Assignment 1

Introduction: Infinite Potential Well

Due: 2022/01/28

1 Directions

Neglect spin in this problem.

(a) Consider one electron in a one-dimensional, infinitely-deep potential well, with a width of 10 in atomic units. Find the electron wavefunctions and allowed energy levels by analytically solving the Schrödinger equation. Plot the first six wave functions and give their associated eigenvalues.

(b) Same as (a), but this time solve the equation numerically using a language of your choice (Python, Matlab, Maple, Mathematica, Basic, C, Pascal, Fortran, assembly, machine code, etc.). Discretize the width of the potential well into at least 10 segments. Finite difference is preferred for solving the equation numerically.

(c) Same as (b), but this time assume that there are two non-interacting electrons in the well. Note that in this case the wave function will have two variables (the positions of the two electrons).

(d) Same as (c), but this time include the Coulomb interaction between the two electrons. Comment on the solutions and their difference with those obtained in part (c).

2 Solution

For this assignment, we need to find the solutions of the following one-dimensional, time-independent Schrödinger's equation:

$$\hat{H}\Phi_n(x_i) = E_n\Phi_n(x_i)$$

For this assignment, $\hat{H}$ is the Hamiltonian operator for a system of electrons described by their position x_i, E_n is the energy eigenvalue, and $\Phi_n(x_i)$ is the wavefunction that we want to solve for. Note that we will have multiple solutions and we are tasked to obtain the first six. Nuclei are ignored for this problem as only electrons and their interactions are involed, so their contribution is not included in the operator. The expanded form of this Hamiltonian operator for N electrons is as follows (in Hartree atomic units where $\hbar = m_e = e^2 = 1$):

$$\hat{H} = \hat{T} + \hat{U} + \hat{V} \qquad \text{total energy of the system}$$

$$\hat{H} = -\sum_{i=1}^{N} \frac{1}{2}\nabla_i^2 + \sum_{i=1}^{N}\sum_{j>i}^{N} \frac{1}{x_{ij}} + \sum_{i=1}^{N} v(x_i)$$

$$\hat{T} = -\sum_{i=1}^{N} \frac{1}{2}\nabla_i^2 \qquad \text{kinetic energy of each electron}$$

$$\hat{U} = \sum_{i=1}^{N}\sum_{j>i}^{N} \frac{1}{x_{ij}} \qquad \text{Coulomb repulsion between electrons}$$

$$\hat{V} = \sum_{i=1}^{N} v(x_i) \qquad \text{potential energy of each electron}$$

Our one-dimensional, infinitely-deep potential well with a width of 10 atomic units gives us the following boundary conditions for each electron's potential energy $v(x_i)$:

$$v(x_i) = \begin{cases} 0 & |x| < 5a \\ \infty & |x| > 5a \end{cases}$$

2.1 Analytical Solution: Single Electron

For the first two steps, we will analytically solve the Schrödinger equation for a single electron bounded within the 10 atomic unit infinite well to obtain the first six wavefunctions and the associated eigenvalues. For a single electron, the Hamiltonian shown above is simplied as follows:

$$\hat{H} = -\frac{1}{2}\nabla_i^2 + v(x_i)$$

We only have one electron, so the summations are dropped and there is no Coulomb interaction to consider. We can then plug this new Hamiltonian into Schrödinger's equation:

$$\hat{H}\Phi_n(x) = E_n\Phi_n(x)$$
$$-\frac{1}{2}\Phi_n''(x) + v(x)\Phi_n(x) = E_n\Phi_n(x)$$

With the following boundary conditions:

$$v(x_i) = \begin{cases} 0 & |x| < 5a \\ \infty & |x| > 5a \end{cases}$$

We know that the wave function will have the following value at the boundaries of the well (the walls):

$$\Phi_n(\pm 5a) = 0$$

Which leaves us with the following equation to solve for within the well:

$$-\frac{1}{2}\Phi_n''(x) = E_n\Phi_n(x)$$

This is a second order linear differential equation. The general solution to this form of the second order linear differential equation is as follows:

$$\Phi_n(x) = Ae^{ikx} + Be^{-ikx}$$

Plugging in our boundary conditions, we get the following relationships:

$$\begin{aligned}\Phi_n(-5a) &= Ae^{-ik5a} + Be^{ik5a} = 0\\ \Phi_n(5a) &= Ae^{ik5a} + Be^{-ik5a} = 0\end{aligned}$$

With some algebra, we can then get the following relationships:

$$\begin{aligned}A + Be^{2ik5a} = 0 &\Longrightarrow A = -Be^{i10ka}\\ A + Be^{-2ik5a} = 0 &\Longrightarrow A = -Be^{-i10ka}\\ -Be^{i10ka} &= -Be^{-i10ka}\\ e^{i10ka} &= e^{-i10ka}\\ e^{i20ka} &= 1\end{aligned}$$

Using Euler's identity, we know that:

$$e^{i2\pi n} = 1$$

Therefore our possible values for k are as follows:

$$\begin{aligned}2\pi n &= 20ka\\ k &= \frac{\pi n}{10a}\end{aligned}$$

Where n is an integer. With this result, we can progress further into defining the wave function as follows:

$$A = -Be^{i10ka}$$
$$A = -Be^{i\pi n}$$
$$A = -B(e^{i\pi})^n$$
$$A = -B(-1)^n$$
$$\boxed{A = -B} \quad \text{for even n}$$
$$\boxed{A = B} \quad \text{for odd n}$$

$$\Phi_n(x) = Ae^{-ikx} - Ae^{ikx} = 2iA\sin(kx)$$
$$\boxed{\Phi_n(x) = 2iA\sin(\frac{\pi nx}{10a})} \quad \text{for even n}$$
$$\Phi_n(x) = Ae^{-ikx} + Ae^{ikx} = 2A\cos(kx)$$
$$\boxed{\Phi_n(x) = 2A\cos(\frac{\pi nx}{10a})} \quad \text{for odd n}$$

We are left with finding the value for the constant A. We can normalize the wavefunction to calculate this value. This represents the fact that the probability of finding the wave function in all space is equal to 1.

$$1 = \int_{-\infty}^{\infty} |\Phi_n(x)|^2 \, dx$$

Since the problem defines the bounds of the wavefunction to within an infinitely-deep potential well of a fixed size, we can use the bounds of the box as the integration limits.

$$1 = \int_{-5a}^{5a} |\Phi_n(x)|^2 \, dx$$

We can now evaluate the integral for both even and odd cases as follows.

