# Experiment 10 : Numerical Solution of the Time-Independent 1-D Schrödinger Equation

JAGAN P [23MPI0008]

Vellore Institute Of Technology

April 24, 2024

## Abstract

This computational project involves the use of the Finite Difference Method to find eigenvalues and eigenfunction , energy and wavefuntion respectively , of a particle in an infinite square well.
The Programming Language Used to solve the equation numerically is Python.

## Introduction

### The Schrödinger Equation

The Schrödinger equation is a linear partial differential equation that governs the wave function of a quantum-mechanical system

Schrödinger's Wave Equation for a 3-Dimensions :

$$-\frac{\hbar^2}{2m}\nabla^2\psi + V\psi = \hat{E}\psi \tag{1}$$

Where ,
$\psi$ is the Wavefunction
$V$ is the Potential Energy
$E$ is the Energy of the Wave

Question 1

Find the Solution To 1-Demensional Schröridenger Wave Equation for a particle in infinite square potential , Numerically

### Mathamatical Formulation of Finite Difference Method

Consider a particle tapped in a 1-D infinite potential well with boundry conditions $0 \leq x \leq L$ Where the potential outside the boundry is infinite and inside the well it is 0
$V(x)$ is the potential

$$V(x) = \begin{cases} \infty & x < 0 \\ 0 & 0 \leq x \leq L \\ \infty & x > L \end{cases}$$

Then $\psi(x)$ is the Wave Equation

$$\psi(x) = \begin{cases} 0 & x < 0 \\ \psi & 0 \leq x \leq L \\ 0 & x > L \end{cases}$$

**Schrödinger Equation for a particle in 1-Dimension :**

$$-\frac{\hbar^2}{2m}\frac{d^2\psi}{dx^2} + V\psi = E\psi \tag{2}$$

This Diffential Equation can be solved with the Finite Difference method.

**Finite Difference Method:**

Let us take a small distance $dx$ , $n$ timees in $L$ , such that

$$dx = \frac{x_n - x_0}{n}$$

These finite difference expressions leads to a system of n+1 linear algebraic equations if the differential equation is linear

$$\ddot{f} = \frac{f(x+dx) - 2(x) + (x-dx)}{(dx)^2}$$

$\frac{d^2\psi}{dx^2}$ Can be represented by

$$\ddot{\psi}(x_i) = \frac{\psi(x_{i+1}) - 2\psi(x_i) + \psi(x_{i-1})}{(dx)^2} \tag{3}$$

Rewritting $(2)$ in terms of $\ddot{\psi}(x_i)$

$$\frac{\hbar^2}{2m}\ddot{\psi}(x_i) + V(x_i)\psi(x_i) = E\psi(x_i) \tag{4}$$

Substituting Eq $(3)$ in Eq $(4)$

$$-\frac{\hbar^2}{2m(dx)^2}\left[\psi(x_{i+1}) - 2\psi(x_i) + \psi(x_{i-1})\right] + V(x_i)\psi(x_i) = E\psi(x_i)$$

Rearranging the equation

$$\left[2 + \frac{2m(dx)^2\ V(x_i)}{\hbar^2}\right]\psi(x_i)\ -\ \psi(x_{i+1}) - \psi(x_{i-1}) = \frac{2m(dx)^2}{\hbar^2}E\psi(x_2)$$

For i = 1 :

$$\left[2 + \frac{2m(dx)^2\ V(x_1)}{\hbar^2}\right]\psi(x_1)\ -\ \psi(x_2) = \frac{2m(dx)^2}{\hbar^2}E\psi(x_2)$$

For i = 2 :

$$\left[2+\frac{2m(dx)^2\,V(x_2)}{\hbar^2}\right]\psi(x_2)\;-\;\psi(x_3)-\psi(x_1)=\frac{2m(dx)^2}{\hbar^2}E\psi(x_2)$$

For i = n-1 :

$$\left[2+\frac{2m(dx)^2\,V(x_{n-1})}{\hbar^2}\right]\psi(x_{n-1})\;-\;\psi(x_{n-2})=\frac{2m(dx)^2}{\hbar^2}E\psi(x_{n-1})$$

We can model these system of eq in a linear system :

$$\begin{bmatrix} 2+\frac{2m(dx^2)}{\hbar^2}V(x_i) & -1 & 0 & 0... \\ -1 & 2+\frac{2m(dx^2)}{\hbar^2}V(x_i) & -1 & 0... \\ ... & ... & ... & -1 \\ ...0 & 0 & -1 & 2+\frac{2m(dx^2)}{\hbar^2}V(x_i) \end{bmatrix} \begin{bmatrix}\psi(x_1)\\ \psi(x_2)\\ ...\\ \psi(x_n)\end{bmatrix} = \frac{2m(dx)^2E}{\hbar^2}\begin{bmatrix}\psi(x_1)\\ \psi(x_2)\\ ...\\ \psi(x_n)\end{bmatrix}$$

Since the Potential is 0 inside the well

$$2+\frac{2m(dx^2)}{\hbar^2}V(x_i)$$

At $V(x_i)=0$

$$2$$

Then the Linear system reduces to :

