Contents

Introduction

The Schrödinger equation is a linear partial differential equation that governs the wave function of a quantum-mechanical system. On applying initial, boundary and potential conditions, it gives us information on how a wave behaves in a quantum realm.
The equation being a second order differential function,it's not always possible to find an analytical solution. Hence we employ numerical methods that involves high level of computation to solve the Schrodinger Equation.

This has already been achieved by Mohandas Pillai, Joshua Goglio, and Thad G. Walkera (Department of Physics, University of Wisconsin-Madison) using Mathematica and they published it in American Journal of Physics[1].

Wolfram Mathematica is a premium paid software system with built-in libraries for several areas of technical computing. Higher cost of Mathematica makes it inaccessible to a student.Hence alternate choice is always preferred.

The aim of this project is to make a python code that does the same using python programming language.

We start by writing down the time-independent schrodinger equation in Hamiltonian form. This looks strikingly similar to a matrix transformation equation where the eigen value is the total energy and the Eigen vector is the wavefunction.
Therefore to find the Eigen vector -which is the wavefunction- we just have to write the hamiltonian and potential in matrix form and solve the matrix equation. This can be done in python by using the numpy.linalg.eig() function in python with the help of the numpy module.

We can do this by first finding the Hamiltonian matrix, which is the sum of Potential energy matrix and Kinetic energy matrix and taking the Eigen vector of that. However, there are some difficulties associated with this as numpy automatically normalises the vectors and we have to undo that to produce the desired results.

Finaly a graph is drawn plotting distance on the X-axis and wavefunction on the Y-axis.

Schrödinger’s Equation[2]

Schrodinger's equation is a fundamental equation in quantum mechanics that describes the behavior of quantum systems, such as atoms and molecules. It is named after Erwin Schrodinger, who developed the equation in 1925.

The equation is a partial differential equation that relates the wave function of a quantum system to its energy. The wave function is a mathematical function that describes the state of a system, and it contains information about the probability of finding a particle at a particular location. The Schrödinger equation tells us how this wave function evolves over time, and it allows us to predict the behavior of quantum systems.

The Schrödinger equation is a linear equation, which means that if we have two solutions, we can add them together to get another solution. This property is essential in quantum mechanics because it allows us to use superposition, which is a fundamental concept in quantum theory. Superposition means that a quantum particle can exist in multiple states simultaneously. For example, an electron can be in a state where it is spinning clockwise and counterclockwise at the same time.

The Schrödinger equation has many applications in physics, chemistry, and engineering. It is used to describe the behavior of atoms, molecules, and other quantum systems. The equation is also used to calculate the energy

levels and wave functions of these systems, which is essential in understanding chemical reactions, the behavior of materials, and the functioning of electronic devices.

Time-Dependent Schrodinger's Equation

The time-dependent Schrodinger equation is a fundamental equation in quantum mechanics that describes the time evolution of a quantum system. It is a partial differential equation that describes how the wave function of a system changes with time. The equation is written in terms of the wave function Ψ, which represents the probability amplitude of finding a particle in a particular position and time.

The time-dependent Schrodinger equation is given by:

$$\frac{-\hbar^2}{2m}\nabla^2\psi(x,t) + V(x,t)\psi(x,t) = -i\hbar\frac{\partial\psi(x,t)}{\partial t}$$

where ħ is the reduced Planck constant
t is time, m is the mass of the particle
∇^2 is the Laplacian operator
V(x, t) is the potential energy of the system

The Laplacian operator represents the curvature of space and is defined as:

$\nabla^2 = (\partial^2/\partial x^2) + (\partial^2/\partial y^2) + (\partial^2/\partial z^2)$
where x, y, and z are the Cartesian coordinates of the particle.

The time-dependent Schrodinger equation is a cornerstone of quantum mechanics, as it provides a way to calculate the time evolution of quantum systems. It has been used to describe a wide range of physical phenomena, including the behavior of atoms, molecules, and solid-state materials. The solution to the time-dependent Schrodinger equation provides information about the probability of finding a particle in a particular position and time, as well as the energy of the system.

