Final Project: Computational Fluid Dynamics

Due Friday May 16th

1 Introduction

The Midterm Project introduced the basic concepts of computational fluid dynamics by examining different schemes for numerical solution of the linear advection equation

$$\frac{\partial y}{\partial t} + a\frac{\partial y}{\partial x} = 0. \tag{1.1}$$

The present project builds on this foundation by exploring techniques for solving Burgers' equation,

$$\frac{\partial y}{\partial t} + y\frac{\partial y}{\partial x} = 0; \tag{1.2}$$

this resembles the linear advection equation, but with the constant wave speed a replaced by the dependent variable y. The resulting non-linearity in y means that discontinuities can arise in initially-smooth flow, significantly complicating the solution process.

Two possible initial states will be used in this project: the 'rising' state with

$$y(x,0) = \begin{cases} 0 & x < -1 \\ \frac{x+1}{2} & -1 < x < 1 \\ 1 & x > 1 \end{cases}, \tag{1.3}$$

and the 'falling' state with

$$y(x,0) = \begin{cases} 1 & x < -1 \\ \frac{1-x}{2} & -1 < x < 1 \\ 0 & x > 1 \end{cases}. \tag{1.4}$$

These are the same initial states considered in §9.2 of the Lecture Notes[1], and so the solutions constructed there — using the method of characteristics — can be used to validate numerical results.

2 Finite-Difference Schemes

The numerical schemes introduced in the Midterm Project were developed by discretizing the linear advection equation onto space and time grids, and then using finite differences to

[1] Observe that in the notes the dependent variable is u, instead of the y used here.

approximate the derivatives appearing in the equation. Alternative finite-difference approximations led to alternative numerical schemes, each with its own strengths and weaknesses.

Modify your Python code from the Midterm Project (or modify Rich's Python code, which is available on request) to solve Burgers' equation (1.2) using each of the FTCS, FTBS and FTFS numerical schemes. To do this, replace a by y at the appropriate grid point; so, for instance, the FTCS scheme becomes

$$y_k^{n+1} = y_k^n - (t^{n+1} - t^n)\, y_k^n \, \frac{y_{k+1}^n - y_{k-1}^n}{x_{k+1} - x_{k-1}}. \tag{2.5}$$

Instead of periodic boundary conditions, use straight extrapolation to calculate values outside the grid bounds:

$$y_{-1}^n \equiv y_0^n \tag{2.6}$$
$$y_K^n \equiv y_{K-1}^n \tag{2.7}$$

Apply your new code to solve Burgers' equation over the domains $-2 \leq x \leq 6$ and $0 \leq t \leq 5$, for both the rising initial state (1.3) and the falling initial state (1.4), and using each of the FTCS, FTBS and FTFS schemes. You should use grids with uniform spacings $\Delta x = 0.01$ and $\Delta t = 0.005$.

Q1 For each of the three schemes, determine whether the scheme is stable or unstable. In light of the results of the stability analyses you undertook in the midterm project, are your findings here expected or surprising?

Q2 For the stable scheme (there should be only one!), plot $y(x)$ snapshots for the rising and falling initial states at $t = 0$, $t = 1$, $t = 2$ and $t = 4$. Compare these snapshots against the analytic solutions presented in §9.2 of the Lecture Notes. Specifically,

i. For the rising initial state, measure the propagation speeds of the head of the ramp (initially at $x = -1$) and the tail of the ramp (initially at $x = 1$). Compare these against the characteristic speeds of the head and tail, shown in Fig. 9.5 of the Notes — how good is the match?

ii. For the falling initial state, confirm that a flow discontinuity develops at $x = 1$, $t = 2$, as predicted in Fig. 9.10 of the Notes.

iii. Again for the falling initial state, determine the propagation speed of the discontinuity for $t > 2$. You'll notice that the speed is different from the $S = 1/2$ shown in Fig. 9.10 (and derived in the accompanying text). With reference to the details of the numerical scheme, can you explain why this is?

Q2 highlights a known deficiency with finite-difference schemes: even if they are stable around flow discontinuities, they often get the propagation speed of the discontinuity *wrong*. This is because finite-different schemes are not numerically conservative — they do not necessarily enforce the strict conservation laws that the underlying fluid equations embody.

