AM 260, Spring 2025
Homework 4

Posted on Tue, May 8, 2025
Hard deadline 11:59 pm, Wed, June 11, 2025

- You are recommended to use LaTex or MS-words like text editors for homework. A scanned copy of a handwritten solutions will still be accepted on condition that your handwriting is clean and well-organized, and your scanned copy is fully readable.
- The hard deadline June 11 is the deadline for the final project as well. Although this homework does not have its own earlier deadline, you are encouraged to finish this homework at least two weeks earlier before you begin the final coding project since the project is highly relevant to this homework set. Just do not wait until the last minute to begin this homework!

---

1. Description

Organize your HW4 tar file to include two parts, Fortran (or C) source codes for PDE solvers and Python (or MATLAB) visualization tools. For example, your tar file includes the following subdirectories:

- Fortran (default) or C implementation for 1D non-linear scalar advection in a Fortran directory, `"HW4/PDE"`,
- Python (default) or MATLAB implementation for data visualization in a Python directory, `"HW4/PyViz"`.

1.1. Fortran (or C) Implementation

In the Fortran part of the project, use Fortran 90 (or above) to implement:

- the first-order Godunov method (FOG)
- the second-order piecewise linear (reconstruction) method (PLM) using
  - non-TVD slope limiters:
    - upwind slope
    - downwind slope
    - centered slope
  - TVD slope limiters:

* minmod
* MC
* van Leer's

and solve a conservative 1D non-linear scalar advection with the Burgers' equation:

$$u_t + \left(\frac{u^2}{2}\right)_x = 0, \quad x_{\text{beg}} \le x \le x_{\text{end}}. \tag{1}$$

Note that, by default, the CFL number is set to be 0.9, and the domain is set as $x_{\text{beg}} = 0$, and $x_{\text{end}} = 1$; however, they all should be adjustable through a runtime parameter file `advect.init` (see below).

Modular Programming: In this homework set, you learn to design your code in a modular way. A set of good example of modular programming of Fortran is available in the AM 129 online lecture note (links are available on the HW4 Canvas page). In particular, have a look at Newton's method example at the end of Chapter 2 therein.

In a nutshell, a sample structure of your code for HW4 can be organized as below:

- `advect.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
  - `advect_init.f90` – this sets up an initial condition for advection including a call to a proper boundary condition and to CFL
  - `advect_update.f90` – this updates the Burgers' equation and calls either FOG or PLM
    * `FOG.f90` – this implements the first-order Godunov scheme
    * `PLM.f90` – this implements the second-order piecewise linear reconstruction scheme. The PLM routine then calls one of the following slope limiter functions:
      · `upwind.f90` – this implements the upwind slope limiter
      · `downwind.f90` – this implements the downwind slope limiter
      · `centered.f90` – this implements the centered slope limiter
      · `minmod.f90` – this implements the minmod slope limiter
      · `mc.f90` – this implements the MC slope limiter
      · `vanLeers.f90` – this implements the van Leer's slope limiter
  - `cfl.f90` – this calls the CFL condition for advection
  - `bc.f90` – this applies boundary conditions, outflow, or periodic

- `write_data.f90` – this writes your results to a standard ASCII file named. For example, the simplest naming convention of such output files could be `"output_methodName_abcd.dat,"` where `abcd` is a four-digit integer (e.g., 0000, 0001, 0002, etc.) to enumerate each output. There is a good example of an output routine in Newton's method example section in Chapter 2 of the AM 129 lecture note. You're strongly encouraged to take a look at the example output routine (as well as other routines there), study them, and use/modify them for your needs.

Makefile: Always compile your code using a makefile. When coding, make sure you utilize useful *debugging Fortran (or C) flags* for easy debugging processes, for instance, with gdb. Later, you run your code with *optimization flags* only after you are fully convinced with the code. See sections on **Fortran Flags** and **Makefiles** as well as Newton's method example section in Chapter 2 in the AM 129 online lecture note for more information.

