12 Finite Volume Methods

When solving a PDE numerically, how do we deal with discontinuous initial data? The Finite Volume method has particular strength in this area. It is commonly used for hyperbolic PDEs whose solutions can spontaneously develop discontinuities as they evolve in time. These solutions are often called shock waves.

Conservation Laws

Consider the conservation law

$$u_t + f(u)_x = 0, \tag{12.1}$$

where u is a (spatially) one-dimensional conserved quantity, and $f(u)$ is the flux of u. The continuous integral formulation of (12.1) states that

$$\frac{d}{dt}\int_a^b u(x,t)dx + \int_a^b f(u)_x \, dx = 0.$$

$\frac{d}{dt}\int_a^b u(x,t)dx$ may be thought of as the time evolution of the total 'mass' of u across the domain $[a,b]$, and is dependent only on the flux through the boundaries, since

$$\frac{d}{dt}\int_a^b u(x,t)dx = f(u(a)) - f(u(b)).$$

This fact is an important idea utilized by finite volume methods, which generally consider the evolution of u not at a given point, but instead in volume-averaged regions. For example, let $\{x_i\}$ be a grid of equally spaced points with spacing Δx, and let C_i be the i-th 'volume' (subinterval) defined by $(x_{i-1/2}, x_{i+1/2})$. We are interested in the evolution of the volume average of u over this interval,

$$U_i^n = \frac{1}{\Delta x}\int_{C_i} u(x,t^n)dx,$$

where $\{t^n\}$ is the time discretization.

The evolution of these volume-averaged quantities will depend only on the flux through the cell edges, so that

$$\frac{d}{dt}\int_{C_i} u(x,t)dx = f(u(x_{i-1/2},t)) - f(u(x_{i+1/2},t)). \tag{12.2}$$

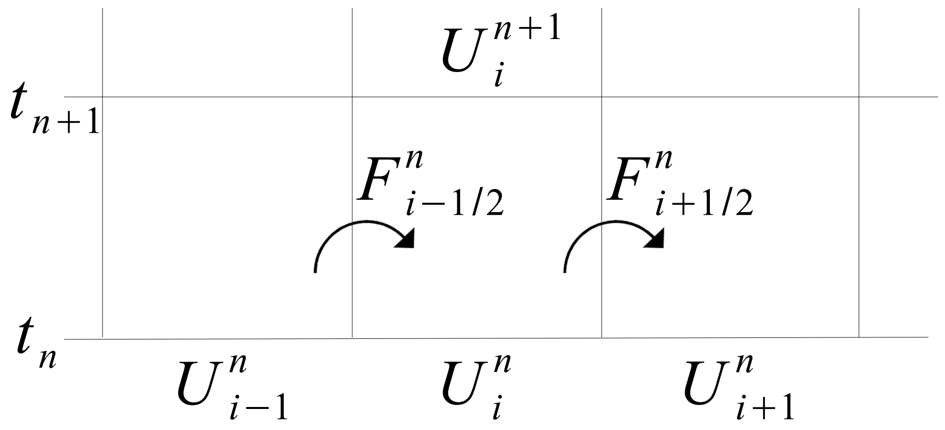

Figure 12.1: A schematic of the fluxes for the finite volume method as indicated by (12.3).

We can then construct a time-stepping method where $\sum_i U_i^n \Delta x$ (the total 'mass' of the system) is conserved from one time step n to the next.

Let $F_{i-1/2}^n = \frac{1}{\triangle t}\int_{t^n}^{t^{n+1}} f(u(x_{i-1/2}, t))\, dt$. Then

$$\begin{aligned} \int_{t^n}^{t^{n+1}} \left[\frac{d}{dt}\int_{C_i} u(x,t)\, dx\right] &= \int_{C_i} u(x, t^{n+1}) - u(x, t^n)\, dx, \\ &= \triangle t \left(F_{i-1/2}^n - F_{i+1/2}^n\right). \end{aligned}$$

Thus, by integrating (12.2) in time, we may approximate the evolution of the cell ('volume') averages with the method

$$U_i^{n+1} = U_i^n - \frac{\Delta t}{\Delta x}\left(F_{i+1/2}^n - F_{i-1/2}^n\right). \tag{12.3}$$

where $U_i^n = \frac{1}{\triangle x}\int_{C_i} u(x, t^n)\, dt$.