For the even case:

$$
\begin{aligned}
1 &= \int_{-5a}^{5a} |\Phi_n(x)|^2 \, dx \\
1 &= 4\,|A|^2 \int_{-5a}^{5a} \sin^2\left(\frac{\pi n x}{10a}\right) dx \\
1 &= 4\,|A|^2 \int_{-5a}^{5a} \frac{1}{2}\left[1 - \cos\left(\frac{\pi n x}{5a}\right)\right] dx \\
1 &= 2\,|A|^2 \int_{-5a}^{5a} \left[1 - \cos\left(\frac{\pi n x}{5a}\right)\right] dx \\
1 &= 2\,|A|^2 \left[x - \frac{5a}{\pi n}\cancel{\sin\left(\frac{\pi n x}{5a}\right)}\right]_{-5a}^{5a} \\
1 &= 2\,|A|^2\, 10a \\
|A|^2 &= \frac{1}{20a} \\
A &= \frac{1}{\sqrt{20a}}, \frac{-i}{\sqrt{20a}}
\end{aligned}
$$

$$\boxed{\Phi_n(x) = \frac{2}{\sqrt{20a}} \sin\left(\frac{\pi n x}{10a}\right)} \qquad \text{for even n}$$

We choose the imaginary value for A so that we can cancel out the imaginary value in the wave equation to obtain a real result. We can perform the same steps for the odd case as follows:

$$
\begin{aligned}
1 &= \int_{-5a}^{5a} |\Phi_n(x)|^2 \, dx \\
1 &= 4\,|A|^2 \int_{-5a}^{5a} \cos^2\left(\frac{\pi n x}{10a}\right) dx \\
1 &= 4\,|A|^2 \int_{-5a}^{5a} \frac{1}{2}\left[1 + \cos\left(\frac{\pi n x}{5a}\right)\right] dx \\
1 &= 2\,|A|^2 \int_{-5a}^{5a} \left[1 + \cos\left(\frac{\pi n x}{5a}\right)\right] dx \\
1 &= 2\,|A|^2 \left[x + \frac{5a}{\pi n}\cancel{\sin\left(\frac{\pi n x}{5a}\right)}\right]_{-5a}^{5a} \\
1 &= 2\,|A|^2\, 10a \\
|A|^2 &= \frac{1}{20a} \\
A &= \frac{1}{\sqrt{20a}}, \frac{-i}{\sqrt{20a}}
\end{aligned}
$$

$$\boxed{\Phi_n(x) = \frac{2}{\sqrt{20a}} \cos\left(\frac{\pi n x}{10a}\right)} \qquad \text{for odd n}$$

In this case, we choose the real value for A so that the wave equation remains real. We can now plot the first six wave functions for $n = 1, 2, 3, 4, 5, 6$ as follows:

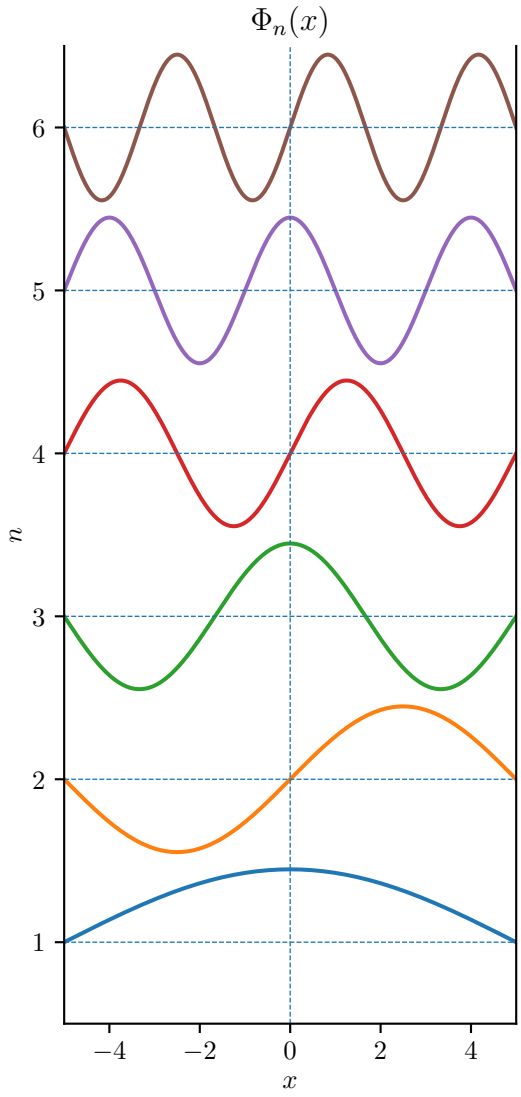


Figure 1: Wave functions for the analytical solution of the one-dimensional, time-independent Schrödinger's describing one electron in an infinitely-deep potential well of width $10a$

For the energy eigenvalues for each of these wavefunctions, we'll need to relate the energy eigenvalue term E_n back to the constant term k that was derived. We can do this by substituting the wave function derived and plug it into the Schrödinger equation that is valid for within the well (where potential energy is equal to 0: $v(x) = 0$) as follows:

n	1	2	3	4	5	6
E_n	E_o	$4E_o$	$9E_o$	$16E_o$	$25E_o$	$36E_o$

Table 1: The first six energy levels calculated for this problem through the analytical method. This will be compared to the following problem's results.

$$
\begin{aligned}
-\frac{1}{2}\Phi_n^{''}(x) &= E_n\Phi_n(x) \\
-\frac{1}{2}\left[\frac{2}{\sqrt{20a}}\sin\left(\frac{\pi nx}{10a}\right)\right]\frac{d^2\Phi_n(x)}{dx^2} &= E_n\frac{2}{\sqrt{20a}}\sin\left(\frac{\pi nx}{10a}\right) \\
-\frac{1}{2}\left[\cancel{\frac{2}{\sqrt{20a}}}\left(-\frac{n^2\pi^2}{100a^2}\right)\cancel{\sin\left(\frac{\pi nx}{10a}\right)}\right] &= E_n\cancel{\frac{2}{\sqrt{20a}}}\cancel{\sin\left(\frac{\pi nx}{10a}\right)} \\
\frac{n^2\pi^2}{200a^2} &= E_n \\
E_o = \frac{\pi^2}{200a^2} & \\
\boxed{E_n = n^2E_o} &
\end{aligned}
$$

