Finite Difference Discretization of Advection Operators

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

Space Weather Simulation Summer School
Boulder, Colorado

Convection in Atmospheric Modeling

Different phenomena and regions of the atmosphere are dominated by **convection processes**, which are in charge of the **transport** of particles –but also of momentum and energy– along the space.

For instance, that is the case of ions: ionization occurs in upper layers of the thermosphere/ionosphere and the ions are then transported to other regions (such as lower altitudes).

However, the numerical approximation of convection processes is **complex** and **challenging**.

Here we will be highlighting some useful discretization techniques to model (linear) advection using a finite difference methods.

Convection-Diffusion Equation in 1D

Consider the convection-diffusion equation: $-\nu\nabla^2 u + c \cdot \nabla u = f$

And, in particular, the following 1D example.

Example 1: $-\nu u_{xx} + c u_x = f, \quad \text{in } (0,1)$ [ConvectionDiffusion1D.py]

with $f = 1$

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

How do we discretize our equation?

Finite Differences Discretization in 1D

Reminder

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

Convection-Diffusion Equation in 1D

Example 1: $-\nu u_{xx} + c u_x = f, \quad \text{in } (0,1)$ [ConvectionDiffusion1D.py]

with $\quad f = 1 \qquad u(0) = u(1) = 0 \qquad \nu = 0.01,\ c = 2$

Employing a centered differences scheme (both for the first and second derivatives), we obtain

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

So we can get the following system $\boldsymbol{A}\widehat{\boldsymbol{u}} = (\boldsymbol{A}_{diff} + \boldsymbol{A}_{adv})\widehat{\boldsymbol{u}} = \boldsymbol{f}$, with

$$\boldsymbol{A}_{diff} = \frac{\nu}{\Delta x^2}\begin{bmatrix} 2 & -1 & 0 & \cdots & 0 \\ -1 & 2 & -1 & \ddots & \vdots \\ 0 & \ddots & \ddots & \vdots & 0 \\ \vdots & \ddots & -1 & 2 & -1 \\ 0 & \cdots & 0 & -1 & 2 \end{bmatrix}, \quad \boldsymbol{A}_{adv} = \frac{c}{2\Delta x}\begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ -1 & 0 & 1 & \ddots & \vdots \\ 0 & \ddots & \ddots & \vdots & 0 \\ \vdots & \ddots & -1 & 0 & 1 \\ 0 & \cdots & 0 & -1 & 0 \end{bmatrix}.$$

Convection-Diffusion Equation in 1D

Example 1: $-\nu u_{xx} + c u_x = f, \quad \text{in } (0,1)$ [ConvectionDiffusion1D.py]

with $f = 1 \qquad u(0) = u(1) = 0$

The problem has analytical solution: $u_a = \frac{1}{c}\left(x - \frac{1 - \exp(cx/\nu)}{1 - \exp(c/\nu)}\right)$

Exercise 1:

a. Solve the problem employing N=128, 256, 512,… How does the discretization error behave? Check convergence!
b. But these are a lot of points… Do we need so many? What happens when N = 64, 32…?
c. The code prints the Péclet number. For which value does the discretization begin to show oscillations?

MIT AEROASTRO

Convection-Diffusion Equation in 1D

The centered differences scheme is unstable and shows oscillations for $Pe = \frac{|c|\Delta x}{\nu}$.

What alternatives do we have?

For example, let's consider the **first-order upwind scheme**. In this case, the advection operator is discretized according to the following splitting

$$cu'_j \approx \frac{c^+}{\Delta x}\left(\widehat{u}_j - \widehat{u}_{j-1}\right) - \frac{c^-}{\Delta x}\left(\widehat{u}_j - \widehat{u}_{j+1}\right)$$

with $c^+ = \max(c,0), \quad c^- = \min(c,0)$

Exercise 1:

d. Implement the first-order upwind method in your code.
   i. Implement it as an additional option (so, add a *flag* and an *if*).
   ii. Try to follow the same structure than of the second-order method.

e. Check convergence. How does the error behave? What do we need for the Péclet?

Convection-Diffusion Equation in 1D

Now consider the unsteady convection-diffusion equation, that is, let's introduce the time derivative term, that is

$$u_t - \nu u_{xx} + c u_x = f$$

Exercise 2: You did this yesterday!

a. Introduce the time derivative using an implicit Euler time integrator.
   i. Consider the same boundary conditions and source term
   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: $\Delta t = 0.1,\ t_f = 3$

b. Check that your code converges to steady-state (same solution/error than before).

Convection-Diffusion Equation in 1D

We have seen that the discretization of the convection term is somehow tricky.

- The first-order upwind scheme is robust and stable but lacks of accuracy.
- The centered scheme is second-order accurate but needs low Péclet to be stable.

Questions

- Does the solution show oscillations in all the domain?
- Could we solve our problem with a more accurate method in those regions where the solution is “smooth” and with a more robust method in those areas with sharp fronts?

Limiters in FD Convection Discretizations

The idea behind the use of limiters is to switch between the high and the low-resolution schemes depending on the behavior of the solution.

To do that, we need

- A sensor to detect how the solution behaves.
- A limiter function that switches between the low and high-order methods.

Implementation of the sensor

We will track the changes in the slope of the solution.

$$\Delta\widehat{u}_j = \widehat{u}_{j+1} - \widehat{u}_j \quad \Rightarrow \quad r_j = \frac{\Delta\widehat{u}_j}{\Delta\widehat{u}_{j-1}}$$

Notice that we are introducing additional states (in this case, the BC values) for consistency.