This formulation guarantees the conservation properties that are so desirable for conservation laws, if the time-averaged fluxes $F_{i-1/2}^n$ can be discretized in a natural way.

The key contribution of finite volume methods is the computation of $F_{i-1/2}^n$. For a truly nonlinear $f(u)$ this can be rather complicated and messy, and typically will involve solving what is usually referred to as the Riemann problem for the conservation law. The interested student can look at [**Le2002**] for a very thorough introduction and discussion on the subject. We will consider the linear problem in one dimension. The analog to higher dimensions is obtained by considering the eigenvector decomposition of any linear system. Nonlinear equations complicate things further.

The linear advection equation and upwinding

The simplest conservation law describes the advection or transport of a quantity. The PDE is given by

$$u_t + a u_x = 0, \tag{12.4}$$

and describes the motion of a concentration of some constituent u by a constant velocity one-dimensional 'wind' $a > 0$.

In higher dimensions this is an important problem in many fields, for example the transport of chemicals in the atmosphere and oceans, proper mixing of various properties in metallurgy, and the passing of information along a network.

Note that whenever $u(x,t)$ is a solution of the advection equation, then $u(x-at,t_0)$ (for any fixed t_0) is also a solution. Thus, if $u(x,0)=u_0(x)$ then the solution for all time can be represented by $u(x,t)=u_0(x-at)$. This is an important property of (12.4), and gives a new meaning to the term advection: this equation merely takes the initial conditions and passively transports them with velocity a.

For this equation the computation of the flux appears straightforward: $F^n_{i-1/2}=a\overline{U}^n_{i-1/2}$ where the $\overline{U}^n_{i-1/2}$ refers to the time average of $U_{i-1/2}$ over the interval t_n to t_{n+1}. Let us determine how to approximate this time average.

Note from Figure 12.2 that when $a>0$ the flux that determines U^{n+1}_i will be dependent on the value of U^n_{i-1}. Thus, one possibility is to approximate the flux by $F^n_{i-1/2}=aU^n_{i-1}$. Using this approximation of the flux together with the flux differencing formula (12.3) yields the first order upwind method, given by

$$U^{n+1}_i = U^n_i - \frac{a\Delta t}{\Delta x}\left(U^n_i - U^n_{i-1}\right).$$

Another way to derive the upwind method is to instead suppose that what we want to do is reconstruct $u(x)$ at each time step n inside each cell $(x_{i-1/2},x_{i+1/2})$from the mean values in that cell and its surrounding neighbors. This reconstructed $\tilde{u}(x)^n$ is then defined piecewise for each cell i. The solution at the next time step can be found as $\tilde{u}^{n+1}(x)=\tilde{u}^n(x-a\triangle t)$ which allows us to determine the fluxes $F^n_{i-1/2}$ once we have settled on a method for determining $\tilde{u}^n(x)$ in each cell. The simplest approach is

$$\tilde{u}^n(x)=U^n_i \text{ for } x\in(x_{i-1/2},x_{i+1/2})$$

This leads to fluxes given by

$$\begin{aligned} F^n_{i-1/2} &= \frac{a}{\triangle t}\int_0^{t_{n+1}-t_n} \tilde{u}^n(x_{i-1/2},t)\,dt, && (12.5)\\ &= \frac{a}{\triangle t}\int_0^{\triangle t} \tilde{u}^n(x_{i-1/2}-at)\,dt, && (12.6)\\ &= aU^n_{i-1}. \end{aligned}$$

The following code solves the problem

$$\begin{aligned} u_t + au_x &= 0, \quad 0<x<1,\\ u(x,t) &= f(x), && (12.7)\\ u(0,t) &= u(1,t), \end{aligned}$$

where

$$f(x)=\exp\left(\frac{-(x-.3)^2}{.005}\right)+\chi_{(.6,.7)}$$

and

$$\chi_{(.6,.7.)} = \begin{cases} 0 & x\le .6\\ 1 & .6<x<.7\\ 0 & x\ge .7\end{cases}$$

