Numerical Solutions to the one dimensional Heat Equation

Zain Ul Abideen[1*]

[1]Department of Space Science, Institute of Space Technology.

[*]Corresponding author. Email: dominuszain@gmail.com

[*]www.github.com/.../NumericalMethods/Finite Difference Methods

Abstract

This study details the procedure for solving the one dimensional heat equation using the finite difference method. The initial conditions and the boundary conditions have to be provided. The final iterative formula for the solution to the partial differential equation was implemented in the Scilab language using nested for loops. The results were visualized by looping over all the rows in the resulting matrix.

1 Introduction

The partial differential equations are different from the ordinary differential equations in the sense that the functions involved here are of more than one variable. The particular differential equation that was solved in this study was the Heat Equation in one dimension. This particular choice was made because the methodology can be applied to solve the schrodinger equation in computational quantum mechanics, as it takes the same form as the Heat Equation. This partial differential equation was solved numerically using the Finite Difference Method. Analytical solutions are also possible but were unsuited for our application. The differential equation was solved on a two dimensional grid by implementing the final iterative formula using nested for loops. The choice for the Scilab language was made because certain design choices made working with matrices in Scilab much easier than alternatives, and because of the fact that Scilab is open-source and cross-platform.

2 Results and Discussions

Let's start our analysis with the one dimensional form of the heat equation:

$$\frac{\partial U}{\partial t} = a^2 \frac{\partial^2 U}{\partial x^2} \tag{1}$$

To convert the above differential equation into a form that can be solved numerically, we need to make substitutions for the first and the second derivative operators. We will use the definition of derivative to make the substitution:

$$\frac{\partial}{\partial t}U(x,t) = \frac{U_x^{t+1} - U_x^t}{\Delta t} \tag{2}$$

Where the subscript gives the position index, and the superscript the time index. This notation will turn out to be a convenient form for iterative solution. It was found that not all expressions for the 2^{nd} derivative produce the correct results. In this study, I will be using the form that did produce correct results:

$$\frac{\partial^2}{\partial x^2}U(x,t) = \frac{U_{x+1}^t - 2U_x^t + U_{x-1}^t}{(\Delta x)^2} \tag{3}$$

Now, making the substitutions into the partial differential equation and expanding:

$$\frac{U_x^{t+1} - U_x^t}{\Delta t} = a^2 \frac{U_{x+1}^t - 2U_x^t + U_{x-1}^t}{(\Delta x)^2} \tag{4}$$

$$U_x^{t+1} - U_x^t = \frac{(a^2)\Delta t}{(\Delta x)^2} U_{x+1}^t - 2U_x^t + U_{x-1}^t \tag{5}$$

Since we are solving for the time evolution, we will isolate the term that provides that. We will also define a place-holder variable λ for the constants.

$$U_x^{t+1} = U_x^t + \lambda(U_{x+1}^t - 2U_x^t + U_{x-1}^t) \tag{6}$$

To solve the above equation iteratively, we need two sets of conditons; the boundary conditions and the initial conditons. The initial conditons would provide with the spatial state at an initial time. The iterative formula will give us the time evolution of that state as given by the partial differential equation. For the initial conditon, we would need a function of the spatial co-ordinate x i.e. $f(x)$. Since the value in next step of time depends upon multiples values in the previous step, the initial conditon would not provide the sufficient details for the iterative process to begin. We would also need the boundary conditions i.e. the values of the positions $-\infty$ and $+\infty$ for all the instants of time. The most common boundary conditions are that the functions vanish at both of those extreme ends for all instants of time. Ofcourse we would approximate these extreme ends with some finite but large numbers. The whole iterative process for the solution of the Heat Equation is detailed in the diagram below. It is clearly evident how we need both the initial and the boundary conditions to start the iterative process, and how the iterative process could be modelled with nested for loops. The number of points in the length dimension are directly proportional to the quality of the plot. They should be set to a sweet number and then left alone. The number of points in the time dimension are directly proportional to the time the 'animation' lasts. It can be set to any number

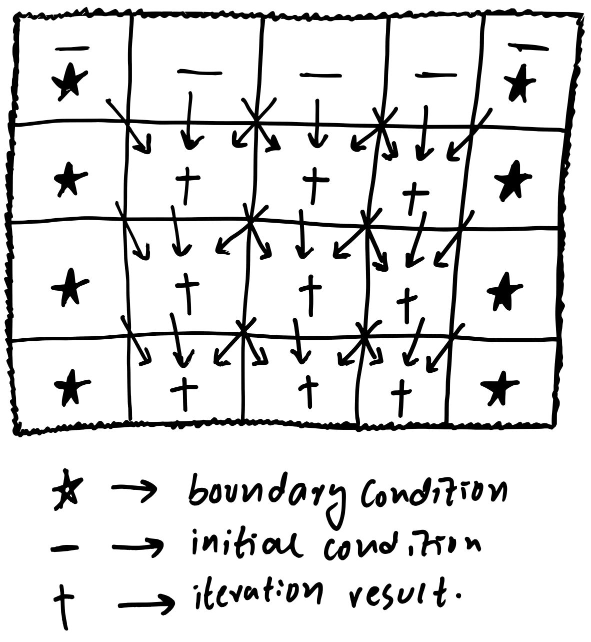

Figure 1: The horizontal blocks give the position co-ordinates, and the vertical blocks give the temporal co-ordinates, with an origin in the upper left corner.

depending upon the nature of study. All the different rows of the resulting matrix were plotted using the *for* loop.