Photo: Wikipedia / Gulumeemee

Taiwan Scimitar Babbler
小彎嘴

ASTR 660 / Class 12

Finite volume methods

Recap of finite difference methods

Last time we used finite differences on a grid to solve the advection and wave equations. The issues we discussed were:

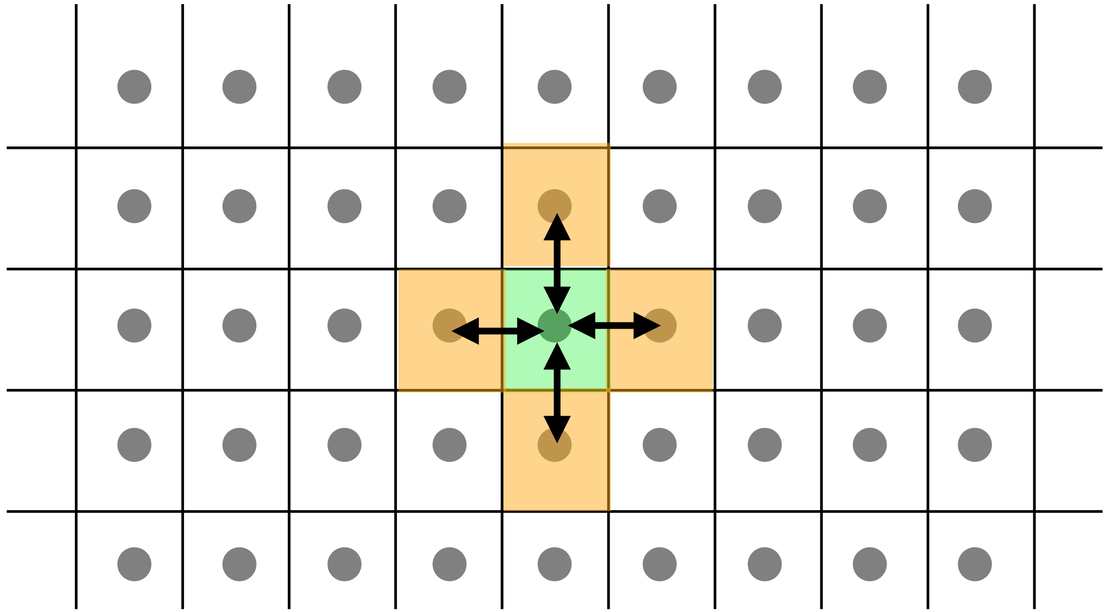
- Stability (numerical errors grow catastrophically);
- Courant condition/ choice of Courant number — the timestep is constrained by the grid spacing;
- Addition of artificial viscosity to counter diffusion due to approximation;
- Directional differences / discontinuities.

We now examine a different, more generally useful numerical approach: the finite volume method.