Notice that this PDE has periodic boundary conditions. Essentially we are evolving the signal around the unit circle. This allows us to evolve the signal much further to test our numerical methods, since we only have to discretize the interval $[0, 1]$ instead of a much larger domain. To see how to implement the boundary conditions, consider a grid $0 = x_0 < x_1 < \ldots < x_{N-1} < x_N = 1$ of evenly spaced points. Since $u(x)$ is periodic then $u(x_N) = u(x_0)$, so it is sufficient to track $x_0, \ldots, x_{N-1}$.

```python
import numpy as np
from matplotlib import pyplot as plt
from math import floor

def upwind(u0, a, xmin, xmax, t_final, nt):
    """ Solve the advection equation with periodic
    boundary conditions on the interval [xmin, xmax]
    using the upwind finite volume scheme.
    Use u0 as the initial conditions.
    a is the constant from the PDE.
    Use the size of u0 as the number of nodes in
    the spatial dimension.
    Let nt be the number of spaces in the time dimension
    (this is the same as the number of steps if you do
    not include the initial state).
    Plot and show the computed solution along
    with the exact solution. """
    dt = float(t_final) / nt
    # Since we are doing periodic boundary conditions,
    # we need to divide by u0.size instead of (u0.size - 1).
    dx = float(xmax - xmin) / u0.size
    lambda_ = a * dt / dx
    u = u0.copy()
    for j in range(nt):
        # The Upwind method. The np.roll function helps us
        # account for the periodic boundary conditions.
        u -= lambda_ * (u - np.roll(u, 1))
    # Get the x values for the plots.
    x = np.linspace(xmin, xmax, u0.size+1)[:-1]
    # Plot the computed solution.
    plt.plot(x, u, label='Upwind Method')
    # Find the exact solution and plot it.
    distance = a * t_final
    roll = int((distance - floor(distance)) * u0.size)
    plt.plot(x, np.roll(u0, roll), label='Exact solution')
    # Show the plot with the legend.
    plt.legend(loc='best')
    plt.show()

# Define the initial conditions.
# Leave off the last point since we're using periodic
# boundary conditions.
nx = 30
nt = nx * 3 // 2
x = np.linspace(0., 1., nx+1)[:-1]
```

```
u0 = np.exp(-(x - .3)**2 / .005)
arr = (.6 < x)  &  (x < .7 )
u0[arr] += 1.

# Run the simulation.
upwind(u0, 1.2, 0, 1, 1.2, nt)
```

Try running the previous code block with `nx` set to 30, 60, 120, and 240. You will notice that the numerical solution diffuses with time. It diffuses especially fast at the points of discontinuity.

Piecewise linear reconstruction and slope limiters

The upwind method is formally only first order, and actually does relatively poorly in terms of actually transporting the initial data with velocity a. You can notice from the example code that the upwind method has errors that are 'diffusive' meaning that the initial data is diffused as time evolves, losing the peaks and fine details. This is because the error for the upwind method is on the order of the second derivative of u which is of a diffusive nature. To get an improved method, consider a better reconstruction inside each cell, i.e.

$$\tilde{u}^n(x) = U_i^n + m_i^n(x - x_i) \text{ for } x \in (x_{i-1/2}, x_{i+1/2}) \tag{12.8}$$

where the slope of this linear reconstruction m_i^n is determined as a function of the neighboring cell averages at time n and U_i^n itself. Then the flux is given by

$$\begin{aligned} F_{i-1/2}^n &= \frac{a}{\triangle t}\int_0^{t_{n+1}-t_n} \tilde{u}^n(x_{i-1/2} - at)\,dt, \\ &= \frac{a}{\triangle t}\int_0^{\triangle t} U_{i-1}^n + m_i^n(x_{i-1/2} - at - x_i), \\ &= a\left(U_{i-1}^n + \frac{m_{i-1}^n}{2}(\triangle x - a\triangle t)\right). \end{aligned} \tag{12.9}$$

One of the most natural approaches is to just estimate the slope depending on the cell i and a neighboring cell $i+1$ or $i-1$. This leads to two popular methods, the Lax-Wendroff method and the Beam-Warming method (that really is the name). The Lax-Wendroff method has a slope chosen as

$$m_i^n = \frac{U_{i+1}^n - U_i^n}{\Delta x}. \tag{12.10}$$

which it turns out is formally second-order accurate. It turns out though that the errors for this method are dispersive, meaning that near very steep gradients, the method will generate very rapid oscillations (due to the third derivative of u not being approximated accurately). Another way to consider how these errors arise is to notice from Figure 12.2 that if the piecewise linear reconstruction is advocated by some positive wind a then there will be places where the discontinuous nature of the reconstruction will introduce spurious maxima or minima into the solution. These become the spurious waves seen in simulations using the Lax-Wendroff method.

