Comparision of Analytical and Numerical Results for 1D Diffusion/Heat Equation

Ayush Singh

October 26, 2024

1 Introduction

The PDE we try to solve is the one which led to the foundation of Fourier transforms. In this report, the author has performed a detailed comparison between the results obtained for a PDE (1D Diffusion Equation) obtained using the analytical methods and the numerical method, FDM.
Both the methods provide an approximation of their own kind. Both the approximations are discussed thoroughly in this report.

2 Our Problem

The problem we are working upon which is known as a 1D Diffusion Equation/1D Conduction Equation/1D Heat Equation is:

A rod of length $L = \pi$ and thermal conductivity $k = 1$ has initial temperature condition $f(x) = 100$. Both the ends are maintained at temperature 0. So, the boundary condition is $u(0) = u(L) = 0$.

The governing partial derivative equation is:

$$k\frac{\partial^2 u}{\partial x^2} = \frac{\partial u}{\partial t} \qquad \forall\ 0 < x < L, t > 0 \tag{1}$$

3 Analytical Method

The analytical method to solve this PDE provides a series solution to us where the approximation is concerned with the number of terms of the series we have considered. The more the terms we consider, more accurate our solution becomes.

Solution: Assuming the solution of PDE to be

$$u(x,t) = X(x)T(t) \tag{2}$$

So, on differentiating the above equation (2), we get the following:

$$\frac{\partial u}{\partial x} = X'T \tag{3}$$

$$\frac{\partial^2 u}{\partial x^2} = X''T \tag{4}$$

$$\frac{\partial u}{\partial t} = XT' \tag{5}$$

Putting values of equations (3), (4), (5) in the (1), we get the following equation.

$$kX''T = XT' \tag{6}$$

Equating the above equation (6) to an arbitrary constant c,

$$kX''T = XT' = c$$

We can now write this as,

$$\frac{X''}{X} = \frac{c}{k} \qquad\qquad \frac{T'}{T} = c$$

We can now write this as

$$X'' - \frac{c}{k}X = 0 \qquad\qquad \frac{T'}{T} - c = 0$$

So,

$$(D^2 - \frac{c}{k})X = 0 \qquad\qquad \int \frac{T'}{T} - \int c = 0 \tag{7}$$

Just solving the equation on the right side of (7)

$$\ln T - ct = c_1$$

$$\therefore\ T = e^{ct+c_1} \tag{8}$$

Now solving the left side of the (7)
and assuming, $\frac{c}{k} = \lambda^2$

(Case 1) For $\lambda^2 < 0$
So, The roots of the equation are $m = \pm\lambda$

$$\therefore\ X_1 = C_2 e^{\lambda x} + C_3 e^{-\lambda x}$$

$$\therefore\ X_1 = c_2 \cosh \lambda x + c_3 \sinh \lambda x \tag{9}$$

Applying boundary conditions on the equation (9), we get $c_2 = c_3 = 0$, so this solution can be regarded as trivial solution.

(Case 2) For $\lambda = 0$
So the roots are $m = 0, 0$

$$\therefore\ X_2 = c_4 + c_5 x \tag{10}$$

On applying the boundary condition on the equation (10), we get $c_4 = c_5 = 0$, so this solution is also a trivial solution.

(Case 3) For $\lambda^2 > 0$
So the roots are $m = \pm i\lambda$

$$\therefore\ X_3 = C_6 e^{\lambda x} + C_7 e^{\lambda x}$$

$$\therefore\ X_3 = c_6 \cos \lambda x + c_7 \sin \lambda x \tag{11}$$

Applying boundary conditions on the equation (11), we get the solution,

$$\lambda = \frac{n\pi}{L}$$

So, the solution of X becomes

$$X(x) = c_7 \sin \frac{n\pi}{L} x \tag{12}$$

Putting the value of λ in the equation (8), we get the solution for T

$$T(t) = c_1 e^{-k \frac{n^2 \pi^2}{L^2} t} \tag{13}$$

Putting the values in the equation (2), We get

$$u(x,t) = (c_1 c_7) \sin\Big(\frac{n\pi}{L} x\Big) e^{-k \frac{n^2 \pi^2}{L^2} t} \tag{14}$$

According to the superposition principle, the solution is a series combination of ∞ number of solutions.

$$\therefore\ u(x,t) = A_n \sin\Big(\frac{n\pi}{L} x\Big) e^{-k \frac{n^2 \pi^2}{L^2} t} \tag{15}$$

