1D Heat Equation

Vishal Indivar Kandala

Applied Intelligent Systems Lab, Texas A&M University, College Station, TX

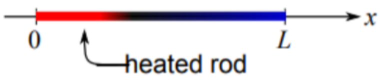
November 2, 2020

Overview

Section 1

Objective

1. Simulate the evolution of the Temperature profile of the 1-D Metal rod shown below, to find the temperature $u(x, t)$ as a function of location and time.



2. The rod is said to be insulated at both ends and no internal heat sources or sinks are present.
3. An initial temperature profile of f(x) is assumed.

1-D Heat Equation

1. Consider an arbitrary segment of the rod, whose width is Δx and which is located at x.

$$\dot{e} = \dot{q}_{left} - \dot{q}_{right} \tag{1}$$

$$\dot{q} = -K_0 \frac{\partial u}{\partial x} \tag{2}$$

2. Fourier's law described in Equation 2 can be applied to find the fluxes on the left and right sides of the segment.

$$c\rho A\Delta x\Big(u(x+\Delta x,t) - u(x,t)\Big) = -K_0 \Delta t A\Bigg(\Big(\frac{\partial u}{\partial x}\Big)_x - \Big(\frac{\partial u}{\partial x}\Big)_{x+\Delta x}\Bigg) \tag{3}$$

1. Simplifying Equation 3 leads to the 1D Heat equation which is a parabolic PDE.

$$\frac{\partial u}{\partial t} = \alpha \frac{\partial^2 u}{\partial x^2} \tag{4}$$

2. This equation is also known as the second order diffusion equation, the co-efficient α is known as the Thermal diffusivity and is a material property.
3. To make use of the Heat equation, Boundary conditions and Initial condition are required.
4. The initial condition in this case is the initial temperature profile $u(x, 0)$.
5. The solution is effected by the boundary conditions which can be determined from the condition that the rod is insulated at both ends.

Section 2

Initial and Boundary conditions

1. At the boundaries, the rod is insulated at both ends i.e the heat flux at both ends is zero.

$$\left(\frac{\partial u}{\partial x}\right)_{x=0} = \left(\frac{\partial u}{\partial x}\right)_{x=L} = 0 \tag{5}$$

2. This kind of boundary condition is known as the Neumann Boundary condition as opposed to holding the ends at a certain constant temperature which is known as the Dirichlet boundary condition.

$$u(x,0) = 0.5\big(sin(x) + cos(x)\big) \tag{6}$$

3. The Initial condition represented in Equation 6 is a sinosuidal condition.

Material

1. The thermal diffusivity α is a function of material properties.

$$\alpha = \frac{K_0}{\rho c} \tag{7}$$

2. **Aluminum** was chosen as the material of the rod, arbitrarily.
3. For this solution, iti is assumed that all the material properties are constant and do not vary with temperature.
4. The specific heat(c) of aluminum at STP is 0.91 $\frac{kJ}{kg-K}$.
5. The density(ρ) of aluminum at STP is 2710 $\frac{kg}{m^3}$
6. The Thermal conductivity(K_0) of Aluminum at STP is 205 $\frac{W}{m-K}$

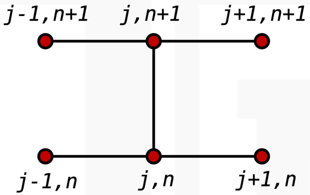
Model

1. The **Finite Difference Method** has been chosen to solve the problem as it is computationally least expensive and is suitable to solve parabolic PDEs with relatively simple boundary conditions.
2. The rod is divided into a grid of points at which function evaluations would be made.
3. The governing equation has been non-dimensionalized and the characteristic length L is assumed to be of unit value.

Section 3

Crank Nicholson Scheme

1. This scheme is semi-implicit in nature, as it involves matrix inversion while the previous timestep information is also considered as part of the solution.



2. The scheme is second order accurate in space and time i.e $\mathcal{O}((\Delta x)^2, (\Delta t)^2)$.

$$\lambda = \frac{\alpha \Delta t}{(\Delta x)^2} \tag{8}$$

3. It is unconditionally stable i.e for any value of the CFL number(λ).

Crank Nicholson Scheme

1. The CN scheme is the most prominent of the Weighted Average schemes.

$$\Theta = 0.5$$

$$u_i^{n+1} = u_i^n + \lambda\left[(1-\Theta)\left(u_{i+1}^{n+1} - 2u_i^{n+1} + u_{i-1}^{n+1}\right) + (\Theta)\left(u_{i+1}^n - 2u_i^n + u_{i-1}^n\right)\right]$$

$$u_i^{n+1} = u_i^n + \frac{\lambda}{2}\left[\left(u_{i+1}^{n+1} - 2u_i^{n+1} + u_{i-1}^{n+1}\right) + \left(u_{i+1}^n - 2u_i^n + u_{i-1}^n\right)\right] \tag{9}$$