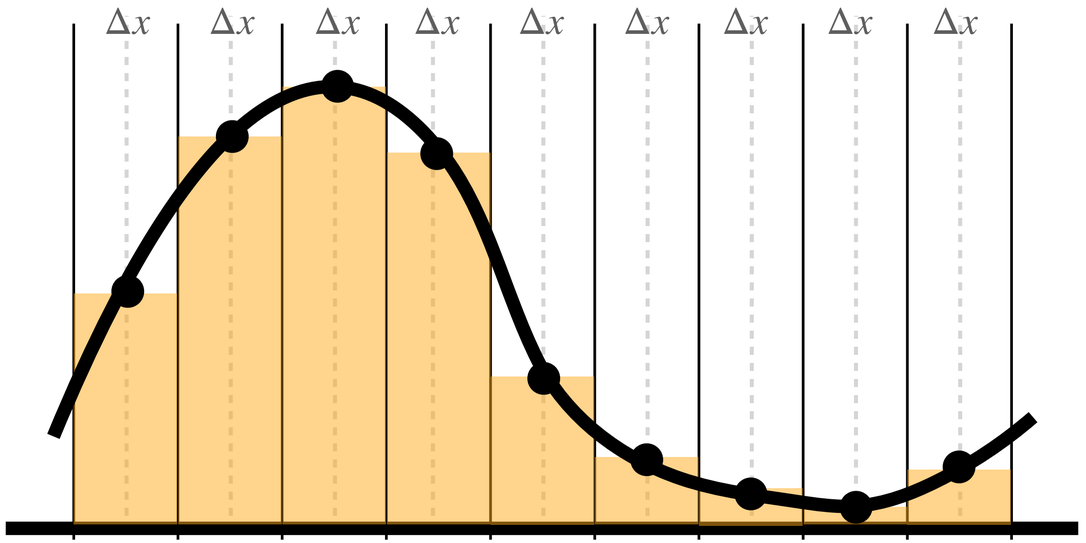
Finite Volume Methods

Each “gridpoint” is the midpoint of a **cell** around it (area in 2-d, volume in 3-d).

Rather the calculating properties only at the gridpoints, we track totals (or averages) of quantities, e.g. mass (density) over cells. We solve **conservative** equations for the flow of quantities into and out of the cells (**fluxes**).



