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

Joshua Bradt
(Dated: March 4, 2016)

Expressions for the Hamiltonian of one and two electrons trapped in a harmonic oscillator potential are derived. Jacobi's rotation algorithm for diagonalizing a system of equations is presented, and an implementation of this algorithm in C is used to find the first few eigenvalues and eigenfunctions of the system. The results are compared to results calculated by a function from the Armadillo linear algebra library [1] and to an analytic solution [2]. In both cases, the results calculated here and the results from other sources are in agreement. Finally, the performance of Jacobi's algorithm is measured and found to be much slower than the algorithm used by Armadillo.

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. [3] For the purposes of this work, we will consider the cases of one or two electrons confined by a harmonic oscillator potential, the theory of which is described in Section II. The one-electron case is a simple example of a particle interacting with an external potential, but the two-electron case is more complex due to the Coulomb repulsion between the two negatively charged particles. That being said, both of these cases admit analytic solutions. The solution to the one-electron problem is derived in most introductory quantum mechanics texts (e.g. Shankar [4, p. 351]), but an exact solution for the two-electron problem was only more recently found by Taut [2].

Instead of analytic solutions, it is often easier to seek a numerical solution. One simple way to do this is Jacobi's rotation algorithm, which is derived in Section III. This method produces correct results, and is relatively easy to implement, but it is also quite inefficient, as shown in Section IV.

II. THEORY

A. One electron case

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

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

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,

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

then the energies are known to be

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

with principal quantum number $n=0,1,2,\dots$ and angular momentum quantum number $l=0,1,2,\dots$. [4]

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 effectively 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 an arbitrary 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 $\alpha=\sqrt{\hbar/m\omega}$ and $\lambda=2E/\hbar\omega$ to get

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

B. Two-electron case

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

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

The potential V must include the interaction of each electron with the well and the Coulomb repulsion between the two electrons. This can be written as

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

where $k=1/4\pi\epsilon_0$.

Equation 5 can be simplified by introducing the electron separation distance $r=|\mathbf{r}_1-\mathbf{r}_2|$ and the 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. This leaves

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

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

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

which can be rearranged to get the equation

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

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

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

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

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

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

III. NUMERICAL SOLUTION

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

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

with step size h. For simplicity, define

$$\begin{aligned} u_i &= u(\rho)\\ u_{i\pm1} &= u(\rho\pm h)\\ \rho_i &= ih\\ V_i &= V(\rho_i).\end{aligned}$$

Using these expressions and discretizing (7) between $\rho=0$ and some arbitrary $\rho_{\max}$ then yields

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

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

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

which suggests treating the problem as a system of linear equations

$$\mathbf{A}\mathbf{u}=\lambda\mathbf{u}, \quad (9)$$

where the tridiagonal matrix $\mathbf{A}$ is defined as

$$\begin{pmatrix} \frac{2}{h^2}+V_1 & -\frac{1}{h^2} & & & \\ -\frac{1}{h^2} & \frac{2}{h^2}+V_2 & -\frac{1}{h^2} & & \\ & -\frac{1}{h^2} & \ddots & \ddots & \\ & & \ddots & \ddots & -\frac{1}{h^2} \\ & & & -\frac{1}{h^2} & \frac{2}{h^2}+V_{N-1} \end{pmatrix} \quad (10)$$

and the vector $\mathbf{u}=\begin{pmatrix}u_1 & u_2 & \ldots & u_{N-1}\end{pmatrix}^T$.