2. Equation 9 can be represented in a matrix form as follows.

$$\boldsymbol{A}u^{n+1} = \boldsymbol{B}u^n + \boldsymbol{b} \tag{10}$$

$$\boldsymbol{b} = \begin{bmatrix} 2\lambda\Delta x\dot{q}_0 & \dots & \dots & \dots & \dots & \dots & \dots & 2\lambda\Delta x\dot{q}_L \end{bmatrix}$$

Crank Nicholson Scheme

$$
\boldsymbol{A} = \begin{bmatrix}
1+\lambda & -\lambda & 0 & \dots & \dots & \dots & \dots & 0 \\
\frac{-\lambda}{2} & 1+\lambda & \frac{-\lambda}{2} & \dots & \dots & \dots & \dots & \vdots \\
0 & \frac{-\lambda}{2} & 1+\lambda & \frac{-\lambda}{2} & \dots & \dots & \dots & \vdots \\
\vdots & & \ddots & \ddots & \ddots & & & \vdots \\
\vdots & & & \ddots & \ddots & \ddots & & \vdots \\
\vdots & & & & \ddots & \ddots & -\lambda & 1+\lambda
\end{bmatrix}
$$

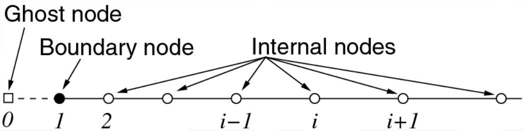
$$
\boldsymbol{B} = \begin{bmatrix}
1-\lambda & \lambda & 0 & \dots & \dots & \dots & \dots & 0 \\
\frac{\lambda}{2} & 1-\lambda & \frac{\lambda}{2} & \dots & \dots & \dots & \dots & \vdots \\
0 & \frac{\lambda}{2} & 1-\lambda & \frac{\lambda}{2} & \dots & \dots & \dots & \vdots \\
\vdots & & \ddots & \ddots & \ddots & & & \vdots \\
\vdots & & & \ddots & \ddots & \ddots & & \vdots \\
\vdots & & & & \ddots & \ddots & \lambda & 1-\lambda
\end{bmatrix}
$$

Crank Nicholson Scheme

1. The scheme takes information from the present time step as well as the previous time step, making use of the maximum amount of information available.
2. However, the scheme does produce oscillations when the gradients are very high in the initial condition or during the first few time steps when this happens.
3. This is numerical dispersion is usually observed when the CFL Number (λ) is significantly higher than 0.5 and is caused by the presence of odd order terms in the modified PDE.

Boundary Condition Implementation

1. To be able to apply the Neumann Boundary conditions described by Equation 5, ghost points are assumed to be present at either ends of the stencil (grid).



2. Using the ghost points, the spatial gradients at the boundaries are computed using a first order central difference approach.

$$\frac{\partial u_0}{\partial x} = \frac{u_1 - u_{-1}}{\Delta x} = 0$$

$$u_{-1} = u_1$$

$$\frac{\partial u_N}{\partial x} = \frac{u_{N+1} - uN - 1}{\Delta x} = 0$$

$$u_{N+1} = u_{N-1}$$

Boundary Condition Implementation

1. The relations between u_{-1}. u_1, u_{N+1} and u_{N-1} are then used to homogenously implement the CN scheme at boundaries.

Rannacher Smoothing

1. To avoid the dispersion caused by crank nicholson scheme, One strategy is to start the first two solutions with an implicit euler formulation and a timestep that is half the original value.
2. Implicit Euler scheme is also unconditionally stable and performs well even when gradients are high.
3. Implicit Euler scheme is first order in space and time i.e $\mathcal{O}((\Delta x),(\Delta t))$.
4. This implementation ensures that the high frequencies at the beginning of the solutio are damped out and then crank nicholson method which is much more accurate can take over.

Section 4

Performance

1. The entire project has been written in Python, with the numpy LinAlg solver to invert matrices.
2. The **np.linalg.solve()** function is of $\mathcal{O}(n^3)$ which is the highest time complexity you will find in the algorithm.
3. On average, a solution with 2000 grid points over 500 time steps takes 28 seconds.
4. This implies that each matrix inversion is taking about 0.06 seconds.

Structure

1. There are 6 python files that make up this project.

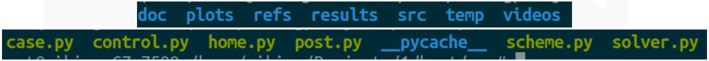


Figure: The structure of the project and the source directory

2. The solutions are solved in the **results** folder
3. The figures produced to create the video are saved in **temp**
4. The plots are docs are stored in their respective namesake folders.
5. The project is hosted on github and can be found **here**.

Setup

1. **control.py** contains all the control parameters such as the grid size, number of time steps and all the variables can be directly altered from here.
2. **case.py** contains material property information as well as the initial condition assumed in the problem, by editing this file, the entire solution can be changed.
3. **scheme.py** contains the pre calculated co-efficient matrices **A**,**B** and the vector **b** for both the Crank Nicholson scheme and the Implicit Euler scheme as both are used in the solution.

function

1. The user executes **home.py** with arguments "-s" and "-u0", these confirm whether a new solution must be generated and also which initial solution to apply.
2. Here, based on the argument, if a new solution is to be generated, then the relevant parameters are pulled from control.py, case.py and scheme.py
3. The parameters are then input into a **solve()** function written in **solver.py** that saves csv results in results folder and returns a 2D array.
4. The solve() function implements Rannacher time stepping with co-efficients of CN scheme and implicit euler scheme that are pulled from scheme.py.
5. In home, this 2D array is passed on to the post processing functions present in **post.py**.

