Finite Difference Methods (FDM) for ODEs and PDEs

This repository contains Python implementations of Finite Difference Methods (FDM) for solving Ordinary Differential Equations (ODEs) and Partial Differential Equations (PDEs) . These methods approximate derivatives using discrete differences, making them useful for numerical solutions in computational mathematics and engineering.

Mathematical Background

First-Order Derivative Approximation

Forward Difference (Explicit Scheme) :

$$\frac{du}{dx} \approx \frac{u(x+h) - u(x)}{h} + \mathcal{O}(h)$$

Backward Difference (Implicit Scheme) :

$$\frac{du}{dx} \approx \frac{u(x) - u(x-h)}{h} + \mathcal{O}(h)$$

Central Difference (Higher Accuracy) :

$$\frac{du}{dx} \approx \frac{u(x+h) - u(x-h)}{2h} + \mathcal{O}(h^2)$$

Second-Order Derivative Approximation

Central Difference Scheme :

$$\frac{d^2u}{dx^2} \approx \frac{u(x+h) - 2u(x) + u(x-h)}{h^2} + \mathcal{O}(h^2)$$

Finite Difference Methods for ODEs

Boundary Value Problems (BVPs)

For the second-order ODE:

$$\frac{d^2u}{dx^2} = f(x), \quad a \leq x \leq b$$

the discrete approximation is:

$$\frac{u_{i+1} - 2u_i + u_{i-1}}{h^2} = f(x_i)$$

Types of BVPs Implemented:

1. Dirichlet BVP :

$$u(a) = \alpha, u(b) = \beta$$

2. Neumann BVP :

$$\frac{du}{dx}\Big|_{x=a} = g_1, \frac{du}{dx}\Big|_{x=b} = g_2$$

3. Mixed BVPs : Combination of Dirichlet and Neumann conditions.

Finite Difference Methods for PDEs

Elliptic PDEs: Laplace and Poisson Equations

1 Elliptic equation:

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0$$

Using the five-point stencil :

$$u_{i,j} = \frac{1}{4}(u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1})$$

For Poisson's equation:

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = f(x, y)$$

2 Parabolic PDEs: Heat Equation

For the one-dimensional heat equation:

$$\frac{\partial u}{\partial t} = \alpha \frac{\partial^2 u}{\partial x^2}$$

1. Explicit Method (FTCS) :

$$u_i^{n+1} = u_i^n + \lambda(u_{i+1}^n - 2u_i^n + u_{i-1}^n)$$

where $\lambda = \frac{\alpha \Delta t}{\Delta x^2}$, with stability condition $\lambda \leq 0.5$.

2. Implicit Method (BTCS) :

$$u_i^{n+1} - \lambda(u_{i+1}^{n+1} - 2u_i^{n+1} + u_{i-1}^{n+1}) = u_i^n$$

3. Crank-Nicholson Method :

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} = \frac{\alpha}{2}\left(\frac{u_{i+1}^{n+1} - 2u_i^{n+1} + u_{i-1}^{n+1}}{\Delta x^2} + \frac{u_{i+1}^n - 2u_i^n + u_{i-1}^n}{\Delta x^2}\right)$$

3 Hyperbolic PDEs: Wave Equation (Explicit Scheme)

For the 1D wave equation :

$$\frac{\partial^2 u}{\partial t^2} = c^2 \frac{\partial^2 u}{\partial x^2}$$

using a second-order explicit central difference scheme:

$$u_i^{n+1} = 2u_i^n - u_i^{n-1} + \lambda^2(u_{i+1}^n - 2u_i^n + u_{i-1}^n)$$

where $\lambda = \frac{c\Delta t}{\Delta x}$ and stability requires $\lambda \leq 1$.

4 Visualization Using Matplotlib

Matplotlib is used to visualize numerical solutions:

ODEs : Solution curves

Elliptic PDEs : Contour plots

Parabolic and Hyperbolic PDEs : Surface and time evolution plots