AMS 209, Fall 2016
Final Project Type B – Numerical PDE: Linear Advection and Diffusion Equation

1. Project Description

There are two parts in the final term project:

- Fortran implementation of a linear algebra solver in a Fortran directory, `"project/PDE"`,
- Python implementation of a run setup, a run scheduler, and a data visualizer, in a Python directory, `"project/PyRun"`.

1.1. Fortran Implementation

In the Fortran part of the project, you are going to implement:

- finite difference schemes (two advection schemes + one diffusion scheme),

in order to solve a linear advection-diffusion 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_a is set to be 1.0 in all cases.

Modular Programming: Please design your code in a modular way. A sample structure of your code can be organized as below:

- `advect_diff.f90` – this is going to be your main driver routine, within which you call the following subroutines:
  - `grid_init.f90` – this sets up the grid configuration
  - `diffuse_init.f90` – this sets up an initial condition for diffusion
  - `diffuse_update.f90` – this updates (only) the diffusion equation
  - `advect_init.f90` – this sets up an initial condition for advection
  - `advect_update.f90` – this updates (only) the advection equation
    - `upwind.f90` – this implements the upwind method
    - `centered.f90` – this implements the centered scheme
  - `cfl.f90` – this calls the CFL condition for diffusion or advection
  - `bc.f90` – this applies boundary conditions, periodic or outflow
  - `check_error.f90` – this checks the exit condition for convergence
  - `write_data.f90` – this outputs your results to a file, `output_N.dat` for a set of $t = \{Nt_{\max}\}_N = \{0.0, 0.2t_{\max}, 0.5t_{\max}, 0.8t_{\max}, t_{\max}\}$, i.e., $N = 0, 0.2, 0.5, 0.8, 1.0$.

Makefile: Please compile your code using a makefile (Executing make with Python is also optional.). When coding, please make sure you use useful *debugging Fortran flags* for easy debugging processes, for instance, with gdb. Later, you run your code with *optimization flags* only after you are convinced with the code. See sections on **Fortran Flags** and **Makefiles** in the lecture note.

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

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

with $\kappa > 0$.

Discretization: Write a Fortran program in order to numerically solve Eq. (2). Use the finite difference scheme in Eq. (21) in the reading material.

Initial condition: The initial condition is described as:

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

Boundary condition: The boundary condition is given so as to hold the temperature u to be zero at $x = 0$ and 100° F at $x = 1$ for $t \geq 0$ (i.e., in Eq. (9) of the reading material, we have $u_0^n = 0°\text{F}$ and $u_{N+1}^n = 100°\text{F}$.).

Material properties Your numerical scheme solves for temporal evolutions of a material diffusivity of copper with $\kappa = 1.156 cm^2/sec$.

Questions:
(a) Choose a time $t_{\max}$ sec at which the temperatures of the materials reach to a steady state solution for $\kappa = 1.156$. Note that you need a criterion to determine $t_{\max}$. The maximum steady-state time step $t_{\max}$ can be determined when the L_1 error E^n is less than a threshold value ϵ:

$$||E^n||_1 = \Delta x \sum_{i=1}^{N} |u_i^n - u_i^{n-1}| < \epsilon. \tag{4}$$

Use $\epsilon = 10^{-4}$. Use the grid sizes of $N = 32$ and 128. Write outputs into files at $t = 0.2t_{\max}, 0.5t_{\max}, 0.8t_{\max}$ and $t_{\max}$.

(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 Eq. (23) of the reading material for the given κ?

(d) What is your value of $t_{\max}$ for $\kappa = 1.156$?

1.1.2. 1D Advection Let $\kappa = 0$ in Eq. (1) now, with $a > 0$. The resulting equation in this case, is the linear scalar advection equation of the form

$$u_t + au_x = 0. \tag{5}$$

Use $a = 1$.

Discretization: Write a Fortran code to implement two different finite difference schemes for advection (see the reading material):

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

Initial condition: An initial condition is given as follow:

- smooth continuous initial profile,

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

Boundary condition: We are going to use a periodic boundary condition for the smooth sine wave advection. Consider using a discrete domain with N cell-centered grid points

$$x_i = (i - \frac{1}{2})\Delta x, \;\; \Delta x = \frac{1}{N}, \;\; i = 1, \cdots, N. \tag{7}$$

Using one layer of guard cell (GC) on each side of the domain, we have one extra GC point through which we will impose the boundary conditions. At the left boundary, we have GC whose coordinate is

$$x_0 = -\frac{\Delta x}{2}, \tag{8}$$

and at the right boundary we get

$$x_{N+1} = (N + \frac{1}{2})\Delta x. \tag{9}$$

With these GCs, the **periodic boundary condition** on the GC regions can be implemented as

$$u_0^n = u_N^n, \tag{10}$$
$$u_{N+1}^n = u_1^n. \tag{11}$$

Questions:
(e) Find a time $t_{\max}$ sec analytically at which the sine wave makes a one periodic cycle to the initial location.

