Finite Difference Discretization of Elliptic Problems: the Heat Equation

Jordi Vila-Pérez, jvilap@mit.edu

Massachusetts Institute of Technology

Space Weather Simulation Summer School

Boulder, Colorado

Heat Equation in Atmospheric Modeling

A number of physical phenomena in the atmosphere are driven by changes in temperature. It is important for atmospheric modeling to be able to characterize heat transfer in these circumstances.

Indeed, the thermosphere receives energy from the Sun, which is responsible for heating the air and creating a certain dynamics (from day to night) in the atmosphere.

You will have a whole class tomorrow about the modeling of radiative transfer.

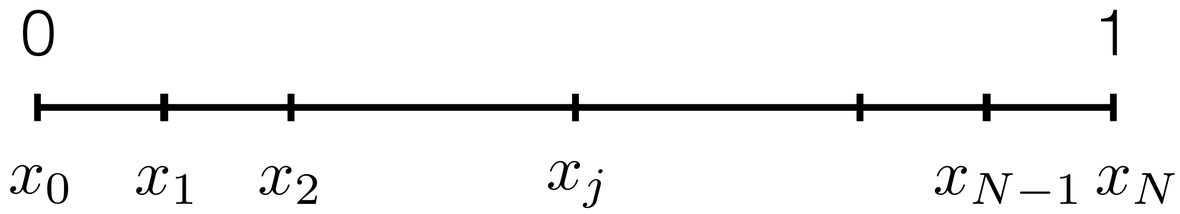
Today, let's focus on learning how to solve the **heat equation**, that is:

$$\rho c_p \frac{\partial T}{\partial t} - \nabla \left(\kappa \nabla T \right) = \dot{Q}$$

using a finite differences (FD) discretization.

Finite Differences Discretization in 1D

We subdivide the interval $(0, 1)$ into $N + 1$ equal subintervals, so $\Delta x = 1/N$.



$$\widehat{u}_j \approx u(x_j), \qquad \text{for } 0 \leq j \leq N$$

Then we approximate their derivatives meeting a certain order of accuracy.

A common approach is based on combining Taylor's series approximation around point x_j, namely

$$u_{j+1} = u_j + u'_j \Delta x + \mathcal{O}(\Delta x^2)$$

Approximation of first-order derivative

$$\Rightarrow \boxed{u'_j = \frac{u_{j+1} - u_j}{\Delta x}} + \boxed{\mathcal{O}(\Delta x)}$$

First-order accuracy

Finite Differences Discretization in 1D

We can combine different Taylor expansions to obtain other approximations with increased accuracy, but also with increased stencil:

$$u_{j+1} = u_j + u'_j \Delta x + \frac{1}{2} u''_j \Delta x^2 + \frac{1}{6} u'''_j \Delta x^3 + \mathcal{O}(\Delta x^4)$$

$$- \qquad u_{j-1} = u_j - u'_j \Delta x + \frac{1}{2} u''_j \Delta x^2 - \frac{1}{6} u'''_j \Delta x^3 + \mathcal{O}(\Delta x^4)$$

$$u_{j+1} - u_{j-1} = 2u'_j \Delta x + \mathcal{O}(\Delta x^3)$$

Approximation of first-order derivative

$$\Rightarrow \boxed{u'_j = \frac{u_{j+1} - u_{j-1}}{2\Delta x}} + \boxed{\mathcal{O}(\Delta x^2)}$$

Second-order accuracy

Finite Differences Discretization in 1D

So we have different approximations depending on the order of accuracy we want to impose in our discretization.

First order derivative

$$u'_j = \frac{u_{j+1} - u_j}{\Delta x}$$

Forward differences

$$u'_j = \frac{u_j - u_{j-1}}{\Delta x}$$

Backward differences

$$u'_j = \frac{1}{\Delta x}\left(-\frac{3}{2}u_j + 2u_{j+1} - \frac{1}{2}u_{j+2}\right)$$

Forward differences

$$u'_j = \frac{1}{\Delta x}\left(\frac{3}{2}u_j - 2u_{j-1} + \frac{1}{2}u_{j-2}\right)$$

Backward differences

$$u'_j = \frac{u_{j+1} - u_{j-1}}{2\Delta x}$$

Centered differences

Second order derivative

$$u''_j = \frac{u_{j-1} - 2u_j + u_{j+1}}{\Delta x^2}$$

Centered differences

First-order accuracy

Second-order accuracy

Poisson Equation in 1D

Consider the Poisson equation: $-\nabla \cdot u = f$.