Limiters in FD Convection Discretizations

Implementation of the limiters

We will implement different limiters, using the sensor we have defined just before.

Notice that there are a bunch of different limiters. Different applications may require different limiters, which are better suited for different behaviors.

Option 1

$$\phi = \frac{2r}{r^2+1},$$

Van Albada-2

Option 2

$$\phi = \max(0, \min(\beta r, 1), \min(r, \beta)), \ \text{for } 1 \leq \beta \leq 2$$

minmod $\quad \text{for } \beta = 1$

superbee $\quad \text{for } \beta = 2$

Sweaty $\quad \text{for } 1 < \beta < 2$

And then the resulting advection operator reads as $\boldsymbol{A}_{adv} = (1-\phi)\boldsymbol{A}_{low} + \phi \boldsymbol{A}_{high}$

Limiters in FD Convection Discretizations

[ConvectionDiffusion1D_limiters.py]

Observations

- Limiters introduce non-linearities, since they are a function of the solution.
- Combined with implicit methods, this implies solving a non-linear system of equations.
- However, note that some of them are not differentiable.
- In this case, we will use a relaxation approach and compute the sensor/limiter from the solution at previous time.
- The method is no longer implicit! But it will behave better (we are not so restricted by time-step) than a fully explicit approach. Look at the printed CFL number.

Note that there are a bunch of little implementation details so as not to mess up with indices.

Convection-Diffusion Equation in 1D

We have implemented the advection operator introducing limiters to switch from lowew to higher-order discretization schemes. [ConvectionDiffusion1D_limiters.py]

Exercise 3:

a. Check how the solution behaves with the different limiters.
   i. Change $\beta = 0, 1, 1.5, 2, \cdots$
   ii. Change the time-step size.
b. Does the solution behave better or worse than when employing the upwind or the centered scheme?
   i. Check the error.
   ii. Check the oscillatory behaviour for different grid resolutions.

Transient Transport of a Pollutant

Now let's consider the transient transport of a pollutant.

Example 2: $u_t - \nu u_{xx} + c u_x = f, \quad \text{in } (0,1)$ [PollutantTransport1D.py]

with $u'(0) = 0,\ u(1) = u_0,$

$$f = 100 \exp\left(-\left(\frac{x-0.8}{0.01}\right)^2\right) \frac{\sin(2\pi t) + |\sin(2\pi t)|}{2},$$

$$\nu = 0.01,\ c = -2$$

Exercise 4:

a. For N=32, change the parameter of the limiter $\beta = 0, 1, 1.5, 2, \cdots$

b. Change the time-step size. Make it larger. Does the solution show any oscillation? Does the solution cross to negative at some point? Why is that?

c. What happens if you lower the viscosity term? Repeat the previous sections until you reach $\nu = 0$

Transient Transport of a Pollutant

Setting our viscosity parameter to 0, we have now a hyperbolic equation:

Example 2: $u_t + cu_x = f, \quad \text{in } (0,1)$

with $u'(0) = 0,\ u(1) = u_0,$

$$f = 100\exp\left(-\left(\frac{x-0.8}{0.01}\right)^2\right)\frac{\sin(2\pi t) + |\sin(2\pi t)|}{2},$$

$$\nu = 0.01,\ c = -2$$

Exercise 4:

d. Consider only a first-order upwind approach. That is, in a new script, remove the limiters and the higher resolution scheme from the script.

Can we implement a higher-order scheme without the need of limiters?

Transient Transport of a Pollutant

Let's consider the **second-order upwind** scheme:

$$cu'_j \approx \frac{c^+}{\Delta x}\left(\frac{3}{2}\widehat{u}_j - 2\widehat{u}_{j-1} + \frac{1}{2}\widehat{u}_{j-2}\right) - \frac{c^-}{\Delta x}\left(\frac{3}{2}\widehat{u}_j - 2\widehat{u}_{j+1} + \frac{1}{2}\widehat{u}_{j+2}\right)$$

Exercise 4:

e. Implement the second-order upwind scheme. How does it behave? Does it oscillate? Change the resolution and the time-step.

f. Did we have to implement anything else? What can make this approach unsuitable? How do we implement boundary conditions?

g. Note that the Neumann boundary condition can be implemented using the upwind formula, instead of a centered scheme.

Review each other's code.

Transient Transport of a Pollutant

Exercise 4:

h. Implement the following boundary conditions: $u'(0) = 0,\ u''(1) = 0.$
   i. We need an additional ghost state on the right (besides the one on the left).
   ii. Then, think of the approximation of the second order derivative:

$$u''_{N+1} \approx \frac{\widehat{u}_N - 2\widehat{u}_{N+1} + \widehat{u}_{N+2}}{\Delta x^2} = 0$$

Review each other's code.
Submit your exercises in your repo account.

Now you have a transport model that implements a zero-derivative condition on the left boundary and constant derivative condition on the right boundary.

You will need that to implement an ion transport model.

A Few Notes

- Higher-order methods are more accurate but may be less stable.
- A first-order upwind method is more dissipative. Diffusion makes things more stable, but may compromise the accuracy of the solution. Structures are smeared.
- Limiters characterize changes on the slope of the approximation and "introduce diffusion" locally. However, the introduce nonlinearities even in the linear case.
- Non-linear advection behaves differently. There are entire courses and research lines just about that.
- It is straightforward to implement a FD method with equal spacing. When dealing with non-uniform grids, we have to care a little bit more about how to build the system. However, highly stretched or distorted grids may display a deteriorated accuracy.

Questions and Comments

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