We started off with the even n wavefunction, but the odd n wavefunction results in the same energy eigenvalues. Therefore our result is valid for both even and odd n. Our first six eigenvalues are as follows:

2.2 Numerical Solution 1: Single Electron

In order to solve the single electron problem numerically, we must discretize the following equation:

$$
-\frac{1}{2}\Phi_n^{''}(x) + v(x)\Phi_n(x) = E_n\Phi_n(x) \qquad\qquad v(x_i) = \begin{cases} 0 & |x| < 5a \\ \infty & |x| > 5a \end{cases}
$$

To accomplish this, we can replace the second derivative of the wave function in the one-dimensional, time-independent Schrödinger's with a second-order centered difference approximation which has the following form:

$$
\frac{d^2f(x)}{dx^2} \approx \frac{f(x+\Delta x) - 2f(x) + f(x-\Delta x)}{\Delta x^2}
$$

Schrödinger's equation then transforms into a discretized version as follows:

$$
-\frac{1}{2}\left[\frac{\Phi_n(x+\Delta x) - 2\Phi_n(x) + \Phi_n(x-\Delta x)}{\Delta x^2}\right] + v(x)\Phi_n(x) = E_n\Phi_n(x)
$$

This solution relies on a step size Δx with the second derivative approximation requiring a previous value $(x-\Delta x)$ and the next value $(x+\Delta x)$ along with the current value x. By taking into account the entire well,

sliced into N segments of size Δx, each segment will represent a different set of the discretized Schrödinger's equation to solve, which together forms a linear set of equations which can be solved with the help of linear algebra. For a discretization of $N = 10$, we have the following matrices involved:

The kinetic energy matrix:

$$\boldsymbol{T_{N=10}} = -\frac{1}{2\Delta x^2}\begin{bmatrix} -2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & -2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -2 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & -2 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -2 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & -2 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -2 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -2 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -2 \end{bmatrix}$$

And the potential energy matrix:

$$\boldsymbol{V_{N=10}} = \begin{bmatrix} v(x) & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & v(x) & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & v(x) & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & v(x) & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & v(x) & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & v(x) & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & v(x) & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & v(x) & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & v(x) & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & v(x) \end{bmatrix}$$

where

$$v(x) = \begin{cases} 0 & |x| < 5a \\ \infty & |x| > 5a \end{cases}$$

Which results in the following system to solve for:

$$\boldsymbol{T}\,|\Phi\rangle + \boldsymbol{V}\,|\Phi\rangle = \boldsymbol{E}\,|\Phi\rangle$$

or simply:

$$\boldsymbol{H}\,|\Phi\rangle = \boldsymbol{E}\,|\Phi\rangle$$

where $|\Phi\rangle$ is the column vector representing each step of the wave function's discretization:

$$|\Phi\rangle = \begin{bmatrix} \Phi_1 \\ \Phi_2 \\ \Phi_3 \\ \Phi_4 \\ \Phi_5 \\ \Phi_6 \\ \Phi_7 \\ \Phi_8 \\ \Phi_9 \\ \Phi_{10} \end{bmatrix}$$

and $\boldsymbol{H}$ is the Hamiltonian matrix composed of $\boldsymbol{T}+\boldsymbol{V}$. By diagonalizing $\boldsymbol{H}$, we will obtain the eigenvectors as well as the eigenvalues of the system. The eigenvalues will provide the energies and the eigenvectors will provide the associated functions for the discretization chosen for the problem.

When simulating using $N = 10$, we obtain the following wavefunctions and energies:

n	1	2	3	4	5	6
E_n	1	3.91898595	8.52047896	14.43169344	21.17373795	28.20041213

Table 2: The first six energy levels calculated for this problem. They roughly match the coefficients that were obtained and listed in Table 1

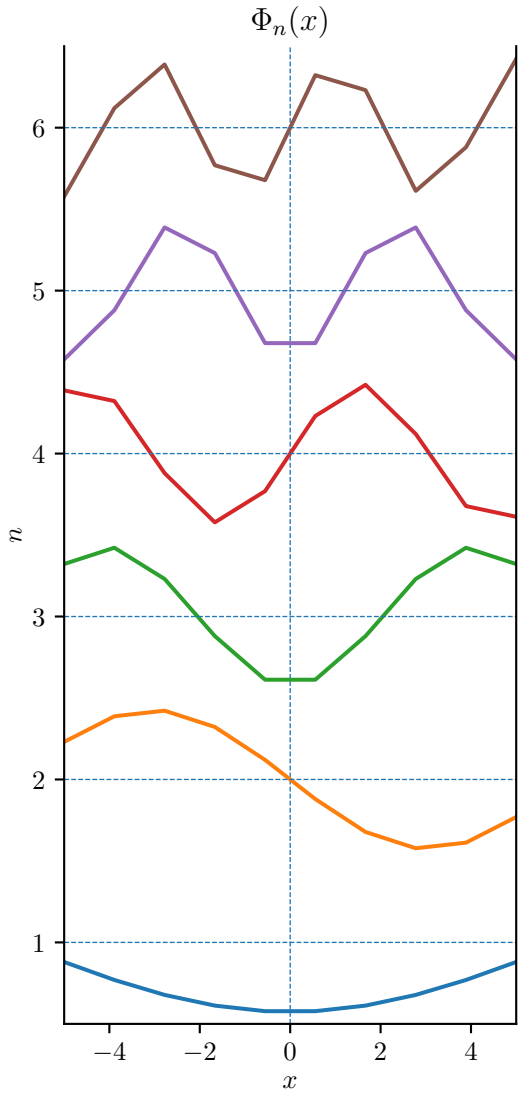


Figure 2: Wave functions for the numerical solution of the one-dimensional, time-independent Schrödinger's describing one electron in an infinitely-deep potential well of width $10a$ and discretization of $N = 10$

