Heat Transfer through a Medium with respect to Space and Time and Comparison of Orders of Accuracy

Joel Samuel Rhine

GWID: 25222529

The George Washington University

Abstract

Analysis of Heat Transfer was carried out in one dimension from one surface to another through a medium. The computations of this problem were carried out using Python, which is a scientific programming language. Python was also used to prepare plots that further aided to understand the results. The discretizing technique used here was, ‘Central Finite Difference in Space’. Two different variations of this method were used, one which provided accuracy results of the second order of the space step, ‘δx’ and the other which provided results of the fourth order. The results were compared and studied accordingly.

Keywords: Heat transfer, one dimension, python.

Table of Contents

Heat Transfer through a Medium with respect to Space and Time and Comparison of Orders of Accuracy

Introduction

Heat transfer in one of the most common natural phenomenon known. The most fundamental understanding is that heat flows from a high temperature sink to a low temperature sink, in which the two temperatures are always relative to each other. Heat flow takes place until a thermal equilibrium is attained. Once this condition is established, there will be no change in the flow unless any change is made in the initial conditions, i.e. the temperatures. For this problem, the two surfaces of a wall are considered. They are kept at constant temperatures of 40 degrees Celsius and 20 degrees Celsius respectively. The wall is of a certain thickness, 'L' through which the heat flow takes place. The thermal diffusivity, 'α' of the wall is taken as a constant value and computed. The wall is assumed to be uniform to avoid any change in its thermal diffusivity.

Method

For the purpose of computation, a Second Degree Central Finite Difference Scheme is undertaken. This is done so because it has the highest accuracy when compared to other finite difference schemes. Its accuracy is of the order of the square of the space step, 'δx'. For example, if the step size is 0.1 then the accuracy of it's computed results would be 0.01 ($0.1^2 = 0.01$). Thus, it is called a 2nd order accuracy method.

The thickness of the wall was first divided into several sections, '*n*'. The thickness of each section was taken as, 'δx' as mentioned above. In every section a node was defined on which the analysis was conducted. This node was at exactly, $\delta x/2$ distance of each section. These nodes were numbers from 0 to '*n*' and stored in '*i*'. If i=0, then it would be the first node under consideration. If i=1, then it would be the second node and so on. Each of the sections between the two surfaces was initialized at a temperature 0 degrees Celsius. The time through which this study was carried out was set at 60 seconds and the time step, '*dt*' was set to 0.1. This meant that the temperatures would initially start at 0 and as time would progress these temperatures would start rising until equilibrium is attained. The representation of this data on a graph after equilibrium would ideally show a straight line from 40 degrees Celsius on the first surface to 20 degrees Celsius on the second surface.

Furthermore, a variation of the 2nd Degree Central Difference Scheme was also undertaken which gave results accurate to the fourth order of the spatial step size, 'δx'. Thus, if δx was taken as 0.1 then the results would be accurate up to its fourth degree, i.e. 0.0001 ($0.1^4 = 0.0001$). Once the results were obtained by 2nd order accuracy and 4th order accuracy, they were compared and studied.

Finite Difference Methods in Space

When we try to solve a problem analytically we usually have a method and it, in most cases, involves a continuous function which results in another continuous function. But, to solve a problem numerically, the function must be first, discretized into individual parts and then solved. This is the basic concept of understanding Finite Difference Methods. These methods are used to find solutions using numerical differentiation and partial numerical differentiation of any degree whenever required. The input can be a continuous function or a set of data. The three types of Finite Difference Methods in Space are Forward, Backward and Central.

These methods are based on the Taylor's series, given by,

$$y(x+h) = y(x) + {y'(x)}/{h} + {y''(x)}/{h^2} + \cdots$$

The terms after ${y'(x)}/{h}$ are truncated which introduces a truncation error $O(h)$.

Thus, the equation reduces to,

$$y(x+h) = y(x) + {y'(x)}/{h} + O(h^{\mathrm{a}})$$

where, 'a' is the order of accuracy.

Using the above equations, the finite difference methods are achieved.

To solve the problem at hand a 2^{nd} Degree Central Finite Difference Scheme was used. As mentioned above, two variations of this method were used and compared.

The first was 2nd Degree 2nd Order Accuracy (a = 2) given by,

for second derivative:

$$y''(x) = {y(x+h) - y(x) + y(x-h)}/{h^2} + O(h^2)$$

for third derivative:

$$y'''(x) = {y(x+2h) - y(x+h) + y(x-h) - y(x-2h)}/{2h^2} + O(h^2)$$

And, the second was 2nd Degree 4th Order Accuracy (a = 4) given by,

$$y''(x) = \frac{-y(x+2h) + 16y(x+h) - 30y(x) + 16y(x-h) - y(x-2h)}{12h^2} + O(h^4)$$

The Heat Equation

The heat equation used to model the problem at hand is given by,

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

Where, 'T' is the temperature in degrees Celsius, 't' is time, 'x' is space and 'α' is the constant of Thermal Diffusivity.

