AM 129, Fall 2023
Final Project Type C – Numerical PDE: Linear Advection-Diffusion Equation

1. Project Description

There are three parts in the final term project:

- A Fortran implementation of finite difference methods (in your git repository)
- A Python tool for visualizing solutions (in your git repository)
- A written report (uploaded to Canvas)

1.1. Fortran Implementation

In the Fortran part of the project you will implement finite difference schemes (three advection schemes + one diffusion scheme), in order to solve the linear advection equation

$$u_t + au_x = \kappa u_{xx}, \quad 0 \le x \le 1, \tag{1}$$

where a is a constant advection velocity and $\kappa \ge 0$ is a constant diffusion coefficient. The CFL number C is set to be 1.0 in all cases.

Modular Programming: Please design your code in a modular way. Consider using this code structure as a starting point:

- `advect_diff.f90` – This is going to be your main driver routine which calls the appropriate routines from the remaining files.
- `setup_module.f90` – This reads the initialization file and sets all run-wide parameters.
- `fd_module.f90` – This holds and updates the solution respecting the grid and boundary condition modules.
- `error_module.f90` – This checks if the diffusion equation has reached steady state
- `output_module.f90` – This writes the solution out to disc.

You are also free to use the functions and subroutines given in the `read_initFile_module.F90` file from the Mathieu functions example code at the end of the Fortran chapter. You may also want to reference the code given for homework 3, as well as the code given for the other project types.

Makefile: Please compile your code using a Makefile. When coding, please make sure you use useful *compiler flags* to support debugging, for instance, with gdb or Valgrind. Later, you can compile and run your code with *optimization flags*, but only after you are convinced with the correctness of your code. See sections on **Fortran Flags** and **Makefiles** in the lecture notes. You are welcome to adapt the Makefiles given with the homework assignments to this scenario.

1.1.1. 1D Diffusion Let $a = 0$ in equation (1). The resulting equation is the classical heat equation (or diffusion equation) of the form

$$u_t = \kappa u_{xx} \tag{2}$$

with $\kappa > 0$.
Discretization: Write a Fortran program to numerically solve equation (2) using the finite difference scheme in equation (21) in the reading material.
Initial condition: The initial condition is:

$$u_0(x) = \begin{cases} 0°\text{F} & \text{for } 0 \le x < 1, \\ 100°\text{F} & \text{for } x = 1. \end{cases} \tag{3}$$

Boundary conditions: The boundary conditions are set to hold the temperature u at zero at the left boundary, $u(0, t) = 0$, and 100° F at the right boundary, $u(1, t) = 100$, for all $t > 0$. Using the ghost points described in equation (9) of the reading material gives $u_0^n = 0$ and $u_{N+1}^n = 100$.
Material properties: Use a thermal diffusivity of $\kappa = 1.156 cm^2/sec$, which is the value for copper.
Questions:
(a) Find a time $t_{\max}$ (sec) at which the temperature of the material reaches a steady state. Declare that the solution is steady when the L_1 norm of the change in the solution is sufficiently small, i.e.

$$\left|\left|\frac{\Delta u^n}{\Delta t}\right|\right|_1 \equiv \frac{1}{N\Delta t}\sum_{i=1}^{N}\left|u_i^n - u_i^{n-1}\right| < \epsilon. \tag{4}$$

where $\epsilon = 10^{-6}$. Determine $t_{\max}$ for grids of size $N = 32$, and $N = 128$. Write data files holding the solution at $t/t_{\max} = 0,\ 0.2,\ 0.5,\ 0.8$ and 1 for each grid resolution.
(b) Is there any difference in solution between the two different grid resolutions, say, in terms of number of steps to reach $t_{\max}$?
(c) What happens if your Δt_{diff} fails to satisfy the CFL condition in equation (23) of the reading material for the given κ?
(d) What is your value of $t_{\max}$ (sec) for $\kappa = 1.156$? What happens to $t_{\max}$ when κ is increased or decreased by a factor of 10? Explain your results.

1.1.2. 1D Advection Let $\kappa = 0$ in equation (1). The resulting equation is the linear advection equation

$$u_t + a u_x = 0. \tag{5}$$

Use an advection velocity of $a = 1$.
Discretization: Implement three different finite difference schemes for advection (see the reading material):

- 1st order accurate upwind scheme in equation (30),
- 2nd order accurate centered scheme in equation (32),

- 2nd order accurate Lax-Wendroff scheme in equation (35).