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 [3]. 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 final matrix $\mathbf{D}$ is diagonal. Naturally, identical transformations must be applied to the eigenvector 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. [3]

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

$$\mathbf{S} = \begin{pmatrix} 1 & 0 & \dots & 0 & \dots & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 & \dots & 0 & \dots & 0 \\ \vdots & \vdots & & \vdots & & \vdots & & \vdots \\ 0 & 0 & \dots & \cos\theta & \dots & \sin\theta & \dots & 0 \\ \vdots & \vdots & & \vdots & & \vdots & & \vdots \\ 0 & 0 & \dots & -\sin\theta & \dots & \cos\theta & \dots & 0 \\ \vdots & \vdots & & \vdots & & \vdots & & \vdots \\ 0 & 0 & \dots & 0 & \dots & 0 & \dots & 1 \end{pmatrix}, \tag{11}$$

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 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

$$b_{qp} = a_{pq}(c^2 - s^2) + (a_{pp} - a_{qq})cs = 0. \tag{12}$$

If we define

$$\tau = \frac{a_{qq} - a_{pp}}{2a_{pq}} \tag{13}$$

$$t = \tan\theta = \frac{s}{c} \tag{14}$$

then (12) can be rearranged and written as

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

which has roots

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

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,

$$t = \begin{cases} \dfrac{1}{\tau + \sqrt{\tau^2+1}} & \tau \geq 0 \\ \dfrac{1}{\tau - \sqrt{\tau^2+1}} = \dfrac{-1}{-\tau + \sqrt{\tau^2+1}} & \tau < 0. \end{cases} \tag{15}$$

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

$$c = \frac{1}{\sqrt{1+t^2}} \quad \text{and} \quad s = tc, \tag{16}$$

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

$$b_{ip} = ca_{ip} - sa_{iq}, \quad i \notin \{p,q\} \tag{17}$$

$$b_{iq} = ca_{iq} + sa_{ip}, \quad i \notin \{p,q\} \tag{18}$$

$$b_{pp} = c^2 a_{pp} - 2csa_{pq} + s^2 a_{qq} \tag{19}$$

$$b_{qq} = c^2 a_{qq} + 2csa_{pq} + s^2 a_{pp} \tag{20}$$

$$b_{pq} = a_{pq}(c^2 - s^2) + (a_{pp} - a_{qq})cs = 0 \tag{21}$$

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 (13), $\tan\theta$ using (15), and $\cos\theta$ and $\sin\theta$ using (16).
3. Calculate the new matrix elements b_{ij} using (17–21).
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.

IV. RESULTS

The algorithm described in Section III was implemented in C and run for both the one- and two-electron cases. This implementation can be found online in the repository at [5] in the file `project2/src/jacobi.c`. Documentation of how to use the code is available online.

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 the program is correct, the eigenvalues and eigenvectors were compared to those found by a divide-and-conquer eigenvalue algorithm provided by the function `eig_sym` from the Armadillo C++ linear algebra library [1]. The first 20 eigenvalues differ between

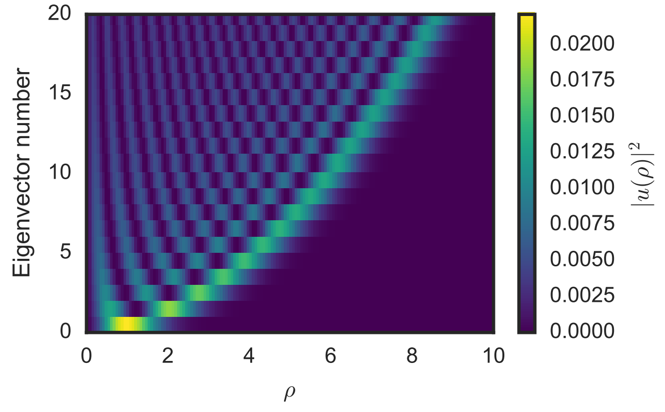

FIG. 1. The probability distributions of the first 20 eigenfunctions of the one-electron system. The distributions are plotted as a function of ρ from left to right, and distributions with increasing energy are plotted from bottom to top. The color corresponds to the magnitude of the probability distribution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

the two codes 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 distribution 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 and drive the distribution toward smaller values of the separation ρ.

The accuracy of the algorithm for the two-electron case can be compared to an exact analytic solution to the problem found by Taut [2]. The code was run for three different values of ω_r which had analytic results listed in Taut. The results of this comparison are shown in Table I.