(f) Choose two grid resolutions of sizes $N = 32$ and 128 as in the diffusion case, and run the two different finite difference schemes. Please make sure you

satisfy the CFL condition for advection in Eq. (19) (or Eq. (29)) of the reading material. Identify any finite difference scheme(s) that work(s) well for the sine wave. You are expected to obtain results that look similar to Figure 2 of the reading material. Identify any good and bad scheme(s) for the smooth sine advection.

(g) Choose your best working scheme for the sine wave and run it, but this time, *with* a large Δt_{adv} that does not satisfy the CFL condition. What do you see?

1.1.3. 1D Advection-Diffusion You now study a full advection and diffusion equation in Eq. (1) with $\kappa > 0$ and $a = 1$. The two PDEs (advection and diffusion) can be combined in a sequential way, i.e., advection first, followed by diffusion. This technique is called the "operator splitting" method where you first solve the advection part using one of the two advection methods, taking u_i^n as an initial condition,

$$u_i^* = u_i^n - a\Delta t D_x u_i^n, \tag{12}$$

followed by the diffusion update, taking the adjectively updated intermediate solution u_i^* as its initial condition,

$$u_i^{n+1} = u_i^* + \kappa\Delta t D_x^2 u_i^*. \tag{13}$$

Note that D_x represents one of the differencing schemes for the first derivative (Eqs. (30) – (37) of the reading material), whereas D_x^2 represents the centered differencing diffusion operator (Eq. (22) of the reading material) for the second derivative.

Questions:
(h) Find analytically a diffusion coefficient $\kappa > 0$ such that $\Delta t_{adv} = \Delta t_{diff}$ on $N = 32$. Use $C_a = 1$. Pick and run your advection method which produced numerical instability (upwind or centered, or both?) in the sine advection in the pure advection mode with $\kappa = 0$. Does non-zero diffusion κ help to suppress the numerical instabilities you observe with $\kappa = 0$?

1.2. Python Implementations

You use Python to produce various plots of the Fortran outputs:

- Run setup and run scheduler: these are optional in Project Type B. Please explore how you can organize your runs better using Python, but again, this is optional.
- Solution visualizer:
  - Plots for Q (a): Produce two cases of the diffusion runs $N = 32, 128$ each of which contains four subfigures using `plt.subplots(2,2,i)`, `i=1,2,3,4` for $t = 0.2t_{\max}$, $0.5t_{\max}$, $0.8t_{\max}$ and $t_{\max}$.

- Plots for Q (f): Produce two cases of the advection runs $N = 32, 128$, each of which runs with both the upwind and the centered methods. Plot all four cases using `plt.subplots(2,2,i), i=1, ..., 4`. Subfigures in the first column (`plt.subplots(2,2,i), i=1,2`) display the sine wave solutions with the upwind scheme for $N = 32, 128$, and subfigures in the second column (`plt.subplots(2,2,i), i=3,4`) show the sine wave solutions with the centered scheme for $N = 32, 128$. In each figure, plot both the initial profile at $t = 0$ and the final profile at $t_{\max}$.
- Plot for Q (g): Plot one with $\Delta t_{adv} < \Delta x/|a|$, and another one with $\Delta t_{adv} > \Delta x/|a|$. You can do this, for instance, using $C_a = 0.9$ and $C_a = 1.2$, where $\Delta t_{adv} = C_a \Delta x/|a|$. Again, in each case, plot both the initial profile at $t = 0$ and the final profile at $t_{\max}$.
- Plot for Q (h): One plot for this.

1.3. LaTeX Report

Write your final report using LaTeX (7-page limit including figures). You have to write three parts in your report:

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

1.4. Website Update

Upload your LaTeX report and your source codes to your website, under a new tab, `"Project"`.

2. Appendix: Example Matlab Code

```
% --------------------------------------------------
% AMS 209 - Fall, 2016
% MATLAB code for 1D heat diffusion
% u_t = kappa *u_xx
% Written by Prof. Dongwook Lee
% AMSC, UCSC
% --------------------------------------------------
clf;
clear all;

%grid resolution
xa=0.;
xb=1.;

N=16;
dx = (xb-xa)/N;
```

```
%discrete domain
x=linspace(0.5*dx,xb-0.5*dx,N);

% fixed BC
g0=0.;
g1=100.;
% IC
u(1) = g0;
u(2:N+1)=0;
u(N+2)=g1;

% diffusion coefficient
kappa=1.156;

% CFL & dt
Ca = 0.8;
dt= this is your CFL satisfying dt
t=0;
tmax=?;



while t<tmax;
    for i=2:N+1;
        % solve heat diffusion for interior points
        % finite difference implementation goes here (one line)
    end

    %update t
    t=t+dt;

    % update BC
    uNew(1)  =g0;
    uNew(N+2)=g1;

    % store your solution array
    u=uNew;


end
```