1-dimensional finite volume method

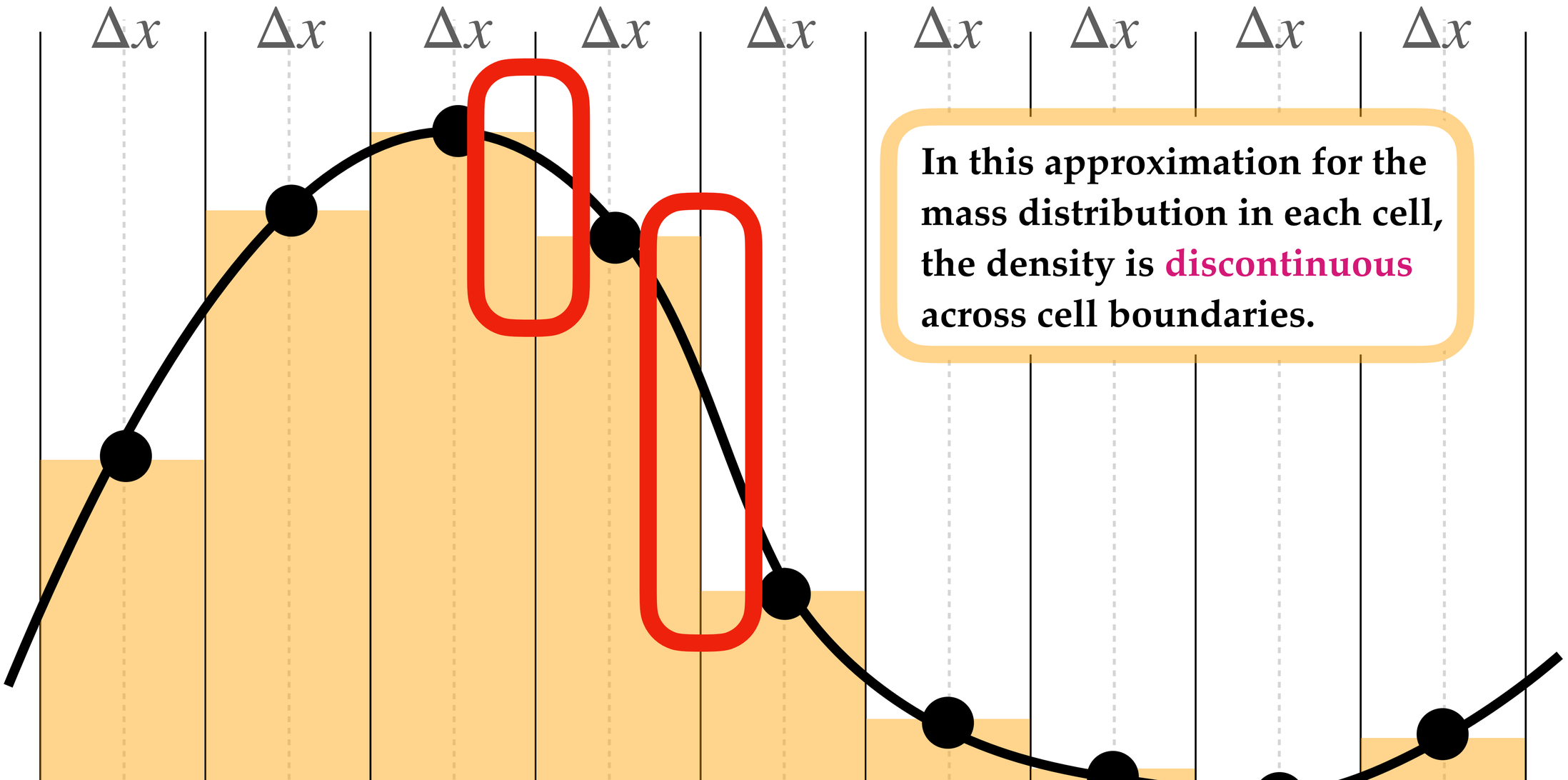


Space is divided into cells of width Δx. Each cell i has some mass m_i in it. We know the mean density of the cell, $\langle \rho \rangle_i = m_i / \Delta x$, but not how the mass is distributed inside the cell.

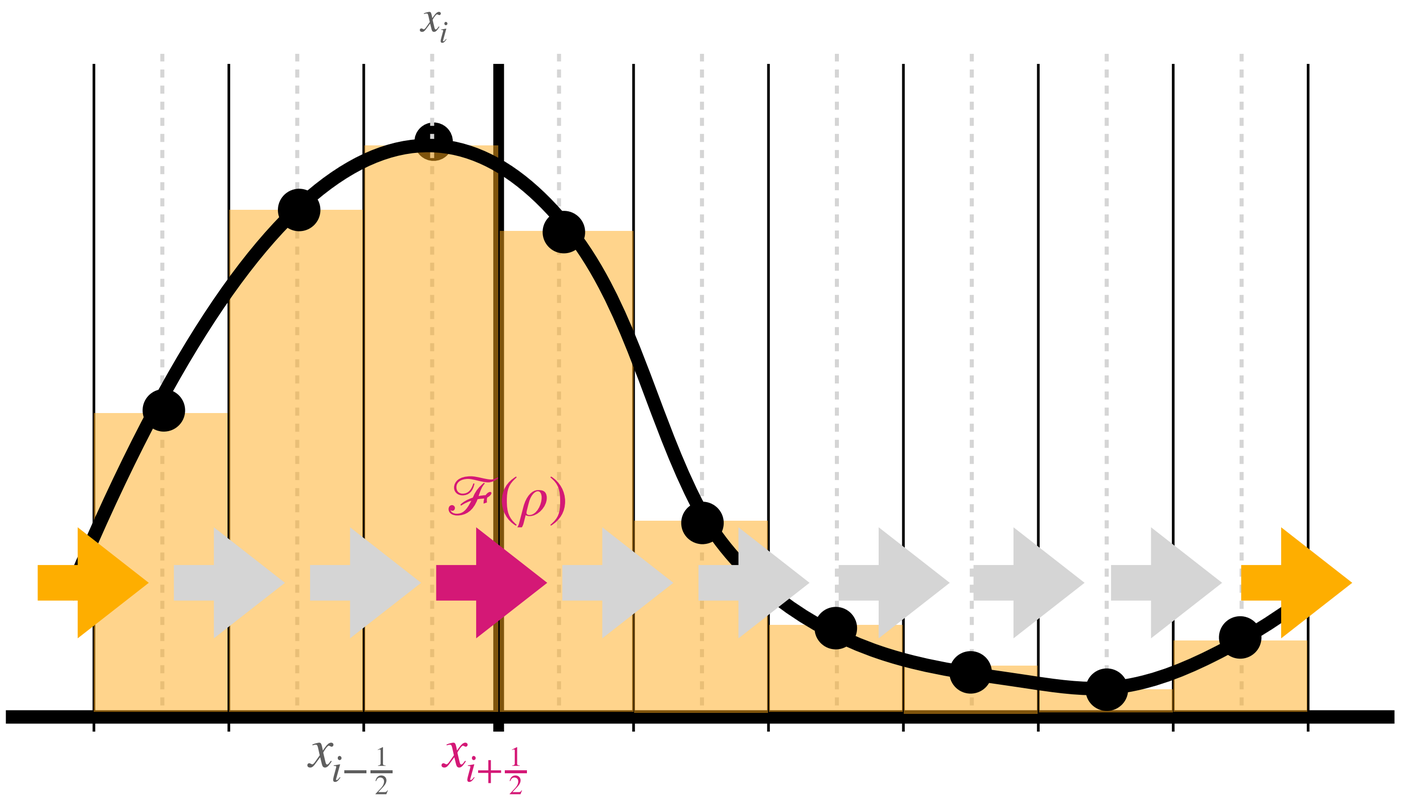
To second order, $\langle \rho \rangle_i = \int_{i-\frac{1}{2}}^{i+\frac{1}{2}} \rho(x)\, dx \approx \rho(x_i)$