In 1D, it reads as the following boundary value problem

$$-u_{xx} = f, \qquad \text{in } (0,1)$$

with

$$u(0) = u(1) = 0$$

$$f \in \mathcal{C}^0$$

Solution properties:

- Existence and uniqueness: the solution always exists and is unique.
- Regularisation: The solution is “smoother” than the source term or initial data.
- Positiveness: Given a non-negative source term, the solution is non-negative.

Poisson Equation in 1D

The FD discretization of the 1D Poisson equation is the following:

$$-\frac{\widehat{u}_{j+1} - 2\widehat{u}_j + \widehat{u}_{j-1}}{\Delta x^2} = f(x_j), \qquad 1 \leq j \leq N-1$$

$$\widehat{u}_0 = \widehat{u}_N = 0$$

Which can be expressed as the following linear system of equations $\boldsymbol{A}\widehat{\boldsymbol{u}} = \boldsymbol{f}$

$$\boldsymbol{A} = \frac{1}{\Delta x^2}\begin{bmatrix} 2 & -1 & 0 & \cdots & 0 \\ -1 & 2 & -1 & \ddots & \vdots \\ 0 & \ddots & \ddots & \vdots & 0 \\ \vdots & \ddots & -1 & 2 & -1 \\ 0 & \cdots & 0 & -1 & 2 \end{bmatrix}, \quad \widehat{\boldsymbol{u}} = \begin{bmatrix} \widehat{u}_1 \\ \widehat{u}_2 \\ \vdots \\ \widehat{u}_{N-2} \\ \widehat{u}_{N-1} \end{bmatrix}, \quad \boldsymbol{f} = \begin{bmatrix} f(x_1) \\ f(x_2) \\ \vdots \\ f(x_{N-2}) \\ f(x_{N-1}) \end{bmatrix}.$$

Poisson Equation in 1D

Example 1: $-u_{xx} = f, \qquad \text{in } (0,1)$ [Poisson1D.py]

$$f = (3x + x^2)\exp(x)$$
$$u(0) = u(1) = 0$$

The problem has analytical solution: $u_a = -x(x-1)\exp(x)$

Exercise 1:

a. Solve the problem employing N=8, 16, 32,… How does the discretization error behave?

b. Compute the convergence rate of the approximation.

c. What happens if we want to implement a different Dirichlet BC? Did we make any assumption? Implement the Dirichlet BC explicitly. Check convergence.

Poisson Equation in 1D

Now we are going to talk about other types of boundary conditions. Let's consider:

Example 2: $-u_{xx} = f, \qquad \text{in } (0,1)$

$$f = 2(2x^2 + 5x - 2)\exp(x)$$

$$u(0) = u'(1) = 0$$

The problem has analytical solution: $u_a = 2x(3 - 2x)\exp(x)$

We want to implement a Neumann boundary condition on the right boundary: $u'_N = 0$

Exercise 2:

a. First of all, consider the new source term and analytical solution.
b. Implement a Neumann boundary condition using a first-order approximation.
c. Implement a Neumann boundary condition using a second-order approximation.

Poisson Equation in 1D

First-order approximation:

$$\frac{1}{\Delta x}\left(\widehat{u}_N - \widehat{u}_{N-1}\right) = 0$$

Second-order approximation:

$$\frac{1}{\Delta x}\left(\frac{3}{2}\widehat{u}_N - 2\widehat{u}_{N-1} + \frac{1}{2}\widehat{u}_{N-2}\right) = 0$$

Once you implement the new boundary conditions:

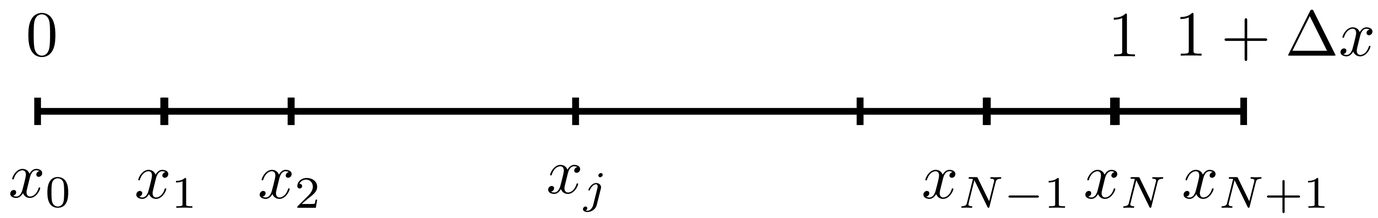
- Remember to check convergence.
- Implement both options with an *if*, introducing the flag *order* to select the order of the BC approximation you want to employ.