Time-Independent Schrodinger's Equation[3]

The time-independent Schrodinger equation is a fundamental equation in quantum mechanics that describes the stationary states of a quantum system. It is a partial differential equation that describes the allowed energy states of a quantum system. The equation is written in terms of the wave function Ψ, which represents the probability amplitude of finding a particle in a particular position.

The time-independent Schrodinger equation is given by:

$$\frac{-\hbar^2}{2m}\nabla^2\psi(x) + V(x)\psi(x) = E\,\psi(x)$$

where ħ is the reduced Planck constant
m is the mass of the particle
∇² is the Laplacian operator
V(x) is the potential energy of the system
E is the energy of the system
x is the position of the particle.

The time-independent Schrodinger equation is an eigenvalue equation, which means that the wave function Ψ is an eigenvector of the Hamiltonian operator H with corresponding eigenvalue E. The solution to the time-independent Schrodinger equation provides information about the allowed energy levels of a quantum system.

The time-independent Schrodinger equation is an important equation in quantum mechanics, as it provides a way to calculate the energy levels of a quantum system. It has been used to describe a wide range of physical phenomena, including the behavior of atoms, molecules, and solid-state materials. The solution to the time-independent Schrodinger equation provides information about the allowed energy levels of a quantum system, which can be used to predict the behavior of the system in various physical situations.

Methods to Solve Schrödinger’s Equation

Solving the Schrödinger equation can provide valuable insights into the properties and behavior of these systems. The equation can be solved analytically or numerically.

Analytical solutions to the Schrödinger equation are obtained by using mathematical techniques such as separation of variables, series expansions, or perturbation theory. Analytical solutions can provide exact solutions that can be expressed in closed-form expressions. However, analytical solutions are often only available for simple systems and can be difficult or impossible to obtain for complex systems.

Numerical solutions to the Schrödinger equation involve approximating the wave function and energy of a system by discretizing space and time. Numerical methods include finite difference methods, finite element methods, and Numerov methods. Numerical solutions can provide accurate solutions for both simple and complex systems. However, numerical methods can be computationally expensive and require significant computational resources.

Both analytical and numerical methods have their advantages and disadvantages. Analytical solutions provide exact solutions and can be used to derive analytical expressions for the wave function and energy. However, analytical solutions are often only available for simple systems and can be challenging to obtain for complex systems. Numerical methods can provide

accurate solutions for both simple and complex systems but can be computationally expensive.

Analytical Methods[5]

1. Separation of variables method

   This method involves assuming that the wave function can be written as a product of functions of different variables, such as position and time, and then solving each function separately.

   Advantage: it is simple, direct, easy to understand, and easy to solve.

   Disadvantage: this is not always the case in complex problems like the boundary value. The separation of variables method becomes convoluted and hard to solve in such instances.

2. Perturbation theory[6]

   This method involves treating the Hamiltonian as a sum of a simple, known Hamiltonian and a perturbation that is small relative to the simple Hamiltonian. The wave function can then be approximated as a series expansion in powers of the perturbation.

   Advantage: perturbation theory is that they provide analytical formulas that describe the influence and role of system parameters on the response.

   Disadvantage: they are limited to relatively narrow, but important, classes of mathematical problems.

3. Variational principle

   This method involves approximating the wave function using a trial function that depends on some adjustable parameters. The parameters are then optimized to minimize the energy of the system.

   Advantage: speed, scalability,novelty.

   Disadvantage: approximate,very little theory around it.

Numerical Methods[7]

1. Finite Difference Method

   This method involves approximating the differential equation by discrete differences, and then solving the resulting algebraic equations using matrix algebra.

   Advantage: it gives higher-order accuracy of the spatial discretization

   Disadvantage: The finite difference method requires a structured grid.

2. Shooting method

   In this method, the Schrodinger equation is transformed into a system of first-order differential equations, which can then be solved using standard numerical methods like the Runge-Kutta method.