If we approximate the mass distribution in a cell as **uniform**, the density in the cell can be represented by the density at the midpoint.

An important point for later



1-d advection

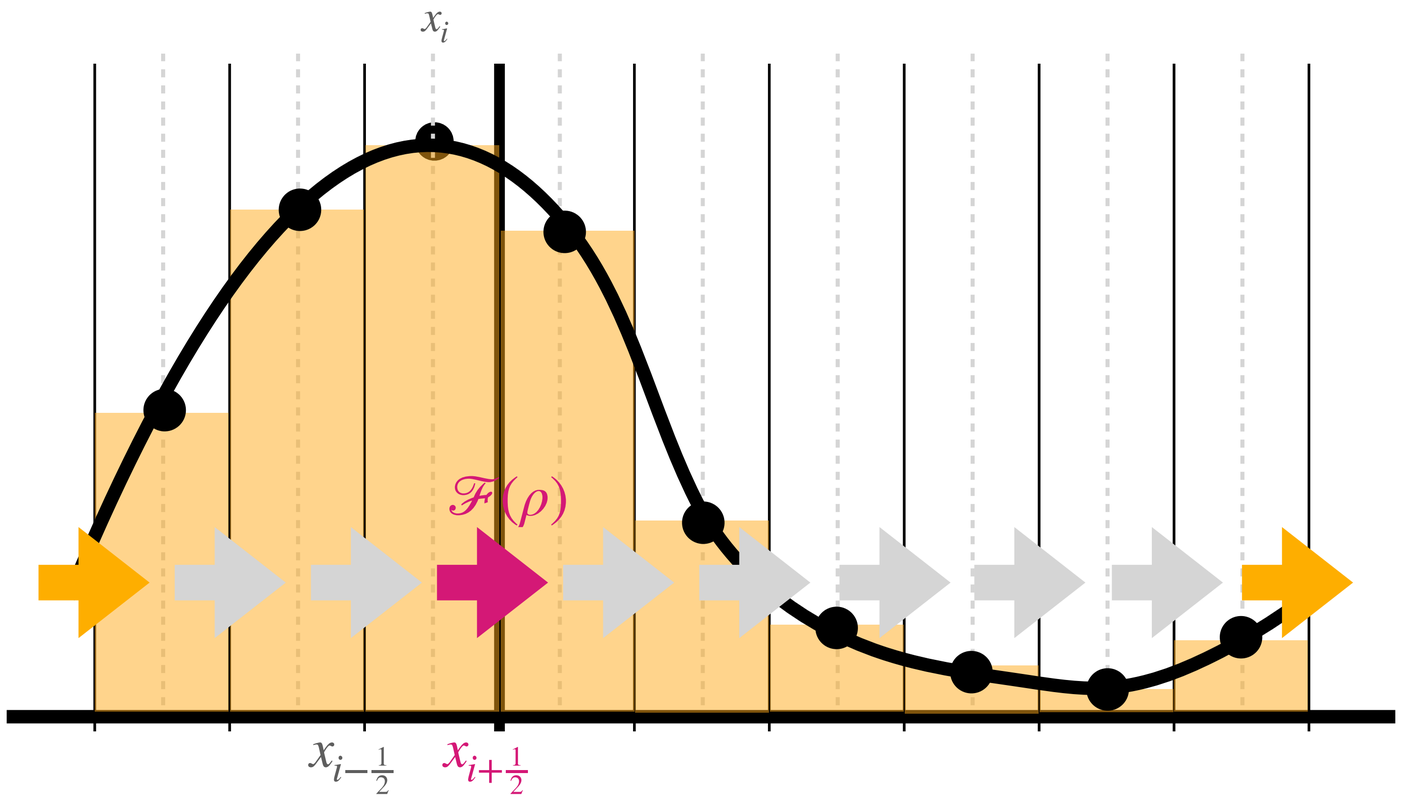


Mass is advected to the right, with periodic boundary conditions.

