Shreyansh – B23PH1024

Lab-3 : Solving ODE / PDE

Aim :- Use numerical methods (Runge-Kutta) to solve ODE/PDE.

About PDE :- Schrödinger Eqn (Time Independent) for particle in a 1-D box :

$$\frac{\partial^2 y}{\partial x^2} - \frac{2m}{\hbar^2}(V-E)y = 0$$

where $V(x) = \begin{cases} 0 & 0<x<L \\ \infty & \text{O.W.} \end{cases}$

$$\therefore E_n = \frac{n^2 h^2}{8mL^2}$$

Procedure :- 1. For $0<x<L$: $\frac{\partial^2}{\partial x^2} y + \frac{2mE}{\hbar^2} y = 0$

$$\therefore \frac{\partial^2}{\partial x^2} y + K^2 y = 0 \qquad K = \sqrt{\frac{2mE}{\hbar^2}}$$

Boundary Cond., $y(0) = y(L) = 0$

11. Let $y_1 = y \quad y_2 = y' = y_1'$

$\therefore$ We get 2 eqn : $y_2 = y_1'$

$$y_2' = -k^2 y_1$$

We do this as can solve 1st order eqn numerically, so for 2nd order we

first convert to 2 1st-order & solve them simultaneously.

iii. So we need to simultaneously solve $y_2 = y_1'$ & $y_2' = -k^2 y_1$ for y_2 & y_1.

iv. Assuming we already know En i.e. K we start with

$$Y = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}, \quad Y_0 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

v. For $x \in \{0, h, 2h, \ldots, L\}$ [h → steps not 6.626×10^{-34} Js]

$$\psi[k] = Y[0]$$

$$K1 = f(x, y)$$

$$K2 = f\left(x + h/2,\ y + h * K_1/2\right)$$

$$K3 = f\left(x + h/2,\ y + h * K_2/2\right)$$

$$K4 = f(x + h,\ y + h * K_3)$$

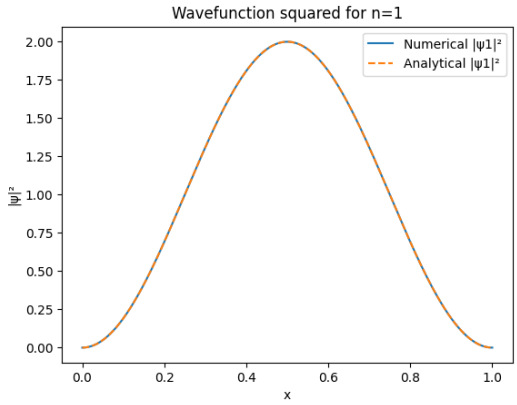
vi. Here $f(x, y) = \begin{bmatrix} y \\ -k^2 y \end{bmatrix}$ ⟶ From the 2 eqn

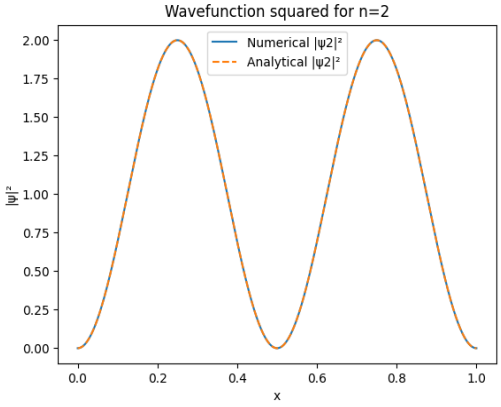
vii. Hence we get $\psi(x)$ for $0, h, 2h, \ldots, L$

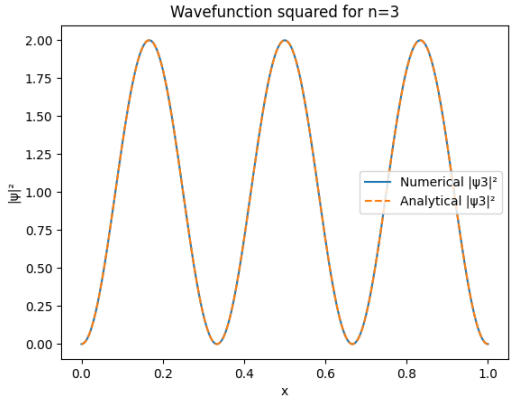
Lab3-Differential_Eqn.py

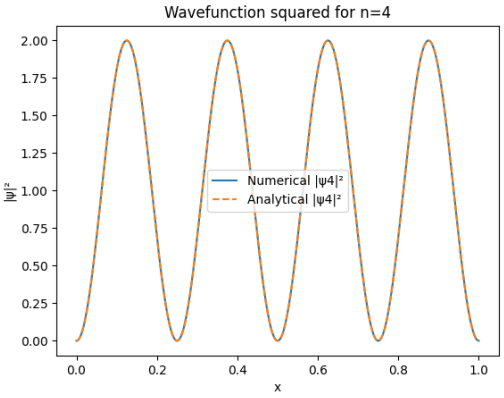
```python
import numpy as np
import matplotlib.pyplot as plt

def f(x, y, k2):
    y1, y2 = y
    return np.array([y2, -k2 * y1])

def TISE_solver(E, m, L, h=1e-3):
    hbar = 6.626e-34 / (2 * np.pi)
    k2 = (2 * m * E) / (hbar**2)

    x_vals = np.arange(0, L + h, h)
    psi_vals = []

    y = np.array([0.0, 1.0])

    for x in x_vals:
        psi_vals.append(y[0])

        K1 = f(x, y, k2)
        K2 = f(x + h/2, y + h*K1/2, k2)
        K3 = f(x + h/2, y + h*K2/2, k2)
        K4 = f(x + h, y + h*K3, k2)

        y = y + (h/6) * (K1 + 2*K2 + 2*K3 + K4)

    psi_vals = np.array(psi_vals)

    norm = np.trapezoid(psi_vals**2, x_vals)
    psi_vals = psi_vals / np.sqrt(norm)

    return np.array(x_vals), psi_vals

def Energy(m, L, n):
    hbar = 6.626e-34 / (2 * np.pi)
    return (n**2 * np.pi**2 * hbar**2) / (2 * m * L**2)

def expected_psi(n, L, x):
    return np.sqrt(2 / L) * np.sin((n * np.pi * x) / L)

mass = 1.0
length = 1.0

for n in [1, 2, 3, 4]:
    x, psi_num = TISE_solver(Energy(mass, length, n), mass, length)
    psi_exp = expected_psi(n, length, x)

    plt.plot(x, np.abs(psi_num)**2, label=f"Numerical |ψ{n}|²")
    plt.plot(x, np.abs(psi_exp)**2, "--", label=f"Analytical |ψ{n}|²")
    plt.title(f"Wavefunction squared for n={n}")
    plt.xlabel("x")
```

```
    plt.ylabel("|ψ|²")
    plt.legend()
    plt.show()

```









$|\Psi_n|^2$ for n=1,2,3,4 for Particle in 1-D box