The above equation can be discretized using the 2nd Degree 2nd Order Central Finite Scheme and can be written as follows,

$$\frac{\partial T}{\partial t} = \alpha \left(\frac{T(i-h) - 2T(i) + T(i+h)}{h^2}\right) + O(h^2)$$

and using the 2nd Degree 4th Order Central Finite Scheme as follows,

$$\frac{\partial T}{\partial t} = \alpha \left(\frac{-T(i+2h) + 16T(i+h) - 30T(i) + 16T(i-h) - T(i-2h)}{h^2}\right) + O(h^4)$$

where, $h = \delta x$ is the width of each section as mentioned before and $\frac{\partial T}{\partial t}$ is the rate at which the temperature, 'T' changes.

These two equations were repeated over a period of 60 seconds through time step, dt = 0.1. Both the solutions were plotted, and a comparison of the final results was performed.

Results and Discussions.

For number of sections, '*n*' taken as 20, thickness, '*L*' as 0.1m, α as 0.0001, temperature of surfaces as 40 and 20 degrees Celsius respectively, we achieve the following results.

From fig 1,2,3 it can be observed that the plot initially began near 0 because the temperature of each of the individual sections were initialized at 0. But as we progressed in time, the plot of both the solutions, i.e. the 2nd order accuracy as well as the 4th order accuracy began rising from the left towards 40 degrees Celsius and from the right towards 20 degrees Celsius. Eventually, from fig 4, when time reached 60 seconds an almost straight line was achieved beginning at 40 degrees on the first surface to 20 degrees on the second surface.

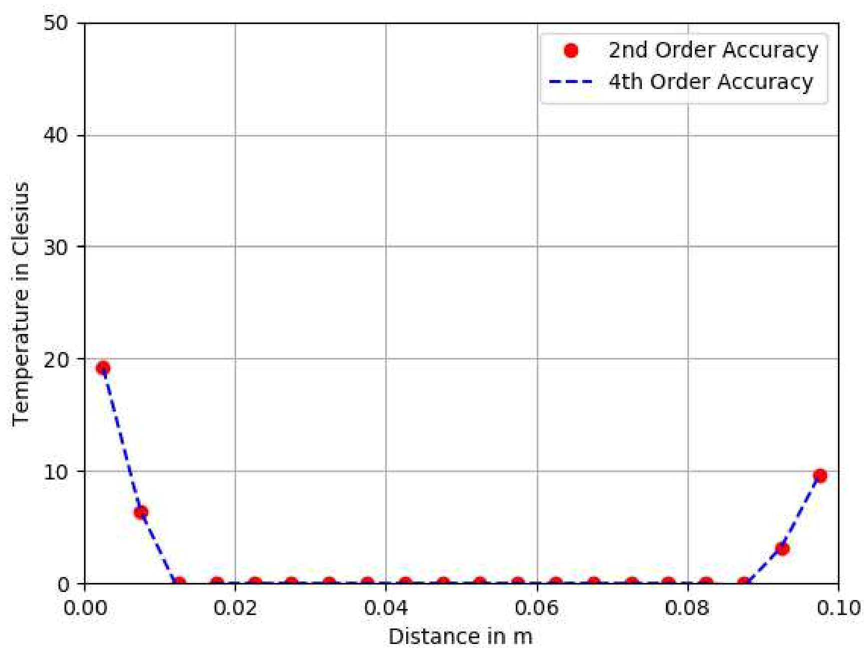

fig 1.

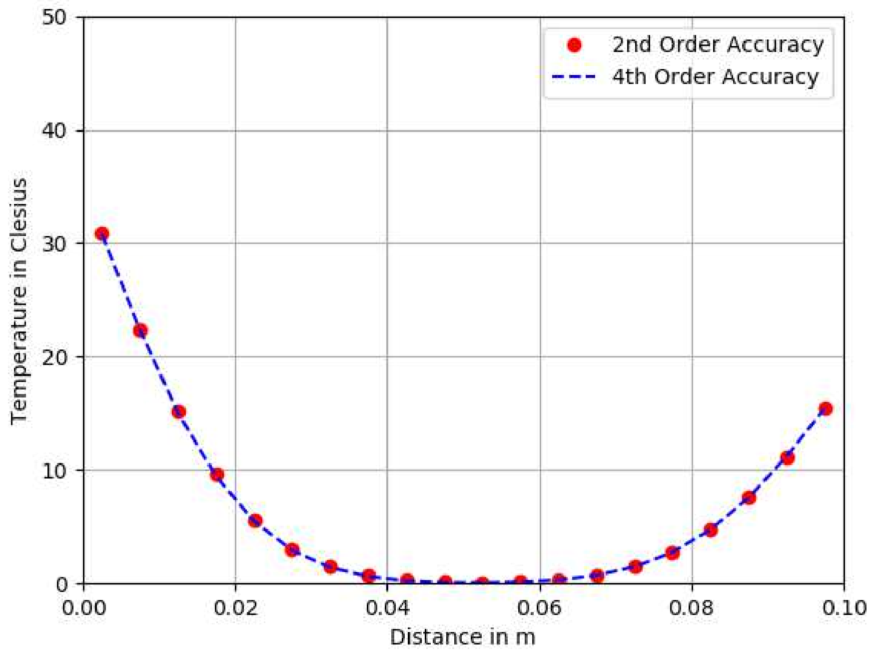
2nd Order Accuracy
4th Order Accuracy
Temperature in Clesius
Distance in m
0
10
20
30
40
50
0.00
0.02
0.04
0.06
0.08
0.10