In a timestep Δt, the mass leaving x_i through the boundary $x_{i+\frac{1}{2}}$ is equal to the mass entering x_{i+1} through the same boundary.

The flux across the boundary is $\mathcal{F}(\rho)$, where $\rho \equiv \rho(x, t)$.

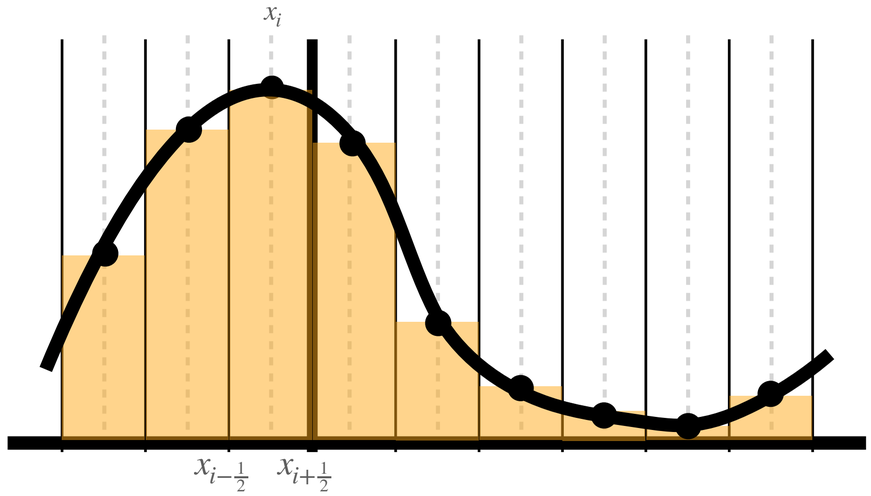
1-d advection



Example: uniform advection $\implies \mathscr{F}(\rho) = a\rho$.

How much mass moves from each cell to the next in time Δt?

1-d advection



How does the mass in cell *i* change over a timestep?

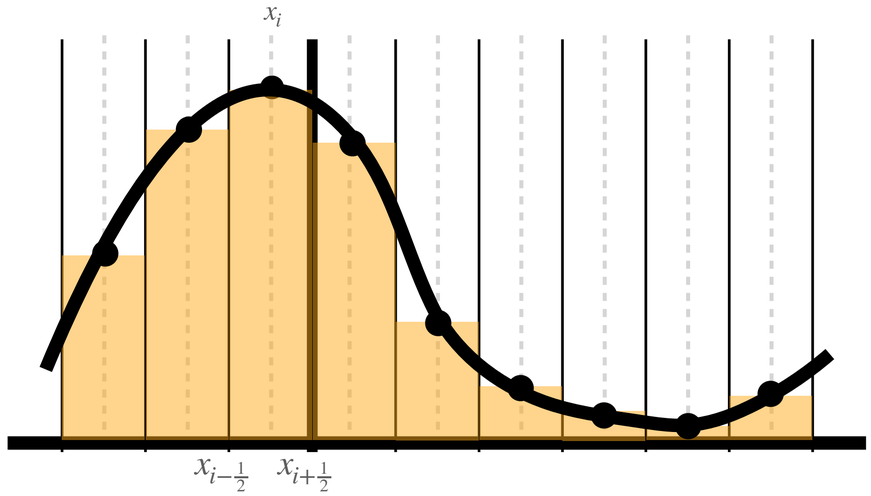
Integrate over a cell and divide by the cell width:

$$\frac{1}{\Delta x}\int_{x_{i-1/2}}^{x_{i+1/2}} \frac{\partial \rho}{\partial t} dx = -\frac{1}{\Delta x}\int_{x_{i-1/2}}^{x_{i+1/2}} \frac{\partial \mathscr{F}}{\partial t} dx$$

On the left, $\frac{1}{\Delta x}\int_{x_{i-1/2}}^{x_{i+1/2}} \frac{\partial \rho}{\partial t} dx = \frac{\partial}{\partial t}\int_{x_{i-1/2}}^{x_{i+1/2}} \frac{\rho dx}{\Delta x} = \frac{\partial \langle \rho \rangle_i}{\partial t}$, the rate of change of the average density in the cell.