2.3 Numerical Solution 2: Two Non-Interacting Electrons

When an additional, non-interacting electron is introduced, the Hamiltonian changes as follows:

$$\hat{H} = -\frac{1}{2}\nabla_1^2 - \frac{1}{2}\nabla_2^2 + v(x_1) + v(x_2)$$

A second electron is introduced into the system, meanining that the wave function will also be composed of two coordinates x_1 and x_2: $\Phi(x_1, x_2)$ The second order differentiation will now have to occur for the wave function with respect to x_1 as well as a second time with respect to x_2. Therefore, using the work outlined previously, the version of the two, non-interacting electron Schrödinger equation is equal to:

$$-\frac{1}{2}\frac{d^2\Phi(x_1,x_2)}{dx_1^2} + -\frac{1}{2}\frac{d^2\Phi(x_1,x_2)}{dx_2^2} + v(x_1)\Phi(x_1,x_2) + v(x_2)\Phi(x_1,x_2) = E\Phi(x_1,x_2)$$

$$v(x_1) = v(x_2) = v(x) = \begin{cases} 0 & |x| < 5a \\ \infty & |x| > 5a \end{cases}$$

Which discretizes as follows:

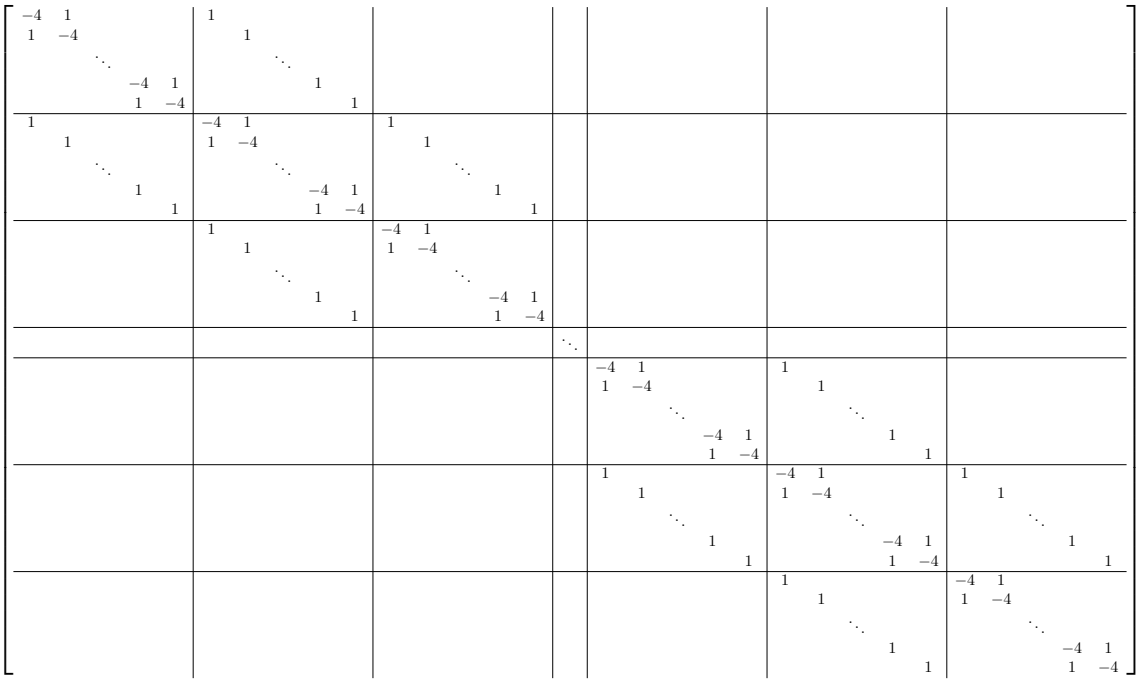
$$-\frac{1}{2}\left[\frac{\Phi(x_1+\Delta x,x_2) - 2\Phi(x_1,x_2) + \Phi(x_1-\Delta x,x_2)}{\Delta x^2}\right] + -\frac{1}{2}\left[\frac{\Phi(x1,x_2+\Delta x) - 2\Phi(x1,x_2) + \Phi(x1,x_2-\Delta x)}{\Delta x^2}\right]$$
$$+2v(x)\Phi(x_1,x_2) = E\Phi(x_1,x_2)$$

$$-\frac{1}{2}\frac{\Phi(x_1+\Delta x,x_2) + \Phi(x1,x_2+\Delta x) - 4\Phi(x_1,x_2) + \Phi(x_1-\Delta x,x_2) + \Phi(x1,x_2-\Delta x)}{\Delta x^2} + 2v(x)\Phi(x_1,x_2) = E\Phi(x_1,x_2)$$

Numerically solving this problem is more complicated than the previous case because now we are solving for two variables: x_1 and x_2. This bears resemblance to solving the one-dimensional problem in two dimensions. To solve this problem, we construct the wave equation column vector as follows:

$$|\Phi\rangle = \begin{bmatrix} \Phi_{0,0} \\ \Phi_{1,0} \\ \vdots \\ \Phi_{N-1,0} \\ \Phi_{N,0} \\ \Phi_{0,1} \\ \Phi_{1,1} \\ \vdots \\ \Phi_{N-1,1} \\ \Phi_{N,1} \\ \vdots \\ \Phi_{0,N} \\ \Phi_{1,N} \\ \vdots \\ \Phi_{N-1,N} \\ \Phi_{N,N} \end{bmatrix}$$

The problem must now take into account the independent positions of the two electrons in the system. This means that we now have $N \times N \times N \times N$ systems of equations to solve for. Using the equation above, the kinetic energy matrix will take the following general form:



This general form is also known as the sparse matrix for the Laplacian differential operator. Obtaining the eigenvalues and the eigenvectors of the system will provide us with the solutions for the equation. Unfortunately, since we now have two electrons with two different possibilities for their current energy levels, it isn't obvious how to discern between the two. For the following numerical answers, the first six energy levels are calculated from lowest to highest, but the specific values for n for each of the electrons is not evident.

n	1	2	3	4	5	6
E_n	1	3.97946748	5.26546441	5.27606611	8.17405936	8.27969253

Table 3: The first six energy levels calculated for this problem, normalized to the first level. All of the solution energies were computed and these are the first six to come out of the sorting process. The associated plots are related to these values through their n values, although the n value doesn't correlate exactly to the actual n for each electron.