Now, applyting the initial condition, $u(x, 0) = f(x)$

$$A_n \sin\Big(\frac{n\pi}{L} x\Big) = f(x)$$

So, the function $f(x)$ is a Fourier Sine Series, where A_n is the coefficient of the Sine term.
By this obervation,

$$A_n = \int_0^L f(x) \sin\Big(\frac{n\pi}{L} x\Big) dx \tag{16}$$

It is given that $f(x) = 100$, So solving the above integral (16), we get,

$$A_n = \frac{200}{n\pi}(1 - (-1)^n) \tag{17}$$

So, the final solution of the problem becomes,

$$u(x,t) = \frac{200}{n\pi}(1 - (-1)^n) e^{-n^2 t} \sin nx \tag{18}$$

4 Numerical Methods using Finite Difference Method (FDM)

In numerical methods, we convert the Partial Differential Equation into a Linear Algebraic Equation. This conversion can be done using many methods such as Finite Difference Method (FDM), Finite Element Method (FEM), Finite Volume Method (FVM), Lattice Boltzmann Method (LBM), etc. This process of conversion is known as Discretization where the conversion of ∂ to Δ creates the first approximation.
We use Numerical Methods when it is almost impossible to solve a problem using the analytical methods.
FDM is the simplest way of discretization where direct conversion is made, e.g. $\partial u \approx u^{i+1} - u^{i-1}$(Using Forward Differencing and Taylor Series expansion).
The second approximation is when the domain is divided in a mesh (or grid) and the equation is solved at every single node.

Discretization:
Using the Taylor Series expansion, the initial PDE (1) can be written as

$$\frac{u_{n+1}^i - u_n^i}{\Delta t} = k\frac{u_n^{i+1} - u_n^i + u_n^{i+1}}{\Delta x^2} \tag{19}$$

Here,
u_n is the current value of function,
u_{n+1} is the future value of the function,
u^i is the value of the function at the current node,
u^{i+1} is the value of the function at the next/east node, and
u^{i-1} is the value of the function at the previous/west node.

Now, re-arranging the equation (19),

$$u_{n+1}^i = u_n^i + \frac{k\Delta t}{\Delta x^2}(u_n^{i+1} - 2u_n^i + u_n^{i+1}) \tag{20}$$

5 Results and Discussions

5.1 Results From Analytical Method

The initial condition in the image 1 is wavy whereas the initial condition in the given problem was a constant function. This is because we have just considered first 25 terms of the series. Considering more and more terms will make the solution more accurate. As the series is solved with time, it becomes accurate. The most accurate result will be given when ∞ terms of series will be considered.

The code to plot the above series is:

```
import numpy as np
import matplotlib.pyplot as plt

# constants for this function
L = np.pi # length of the rod
k = 1 # thermal diffusivity
totalT = .5 # total time to simulate

n = 5 # number of terms to include from the series
    solution
nx = 20 # number of points to plot
nt = 50 # number of points to plot

x = np.linspace(0, L, nx)
t = np.linspace(0, totalT, nt)

# Initial Conditions
u = 100*np.ones(nx)

def calculate_y(xi, ti):
    global u, n, L, k
    u = np.zeros(nx)
    for i in range(1, n-1):
        u += (200/(i*np.pi))*(1-(-1)**i)*np.sin(i*np.pi*
            xi/L)*np.exp(-k*(i*np.pi/L)**2*ti)

# print(y)

# plot surface
X, T = np.meshgrid(x, t)
U = np.zeros((nt, nx))

for i in range(nt):
    calculate_y(x, t[i])
    U[i, :] = u
    y = np.zeros(nx)

fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.plot_surface(X, T, U)
ax.set_xlabel('x')
ax.set_ylabel('t')
ax.set_zlabel("u(x,t)")
plt.title(r"Heat (1D Diffusion) Equation PDE: $k\frac{\
    partial^2u}{\partial x^2} = \frac{\partial u}{\partial
    t}$")
```

```
plt.show()
```