3 Finite-Volume Schemes

Finite-volume schemes are an alternative numerical approach to solving fluid equations, which benefit from being exactly[2] conservative. As a result, they do a very good job of correctly predicting the propagation speed of discontinuities — and this is why they are so common in astrophysical applications, where shocks and discontinuities often crop up.

Finite-difference schemes consider the time evolution of the dependent variable y at discrete points x_k. In contrast, finite-volume schemes consider the evolution of spatial averages of the dependent variable over cells of finite extent. The average in the k'th cell at time t^n is defined as

$$\overline{y}_k^n \equiv \frac{1}{x_{k+1/2} - x_{k-1/2}} \int_{x_{k-1/2}}^{x_{k+1/2}} y(x, t^n)\,\mathrm{d}x, \tag{3.8}$$

where $x_{k-1/2}$ denotes the location of the interface between cells $k-1$ and k, and likewise $x_{k+1/2}$ the location of the interface between cells k and $k+1$. (These half-integer indices are for notational convenience only; when writing a code, we have to use, e.g., `x[k]` to store $x_{k-1/2}$ and `x[k+1]` to store $x_{k+1/2}$).

To determine the time evolution of the $\overline{y}$, consider the scalar conservation law

$$\frac{\partial y}{\partial t} + \frac{\partial f}{\partial x} = 0. \tag{3.9}$$

Both the linear advection equation (1.1) and Burgers' equation (1.2) are instances of this, with fluxes $f(y) = ay$ and $f(y) = y^2/2$, respectively. Recall from the Notes that this equation, representing conservation of the dependent variable y in differential form, has an associated integral form

$$\int_{x_\mathrm{a}}^{x_\mathrm{b}} y(x, t_\mathrm{b}) - y(x, t_\mathrm{a})\,\mathrm{d}x + \int_{t_\mathrm{a}}^{t_\mathrm{b}} f[y(x_\mathrm{b}, t)] - f[y(x_\mathrm{a}, t)]\,\mathrm{d}t = 0, \tag{3.10}$$

where $(x_\mathrm{a}, x_\mathrm{b})$ is an arbitrary spatial interval and $(t_\mathrm{a}, t_\mathrm{b})$ an arbitrary time interval. If we make the choices $(x_\mathrm{a}, x_\mathrm{b}) = (x_{k-1/2}, x_{k+1/2})$ and $(t_\mathrm{a}, t_\mathrm{b}) = (t^n, t^{n+1})$, then this integral form becomes

$$(x_{k+1/2} - x_{k-1/2})(\overline{y}_k^{n+1} - \overline{y}_k^n) + (t^{n+1} - t^n)(\overline{f}_{k+1/2}^{n+1/2} - \overline{f}_{k-1/2}^{n+1/2}) = 0, \tag{3.11}$$

where $\overline{y}_k^n$ is the cell-average of y introduced before, and

$$\overline{f}_{k-1/2}^{n+1/2} \equiv \frac{1}{t^{n+1} - t^n} \int_{t^n}^{t^{n+1}} f[y(x_{k-1/2}, t)]\,\mathrm{d}x \tag{3.12}$$

is the time average over the interval (t^n, t^{n+1}) of the flux f at the cell interface $x_{k-1/2}$. Rearranging this equation, we arrive at a recipe for calculating the time evolution of $\overline{y}$:

$$\overline{y}_k^{n+1} = \overline{y}_k^n - \frac{t^{n+1} - t^n}{x_{k+1/2} - x_{k-1/2}}(\overline{f}_{k+1/2}^{n+1/2} - \overline{f}_{k-1/2}^{n+1/2}). \tag{3.13}$$

[2]Well, to within the numerical precision of the computer they're running on.

For uniform grids in space and time, this simplifies to

$$\overline{y}_k^{n+1} = \overline{y}_k^n - \frac{\Delta t}{\Delta x}(\overline{f}_{k+1/2}^{n+1/2} - \overline{f}_{k-1/2}^{n+1/2}). \tag{3.14}$$

These latter equations have a very simple physical explanation: the change in $\overline{y}_k$ arises from a inflow or outflow of y at either of the cell interfaces, as represented by the fluxes in the second term on the right-hand side. What's really neat about this approach is that the flow of y out of cell k through the $x_{k-1/2}$ interface is exactly matched by the flow of y *into* cell $k-1$, because this flow is represented by the same flux value $f_{k-1/2}$. Thus, our finite-volume scheme is guaranteed to be conservative!