   Advantage: it takes advantage of the speed and adaptivity of methods for initial value problems.

Disadvantage: it is not as robust as finite difference or collocation methods

3. Numerov method

   This is a variant of the finite difference method that is particularly well-suited for solving second-order differential equations like the Schrodinger equation.

   Advantage: it gives higher-order accuracy of the spatial discretization

   Disadvantage: The finite difference method requires a structured grid.

In conclusion, the Schrödinger equation can be solved analytically or numerically. Analytical solutions provide exact solutions but are often only available for simple systems. Numerical solutions can provide accurate solutions for both simple and complex systems but can be computationally expensive. The choice of method depends on the complexity of the system and the available computational resources.

Matrix Numerov Method

The time-independent 1-D Schrödinger equation is

$$\frac{-\hbar^2}{2m}\frac{\partial^2 \psi}{\partial x^2} + V\,\psi = E\,\psi \qquad (1.0)$$

which can be rearranged to

$$\frac{\partial^2 \psi}{\partial x^2} = \frac{-2m}{\hbar^2}[E - V(x)]\,\psi(x) \qquad (1.1)$$

which is of the form

$$\psi^{(2)}x = f(x)\,\psi(x) \qquad (1.2)$$

Taylor series expansions of the wave function ψ(x) gives

$$\psi(x \pm d) = \psi(x) \pm d\,\psi^{(1)}(x) + \frac{1}{2!}d^2\,\psi^{(2)}(x) \pm \frac{1}{3!}d^3\,\psi^{(3)}(x) + \frac{1}{4!}d^4\,\psi^{(4)}(x) + \ldots \qquad (1.3)$$

It follows that

$$\psi(x+d) + \psi(x-d) = 2\psi(x) + d^2\psi^{(2)}(x) + \frac{1}{12}d^4\psi^{(4)}(x) + O(d^6) \qquad (1.4)$$

which can be rearranged as

$$\psi^{(2)}(x) = \frac{\psi(x+d) + \psi(x-d) - 2\psi(x)}{d^2} - \frac{1}{12}d^2\psi^{(4)}(x) + O(d^4). \qquad (1.5)$$

substituting (1.5) in (1.2),

$$\psi^{(4)}(x) = \frac{f(x+d)\psi(x+d) + f(x-d)\psi(x-d) - 2f(x)\psi(x)}{d^2} + O(d^2). \tag{1.6}$$

substitution of (1.6) to (1.5) yields...

$$f_i\psi_i = \frac{\psi_{i+1} + \psi_{i-1} - 2\psi_i}{d^2} - \frac{1}{12}(f_{i+1}\psi_{i+1} + f_{i-1}\psi_{i-1} - 2f_i\psi_i) \tag{1.7}$$

where

$$f_{i-1} \equiv f(x-d), \quad f_i \equiv f(x), \quad f_{i+1} \equiv f(x+d),$$
$$\psi_{i-1} \equiv \psi(x-d), \quad \psi_i \equiv \psi(x), \quad \psi_{i-1} \equiv \psi(x-d).$$

rearranging (1.6), we have

$$\frac{\psi_{i+1} + \psi_{i-1} - 2\psi_i}{d^2} = \frac{1}{12}(f_{i+1}\psi_{i+1} + f_{i-1}\psi_{i-1} + 10f_i\psi_i). \tag{1.8}$$

recall that

$$f_{i-1} = -\frac{2m}{\hbar^2}(E - V_{i-1}), \quad f_i = -\frac{2m}{\hbar^2}(E - V_i), \quad f_{i+1} = -\frac{2m}{\hbar^2}(E - V_{i+1}), \tag{1.9}$$

then we have

$$-\frac{\hbar^2}{2m}\frac{\psi_{i-1} - 2\psi_i + \psi_{i+1}}{d^2} + \frac{V_{i-1}\psi_{i-1} + 10V_i\psi_i + V_{i+1}\psi_{i+1}}{12} = E\frac{\psi_{i-1} + 10\psi_i + \psi_{i+1}}{12}. \tag{2.0}$$