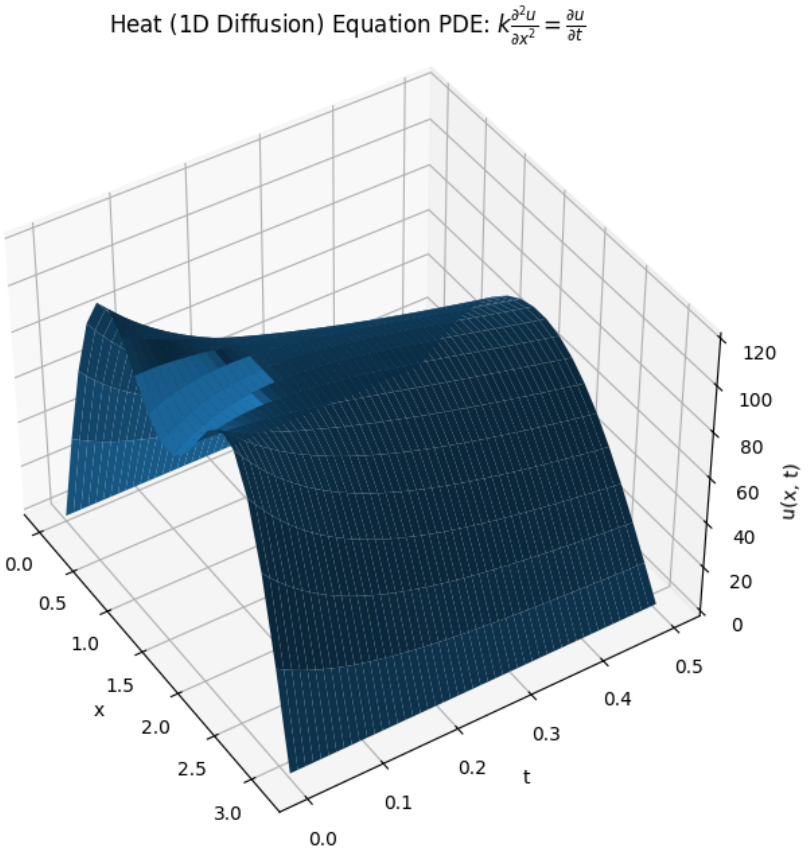

Figure 1: Surface Plot of the Solution

5.2 Results from FDM

The CFD code for the discretized equation (4) is:

```
import numpy as np                    # here we load
    numpy
from matplotlib import pyplot as plt  # here we load
    matplotlib
```

```
# Initial
l = np.pi # total length
k = 1 # thermal diffusivity

nx = 81  # number of small elements, try changing this
    number from 41 to 81 and Run All ... what happens?
dx = l / (nx-1) # width of each element
nt = 1500    # number of times the simulations will run,
    is the number of timesteps we want to calculate
courant_number = 0.3
dt = courant_number * (dx**2) / k

# Intital Conditions
u = 100*np.ones(nx)

un = 100*np.ones(nx) # initialize a temporary array

plt.plot(np.linspace(0, l, nx), un)

for n in range(nt):  # loop for values of n from 0 to nt,
     so it will run nt times
     un = u.copy() # copy the existing values of u into un
     u[1:-1] = un[1:-1] + ((k*dt/dx**2)*(un[2:] -2*un
         [1:-1] + un[0:-2]))

     # Boundary Conditions
     u[0] = 0 # u(0, t) = 0
     u[-1] = 0 # u(l, t) = 0


plt.plot(np.linspace(0, l, nx), u)
plt.show()
```

5.3 Comparison of Results

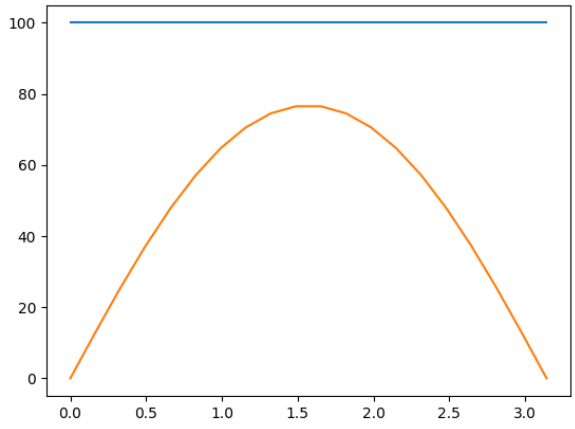

Figure 2: Solution of problem using analytical method in 2D Plot after time t

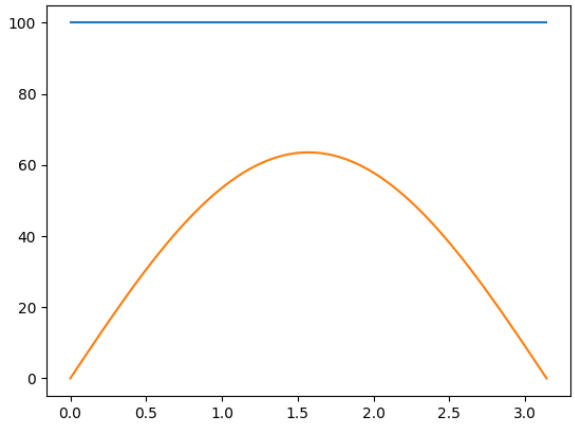

Figure 3: Solution of problem using numerical method, FDM, in 2D Plot after nt iterations

Here, the blue graph shows the initial condition and the orange graph shows the value of function after time t.

5.4 Conclusion

For such a simple equation both the methods gave a completely similar result. We can validate and verify our results using this study because both the methods are completely independent and similar results show that the solution of the problem was calculated correctly.
Numerical methods are usually used only when the solution is impossible to be calculated by the analytical methods and very expensive to be determined by the experimental methods. For e.g., the Navier-Stoke's equations in 1D has a single equation with 2 unknowns, which makes it impossible to solve analytically. In such a case, we solve it using algorithms like SIMPLE and PISO where a Poisson's equation form has to be derived for pressure correction.