Review each other's code.

Poisson Equation in 1D

Introduction of a ghost state

We enlarge our physical domain one more cell, so that we can compute boundary conditions based on fluxes (derivatives) more accurately.



Then we can apply a centered differences approximation of the derivative.

$$\frac{1}{2\Delta x}\left(\widehat{u}_{N+1} - \widehat{u}_{N-1}\right) = 0$$

Exercise 2:

d. How does it behave? Is it more or less accurate? (Check convergence!)

Again, time to review each other's code.

Heat Equation in 1D

Let's introduce a time derivative into the 1D Poisson equation, so we obtain the full heat equation in 1D.

$$u_t - u_{xx} = f, \qquad \text{in } (0,1)$$

Exercise 2:

e. Introduce the time derivative using an implicit Euler time integrator.

   i. Consider the same boundary conditions and source term than in the previous exercise.

   ii. Use an initial condition: $u^0 = 0$

   iii. Initialize the solution variable and implement a for loop to compute the approximation at each time step.

   iv. Define the time-steps at the beginning of the script (like the domain discretization) employing: $\Delta t = 1/24,\ t_f = 3$

Heat Equation in 1D

Let's introduce a time derivative into the 1D Poisson equation, so we obtain the full heat equation in 1D.

$$u_t - u_{xx} = f, \qquad \text{in } (0,1)$$

Exercise 2:

f. Consider now an unsteady source term: $f = 2(2x^2 + 5x - 2)\exp(x)|\cos(\pi t)|$

With the initial condition: $u^0 = x(3 - 2x)\exp(x)$

Review each other's code.

Heat Equation in 1D Spherical Coordinates

So far, we have been solving the 1D heat equation assuming Cartesian coordinates, that is:

$$u_t - \nabla^2 u = f \quad \Rightarrow \quad u_t - u_{xx} = f$$

However, in the space weather context we will be dealing with domains around a planet, for example the Earth. To make a 1D approximation in this case, we assume angular symmetry in the behavior of the corresponding quantity (for instance, temperature) and we will solve our equation in the radial direction.

Reminder of your Calculus class

$$\nabla^2 u = u_{rr} + \frac{2}{r} u_r \qquad \nabla \cdot u = u_r + \frac{2u}{r}$$

So $$\boxed{u_t - u_{rr} - \frac{2}{r} u_r = f, \quad \text{in } 0 < r_0 \leq r \leq r_1}$$

Heat Equation in 1D Spherical Coordinates

Example 3: $-u_{rr} - \frac{2}{r}u_r = f, \quad \text{in } (r_0, r_0+1)$ [PoissonSpherical1D.py]

$$f = 2\left(2\frac{A(\tilde{r})}{r} + B(\tilde{r})\right)\exp(\tilde{r}), \qquad u(r_0) = u_0, \quad u'(r_0+1) = 0$$

with: $A(\tilde{r}) = 2\tilde{r}^2 + \tilde{r} - 3,$

$B(\tilde{r}) = 2\tilde{r}^2 + 5\tilde{r} - 2,$

$\tilde{r} = r - r_0$

The problem has analytical solution: $u_a = 2\tilde{r}(3 - 2\tilde{r})\exp(\tilde{r}) + u_0$

Exercise 3:

a. Check convergence.

Heat Equation in 1D Spherical Coordinates

Consider now the transient version of the problem:

Example 3: $$u_t - u_{rr} - \frac{2}{r}u_r = f, \quad \text{in } (r_0, r_0+1)$$

$$f = 2\left(2\frac{A(\tilde{r})}{r} + B(\tilde{r})\right)\exp(\tilde{r})|\cos(\pi t)|, \qquad u(r_0) = u_0, \quad u'(r_0+1) = 0$$

Exercise 3:

b. Implement the unsteady version of the problem, considering:

i. Initial condition: $u^0 = \tilde{r}(3-2\tilde{r})\exp(\tilde{r}) + u_0$

ii. Solve it for: $\Delta t = 1/24,\ t_f = 3$

Review each other's code.

Submit your exercises in your repo account.

Questions and Comments

Jordi Vila-Pérez, jvilap@mit.edu
Massachusetts Institute of Technology