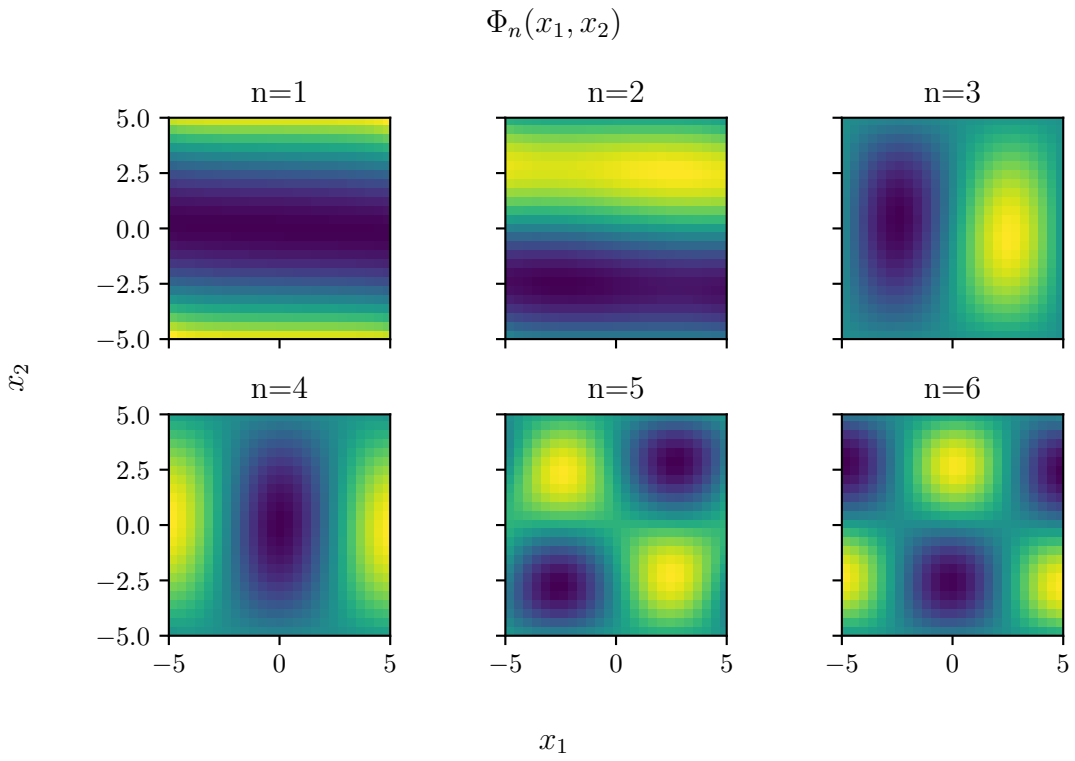


Figure 3: Wave functions for the numerical solution of the one-dimensional, time-independent Schrödinger's describing two non-interacting electrons in an infinitely-deep potential well of width $10a$ and discretization of $N = 25$. Each plot represents two electrons, with the x and y axis corresponding to the position of each electron in the well: $\Phi_n x_1, x_2$.

2.4 Numerical Solution 3: Two Electrons using Coulomb Interaction

For this last problem, we will now introduce the Coulomb repulsion between the two electrons in the system. This adds a third term to the Hamiltonian as follows:

$$\hat{H} = -\frac{1}{2}\nabla_1^2 - \frac{1}{2}\nabla_2^2 + \frac{1}{||\mathbf{x_1} - \mathbf{x_2}||} + v(x_1) + v(x_2)$$

This is straightforward to implement. The magnitude of the distance between the two electrons is

n	1	2	3	4	5	6
E_n	1	1.36143053	8.65467932	1.93060894	2.30840873	2.60201186

Table 4: The first six energy levels calculated for this problem, normalized to the first level. All of the solution energies were computed and these are the first six to come out of the sorting process. The associated plots are related to these values through their n values, although the n value doesn't correlate exactly to the actual n for each electron.

computed as follows:

$$\frac{1}{||\mathbf{x_1} - \mathbf{x_2}||} = \frac{1}{\sqrt{(x_2 - x_1)^2}}$$

In the numerical solution, this is implemented as a matrix with a diagonal, where every element of the diagonal computes the Coulomb repulsion for the given x_1 and x_2 coordinates of each electron. For each row of the matrix, we know what discretization slice is assigned to it.