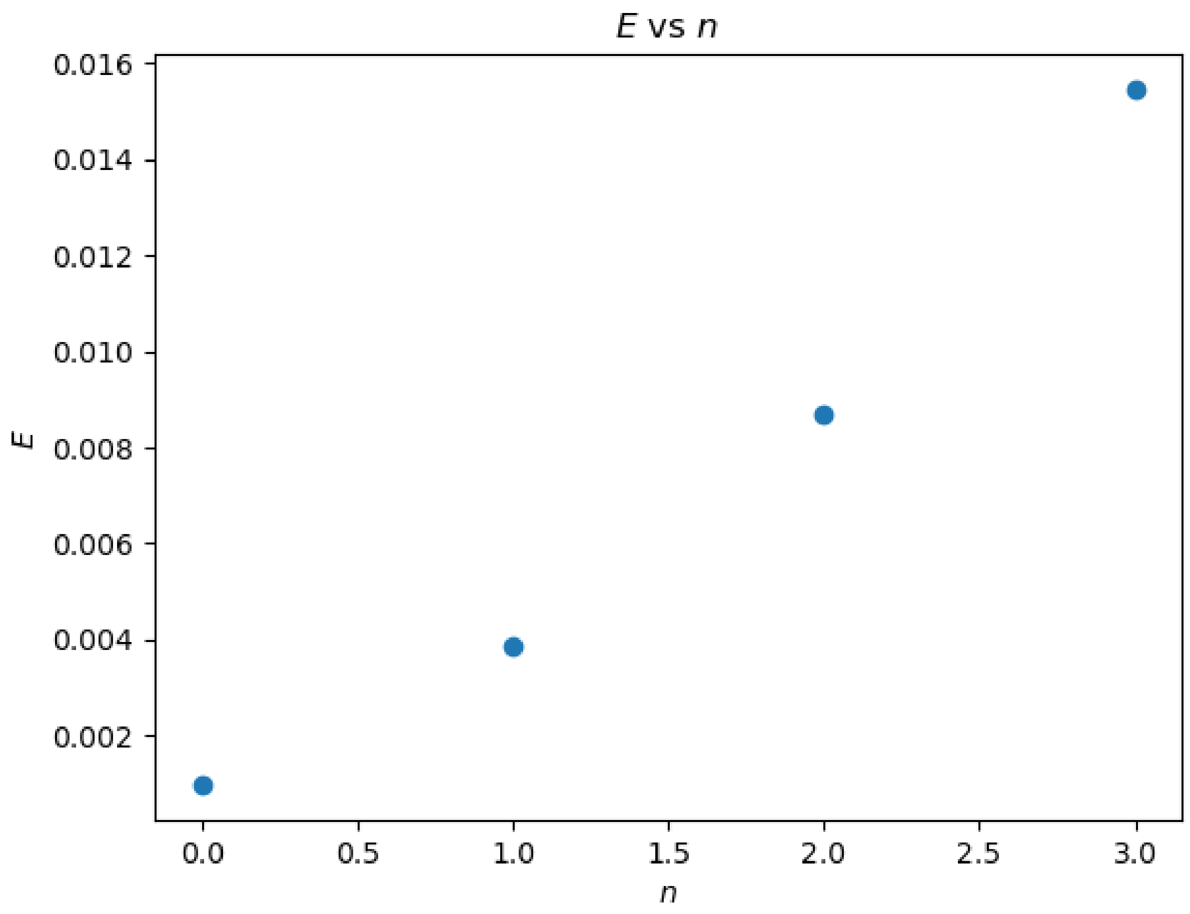
$$\begin{bmatrix} 2 & -1 & 0 & 0... \\ -1 & 2 & -1 & 0... \\ ... & ... & ... & -1 \\ ...0 & 0 & -1 & 2 \end{bmatrix} \begin{bmatrix}\psi(x_1)\\ \psi(x_2)\\ ...\\ \psi(x_n)\end{bmatrix} = \frac{2m(dx)^2E}{\hbar^2}\begin{bmatrix}\psi(x_1)\\ \psi(x_2)\\ ...\\ \psi(x_n)\end{bmatrix}$$