ω_r	$\rho_{\max}$	Exact	Calculated	Difference
0.25	10	1.250	1.250	0.000
0.05	15	0.3500	0.3510	0.0010
0.018 27	30	0.1644	0.1644	0.0000

TABLE I. Comparison of calculated eigenvalues of the two-electron system to an analytic solution for three values of the parameter ω_r. The exact values are from Taut [2]. (Note that the solutions printed in [2] are a factor of 2 less than the ones shown here. This is due to a difference in how the expressions are derived in that paper, and it has been accounted for already in this table.)

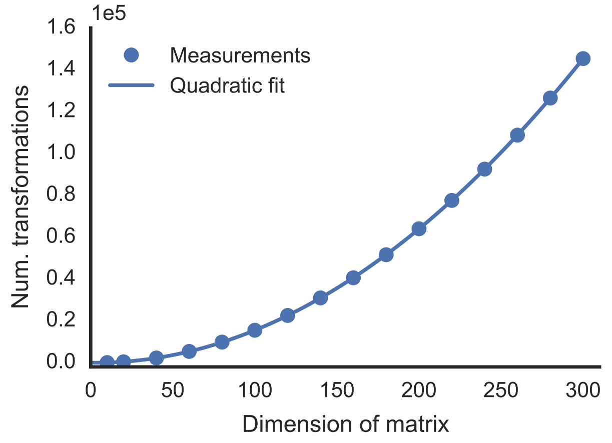

FIG. 2. Scaling of the number of similarity transformations required as a function of the dimension of the matrix. The fit shown is the quadratic function $N_{\text{trans.}} = 1.647N^2 - 9.496N + 7.899$.

The Jacobi code reproduced the analytic eigenvalues almost exactly.

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, where N is the number of grid points (or the dimension of the matrix). 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

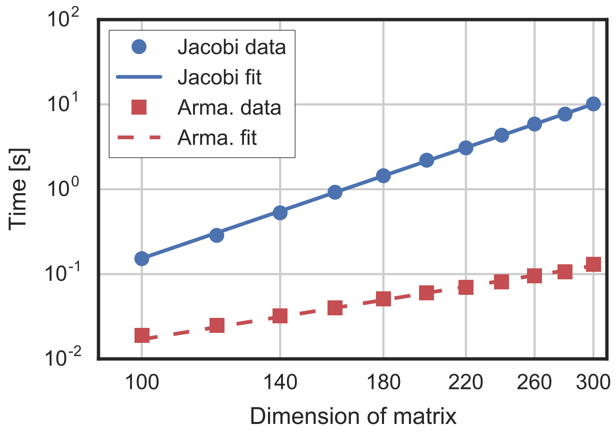

FIG. 3. Time taken by each solver as a function of the dimension of the matrix, shown on a log-log scale. Each set of data was fit with a power law function $y = ax^b$. The fit shown for the Jacobi solver data is the function $t = (3.60 \times 10^{-9})N^{3.814}$, while the fit to the Armadillo data is $t = (3.96 \times 10^{-6})N^{1.817}$. 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.

1 ms, so it cannot effectively measure the execution time of either algorithm for small N.

V. CONCLUSION

While Jacobi's rotation algorithm is a straightforward and easily implemented method for finding the eigenvalues and eigenvectors of a matrix, it is also particularly slow and computationally inefficient. That was apparent in both the timing results, where this algorithm scaled as nearly a quartic function of the dimension of the matrix, and in the number of transformations performed, which scaled quadratically with the dimension of the matrix. Furthermore, the matrix being diagonalized here, which represented the Hamiltonian of one or two electrons confined to a harmonic potential well, was tridiagonal to begin with, which suggests that the algorithm could perform even worse for a general symmetric matrix.

The performance of the algorithm could be improved by parallelizing parts of it. According to a time profile created by Apple's *Instruments* program, the vast majority of the time spent in the algorithm is taken up by finding the largest off-diagonal element of the matrix. This would therefore be a good place to start with multithreading.

That being said, the most obvious way to improve the performance of this process is to simply use a different, more efficient algorithm. The divide-and-conquer algorithm used by Armadillo's `eig_sym` function would be a good candidate as its run time grew as $O(N^{1.817})$, which is significantly better than the Jacobi algorithm.