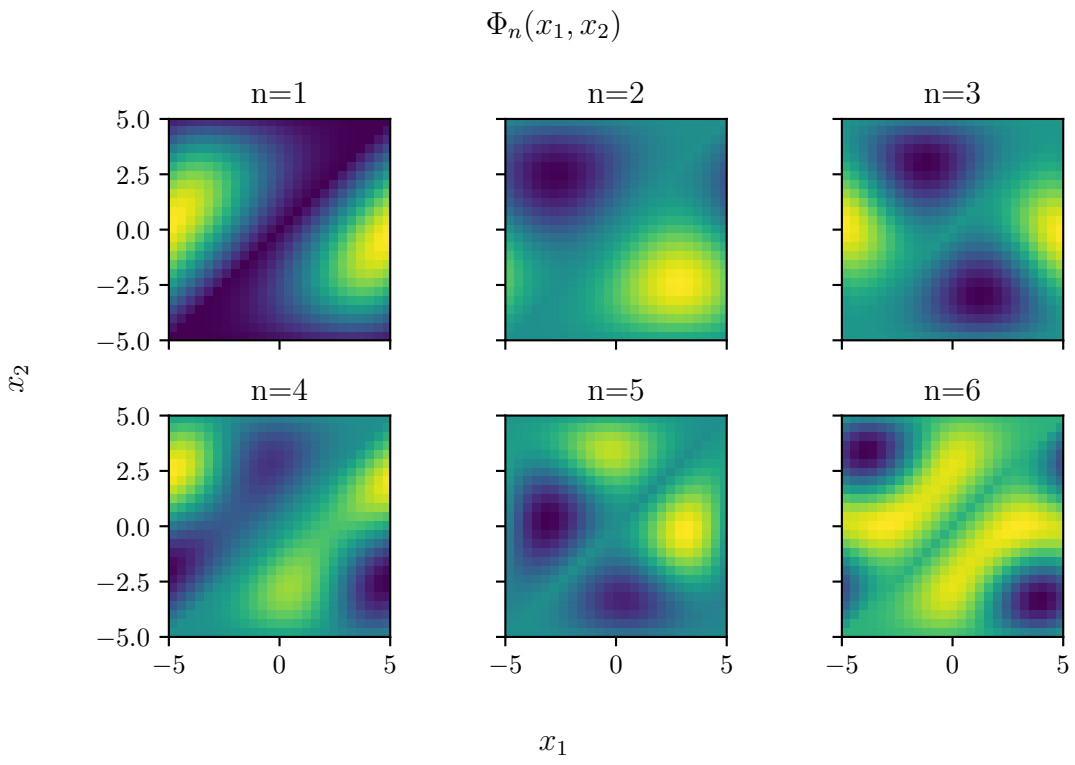


Figure 4: Wave functions for the numerical solution of the one-dimensional, time-independent Schrödinger's describing two interacting electrons (Coulomb repulsion taken into account) in an infinitely-deep potential well of width $10a$ and discretization of $N = 25$. Each plot represents two electrons, with the x and y axis corresponding to the position of each electron in the well: $\Phi_n(x_1, x_2)$.

3 Discussion

- The analytical solution obtains exact answers, but the problem has been greatly simplified. I don't believe exact answers can be easily obtained for more complicated.
- I am unsure of what causes the discretized solution to switch the orientation of the wave function, however, I do recognize that the inverse of the wavefunction is a valid solution to the problem that is presented. By modifying the value for N, I was able to get both orientations of the wavefunction. The likely culprit is the diagonalization algorithm on the Hermitian matrix settling on one solution versus another.
- In the two electron case, it makes sense that the electrons would each have their own energy level, which would lead to NxM different possibilities for the first N=M energy levels of each. However, I am unsure how to extract a specific combination of energy levels from the numerical data. When sorting the energy values from low to high, we are sorting the combination of N and M electrons for both electrons.
- In the two electron case, I expected there to be symmetry in both axes. Especially in the lowest energy case, where I expected a dome-shaped response. I am unsure why we see a linear gradient type of response. This response was evident as the discretization level was increased, so there is likely an error in the implementation or the Hamiltonian.
- When comparing the wavefunctions between the last two numerical solutions, we see that the Coulomb repulsion between the two electrons is highest at $x_1 = x_2$. This manifests as a diagonal line along $x_1 = x_2$ which splits the original wavefunction. Ignoring the sign change between the two wavefunctions, the original wavefunction shape is preserved except for the diagonal where the Coulomb attraction is highest.