which can be written as

$$-\frac{\hbar^2}{2m}A\psi + BV\psi = EB\psi \tag{2.1}$$

where

$$A = \frac{1}{d^2}\begin{pmatrix} -2 & 1 & 0 & 0 & 0 & \cdots \\ 1 & -2 & 1 & 0 & 0 & \cdots \\ 0 & 1 & -2 & 1 & 0 & \cdots \\ 0 & 0 & 1 & -2 & 1 & \cdots \\ 0 & 0 & 0 & 1 & -2 & \ddots \\ \vdots & \vdots & \vdots & \vdots & \ddots & \ddots \end{pmatrix}, \quad B = \frac{1}{12}\begin{pmatrix} 10 & 1 & 0 & 0 & 0 & \cdots \\ 1 & 10 & 1 & 0 & 0 & \cdots \\ 0 & 1 & 10 & 1 & 0 & \cdots \\ 0 & 0 & 1 & 10 & 1 & \cdots \\ 0 & 0 & 0 & 1 & 10 & \ddots \\ \vdots & \vdots & \vdots & \vdots & \ddots & \ddots \end{pmatrix},$$

$$V = \begin{pmatrix} V_1 & 0 & 0 & 0 & 0 & \cdots \\ 0 & V_2 & 0 & 0 & 0 & \cdots \\ 0 & 0 & V_3 & 0 & 0 & \cdots \\ 0 & 0 & 0 & V_4 & 0 & \cdots \\ 0 & 0 & 0 & 0 & V_5 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots & \ddots \end{pmatrix}, \quad \psi = \begin{pmatrix} \psi_1 \\ \psi_2 \\ \psi_3 \\ \psi_4 \\ \psi_5 \\ \vdots \end{pmatrix}.$$

multiplying both sides of equation (2.1) by B^{-1} , we get

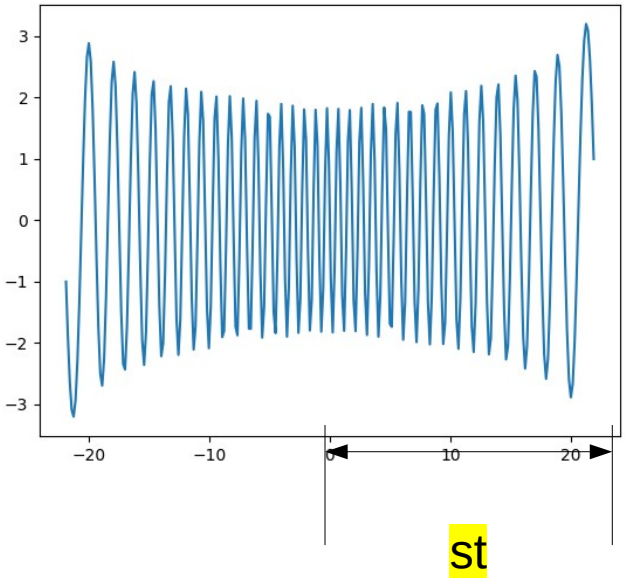
$$-\frac{\hbar^2}{2m}B^{-1}A\psi + V\psi = E\psi \qquad (2.2)$$

which is of the form

$$H\psi = E\psi, \quad H = -\frac{\hbar^2}{2m}B^{-1}A + V$$

where the eigenvector of the Hamiltonian matrix is the time independent wavefunction and the eigenvalues are their corresponding energies.

Determining the Grid



We have to find the scale and number of divisions to make a set of points on the X-axis.

The farthest point needed(st) is the point corresponding to the maximum potential energy which is the Total energy of the system(em).

$V(xmax) = em$

$v(st) = em$

$V(st)$ - $em = 0$

We have to solve this equation to find st.