fig 2.

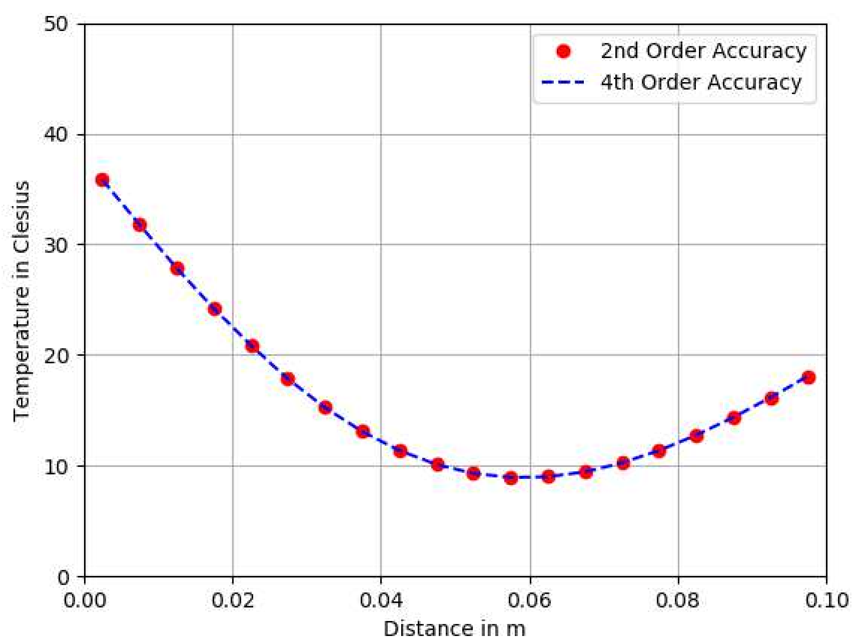
2nd Order Accuracy
4th Order Accuracy
Temperature in Clesius
Distance in m
0
10
20
30
40
50
0.00
0.02
0.04
0.06
0.08
0.10

fig 3.

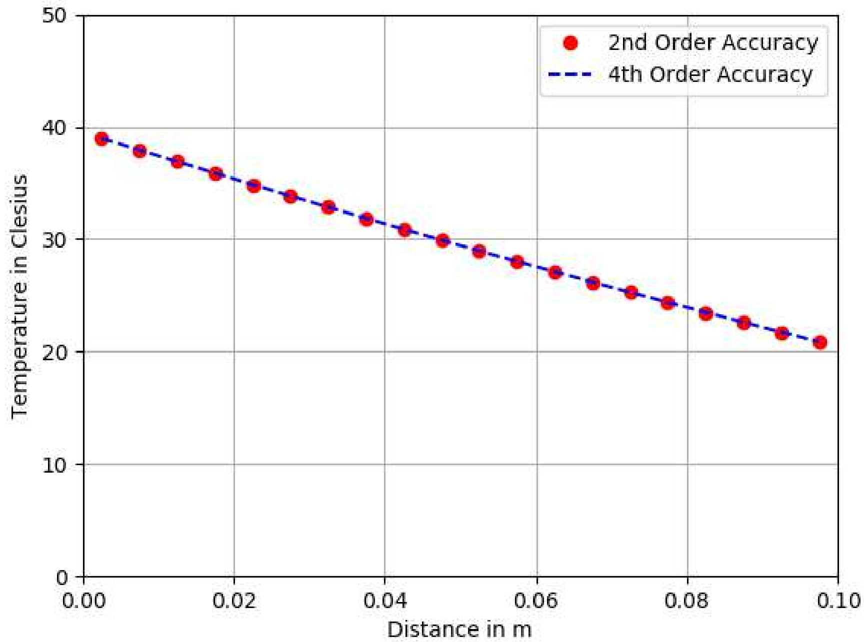
50
40
30
20
10
0
0.00
0.02
0.04
0.06
0.08
0.10
Temperature in Clesius
Distance in m
2nd Order Accuracy
4th Order Accuracy

fig 4.

