heat diffusion of a 1D Rod

Nikki Rider

F2020

updated 2022

1D heat equation

(a) $$\frac{\partial T}{\partial t} = K \frac{\partial^2 T}{\partial x^2}$$

one dimensional
rod of length L
w/ endpoints held at $T=0$

given initial temp at each x on the rod and boundary conditions, how does T evolve in time?

---

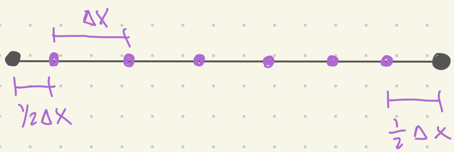
discretize space & time

$x_j = (j - \frac{1}{2})\Delta x$, $t_n = n\Delta t$

$j = 1, 2, \dots J$ $\quad\quad$ $n = 0, 1, 2, \dots N-1$

in the functions for each of the 3 methods to solve the PDE you are given a 1D array "x" which contains your grid points, this ranges from $\left(-\frac{L}{2} + \frac{1}{2}\Delta x \, , \, \frac{L}{2} - \frac{1}{2}\Delta x\right)$

the physical endpoints of the rod are not included (we will set them up later)

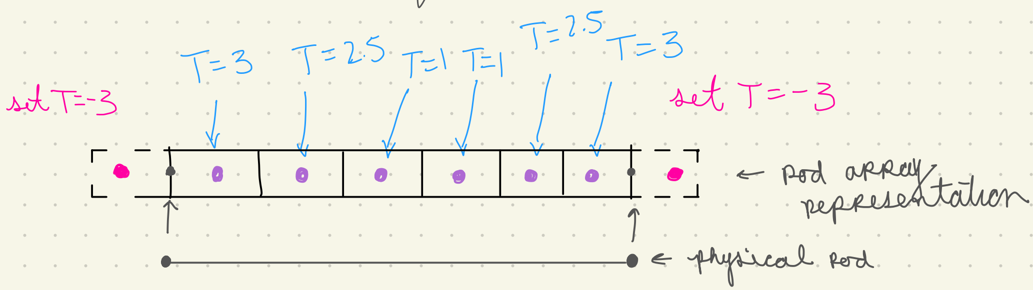


for our boundary condition, we want the physical ends of the rod held at $T = 0$

while we could use a grid w/ support points on the physical rod ends and set the T value there to zero,

this would cause an unphysical temperature gradient between the end points and their neighboring cells

we remedy this by introducing "ghost cells" we will set the values of the "ghost cells" to be the same magnitude but opposite sign of their neighboring cell. for example:

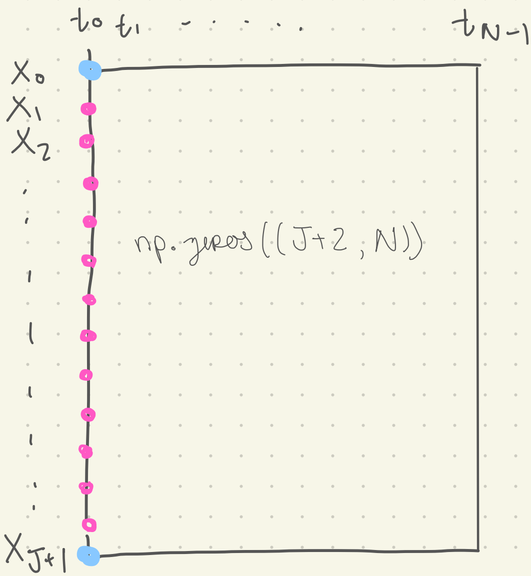


the physical end points are then the average value of the cells that sandwich them, which is zero.

there is already a function which does this, "dirchelet" that is passed in w/ the function handle "fBNC"

(at the end of each solver function we will cut off the rows of ghost cells before returning our temp array) set up an array to store temp values at all support points at all times

$T =$



initial temp (distribution)

temp at rod ends

we want to fill this array w/ T at all x & t

use your updated x grid to then assign your initial condition for the temperature of the rod

initial temp (distribution)