A solution to this dilemma between balancing the diffusive and dispersive errors comes from constructing slopes m_i^n that ensure no such non-monotonic transport takes place. The basic idea is to constrain the slope so that the reconstructed piecewise linear function $\tilde{u}^n(x)$ will not generate

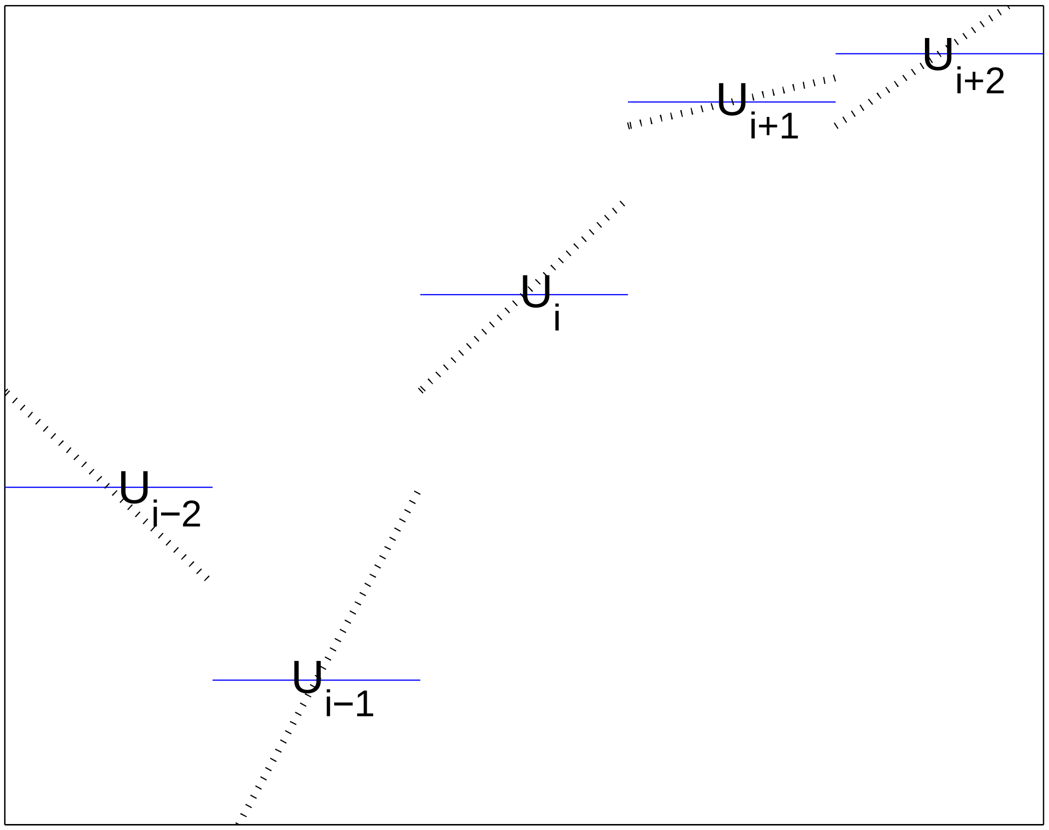

Figure 12.2: The piecewise linear reconstruction for the upwind and Lax-Wendroff methods. The solid lines represent the simplest reconstruction of the cell averages leading to the upwind method, and the dashed lines are those whose slope is obtained via the Lax-Wendroff method. Note that the LW method introduces a spurious maximum at $i + 3/2$ (the cell edge between U_{i+1} and U_{i+2}) and the minimum at $i - 3/2$ will be unphysical exaggerated. The upwind method avoids this difficulties, but clearly loses a significant amount of the available information. This provides the motivation for the slope limiters.