The step size needed(ds) is proportial to the wavelength

$$\lambda = \frac{h}{P}$$

since,

$$E^2 = \frac{P}{2m}$$

$$P = \sqrt{2Em}$$

Therefore,

$$\lambda = \frac{h}{\sqrt{2Em}}$$

if both h and m is taken to be 1,

$$ds = \frac{1}{\sqrt{2em}}$$

where em is the total energy.

Having found st and ds, the number of grid points,

$$n = 2\left(\frac{st}{ds} + 4\pi\right)$$

Algorithm and Code

1.Define the potential function and Energy

```
#Define the potential and Energy
def v(s): return abs(s)
em = 20.
```

Here we give modulus function as the potential and Total energy to be 20.

2.Determine the grid

```
#determine grid
def f(y): return abs(y)-em
st = float(fsolve(f,em)) #findroot
ds= 1/(sqrt(2*em))
n = int(2*((st/ds)+4*pi))
s = [(-ds*(n+1)/2)+ds*i for i in range(1,n+1)]
```

Solve for the point of maximum potential energy and save the value as st. Find ds and n. Construct the list s to set the X-axis.

3.Calculate KE Matrix

```
#calculate KE matrix
def one(n,d): return np.diagflat([1 for i in range (n-abs(d))], d)
A = (one(n,-1)- 2*one(n,0) +one(n,1))/ds**2
B = (one(n,-1)+ 10*one(n,0) +one(n,1))/12.
Bi = inv(B)
K= np.matmul(A, -Bi)/2
```

Contruct a tridiagonal matrix A with -2 in the central diagonal and 1 in the adjacent diagonals. Similarly a matrix with central diagonal 10 is constructed and saved as B. Find the inverse of B and multiply it with A. Divide them by 2 and save a K.

4.Calculate Hamiltonian and eigenvector

```
#Hamiltonian
V= np.diagflat([v(s[i]) for i in range(n)])
H= V+K
eval, evec = eig(H)
```

Calculate the potential at each point and construct a list. Make a diagonal matrix V with this list as the diagonal. Add the matrix V to matrix K to get the Hamilton matrix,H. Find the Eigenvector of H and save it as evec

5.Denormalize and plot the graph

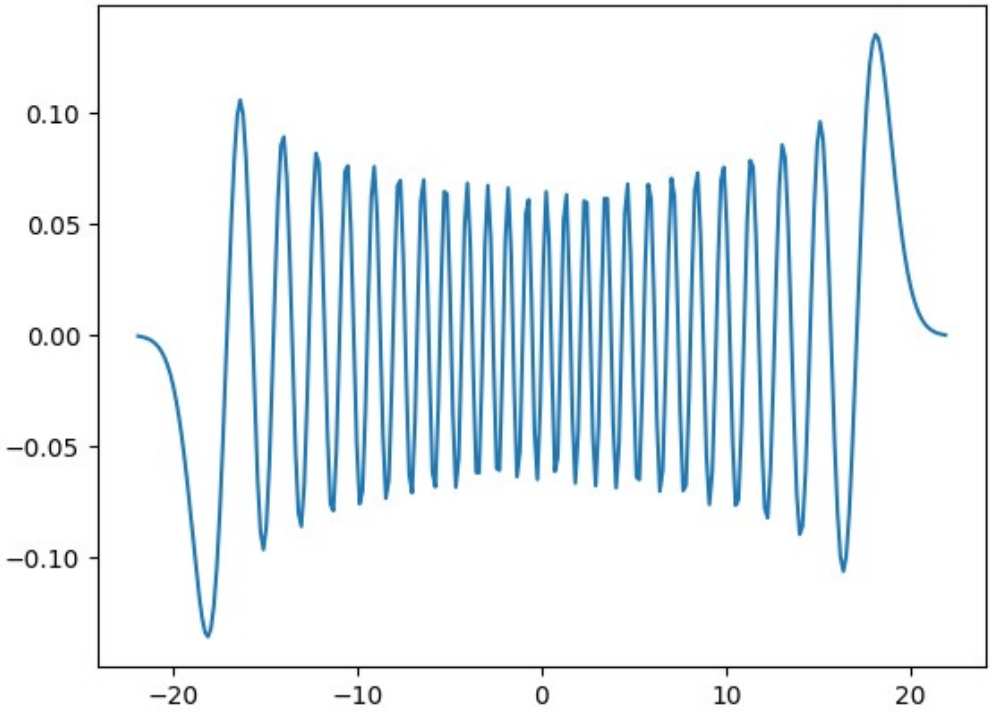
```
#Denormalize the vector
evec = np.transpose(evec)

#Show the plot
plt.plot(s,evec[-20])
plt.show()
```

