Interactive 2D Heat Equation Simulator

PHYS129L Final Project

Landon Osborne - *Physics BS*

March 21, 2023

Introduction

The 2D Heat Equation Simulator is a computational physics project that uses Python to simulate the evolution of temperature in a closed system. It computes the heat equation numerically using the forward time central space (FTCS) finite-difference method. The project is intended to be an interactive user-oriented way to model temperature changes in a variety of physical systems over time. The user has the ability to control multiple different aspects of the system, including internal temperature, thermal diffusivity, boundary conditions, and inputting shapes to model.

Installation

To run the 2D Heat Equation Simulator you'll need to have Python 3 installed on your computer, along with the following python packages: numpy, matplotlib, PIL, numba, and curses. Here's an example of how to install the required Python libraries in the command line with apt-get

```
sudo apt-get update
sudo apt-get install python3-numpy
sudo apt-get install python3-matplotlib
sudo apt-get install PIL
sudo apt-get install numba
sudo apt-get install curses
```

The 2D Heat Equation Simulator allows the user to control the simulation with a curses text based user-interface. It then calculates iterations of the heat equation by performing FTCS finite-difference operations on a grid of numpy arrays with the temperature values. The results of these calculations are plotted, animated, and displayed for the user with matplotlib.

How It Works

The Heat Equation

The heat equation is a partial differential equation that models how heat diffuses in a system. It is given by the equation

$$\frac{\partial T}{\partial t} = \alpha \Delta T \tag{1}$$

where T is temperature at a given point, t is time, α is the thermal diffusivity, and Δ is the Laplacian function. It states that for a given point, the rate at which the temperature changes is proportional to the difference in the surrounding temperatures. For our simulation the 2D cartesian form of the heat equation can be used:

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

Finite-Difference Method

In order to calculate the heat equations evolution over time the FTCS finite-difference method of numerical analysis is used to approximate the derivatives of the heat equation. Using this method, the first and second derivatives can be approximated as:

$$f'(c) \approx \frac{f(c+h) - f(c)}{h}$$

$$f''(c) \approx \frac{f(c+h) - 2f(c) + f(c-h)}{h^2}$$

Then we can transform the continuous variables x, y, t to the discrete form i, j, k where $x = i\Delta x, y = j\Delta y, t = k\Delta t$.

$$u_{i,j}^k = T(x, y, t)$$

This represents a two dimensional grid of temperature values that change over time. Using this form we can convert equation 2 to the finite difference form

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

Next we solve for the change in temperature and assume that $\Delta x = \Delta y$, which brings us to the final form of the equation

$$u_{i,j}^{k+1} = u_{i,j}^{k} + \alpha \frac{\Delta t}{\Delta x^2} \left(u_{i+1,j}^{k} + u_{i-1,j}^{k} + u_{i,j+1}^{k} + u_{i,j-1}^{k} - 4u_{i,j}^{k} \right) \tag{3}$$

This equation is applied over all values in the temperature grid to create the next iteration of the system temperature values. A visualization of the grid and how the values influence each other can be seen below in figure 1.

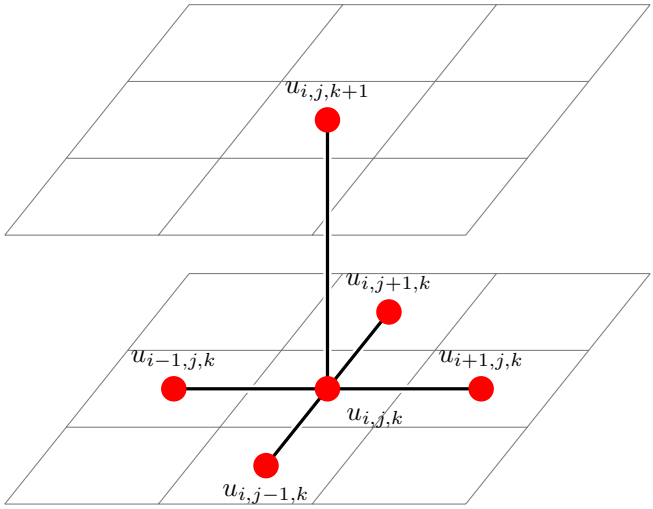

Figure 1: Nodal model showing how points in the finite-difference calculation relate to those in the next iteration. The temperature value at a point and its neighbors all influence the temperature at that point in the next iteration.

To simplify things computationally we keep $\Delta t = 1$ second and $\Delta x = 0.01$ meters, defining the simulation scale. This leaves us with 5 options that the user can choose at the start of the simulation; size (cm), runtime (s), thermal diffusivity (m^2s^{-1}), the system boundary conditions, and the initial temperature distribution. Combining this with the ability for the user to insert different object shapes into the system (with specified thermal diffusivity and initial temperature) and the number of options for testing temperature simulations is effectively limitless in combinations.

Results

The 2D Heat Equation Simulator generates plots of the temperature distribution of the system over a range of time steps, allowing for visualization of how the temperature changes over time. You can use the simulator to calculate the heat flux, heat capacity, and other thermodynamic quantites of systems with various shapes and thermal properites. By varying the input parameters and the customizable object shapes you can study the effects of different boundary conditions, heat sources, heat sinks on the resulting simulated temperature distribution.

Due to computational restraints the simulation size is limited to 10 m^2. At this scale the number of computations per iteration is too large for the cpu to calculate any long period of time quickly. But at smaller scales the number of calculations only increases linearly for runtime so it is possible to observe long term system behaviors.