$T[1:J+1, 0] = fINC(x)$

(you will set boundary conditions as well but that will go in your time loop)

for time n:
$$\begin{cases} T[0,n] = fBNC(0, T[:,n]) \\ T[J+1,n] = fBNC(1, T[:,n]) \end{cases}$$

i_side = 0 for LHS of rod

i_side = 1 for RHS of rod

the above goes in all 3 solver functions then...

discretize heat equation $\frac{\partial T}{\partial t} = \kappa \frac{\partial^2 T}{\partial x^2}$

(1) forward-time centered-space (ftcs)

(an explicit scheme)

$$\frac{T_j^{n+1} - T_j^n}{\Delta t} = \kappa \frac{\left(T_{j-1}^n - 2T_j^n + T_{j+1}^n\right)}{(\Delta x)^2}$$

$$\rightarrow T_j^{n+1} = T_j^n + \underbrace{\frac{\kappa \Delta t}{(\Delta x)^2}}_{\equiv \alpha}\left(T_{j-1}^n - 2T_j^n + T_{j+1}^n\right)$$

$$* \quad \boxed{T_j^{n+1} = T_j^n + \alpha\left(T_{j-1}^n - 2T_j^n + T_{j+1}^n\right)}$$

use * in ftcs function to solve for T^{n+1} (the next column) given T^n (the current column)

$$T_j^{n+1} = \alpha T_{j-1}^n + (1-2\alpha)T_j^n + \alpha T_{j+1}^n$$

the 2 following methods, implicit and CN will use their matrix form in the code though corresponding matrix eq:

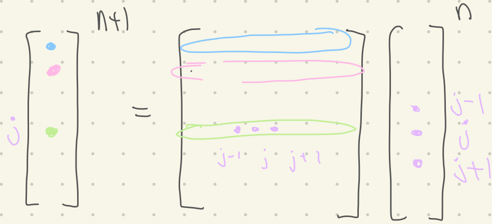
$$\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix}\begin{bmatrix}x\\ y\end{bmatrix} = \begin{bmatrix}1\cdot x+2y\\ \\ \end{bmatrix}$$

$$\underline{T}^{n+1} = \underline{A}\;\underline{T}^{n}$$

vector for time n

vector for time n+1

finding a row of $\underline{A}$ .... match coefficients



$$\begin{array}{cccccc} [\,\cdots & \alpha & (1-2\alpha) & \alpha & \cdots\,] \\ & j-1 & j & j+1 & \end{array}$$

---

assume fourier solution to explore stability

$$T_j^n \equiv \left(\xi(k)\right)^n e^{ikj\Delta x}$$

solution blows up if $|\xi(k)| > 1$

→ plug T_j^n fourier soln into eq * on previous pg

→ solve for $\xi(k)$

→ determine limit on α for $|\xi(k)| < 1$

remember: $\cos x = \frac{1}{2}\left(e^{ix} + e^{-ix}\right)$ & $2\cos x - 2 = -4\sin^2\left(\frac{x}{2}\right)$

& $0 \leq \sin^2\theta \leq 1$

(2) implicit scheme

(use FTCS, but w/ n+1 on RHS)

$$\frac{T_j^{n+1} - T_j^n}{\Delta t} = \kappa \frac{\left(T_{j-1}^{n+1} - 2T_j^{n+1} + T_{j+1}^{n+1}\right)}{(\Delta x)^2}$$

$$\rightarrow \quad T_j^{n+1} = T_j^n + \alpha\left(T_{j-1}^{n+1} - 2T_j^{n+1} + T_{j+1}^{n+1}\right)$$

move all n+1 terms to LHS

$$T_j^{n+1} - \alpha\left(T_{j-1}^{n+1} - 2T_j^{n+1} + T_{j+1}^{n+1}\right) = T_j^n$$

corresponding matrix eq:

different $\underline{\underline{A}}$ than FTCS

$$\boxed{\underline{\underline{A}}\,\underline{T}^{n+1} = \underline{T}^n}$$

there is a "tridiag" function which solves for T^{n+1} given the diagonals of A and T^n already written for you to use in the implicit and CN functions

tridiag(a, b, c, r)

