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 (three advection schemes + one diffusion scheme),

in order to solve a linear advection-diffusion equation

\[ u_t + au_x = \kappa u_{xx}, \quad 0 \leq x \leq 1, \]

(1)

where \( a \) is a constant advection velocity and \( \kappa \geq 0 \) is a constant diffusion coefficient.

Modular Programming: In Project Type B, this is optional, but you are always welcome to design your code in a modular way.

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

\[ u_t = \kappa u_{xx} \]

(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 project description.

Initial condition: The initial condition is described as:

\[ u_0(x) = \begin{cases} 
0^\circ F & \text{for } 0 \leq x < 1, \\
100^\circ F & \text{for } x = 1.
\end{cases} \]

(3)
**Boundary condition:** The boundary condition is given so as to hold the temperature $u$ to be zero at $x = 0$ and $100^\circ$ F at $x = 1$ for $t \geq 0$ (i.e., in Eq. (9) of the project description, we have $u^n_0 = 0^\circ$F and $u^n_{N+1} = 100^\circ$F.).

**Material properties** Your numerical scheme solves for temporal evolutions of two different material diffusivities:

(i) iron with $\kappa = 0.230 cm^2/sec$,
(ii) copper with $\kappa = 1.156 cm^2/sec$.

**Questions:**
(a) Choose a time $t_{\text{max}}$ sec at which the temperatures of the materials reach to a steady state solution for each $\kappa$. Note that you need a criterion to determine $t_{\text{max}}$ for a given $\kappa$. The maximum steady-state time step $t_{\text{max}}$ can be determined when the $l_1$ error $E^n$ is less than a threshold value $\epsilon$:

$$||E^n||_1 = \Delta x \sum_{i=1}^{N} |u^n_i - u^{n-1}_i| < \epsilon. \quad (4)$$

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

(b) Is there any difference in solution between the two different grid resolutions, say, in terms of number of steps to reach $t_{\text{max}}$?

(c) What happens if your $\Delta t_{adv}$ fails to satisfy the CFL condition in Eq. (21) of the project description for each $\kappa$?

(d) What are your values of $t_{\text{max}}$ for two different materials?

**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

$$u_t + au_x = 0. \quad (5)$$

Use $a = 1$.

**Discretization:** Write a Fortran code to implement three different finite difference schemes for advection (see the project description):

- 1st order accurate upwind scheme in Eq. (29),
- 1st order accurate downwind scheme in Eq. (30),
- 2nd order accurate centered scheme in Eq. (31).

**Initial condition:** Two initial conditions are given as follows:
• the first case with smooth continuous initial profile,

\[ u(x, 0) = \sin(2\pi x), \quad (6) \]

and

• the second case with discontinuous shock initial profile,

\[ u(x, 0) = \begin{cases} 
1 & \text{for } x < x_0, \\
-1 & \text{for } x > x_0,
\end{cases} \quad (7) \]

where \( x_0 = 0.3 \) is the location of the initial discontinuity.

**Boundary condition:** We are going to use two different types of boundary conditions,

• periodic condition for the smooth IC in Eq. (6)

• outflow condition (or Dirichlet condition, \( \frac{\partial}{\partial x} = 0 \)) for the discontinuous IC in Eq. (7)

Consider using a discrete domain with \( N \) cell-centered grid points

\[ x_i = (i - \frac{1}{2})\Delta x, \quad \Delta x = \frac{1}{N}, \quad i = 1, \cdots, N. \quad (8) \]

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

\[ x_0 = -\frac{\Delta x}{2}, \quad (9) \]

and at the right boundary we get

\[ x_{N+1} = (N + \frac{1}{2})\Delta x. \quad (10) \]

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

\[ u^n_0 = u^n_N, \quad (11) \]
\[ u^n_{N+1} = u^n_1. \quad (12) \]

The **outflow condition** is given as

\[ u^n_0 = u^n_1, \quad (13) \]
\[ u^n_{N+1} = u^n_N. \quad (14) \]

**Questions:**

(e) Find a time \( t_{\text{max}} \) sec analytically at which the sine wave makes a one periodic cycle to the initial location in the first case; and at which the initial discontinuity travels to reach at \( x = 0.8 \) in the second case.
(f) Choose two grid resolutions of sizes $N = 32$ and 128 as in the diffusion case, and run the three different finite difference schemes. Please make sure you satisfy the CFL condition for advection in Eq. (18) of the project description. Identify any finite difference scheme(s) that work(s) well for the sine wave, and for the discontinuous profile, respectively. You are expected to obtain results that look similar to Figure 2 in the project description using some of the three schemes. Some schemes might not be stable enough and oscillate, distorting the original profiles in both the sine and discontinuous cases. Identify any good and bad schemes for each initial condition.

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

1.1.3. 1D Advection-Diffusion We now study a full advection and diffusion equation in Eq. (1) with $\kappa > 0$ and $a = 1$.

Questions:

(h) Find analytically a diffusion coefficient $\kappa > 0$ such that $\Delta t_{adv} = \Delta t_{diff}$ on $N = 32$. Assume $C_a = 1$. Pick and run one unstable run which produced oscillations in the discontinuous initial profile case in the pure advection with $\kappa = 0$. Does non-zero diffusion $\kappa$ help to reduce numerical oscillations?

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.

- Solution visualizer:
  - Plots for Q (a): Produce two figures of the diffusion runs $N = 32, 128$ which of each 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 figures of the advection runs $N = 32, 128$ which of each contains `plt.subplots(3,2,i)`, $i=1, \ldots, 6$. Subfigures in the first column (`plt.subplots(3,2,i)`, $i=1, 2, 3$) display the sine wave solutions, and subfigures in the second column (`plt.subplots(3,2,i)`, $i=4, 5, 6$) show the discontinuous solutions, where solutions of the three different methods (upwind, downwind, centered) appear in rows.
  - Plot for Q (g): Plot one with $\Delta t_{adv} \leq C_a \Delta x / |a|$, and another one with $\Delta t_{adv} > C_a \Delta x / |a|$, where $0 < C_a \leq 1$ is a CFL number. You can do this, for instance, using $C_a = 0.9$ and $C_a = 1.2$.
  - 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**

```matlab
% AMS 209 - Fall, 2015
% 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;
ul(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