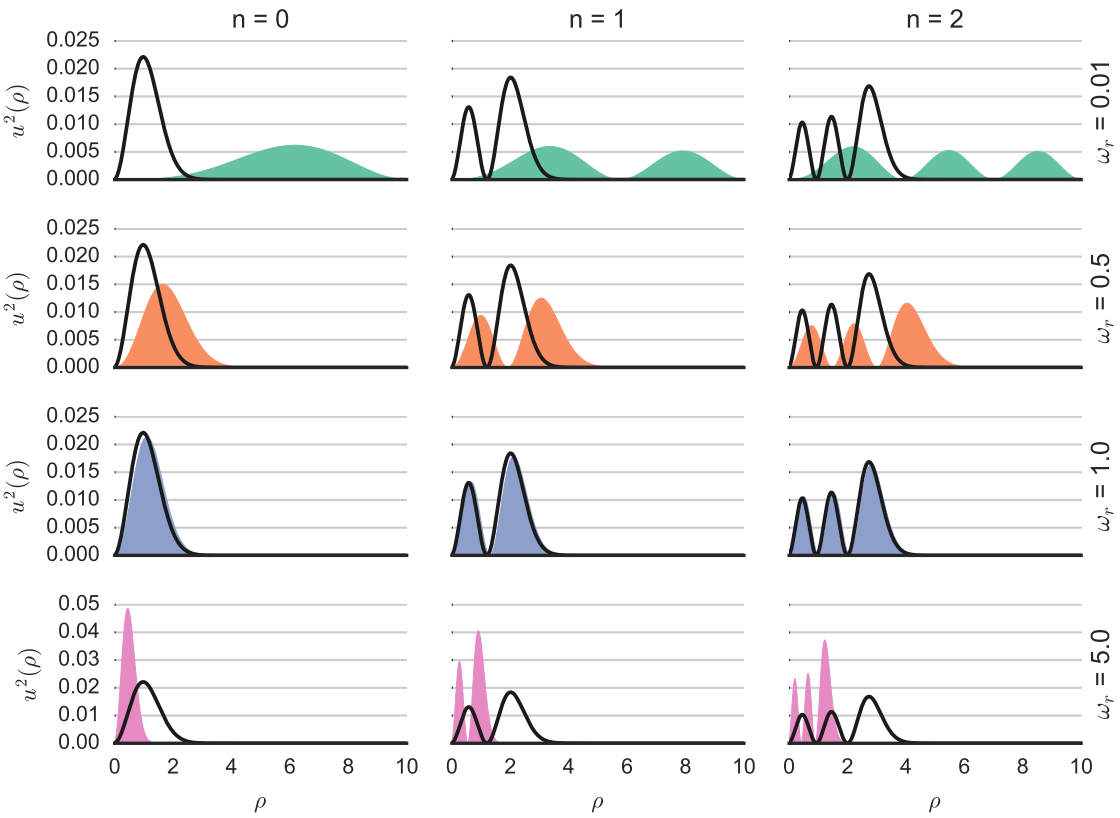

FIG. 4. The three lowest-energy two-electron probability distributions for four values of ω_r, plotted in solid colors. The energy (or the magnitude of the eigenvalue) increases across columns from left to right, while the value of ω_r increases across rows from top to bottom. The solid black line on each plot is the corresponding one-electron probability distribution. The one- and two-electron distributions are very similar for $\omega_r = 1.0$, but they diverge from each other as ω_r moves away from 1.0.