On the right, $-\frac{1}{\Delta x}\int_{x_{i-1/2}}^{x_{i+1/2}} \frac{\partial \mathscr{F}}{\partial t} dx = -\frac{1}{\Delta x}\frac{\partial}{\partial t}\left[\mathscr{F}_{i+1/2} - \mathscr{F}_{i-1/2}\right]$, by the divergence theorem. $\iiint_V \nabla \cdot f \, dV = \oint f \cdot \hat{\nu} \, dS$

1-d advection



We have $\dfrac{\partial \langle \rho \rangle_i}{\partial t} = -\dfrac{1}{\Delta x}\dfrac{\partial}{\partial t}\left[\mathscr{F}_{i+1/2} - \mathscr{F}_{i-1/2}\right]$.

The change in the mean density of the cell is the difference between flux in and flux out.

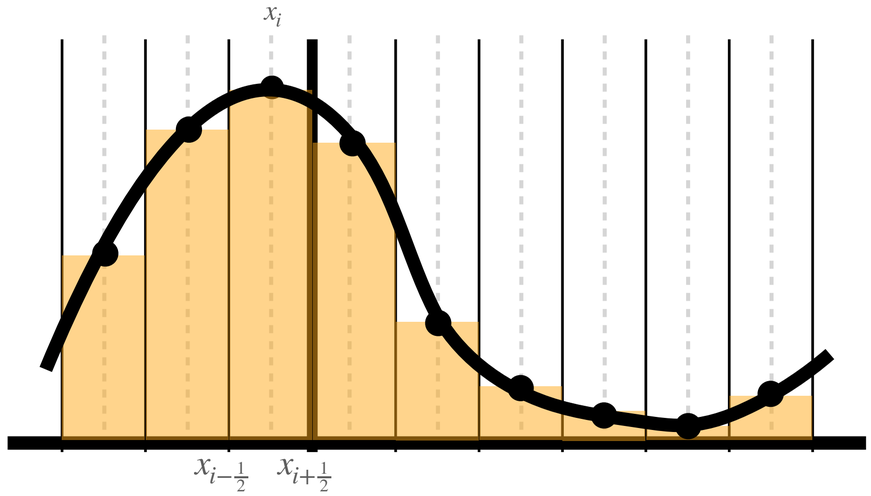
To find the total change over some time, we need to integrate. On the left side we have:

$$\int_{t_n}^{t_{n+1}} \frac{\partial \langle \rho \rangle_i}{\partial t}\, dt = \langle \rho \rangle_i^{n+1} - \langle \rho \rangle_i^{n} \quad (= \rho_i^{n+1} - \rho_i^{n})$$

$\Longrightarrow$ **We can compute the change in the average density in a cell (LHS) knowing the time-averaged fluxes across its boundaries (RHS).**

Summary so far

$$\langle \rho \rangle_i^{n+1} = \langle \rho \rangle_i^n - \frac{1}{\Delta x} \frac{\partial}{\partial t} \left[\mathscr{F}_{i+1/2} - \mathscr{F}_{i-1/2} \right]$$



If we can integrate expressions for the fluxes, this is exact (to roundoff precision).