4 Code Listings and Data

4.1 Python Code Listing

The following is the code written in Python to generate the solutions and plots used in this report.

```
"""
MIT License

Copyright (c) [2022] [Michel Kakulphimp]

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
"""

import matplotlib
import matplotlib.pyplot as plt
import numpy as np
import math

# Matplotlib export settings
matplotlib.use("pgf")
matplotlib.rcParams.update({
    "pgf.texsystem": "pdflatex",
    "font.size": 10 ,
    "font.family": "serif",  # use serif/main font for text elements
    "text.usetex": True,     # use inline math for ticks
    "pgf.rcfonts": False     # don't setup fonts from rc parameters
})

# potential energy function for an infinitely-deep potential well 10 atomic units wide
def potential_energy_func(x):
    if abs(x) > 5:
        # return np.Inf
        return 1e10
    else:
        return 0

# colomb repulsion energy potential
def colomb_repulsion_func(x1, x2):
    if x1 == x2:
        return 1e10
    else:
        return 1.0/math.sqrt((x2-x1)**2)

# the analytically derived 1-D wave function
def wave_func(a, n, x):

    if (n % 2) == 0:
        # even
        return (2.0/math.sqrt(20.0*a))*math.sin((math.pi*n*x)/(10*a))
```

```
    else:
        # odd
        return (2.0/math.sqrt(20.0*a))*math.cos((math.pi*n*x)/(10*a))

# the analytical derived 1-D wave function calculated for a range of x coordinates
def wave_func_fromiter(offset, a, n, x_coords):
    return np.fromiter((wave_func(a, n, xi) + offset for xi in x_coords), x_coords.dtype)

def main():

    """
    Analytical plot
    """

    x_coords = np.linspace(-5, 5, 1000)
    y_coords_set = []
    for i in range(6):
        y_coords_set.append(wave_func_fromiter(i+1, 1, i+1, x_coords))

    fig, ax = plt.subplots(1, 1, gridspec_kw={'width_ratios':[1], 'height_ratios':[1]})
    for i, y_coords in enumerate(y_coords_set):
        ax.plot(x_coords, y_coords)
        ax.axhline(y=i+1, linewidth=0.5, linestyle='dashed')
    ax.axvline(x=0, linewidth=0.5, linestyle='dashed')
    ax.set_ylim([0.5,6.5])
    ax.set_xlim([-5,5])
    ax.set_title('$\Phi_n(x)$')
    ax.set_xlabel('$x$')
    ax.set_ylabel('$n$')
    ax.spines['top'].set_visible(False)

    fig.set_size_inches(3,6)
    fig.tight_layout()

    fig.savefig('analytical-wave-function-plot.pgf')
    fig.savefig('analytical-wave-function-plot.png')

    """
    Numerical solution 1
    """

    start = -5
    end = 5
    N = 10
    print('Numerical solution 1: start=%d, end=%d, N=%d' % (start, end, N))

    # create our x-axis coordinates array using our limits
    x_coords = np.linspace(start, end, N)
    print('X coordinates:')
    print(x_coords)
    # get delta X
    delta_x = x_coords[1] - x_coords[0]

    # generate kinetic energy matrix
    kinetic_energy_matrix = np.zeros((N,N))
    # fill in the kinetic energy matrix with the coefficients of the
    # second-order centered difference approximation
    for i in range(N):
        for j in range(N):
            if i==j:
                # diagonal is -2 for the -2\Phi(x) term
                kinetic_energy_matrix[i,j]= -2
            elif np.abs(i-j)==1:
                # right outside the diagonal we have 1 on either side for the \Phi(x + \
                    Delta x) and \Phi(x - \Delta x) terms
                kinetic_energy_matrix[i,j]=1

    # generate potential energy matrix
```

```
potential_energy_matrix = np.zeros((N,N))
# fill in the potential energy values along the diagonal using the x values for the
    problem
for i in range(N):
    for j in range(N):
        if i==j:
            potential_energy_matrix[i,j]= potential_energy_func(x_coords[i])

# construct the hamiltonian matrix for solving
hamiltonian_matrix = -kinetic_energy_matrix/(2*delta_x**2) + potential_energy_matrix

# solve our problem, obtain the eigenvectors and eigenvalues
eigenvals, eigenvectors = np.linalg.eig(hamiltonian_matrix)
print('eigenvals:' )
print(eigenvals)
print('eigenvectors:' )
print(eigenvectors)
# sort the eigenvalues from low to high and get their indices
sorted_eigenval_indices = np.argsort(eigenvals)
# get the first six indices, for the first six waveforms
sorted_eigenval_indices = sorted_eigenval_indices[0:6]
# get the eigenvalues, which will be out energies the first value will be
# Eo, so we can divide all the values by this amount to obtain the
# coefficient to compare with the analytical solution
energies=(eigenvals[sorted_eigenval_indices]/eigenvals[sorted_eigenval_indices][0])
print('energies:')
print(energies)
print('sorted␣eigenvalue␣indices')
print(sorted_eigenval_indices)

# plot the results
fig, ax = plt.subplots(1, 1)
ax.axvline(x=0, linewidth=0.5, linestyle='dashed')
ax.set_ylim([0.5,6.5])
ax.set_xlim([-5,5])
for i in range(len(sorted_eigenval_indices)):
    y_coords = []
    y_coords = np.append(y_coords,eigenvectors[:,sorted_eigenval_indices[i]])
    y_coords = [y_coords + (i + 1) for y_coords in y_coords]
    ax.plot(x_coords, y_coords)
    ax.axhline(y=i+1, linewidth=0.5, linestyle='dashed')
ax.set_title('$\Phi_n(x)$')
ax.set_xlabel('$x$')
ax.set_ylabel('$n$')
ax.spines['top'].set_visible(False)

fig.set_size_inches(3,6)
fig.tight_layout()

fig.savefig('numerical-solution-1-wave-function-plot.pgf')
fig.savefig('numerical-solution-1-wave-function-plot.png')

"""
Numerical solution 2
"""

start = -5
end = 5
N = 25
print('Numerical␣solution␣2:␣start=%d,␣end=%d,␣N=%d' % (start, end, N))

# create our x1 and x2 axis coordinates array using our limits
x_coords = np.linspace(start, end, N)
print('X␣coordinates:')
print(x_coords)
# get delta X
delta_x = x_coords[1] - x_coords[0]
```

```
    # generate kinetic energy sparse matrix
    # N = 4 example:
    # (*) phi(1,0) + phi(0,1) -4*phi(1,1) + phi(2,1) + phi(1, 2)
    # +---------+---------+---------+---------+
    # |-4 1 0 0 | 1 0 0 0 | 0 0 0 0 | 0 0 0 0 | phi(0,0)
    # | 1-4 1 0 | 0 1 0 0 | 0 0 0 0 | 0 0 0 0 | phi(1,0)
    # | 0 1-4 1 | 0 0 1 0 | 0 0 0 0 | 0 0 0 0 | phi(2,0)
    # | 0 0 1-4 | 0 0 0 1 | 0 0 0 0 | 0 0 0 0 | phi(3,0)
    # +---------+---------+---------+---------+
    # | 1 0 0 0 |-4 1 0 0 | 1 0 0 0 | 0 0 0 0 | phi(0,1)
    # | 0 1 0 0 | 1-4 1 0 | 0 1 0 0 | 0 0 0 0 | phi(1,1) (*)
    # | 0 0 1 0 | 0 1-4 1 | 0 0 1 0 | 0 0 0 0 | phi(2,1)
    # | 0 0 0 1 | 0 0 1-4 | 0 0 0 1 | 0 0 0 0 | phi(3,1)
    # +---------+---------+---------+---------+
    # | 0 0 0 0 | 1 0 0 0 |-4 1 0 0 | 1 0 0 0 | phi(0,2)
    # | 0 0 0 0 | 0 1 0 0 | 1-4 1 0 | 0 1 0 0 | phi(1,2)
    # | 0 0 0 0 | 0 0 1 0 | 0 1-4 1 | 0 0 1 0 | phi(2,2)
    # | 0 0 0 0 | 0 0 0 1 | 0 0 1-4 | 0 0 0 1 | phi(3,2)
    # +---------+---------+---------+---------+
    # | 0 0 0 0 | 0 0 0 0 | 1 0 0 0 |-4 1 0 0 | phi(0,3)
    # | 0 0 0 0 | 0 0 0 0 | 0 1 0 0 | 1-4 1 0 | phi(1,3)
    # | 0 0 0 0 | 0 0 0 0 | 0 0 1 0 | 0 1-4 1 | phi(2,3)
    # | 0 0 0 0 | 0 0 0 0 | 0 0 0 1 | 0 0 1-4 | phi(3,3)
    # +---------+---------+---------+---------+