Initial condition: Two initial conditions are given as follows:

- Smooth continuous initial profile,

$$u(x,0) = \sin(2\pi x). \tag{6}$$

- Discontinuous initial profile

$$u_0(x) = \begin{cases} -1, & 0 \le x < 1/3,\ 2/3 < x \le 1, \\ 1, & 1/3 \le x \le 2/3. \end{cases} \tag{7}$$

Boundary conditions: We are going to use periodic boundary conditions for all advection schemes and choices of initial condition. Consulting again equation (9) from the reading material, these boundary conditions can be implemented by setting

$$u_0^n = u_N^n, \tag{8}$$
$$u_{N+1}^n = u_1^n, \tag{9}$$

before each time step.
Questions:
(e) Find a time $t_{\max} > 0$ (sec) analytically at which the solution returns to its initial condition.
(f) Use two grid resolutions of sizes $N = 32$ and $N = 128$. Run the three different finite difference schemes against both initial conditions (6 total runs at each resolution). Ensure that the CFL condition is satisfied (equation (19) of the reading material). Identify which scheme(s) that work best for the sine wave. You are expected to obtain results that look similar to Figure 2 of the reading material. Identify good and bad scheme(s) given the discontinuous initial condition.
(g) Choose your best working schemes for the smooth and discontinuous cases, and now run them *with* a large Δt_{adv} that does not satisfy the CFL condition. What do you see?

1.2. Python Implementation

Visualize the data files produced in the above runs using `Matplotlib`.

- Plots for Q (a): Produce one figure for each resolution $N = 32, 128$, each of which contains four subplots (using `plt.subplots(2,2,i)`, `i=1,2,3,4`) to show the solution at times $t/t_{\max} = \{0,\ 0.2,\ 0.5,\ 0.8,\ 1\}$.

- Plots for Q (f): Produce one figure for each resolution $N = 32, 128$, each of which has 6 subplots for the different advection scheme/initial condition combinations. In each subplot show the initial and final profiles, at times $t = 0$ and $t = t_{\max}$ respectively, with different line colors. Make this choice of line coloring consistent over all plots.

- Plot for Q (g): Produce one figure with two subplots, one for each initial condition. On each subplot show the initial profile, and two final profiles. Of the two final profiles, one should use $\Delta t_{adv} = 0.9\Delta x/|a|$, and the other should use $\Delta t_{adv} = 1.2\Delta x/|a|$.

1.3. Report

The final report has a 7-page limit including figures, and should include the following sections:

- Abstract
- Body: methods, results, findings, comments, etc.
- Conclusion

Submit your report through your git repository in PDF format.

1.4. Extra Credit: 1D Advection-Diffusion

You will extend your Fortran code to implement the splitting method described here to solve the full advection and diffusion equation (1) with $\kappa > 0$ and $a = 1$.

The two spatial operators (advection and diffusion) can be applied sequentially, i.e., advection first, then diffusion. This technique is called "operator splitting". One can show that this approximates the underlying equation to first order accuracy in time (ignoring other sources of error). An easy refinement of this is the *Strang splitting method* where instead you advance a half time step of advection, then a full time step of diffusion, followed by another half time step of advection. Starting from u_i^n this proceeds as,

$$u_i^* = u_i^n - a\frac{\Delta t}{2} D_x u_i^n, \tag{10}$$

followed by the diffusion update which takes u^* as input

$$u_i^{**} = u_i^* + \kappa \Delta t D_x^2 u_i^*, \tag{11}$$

finalized by another half step of advection

$$u_i^{n+1} = u_i^{**} - a\frac{\Delta t}{2} D_x u_i^{**}. \tag{12}$$

Note that D_x represents one of the differencing schemes for the first derivative (equations (30) – (37) of the reading material), and D_x^2 represents the centered difference diffusion operator (equation (22) of the reading material).

Questions:
(h) Find analytically a diffusion coefficient $\kappa > 0$ such that $\Delta t_{adv} = \Delta t_{diff}$ on $N = 32$. Use $C = 1$. Pick and run your advection method(s) which produced numerical instability in the discontinuous advection in the pure advection mode with $\kappa = 0$. Does non-zero diffusion κ help to suppress the numerical instabilities you observe with $\kappa = 0$? Plot the solutions you generate here to support your observations.