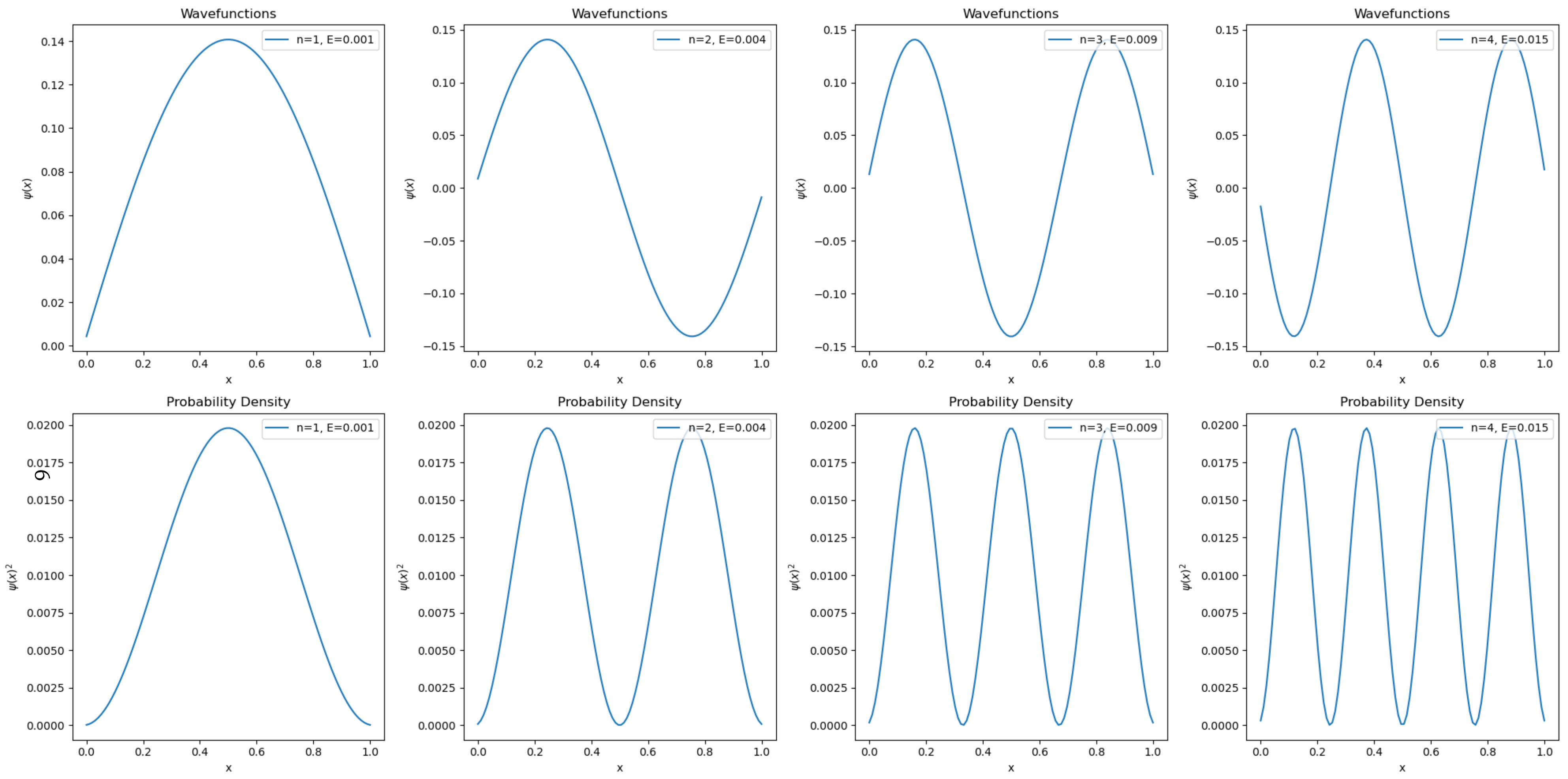
## PYTHON CODE :

Now We can find the eigen value $E$ and the eigenvector $\psi(x)$ with the Following Python code

Listing 1: Python Code

```
# Required Libraries

import numpy as np                            # To create matrix
import matplotlib.pyplot as plt               # To Plot graph
from scipy.linalg import eigh_tridiagonal     # To find the
                                              # eigenvalues and
                                              # eigenvectors




# Constants
L = 1.0 # Maximum Length
N = 100 # number of points in the grid
dx = L / (N - 1)  # grid spacing
x = dx * np.arange(N)




#Hermitian Matrix
diagonal = 2*np.ones(N)
off_diagonal = -1 * np.ones(N - 1)
H = np.diag(diagonal) +  np.diag(off_diagonal, 1) + np.diag(
   off_diagonal, -1)




# Solve for energy eigenvalues and wavefunctions
E, psi = eigh_tridiagonal(diagonal , off_diagonal)



# Wavefuntion and Probability Desity plot
fig, axs = plt.subplots(2, 4, figsize=(20, 10))
for i in range(4):
    axs[0, i].plot(x, psi[:, i], label=f"n={i+1}, E={E[i]:.3f
       }")
    axs[0, i].set_xlabel("x")
    axs[0, i].set_ylabel("$\psi(x)$")
    axs[0, i].set_title("Wavefunctions")
    axs[0, i].legend(loc='upper right')

    axs[1, i].plot(x, psi[:, i]**2, label=f"n={i+1}, E={E[i
        ]:.3f}")
    axs[1, i].set_xlabel("x")
    axs[1, i].set_ylabel("$\psi(x)^2$")
    axs[1, i].set_title("Probability Density")
    axs[1, i].legend(loc='upper right')

plt.tight_layout()
plt.show()

#Energy plot
plt.scatter(x[0:4], E[0:4])
plt.title("$E$ vs $x$ ")
plt.xlabel("$x$")
plt.ylabel("$E$")
plt.show()
```




Wavefunctions
n=1, E=0.001
n=2, E=0.004
n=3, E=0.009
n=4, E=0.015
$\psi(x)$
Probability Density
$\psi(x)^2$
x