[1] C. Sanderson, *NICTA*, Tech. Rep. (2010).
[2] M. Taut, Physical Review A **48**, 3561 (1993).
[3] M. Hjorth-Jensen, "Computational Physics MSU," `https://github.com/CompPhysics/ComputationalPhysicsMSU` (2016).
[4] R. Shankar, *Principles of Quantum Mechanics*, 2nd ed. (Springer US, Boston, MA, 1994).
[5] J. Bradt, "Comp-phys repository," `https://github.com/jbradt/comp-phys` (2016).

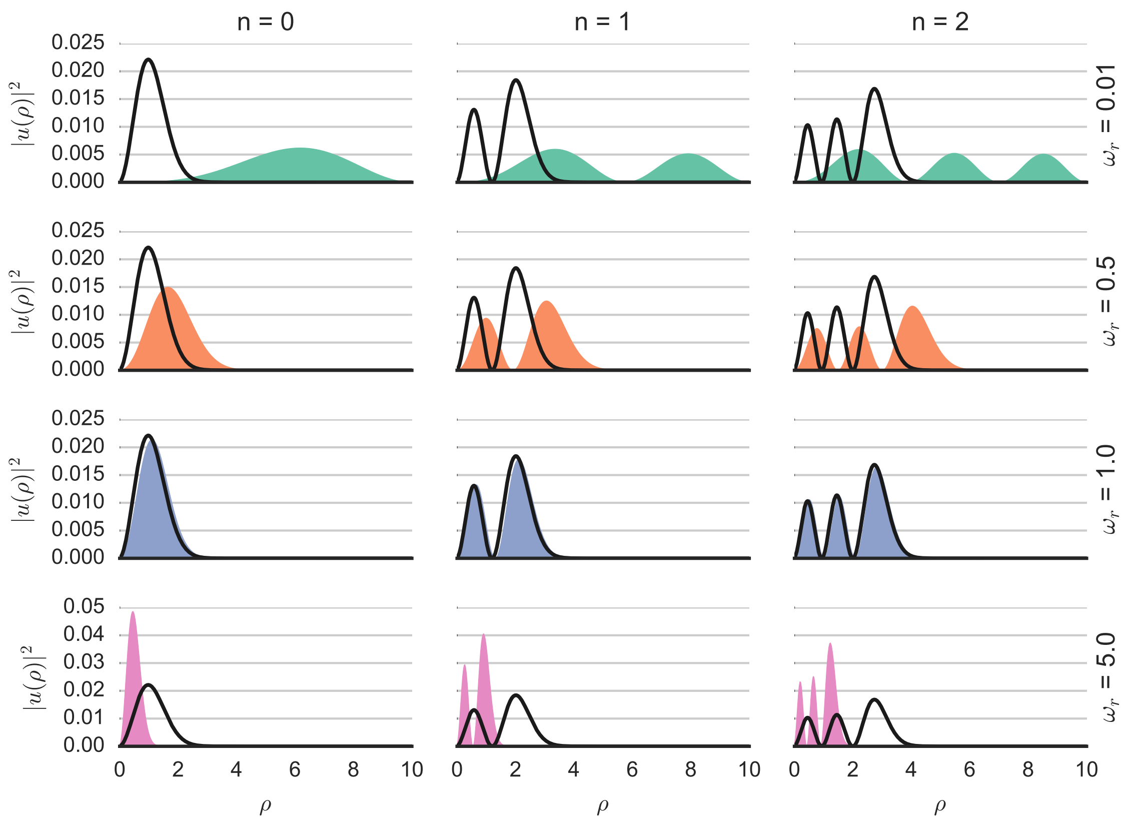

FIG. 4. The three lowest-energy two-electron probability distributions for four values of ω_r, plotted as the filled curves. The energy (or the magnitude of the eigenvalue) increases across columns from left to right, while the value of ω_r increases across rows from top to bottom. The solid black curve 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.