Section 5

Time Evolution of Solutions

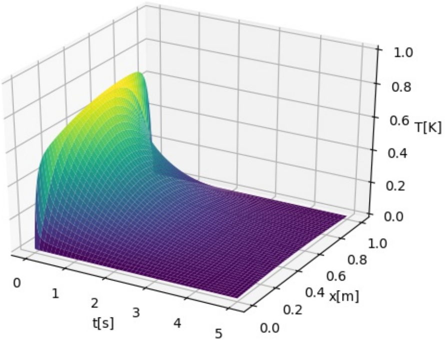


Figure: Time evolution for u(x,0)=0.5(sin(x)+cos(x))

Time Evolution of Solutions

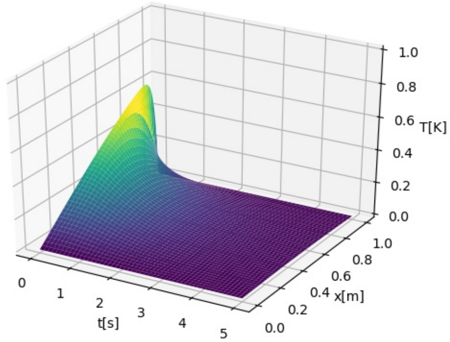


Figure: Time evolution for u(x,0)=0.8(sin(x))

Time Evolution of Solutions

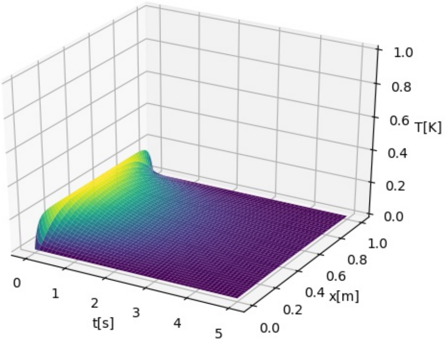


Figure: Time evolution for u(x,0)=0.2 i.e constant

Snapshots of Solutions

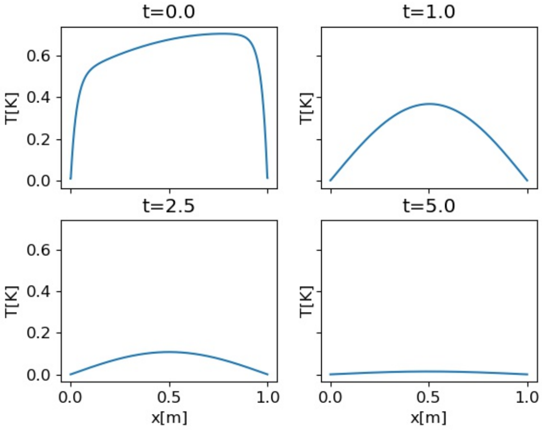


Figure: Snapshots of u(x,t) for u(x,0)=0.5(sin(x)+cos(x))

Snapshots of Solutions

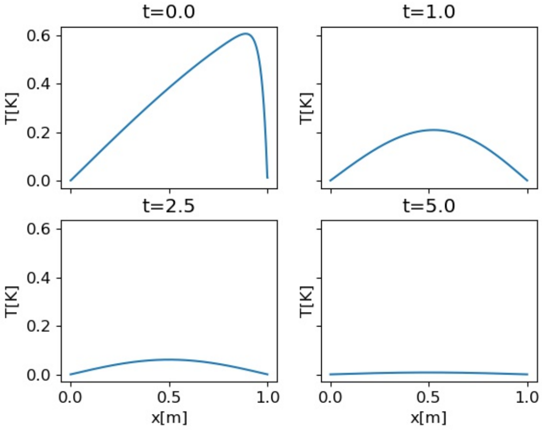


Figure: Snapshots of u(x,t) for u(x,0)=0.8(sin(x))

Snapshots of Solutions

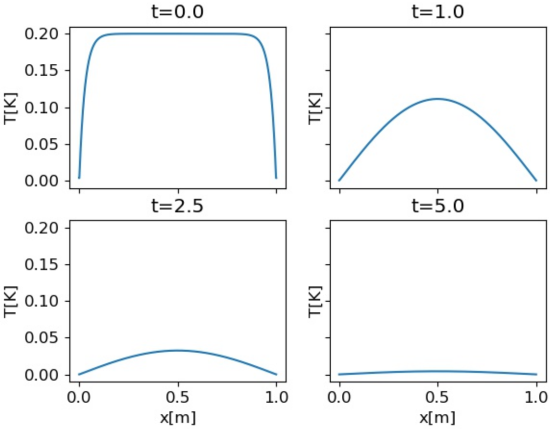


Figure: Snapshots of u(x,t) for u(x,0)=0.2 i.e constant

Thank You!