Transpose the eigen vectors. Plot a graph with s as the X-axis and the eigenvector evec associated with the given energy value as the Y-axis. For example, to get the eigenvector for energy m, plot evec[-m].

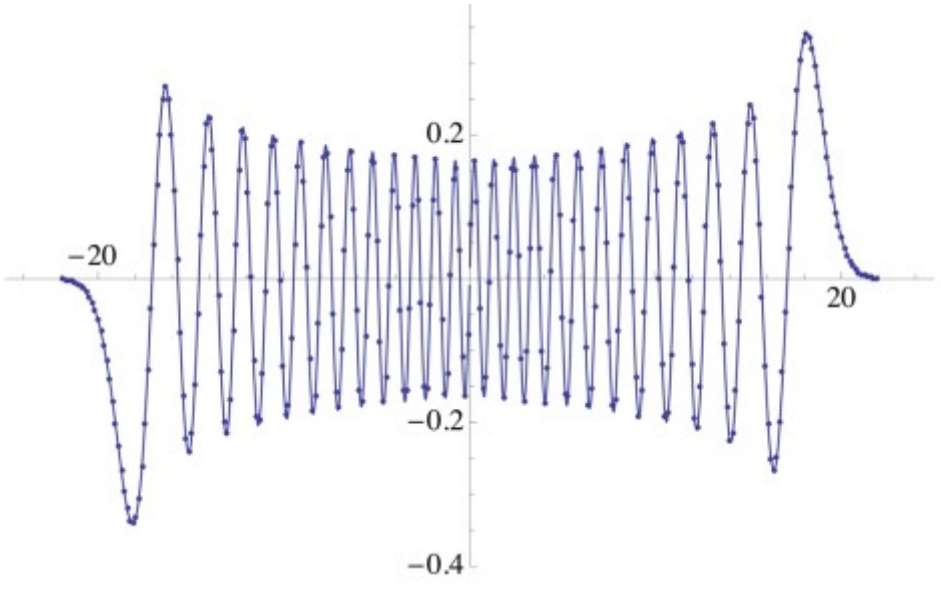
```
from math import sqrt,pi
from scipy.optimize import fsolve
import numpy as np
from numpy.linalg import eig
from numpy.linalg import inv
import matplotlib.pyplot as plt


#Define the potential and Energy
def v(s): return abs(s)
em = 20.

#determine grid
def f(y): return abs(y)-em
st = float(fsolve(f,em)) #findroot
ds= 1/(sqrt(2*em))
n = int(2*((st/ds)+4*pi))
s = [(-ds*(n+1)/2)+ds*i for i in range(1,n+1)]

#calculate KE matrix
def one(n,d): return np.diagflat([1 for i in range (n-abs(d))], d)
A = (one(n,-1)- 2*one(n,0) +one(n,1))/ds**2
B = (one(n,-1)+ 10*one(n,0) +one(n,1))/12.
Bi = inv(B)
K= np.matmul(-Bi, A)/2

#Hamiltonian
V= np.diagflat([v(s[i]) for i in range(n)])
H= V+K
eval, evec = eig(H)

#Denormalize the vector
evec = np.transpose(evec)

#Show the plot
plt.plot(s,evec[-20])
plt.show()
```

Result

Python:



Mathematica:



WHY PYTHON OVER MATHEMATICA

As a student interested in scientific computation, both Mathematica and Python are popular choices for developing scientific applications. Here's a comparison between the two from the perspective of a student:

Ease of Use:
Python has a simpler and more straightforward syntax compared to Mathematica, making it easier to learn and use. This simplicity, along with the vast documentation and user community, makes it an attractive choice for students new to programming. Mathematica, on the other hand, has a steeper learning curve due to its extensive built-in functions and unique syntax.