4 Godunov's Method

The question of how to calculate the interface fluxes (cf. eqn. 3.12) lies at the heart of all finite-volume schemes. One of the most popular approaches was pioneered by Sergei Godunov. His idea was to treat each cell interface as an actual flow discontinuity, separating uniform fluid states on either side of the interface. The evolution of the discontinuity is then modeled using the Rankine-Hugoniot relations, ultimately leading to expressions for the interface flux.

For scalar conservation laws (3.9), a discontinuity between uniform left ('L') and right ('R') states evolves away from its starting position according to the jump relation

$$S(y_{\rm R} - y_{\rm L}) = f(y_{\rm R}) - f(y_{\rm L}). \tag{4.15}$$

Solving for the propagation speed of the discontinuity,

$$S = \frac{f(y_{\rm R}) - f(y_{\rm L})}{y_{\rm R} - y_{\rm L}}. \tag{4.16}$$

If $S > 0$ then the discontinuity moves toward the right, and the fluid state at the starting position is the left state $y_{\rm L}$. Conversely, if $S < 0$ then the discontinuity moves toward the left, and the fluid state at the starting position is the right state $y_{\rm R}$.

Let's apply this reasoning to the interface located at $x_{k-1/2}$, between the $k-1$ and k cells. At time t^n, the left and right states are y_{k-1}^n and y_k^n. From these states we calculate the discontinuity speed

$$S_{k-1/2}^n = \frac{f(y_k^n) - f(y_{k-1}^n)}{y_k^n - y_{k-1}^n}. \tag{4.17}$$

For $t > t^n$, the state at the interface is then

$$y(x_{k-1/2}, t) = \begin{cases} y_{k-1}^n & S_{k-1/2}^n > 0, \\ y_k^n & S_{k-1/2}^n < 0. \end{cases} \tag{4.18}$$

Substituting these expressions into the integrand of eqn. (3.12), we finally arrive at an expression for the time-averaged interface flux:

$$\overline{f}^{n+1/2}_{k-1/2} = \begin{cases} f(y^n_{k-1}) & S^n_{k-1/2} > 0, \\ f(y^n_k) & S^n_{k-1/2} < 0. \end{cases} \tag{4.19}$$

Use the foregoing analysis — specifically, eqns. (3.14), (4.17) and (4.19) — to develop a finite-volume, Godunov-method Python code for solving Burgers' equation. For boundary conditions, implement the same extrapolation conditions as before (cf. eqn. 2.6).

Apply the code to the same cases as before, covering the domains $-2 \leq x \leq 6$ and $0 \leq t \leq 5$ for both the rising initial state (1.3) and the falling initial state (1.4). You should find that your code is stable for either initial state.

Q3 Plot $\overline{y}(x)$ snapshots for the rising and falling initial states at $t = 0$, $t = 1$, $t = 2$ and $t = 4$. Following the same three steps as in Q2, compare these snapshots against the analytic solutions presented in §9.2 of the Lecture Notes. Is the propagation speed of the discontinuity now correctly reproduced?

Q4 For both initial states, calculate the integrated quantity

$$Y(t) \equiv \int y(x,t)\,\mathrm{d}x = \sum_{k=0}^{K-1} \overline{y}(t)\varDelta x \tag{4.20}$$

at each time step. This represents the total amount of the quantity y in the calculation domain. Plot $Y(t)$ as a function of time — can you explain why it decreases (increases) steadily with time for the rising (falling) initial state, and why in particular $|\mathrm{d}Y/\mathrm{d}t| = 1/2$?

Q5 A restriction on the application of Godunov's method is that, across a time step of extent $\varDelta t$, the discontinuity from one interface shouldn't be able to reach one of the neighboring interfaces (otherwise, eqn. 4.18 becomes invalid). Use this restriction to determine an upper limit on the time steps used in your code. Confirm (with suitable plots) that when run with $\varDelta t$ above this limit, on either initial state, your code becomes unstable.