Solving the 2D Heat Equation Using Finite Difference Method (FDM)

Introduction

The objective of this project is to solve the two-dimensional heat equation using the finite difference method (FDM). The heat equation, a partial differential equation, is commonly used in modeling heat transfer. This project demonstrates the implementation, simulation results, and challenges faced during the development process.

Problem Description

The 2D heat equation is given by:

$$\frac{\partial T}{\partial t} = \alpha \left(\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} \right),$$

where:

- $T(x, y, t)$ is the temperature at position (x, y) and time t.
- α is the thermal diffusivity.

The computational domain is initialized with uniform boundary conditions, while a hot spot is introduced in the center. The simulation evolves the temperature distribution over time.

Numerical Method

The finite difference method (FDM) is used to discretize the heat equation. The spatial derivatives are approximated using central differences, and the time derivative is handled using explicit forward differences. The discrete equation is:

$$T_{i,j}^{n+1} = T_{i,j}^{n} + \Delta t \alpha \left(\frac{T_{i+1,j}^{n} - 2T_{i,j}^{n} + T_{i-1,j}^{n}}{\Delta x^2} + \frac{T_{i,j+1}^{n} - 2T_{i,j}^{n} + T_{i,j-1}^{n}}{\Delta y^2} \right),$$

where Δt, Δx, and Δy are the time step and spatial grid sizes, respectively.

Implementation Details

Code Structure

The code was implemented in Python, with modular functions for setting initial conditions, applying boundary conditions, and iterating the solution over time. Key libraries used include:

- `numpy` for numerical computations.
- `matplotlib` for visualization and animation.

Boundary Conditions

Dirichlet boundary conditions were applied, with the domain edges maintained at a constant temperature of zero.

Challenges Faced

Ensuring stability required careful selection of the time step Δt to satisfy the Courant-Friedrichs-Lewy (CFL) condition:

$$\Delta t \leq \frac{\Delta x^2 \Delta y^2}{2\alpha(\Delta x^2 + \Delta y^2)}.$$

Results

Simulation Outputs

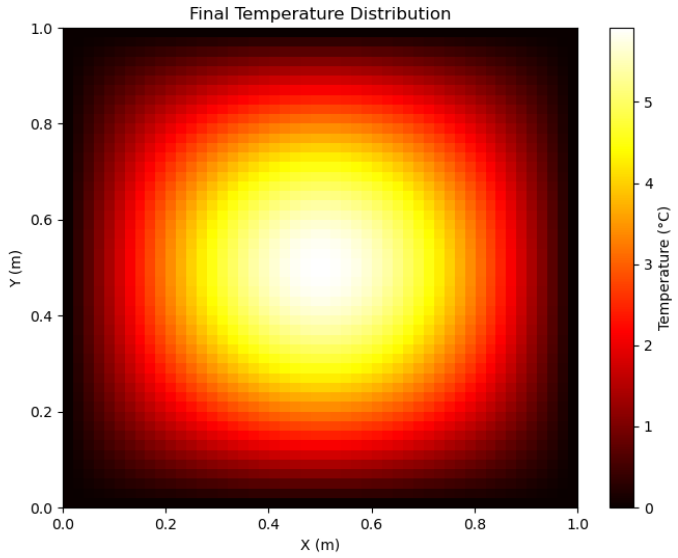

Figure 1: Final temperature distribution after 5000 time steps.

Animation

An animation of the temperature evolution over time was generated to visualize the heat diffusion dynamics. This highlights the smoothing effect of heat diffusion as the temperature stabilizes.

Discussion

The simulation effectively models heat diffusion in two dimensions. The results demonstrate the gradual dissipation of heat from the central hot spot, adhering to physical expectations. While the explicit FDM approach is straightforward, it is computationally intensive for finer grids or longer simulation times due to stability constraints on Δt.

Conclusion

This project implemented a 2D heat equation solver using the finite difference method. The results highlight the utility of FDM for solving PDEs in fluid dynamics. Future work could explore implicit methods to overcome stability constraints and enable larger time steps.

Code Documentation

Structure and Modules

- **Initialization**: Sets up the spatial grid and initial conditions.
- **Boundary Conditions**: Ensures Dirichlet boundary conditions are applied.
- **Time-Stepping Function**: Updates the temperature distribution iteratively.
- **Visualization**: Generates plots and animations of the results.

Dependencies

- `numpy`: Numerical computations.
- `matplotlib`: Visualization.