`advect.init`: Given that you're going to run your code in a various combination of different runtime parameters (e.g., CFL number, reconstruction order, slope limiter, grid size, etc.), it will be handy to provide such a set of runtime parameters in a separate runtime parameter file called `advect.init`. In each run, your code will then read in the parameter values from `advect.init` and use them to solve the Burgers' equation. The benefit of doing this is that those input parameters are read-in at the beginning of each execution and are used for a given run; otherwise you would need to re-compile your codes every time when you wish to change one or more runtime parameters for next simulations.

An example of a runtime parameter file called `rootFinder.init` is given in the AM 129 lecture note along with the routine `setup_module.F90` which reads-in the values from `rootFinder.init`. Study the example implementations for this homework because this runtime parameter file will be important in the final term project as well. The sample implementations in the AM 129 lecture note can be used directly or with some modifications.

1.2. Model Equation

We solve the 1D non-linear scalar advection using the Burgers' equation:

$$u_t + \left(\frac{u^2}{2}\right)_x = 0. \tag{2}$$

Discretization: Write a Fortran program to implement the first-order Godunov method (FOG), and the second-order piecewise linear method (PLM). The PLM implementation employs one of the six slope limiters as mentioned above.

Initial condition (IC): In Chapter 7 of the course lecture note, there are 10 Examples of different initial conditions, Eqs. (7.38) – (7.47), plus one more in Part (a) of Problem 1 in Eq. (7.48) – a total of 11 initial conditions. The main goal in this homework is to see if your code can pass all these 11 setups using both FOG and PLM.

Boundary condition (BC): An outflow condition is to be used for all ten example cases, while a periodic condition is to be used for Part (a) of Problem 1 in Eq. (7.48).

1.3. Python (or MATLAB) Implementations

Use Python (or MATLAB) to produce various plots from the Fortran (or C) outputs. To do this, see AM 129 lecture note to learn how to use matplotlib to plot ASCII format data.

1.4. LaTeX Report

Write your final report using LaTeX (20-page limit including figures using 11 font size on a single space format). You have to write three parts in your report:

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

Questions:

Address the following questions in your LaTeX report. In this report, you are going to maximize the use of plots, clearly describing the results with analytical discussions. To do this, each plot has to be displayed with an informative caption that clearly explains what you're showing (e.g., initial condition, runtime parameters, grid resolution, etc.). In all numerical results, use different colors and symbols for different runs. For example,

- `plt.plot(x, u, color="green", linewidth=2.5, linestyle="-", marker='d',markersize=10)` in Python
- `plot(x,u, 'ro:','LineWidth',1.2)` in Matlab

A brief discussion after each plot should be given to illustrate what you demonstrate.

(a) Run all 11 cases using both FOG and PLM up to $t_{\max} = 0.3$ with the grid resolution of $N = 32$ and CFL=0.9. For PLM, try all six different slope limiters on your numerical experiments but report (i.e., show plots) only the runs with three TVD slope limiters. You are going to display four plots including one from FOG and three from the TVD slope limiters using PLM.

(b) Try out other combinations with smaller/larger CFL, smaller/higher grid resolutions, shorter/longer advection time $t_{\max}$, etc. Select a couple of interesting cases that can represent your findings on different combinations of numerical parameters. Describe what you see.

(c) Change the initial condition of the nonlinear sine advection in Part (a) of

Problem 1 in Eq. (7.48) to $u(x,0) = u_0(x) = 0.5\sin(2\pi x) + 1.5$. Compared to the previous case with $u(x,0) = u_0(x) = \sin(2\pi x)$, this IC will add one more flow dynamics – advection – and turn the solution progressively into a shock as the overall profile simultaneously moves to the right (Note: can you make it to the left instead?). Compare the solutions of FOG and PLM with the three TVD limiters at $t_{\text{max}} = 1.5$ using CFL=0.9. You may overplot all four cases in one figure as long as you use distinctive color schemes and symbols.

(d) Design two new initial conditions on a periodic domain in such a way that the solution forms a rarefaction on the left and a shock on the right. Once formed, the overall profile is to be advected to the left of the domain in your first IC configuration and to the right in your second IC configuration. Describe your initial conditions and explain why your configurations should do the expected flow dynamics. Solve them using both FOG and PLM with a choice of your favorite TVD slope limiter. Compare your FOG and PLM solutions at t_{max} which is to be chosen large enough to display the anticipated temporally evolved profile of the solutions.