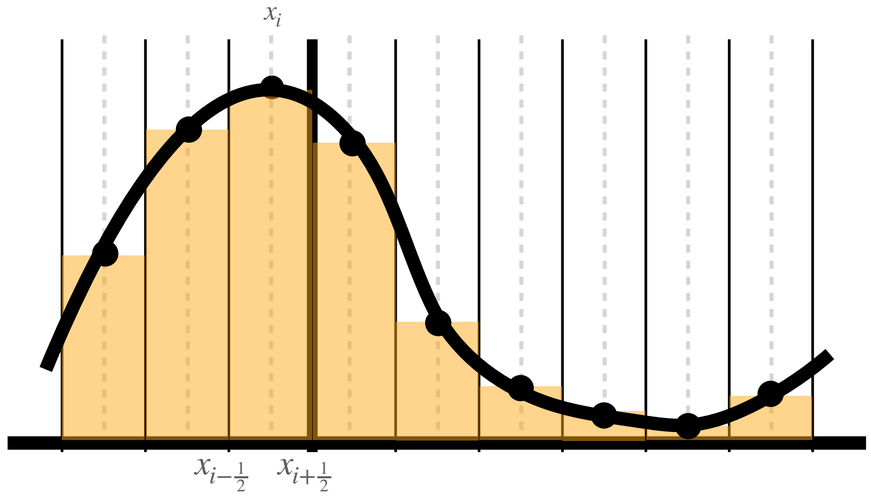
If we don't know the fluxes exactly (usually the case), then we have to approximate them. Remember, we are representing the state of the fluid (in this case, only density) at the **midpoint** of the cell, but we want to know the fluxes on the **boundaries.**

We have to average the fluxes in space and time, with finite differencing (again).

There is another method involving ODE solvers, which we won't talk about: see e.g. Zigale.

Much of the effort in FVMs is about reducing the error that comes from the averaging.

Evolution in time



For time, we can apply the same logic we used for the cells in space: replace the average flux over a timestep with the flux at the midpoint of the timestep, e.g. $\langle \mathscr{F} \rangle_{i+1/2}^{n+1/2} = \dfrac{1}{\Delta t} \displaystyle\int_{t_n}^{t_{n+1}} \mathscr{F}_{i+1/2}\, dt.$

This gives the following ***conservative update scheme***:

$$\langle \rho \rangle_i^{n+1} = \langle \rho \rangle_i^n + \frac{\Delta t}{\Delta x} \left[\langle \mathscr{F} \rangle_{i-1/2}^{n+1/2} - \langle \mathscr{F} \rangle_{i+1/2}^{n+1/2} \right]$$

We could have made other choices for the time average; we will come back to this when we look at the more general case.

An algorithm for advection

1. Compute fluxes for cells at timestep $n+1/2$, using values at the current timestep n.

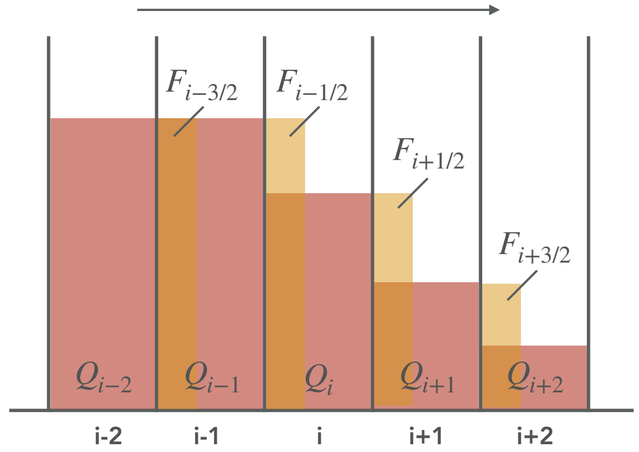
2. Treat the newly computed fluxes as the time-average fluxes over the timestep, and use them to update the density of each cell to the new timestep, $n+1$.

This should work OK ***as long as*** *our timestep is not so big that more than one cell advects across the boundary, i.e.* $a\Delta t < \Delta x$. *Sounds familiar?*

$$\langle \mathscr{F} \rangle_{i-1/2}^{n+1/2} = \mathscr{F}_{i-1/2}^{n} = a\rho_{i-1}^{n} = a\rho_{i-1}$$

$$\langle \mathscr{F} \rangle_{i+1/2}^{n+1/2} = \mathscr{F}_{i+1/2}^{n} = a\rho_{i+1/2}^{n} = a\rho_{i}$$

$$\langle \rho \rangle_{i}^{n+1} = \langle \rho \rangle_{i}^{n} + \frac{a\Delta t}{\Delta x}\left[\rho_{i-1} - \rho_{i}\right]$$



