Solving the 2D Heat Equation with a Source Term Using Finite Difference in MATLAB

1 Project Overview

This project aims to numerically solve the **two-dimensional heat equation** with a source term using the **finite difference method (FDM)** in MATLAB. The heat equation describes the temperature distribution in a given domain over time, incorporating both diffusion and an external heat source. You should develop a MATLAB script that implements an explicit finite difference scheme, visualize the results, and analyze the impact of various parameters.

2 Learning Objectives

By completing this project, you will:

- Understand the fundamental principles of heat diffusion and the governing partial differential equation (PDE).
- Learn to discretize PDEs using the FDM for spatial and temporal derivatives.
- Develop MATLAB scripts to solve and visualize temperature distributions over time.
- Gain experience in debugging numerical codes and validating results against theoretical expectations.

3 Mathematical Formulation

The **2D heat equation with a source term** is given by:

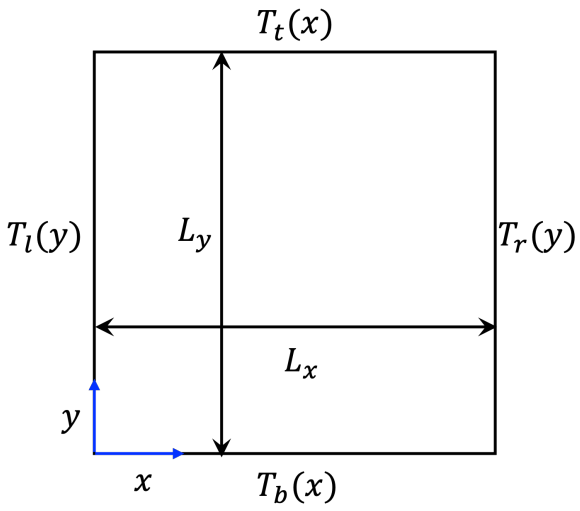
$$\frac{\partial T}{\partial t} = \alpha \left(\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} \right) + S(x, y, t), \tag{1}$$

where:

- $T(x, y, t)$ is the temperature at position (x, y) and time t,
- $\alpha = \frac{k}{\rho c_p}$ is the thermal diffusivity,
- k is the thermal conductivity,
- ρ is the density,
- c_p is the specific heat capacity,
- $S(x, y, t)$ is a heat source term.

4 Project Steps

The project will be divided into the following key steps:



4.1 Problem Definition and Setup

- Define a **rectangular computational domain** ($0 \leq x \leq L_x, 0 \leq y \leq L_y$).
- Set the **initial condition**: $T(x, y, 0) = f(x, y)$, e.g., a uniform temperature or a localized hot spot.
- Implement **Dirichlet boundary conditions**, as in the schematic shown in Figure 4.1.

4.2 Discretization Using Finite Difference

The spatial derivatives are discretized using the **central difference scheme**:

$$\frac{\partial^2 T}{\partial x^2} \approx \frac{T_{i+1,j} - 2T_{i,j} + T_{i-1,j}}{\Delta x^2}, \tag{2}$$

$$\frac{\partial^2 T}{\partial y^2} \approx \frac{T_{i,j+1} - 2T_{i,j} + T_{i,j-1}}{\Delta y^2}. \tag{3}$$

The time derivative is discretized using:

- **Explicit Euler scheme** (forward difference in time):

$$\frac{\partial T}{\partial t} \approx \frac{T_{i,j}^{n+1} - T_{i,j}^{n}}{\Delta t}, \tag{4}$$

  leading to the explicit update formula:

$$T_{i,j}^{n+1} = T_{i,j}^{n} + \alpha \Delta t \left(\frac{T_{i+1,j}^{n} - 2T_{i,j}^{n} + T_{i-1,j}^{n}}{\Delta x^2} + \frac{T_{i,j+1}^{n} - 2T_{i,j}^{n} + T_{i,j-1}^{n}}{\Delta y^2} \right) + S_{i,j} \Delta t. \tag{5}$$

- **Implicit schemes** (e.g., Crank-Nicolson) for better stability.

4.3 Implementing the MATLAB Code

- Define parameters: $L_x, L_y, \Delta x, \Delta y, \Delta t, \alpha, S(x, y, t)$.
- Create **grid points** using `meshgrid`.
- Initialize the **temperature matrix**.
- Implement a **time-marching loop** to update $T(x, y, t)$.
- Apply **boundary conditions** at each time step.

4.4 Implementing Different Source Terms

Possible source terms $S(x, y, t)$ include:

- Constant heat source.
- Gaussian heat source:

$$S(x, y) = S_0 e^{-\frac{(x-x_0)^2+(y-y_0)^2}{\sigma^2}}. \tag{6}$$

- Time-dependent source.

4.5 Visualizing and Analyzing Results

- Use MATLAB's `surf`, `contourf`, or `imagesc` to visualize temperature evolution.
- Compare results for different boundary conditions and source terms.

5 Expected Outcomes

- MATLAB script implementing 2D heat diffusion with a source term.
- Visualizations showing temperature evolution over time.

6 Note:

Take as much time as you need, but don't use ChatGPT!