a, b, and c are the diagonals of A — 1D array length J

r is the RHS vector, T^n the n^{th} column of T

(3) Crank-Nicholson

dr. hutsch's lecture notes page 70

5.2.7 Crank-Nicholson

While we now have an unconditionally stable method to solve the diffusion equation, the disadvantage of the fully implicit approach is the same as in the case for ODEs: The decaying (true) solution will be always **below** the calculated one, i.e. while the result for $t \to \infty$ will be 0, the intermediate time evolution will not be correct, typical for a first-order in time scheme. The solution in terms of the diffusion equation is just to **average** the explicit and implicit update, resulting in the **Crank-Nicholson method**,

$$-\frac{\alpha}{2}u_{j-1}^{n+1} + (1+\alpha)\,u_j^{n+1} - \frac{\alpha}{2}u_{j+1}^{n+1} = \frac{\alpha}{2}u_{j-1}^{n} + (1-\alpha)u_j^{n} + \frac{\alpha}{2}u_{j+1}^{n}. \tag{5.59}$$

This is again of the form $\boldsymbol{Ax} = \boldsymbol{b}$, and thus can be solved in the same way as the implicit method. The Crank-Nicholson method is the standard workhorse for diffusion equations – it usually gives good results at moderate computational cost. In your homework you will discuss the stability properties of the method.

$$-\frac{\alpha}{2}T_{j-1}^{n+1} + (1+\alpha)T_j^{n+1} - \frac{\alpha}{2}T_{j+1}^{n+1} = \frac{\alpha}{2}T_{j-1}^{n} + (1-\alpha)T_j^{n} + \frac{\alpha}{2}T_{j+1}^{n}$$

we want to solve for T^{n+1}, we have T^n

$$\boxed{\underline{\underline{A}}\;\underline{T}^{n+1} = \underline{\underline{B}}\;\underline{T}^{n}}$$

use the tridiag function where the RHS vector, r, is matrix/vector product $\underline{\underline{B}}\,\underline{T}^n$

(you can write $\underline{\underline{B}}\,\underline{T}^n$ out explicitly using the green index version above, & input that as r in tridiag)

```
bvec[:]     = (1.0-alpha)*y[1:J+1,n]+0.5*alpha*(y[2:J+2,n]+y[0:J,n])
```

$$(BT)_j^n = \frac{\alpha}{2}T_{j-1}^{n} + (1-\alpha)T_j^{n} + \frac{\alpha}{2}T_{j+1}^{n}$$

analytic solution

hancock 2006

and the solution becomes

$$u(x,t) = \frac{4u_0}{\pi} \sum_{n=1}^{\infty} \frac{\sin((2n-1)\pi x)}{(2n-1)} \exp\left(-(2n-1)^2 \pi^2 t\right). \tag{26}$$

see hancock 2006 for derivation, they derive this expression for the temperature $u(x,t)$ of a rod starting at $x=0$, ending at $x=1$.

when implementing this in the analytical soln. function, take the sum to N terms (we can't sum infinitely many terms on a computer clearly)

we also will need to shift the x in eq (26) because our rod does not necessarily start at $x=0$, but "xmin".

i recomend using another index for the sum as in the code we use "n" for the index of our columns in T

T^{n+1} $\quad j = 0, 1, 2 \ldots J-1 \quad J = 20 \quad T^n$

$$\begin{array}{c} \\ \\ \\ \\ \\ \end{array}\begin{bmatrix} \\ \\ \\ \vdots \\ T_j^{n+1} \\ \vdots \\ \\ \end{bmatrix}\begin{array}{l} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_j \\ \vdots \\ x_{J-1} \end{array} = \begin{array}{c} \begin{array}{ccccccccc} & & & j-1 & j & j+1 & & & \end{array} \\ \begin{bmatrix} & & & & & & & & \\ 0 & 0 & 0 & \alpha & 1-2\alpha & \alpha & 0 & 0 & 0 \\ & & & & & & & & \end{bmatrix} \end{array} \begin{bmatrix} \\ \\ \\ T_{j-3} \\ T_{j-2} \\ T_{j-1} \\ T_j \\ T_{j+1} \\ \\ \\ \end{bmatrix}\begin{array}{l} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_{j-1} \\ x_j \\ x_{j+1} \\ \vdots \\ x_{J-1} \end{array}$$

$$T_j^{n+1} = \alpha T_{j-1}^n + (1-2\alpha) T_j^n + \alpha T_{j+1}^n$$

$$T[j, n+1] =$$