    # Sparse matrix is generated using identiy matrices, concatenating columns
    # and rows, and adding it all together.
    main_diagonal = np.identity(N*N)*-4.0
    upper_side_diagonal = np.r_[np.c_[np.zeros((N*N-1,1)), np.identity(N*N-1)], np.zeros((1,
        N*N))]
    lower_side_diagonal = np.r_[np.zeros((1,N*N)), np.c_[np.identity(N*N-1), np.zeros((N*N
        -1,1))]]
    upper_outer_identity = np.r_[np.c_[np.zeros((N*N-N,N)),np.identity(N*N-N)], np.zeros((N,
         N*N))]
    lower_outer_identity = np.r_[np.zeros((N, N*N)), np.c_[np.identity(N*N-N), np.zeros((N*N
        -N,N))]]
    kinetic_energy_sparse_matrix = main_diagonal + upper_side_diagonal + lower_side_diagonal
    kinetic_energy_sparse_matrix = kinetic_energy_sparse_matrix + upper_outer_identity +
        lower_outer_identity

    # generate potential energy matrix
    potential_energy_matrix = np.zeros((N*N,N*N))
    # fill in the potential energy values along the diagonal using the x values for the
        problem
    for i in range(N*N):
        for j in range(N*N):
            if i==j:
                # each N block
                potential_energy_matrix[i,j] = 2*potential_energy_func(x_coords[i%N])

    # construct the hamiltonian matrix for solving
    hamiltonian_matrix = -kinetic_energy_sparse_matrix/(2*delta_x**2) +
        potential_energy_matrix

    # solve our problem, obtain the eigenvectors and eigenvalues
    eigenvals, eigenvectors = np.linalg.eig(hamiltonian_matrix)
    print('eigenvals:' )
    print(eigenvals)
    print('eigenvectors:' )
    print(eigenvectors)

    # now to organize our results
    # each N rows will represent the results for the other dimension's n
    # for example, the first N rows will correspond to \Phi(0,0) through \Phi(N,0)
    for n in range(N):
        # sort the eigenvalues from low to high and get their indices
        # get the first six indices, for the first six waveforms
        # add n*N to each index to properly index the original eigenvals since we slice it
```

```
            up
        # sorted_eigenval_indices.append([x+n*N for x in np.argsort(eigenvals[n:n+N+1][0:6])
            ])
        sorted_eigenval_indices = np.argsort(eigenvals)
        # get the eigenvalues, which will be out energies the first value will be
        # Eo, so we can divide all the values by this amount to obtain the
        # coefficient to compare with the analytical solution
        energies = ((eigenvals[sorted_eigenval_indices]/eigenvals[sorted_eigenval_indices
            ][0]))

    print('energies:')
    print(energies)
    print('sorted␣eigenvalue␣indices')
    print(sorted_eigenval_indices)

    # plot the results
    plots_row = 2
    plots_col = 3
    fig, axes = plt.subplots(plots_row, plots_col, sharex=True, sharey=True)
    for i in range(plots_row * plots_col):
        axes[i//plots_col,i%plots_col].set_title('n=%d' % (i + 1))
        axes[i//plots_col,i%plots_col].set_xlim([-5,5])
        axes[i//plots_col,i%plots_col].set_ylim([-5,5])
        # axes[i//plots_col,i%plots_col].contourf(x_coords, x_coords, eigenvectors[:,
            sorted_eigenval_indices[i]].reshape((N,N)))
        axes[i//plots_col,i%plots_col].imshow(eigenvectors[:,sorted_eigenval_indices[i]].
            reshape((N,N)), origin='lower', interpolation="none", extent=[start,end,start,
            end])
    fig.suptitle('$\Phi_n(x_1,␣x_2)$')
    fig.supxlabel("$x_1$")
    fig.supylabel("$x_2$")

    fig.set_size_inches(6,4)
    fig.tight_layout()

    fig.savefig('numerical-solution-2-wave-function-plot.pgf')
    fig.savefig('numerical-solution-2-wave-function-plot.png')

    """
    Numerical solution 3
    """

    # generate colomb repulsion matrix
    colomb_repulsion_matrix = np.zeros((N*N,N*N))
    # fill in the potential energy values along the diagonal using the x values for the
        problem
    for row in range(N*N):
        for col in range(N*N):
            if row==col:
                # each N block
                colomb_repulsion_matrix[row,col] = colomb_repulsion_func(row%N, col//N)
                # print('colomb_repulsion_func(%d,%d)' % (row%N, col//N))

    # construct the hamiltonian matrix for solving
    hamiltonian_matrix = -kinetic_energy_sparse_matrix/(2*delta_x**2) +
        colomb_repulsion_matrix

    # solve our problem, obtain the eigenvectors and eigenvalues
    eigenvals, eigenvectors = np.linalg.eig(hamiltonian_matrix)
    print('eigenvals:' )
    print(eigenvals)
    print('eigenvectors:' )
    print(eigenvectors)

    # now to organize our results
    # each N rows will represent the results for the other dimension's n
    # for example, the first N rows will correspond to \Phi(0,0) through \Phi(N,0)
    for n in range(N):
```

```
        # sort the eigenvalues from low to high and get their indices
        # get the first six indices, for the first six waveforms
        # add n*N to each index to properly index the original eigenvals since we slice it
            up
        # sorted_eigenval_indices.append([x+n*N for x in np.argsort(eigenvals[n:n+N+1][0:6])
            ])
        sorted_eigenval_indices = np.argsort(eigenvals)
        # get the eigenvalues, which will be out energies the first value will be
        # Eo, so we can divide all the values by this amount to obtain the
        # coefficient to compare with the analytical solution
        energies = ((eigenvals[sorted_eigenval_indices]/eigenvals[sorted_eigenval_indices
            ][0]))

    print('energies:')
    print(energies)
    print('sorted␣eigenvalue␣indices')
    print(sorted_eigenval_indices)

    # plot the results
    plots_row = 2
    plots_col = 3
    fig, axes = plt.subplots(plots_row, plots_col, sharex=True, sharey=True)
    for i in range(plots_row * plots_col):
        axes[i//plots_col,i%plots_col].set_title('n=%d' % (i + 1))
        axes[i//plots_col,i%plots_col].set_xlim([-5,5])
        axes[i//plots_col,i%plots_col].set_ylim([-5,5])
        # axes[i//plots_col,i%plots_col].contourf(x_coords, x_coords, eigenvectors[:,
            sorted_eigenval_indices[i]].reshape((N,N)))
        axes[i//plots_col,i%plots_col].imshow(eigenvectors[:,sorted_eigenval_indices[i]].
            reshape((N,N)), origin='lower', interpolation="none", extent=[start,end,start,
            end])
    fig.suptitle('$\Phi_n(x_1,␣x_2)$')
    fig.supxlabel("$x_1$")
    fig.supylabel("$x_2$")

    fig.set_size_inches(6,4)
    fig.tight_layout()

    fig.savefig('numerical-solution-3-wave-function-plot.pgf')
    fig.savefig('numerical-solution-3-wave-function-plot.png')

if __name__ == "__main__":
    main()
```

5 References

These aren't citing anything, but they were useful in helping me figure out this assignment.

- https://medium.com/modern-physics/finite-difference-solution-of-the-schrodinger-equation-c49039d161a8
- https://www.12000.org/my_notes/mma_matlab_control/KERNEL/KEse82.htm
- https://digitalcommons.calpoly.edu/cgi/viewcontent.cgi?article=1119&context=physsp