There are 3 simulation modes in the program, Object simulation mode, Step mode, and Sandbox Mode. Each mode is fully customizable and have different purposes. The Object simulation mode is the most straightforward, it simulates a shape in a box for a set amount of time. It then creates plots of the final temperature state, and an animation of how the temperature changed over time. The Step mode allows for user to step through the simulation at fixed time increments. Finally the Sandbox mode is the same as the Object mode except is has some extra options in setting and manipulating the system temperatures.

Examples

Object simulation mode

As shown in figure 2, the Object mode has a wide variety of choices and is fully customizable. It includes a surface plot and colormap of the final system temperatures, an animation of the temperatures over time, and a plot of the average object temperature.

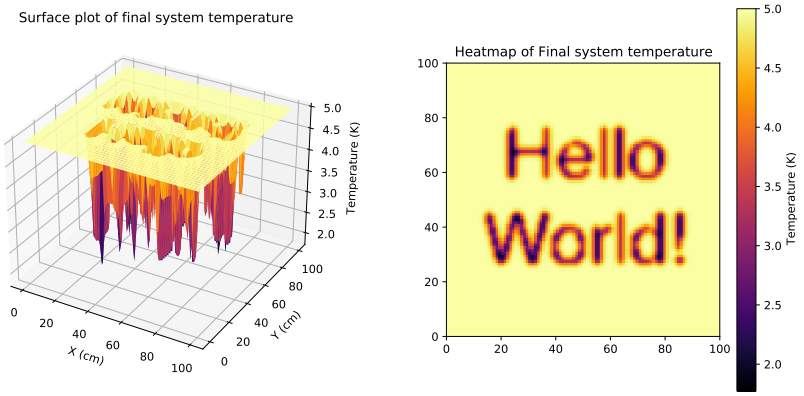

Figure 2: A simple example plot created in Object mode. For this program the runtime was 60 seconds, object temperature was 1 K and background and border temperatures at 5 K. All other simulation values were the default values.

Step mode

Step mode allows for the user to input arbitrarily large runtimes, as it does not store the system animation. This mode only displays the final system temperature state, as shown in figure 3.

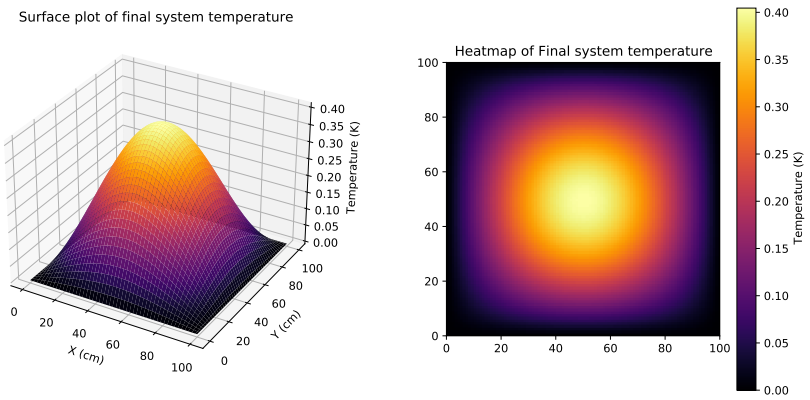

Figure 3: Example plot created in Step mode showing the temperature of the 9-star shape. For this program the runtime was 100,000 seconds and object temperature was set to 1. All other simulation values were either 0 or the default value. As clearly seen in the figure, after 100,000 seconds the temperature has decreased quite significantly.

Sandbox mode

Sandbox mode has all the same customization options as Object simulation mode, but also includes some bonus conditions to initialize such as adding random temperature distributions, shown in figure 4

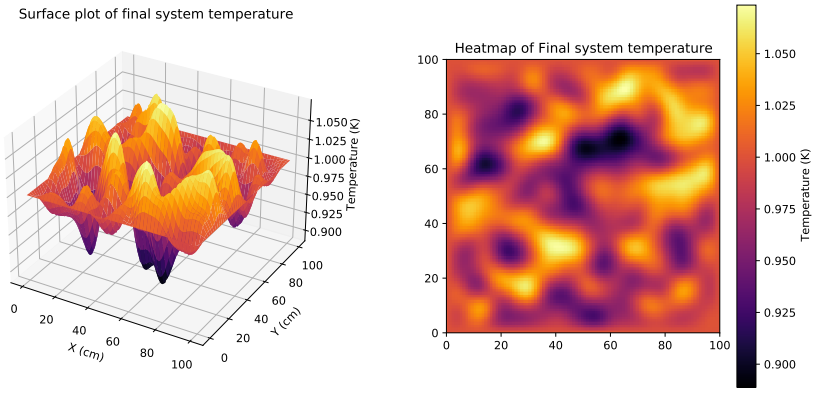

Figure 4: Example plot created in Sandbox mode showing the final state of a random temperature distribution from -1 to 1. For this program the runtime was 1,000 seconds and object temperature was set to 1. All other simulation values were either 0 or the default value.

Conclusion

This project was a lot of fun to develop and fine tune. I especially wanted to focus on creating a user-oriented and interactive program that has infinite different possibilities of settings. It provides an interesting and easy way to model the 2D heat equation and provides some insight into how these physical systems might evolve. Ultimately some of the realistic physical modeling was sacrificed to improve the ease of use and make the program more flexible, but it retains its focus and provides a solid demonstration of numerical analysis of the heat equation.