Functionality:
Mathematica offers a broad range of built-in functions and libraries for scientific computation, including symbolic and numerical computation, graphing, and data analysis. In contrast, Python provides a vast array of third-party libraries and modules, such as NumPy, SciPy, Pandas, and Matplotlib, that provide functionality for scientific computation.

Visualization:
Both Mathematica and Python have excellent visualization capabilities. Mathematica has a comprehensive collection of built-in visualization functions that can create 2D and 3D graphics, animations, and interactive visualizations. Python's Matplotlib library is a popular choice for data visualization, with support for various types of plots, animations, and interactive visualizations.

Community:

Python has a large and active community of developers, researchers, and users who contribute to the development and documentation of the language and its libraries. This community provides support, tutorials, and forums for students to learn and grow their skills in Python. Mathematica also has a sizeable community of users, but its closed-source nature can limit its accessibility to some students.

Price:
Python is open-source and free to use, making it an attractive choice for students on a budget. Mathematica, on the other hand, is a premium software with a high price tag, which can make it less accessible to students.

In conclusion, both Mathematica and Python are powerful tools for scientific computation, and choosing between them depends on the specific needs and preferences of the student. Python's ease of use, extensive third-party libraries, and active community make it an attractive choice for students, while Mathematica's built-in functions, unique syntax, and excellent visualization capabilities make it a popular choice among researchers and professionals in the scientific community.

CONCLUSION

Schrodinger Equation is the central equation of Modern Physics and the foundation of Quantum Mechanics.
The equation being so complex and yet so elegant has been applied to solve breakthrough phenomenons of the Physical world like Quantum Tunneling and Quantum Entanglement.
The ability to plot it's complex wave functions in a programming language that is so simple and free can create a much easier opportunity for students around the world to get into the world of Quatumn Physics.
The Python code that has been written to plot the wave function can be easily modified to fit different kinds of potential functions like the Simple Harmonic Oscillator or Particle in a Box problems. Adding higher order terms from the Matrix Numerov Method gives us access to much higher accuracy in results that we can obtain.

References

[1] *Matrix Numerov Method*
Am. J. Phys. 80, 1017 (2012); doi: 10.1119/1.4748813

[2] *Schrodinger equation*
Sections 5.1 to 5.7 of Modern Physics by Kenneth Krane

[3]Griffiths, D. J. (2005). Introduction to quantum mechanics (2nd ed.). Pearson Prentice Hall. Chapter 2.

[4] Holmes, Mark H. (2013). Introduction to perturbation methods (2nd ed.). New York: Springer. ISBN . OCLC 821883201.

[5] "Quantum Mechanics: Concepts and Applications" by Nouredine Zettili.

[6] "Quantum Mechanics" by Leonard Schiff.

[7] "Numerical Methods for Schrödinger Equations" by Eitan Tadmor.

[8] A Guide to Feynman Diagrams in the Many-Body Problem" by Richard D. Mattuck.

Bibliography

Giovanni Moruzzi - Essential Python for the Physicist-Springer (2020)

Guttag Introduction to Computation and Programming Using Python Revided Expanded The MIT Press 2013

Mark Newman - Computational Physics-(2012)

pythonForEducation_Ver3

Swaroop C H-A Byte of Python

VI_Landau_Computational_Physics_Extended_Ebook

VVI_ Landau_ Manuel Jose Paez - Computational Problems for Physics_ With Guided Solutions Using Python

Numpy documentation
https://numpy.org/doc/

Wolfram language syntax
https://reference.wolfram.com/language/guide/Syntax.html