Table 1

$Node$	ΔT
1	0
2	0
3	0.00014242
4	0.0001849
5	0.00022326
6	0.00025662
7	0.00028426
8	0.00030554
9	0.00032
10	0.00032732
11	0.00032732
12	0.00032
13	0.00030554
14	0.00028426
15	0.00025662
16	0.00022326
17	0.0001849
18	0.00014242
19	0
20	0

Difference in results between 2^{nd} and 4^{th} order Accuracy.

It can be noted from Table 1, that the difference in the final values of both the methods is of the order 1E-4. This difference cannot be observed by the naked eye but may prove fatal in processes which require higher accuracy. Thus, in conclusion, for the given constant conditions the entire analysis of heat transfer as well as the comparison between the two variations of the 2^{nd} Degree Finite Difference Scheme was carried out successfully.

References

1. Brakensiek, D. L. (1967). Finite differencing methods. *Water Resources Research*, *3*(3), 847-860. Retrieved 12 10, 2017, from http://onlinelibrary.wiley.com/doi/10.1029/wr003i003p00847/pdf

2. Li, H., Li, S. (2004). Spatial Discretization. Los Alamos National Laboratory. Retrieved 12 11, 2017, from http://www.phy.pku.edu.cn/~lei/cp/SpacialD.pdf

3. Wadsworth, M., Wragg, A., & Haselgrove, C. B. (1964). The numerical solution of the heat conduction equation in one dimension. *Mathematical Proceedings of the Cambridge Philosophical Society*, *60*(04), 897-907. Retrieved 12 10, 2017, from http://journals.cambridge.org/abstract_s0305004100038366

Relevant Code

MAE_6286_Final_Project_Joel_Rhine_2522252...

```

import numpy as np
import matplotlib.pyplot as plt

L = 0.1 # (m), Thickness of the wall
n = 20 # (Constant), Number of equal sections
alpha = 0.0001 # (Constant), Thermal Diffusivity
T0 = 0 # (C), Temperature of sections between the two surfaces
T1s = 40 # (C), Temperature of Surface 1
T2s = 20 # (C), Temperature of Surface 2
dx = L / n # (Constant), Width of each Section
t_final = 60 # (s), Time cycle of 60 secs
dt = 0.1 # (Constant), Time Step

x = np.linspace(dx / 2, L - dx / 2, n) # Defining the positions of nodes at the center of each section

T_2 = np.ones(n) * T0 # Initializing Temperatures T_2 to 0
T_4 = np.ones(n) * T0 # Initializing Temperatures T_4 to 0

dTdt_2 = np.empty(n)
dTdt_4 = np.empty(n)
```

```
t = np.arange(0, t_final, dt) # Initializing Time Cycle

# Calculation of Temperature values over Time

for j in range(1, len(t)):
    plt.clf()

    # Second Order Accuracy

    dTdt_2[0] = alpha * ((T1s - T_2[0]) / dx ** 2 + (T_2[1] - T_2[0]) / dx ** 2)
    dTdt_2[n - 1] = alpha * ((T_2[n - 2] - T_2[n - 1]) / dx ** 2 + (T2s - T_2[n - 1]) / dx ** 2)

    for i in range(1, n - 1):
        dTdt_2[i] = alpha * (-(T_2[i] - T_2[i - 1]) / dx ** 2 + (T_2[i + 1] - T_2[i]) / dx ** 2)

    # Fourth Order Accuracy

    dTdt_4[0] = dTdt_2[0]
    dTdt_4[n-1] = dTdt_2[n-1]

    dTdt_4[1] = dTdt_2[1]
    dTdt_4[n-2] = dTdt_2[n-2]
    for i in range(2, n-2):
        dTdt_4[i] = alpha*((-T_2[i+2] + 16*T_2[i+1] - 30*T_2[i] + 16*T_2[i-1] - T_2[i-2])/(12*dx**2))

```

```
    T_2 += dTdt_2 * dt

    T_4 += dTdt_4 * dt

# Plotting of Values

    plt.grid()
    plt.plot(x, T_2, color = 'r', marker = 'o', ls = ' ')
    plt.plot(x, T_4, color = 'b', ls = '--')
    plt.legend(['2nd Order Accuracy','4th Order Accuracy'])
    plt.xlabel('Distance in m')
    plt.ylabel('Temperature in Clesius')
    plt.axis([0, L, 0, 50])
    plt.pause(0.05)


# Difference in Accuracy of Equilibrium Results

T = np.empty(n)
T = (T_2 - T_4)
print(T)
```