unphysical extremal values when it is advocated by some finite wind a. The Minmod limiter chooses the slope as

$$m_i^n = minmod\left(\frac{U_i^n - U_{i-1}^n}{\Delta x}, \frac{U_{i+1}^n - U_i^n}{\Delta x}\right) \tag{12.11}$$

where

$$minmod(a, b) = \begin{cases} a \text{ if } |a| < |b| \text{ and } ab > 0 \\ b \text{ if } |b| < |a| \text{ and } ab > 0 \\ 0 \text{ if } ab < 0. \end{cases} \tag{12.12}$$

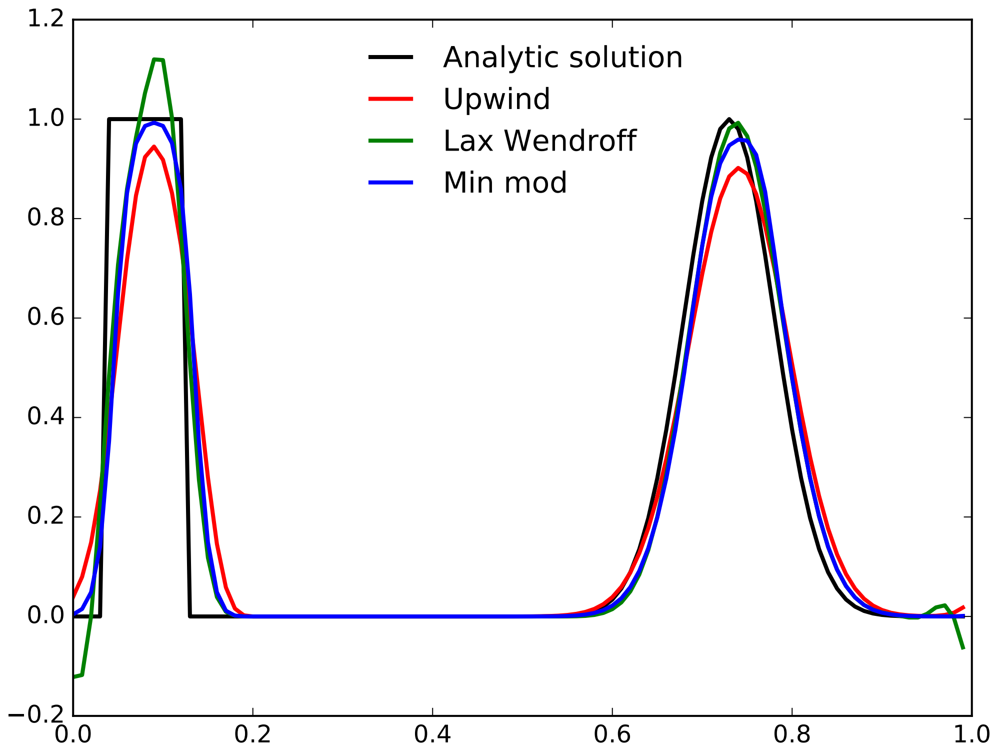

Figure 12.3: Solutions of (12.7) at time $t = 1.2$ using various methods. Here the advection coefficient is $a = 1.2$, and there are $N = 100$ subintervals in space, 150 subintervals in time.

Problem 1. Implement the Lax Wendroff method and use it to solve (12.7). For $N = 30, 60, 120, 240$, plot the analytic solution and the Lax-Wendroff solution. (You should have 4 separate plots, each with 3 graphs.) You should be able to tell that the Lax Wendroff method approximates the smooth portion of the signal much better, as it does not struggle with diffusion. Unfortunately, it has some difficulty with the discontinuous portion, where unphysical oscillations are seen. Recall that we saw something similar in the waves lab when there were discontinuous initial conditions.

Hint: Use equations 12.9 and 12.3.

Problem 2. Implement the Minmod method and use it to solve (12.7). For $N = 30, 60, 120, 240$, plot the anaytic solution and the Minmod solution. (You should have 4 separate plots, each with 2 graphs.) Be sure to vectorize the minmod operation.

Hint: Use equations 12.9 and 12.3.