Godunov scheme for advection

Updated state

Time-averaged flux

Initial state

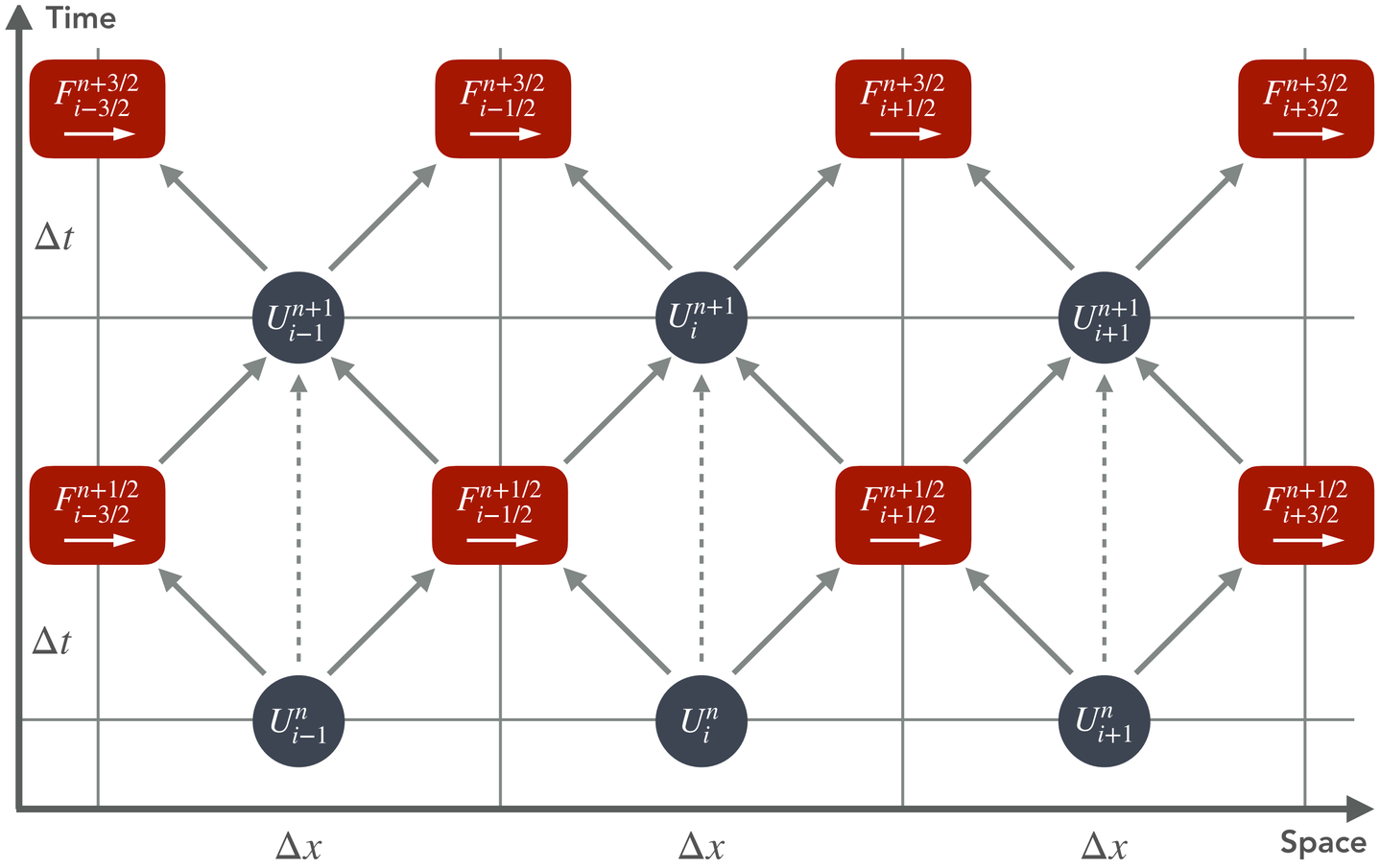
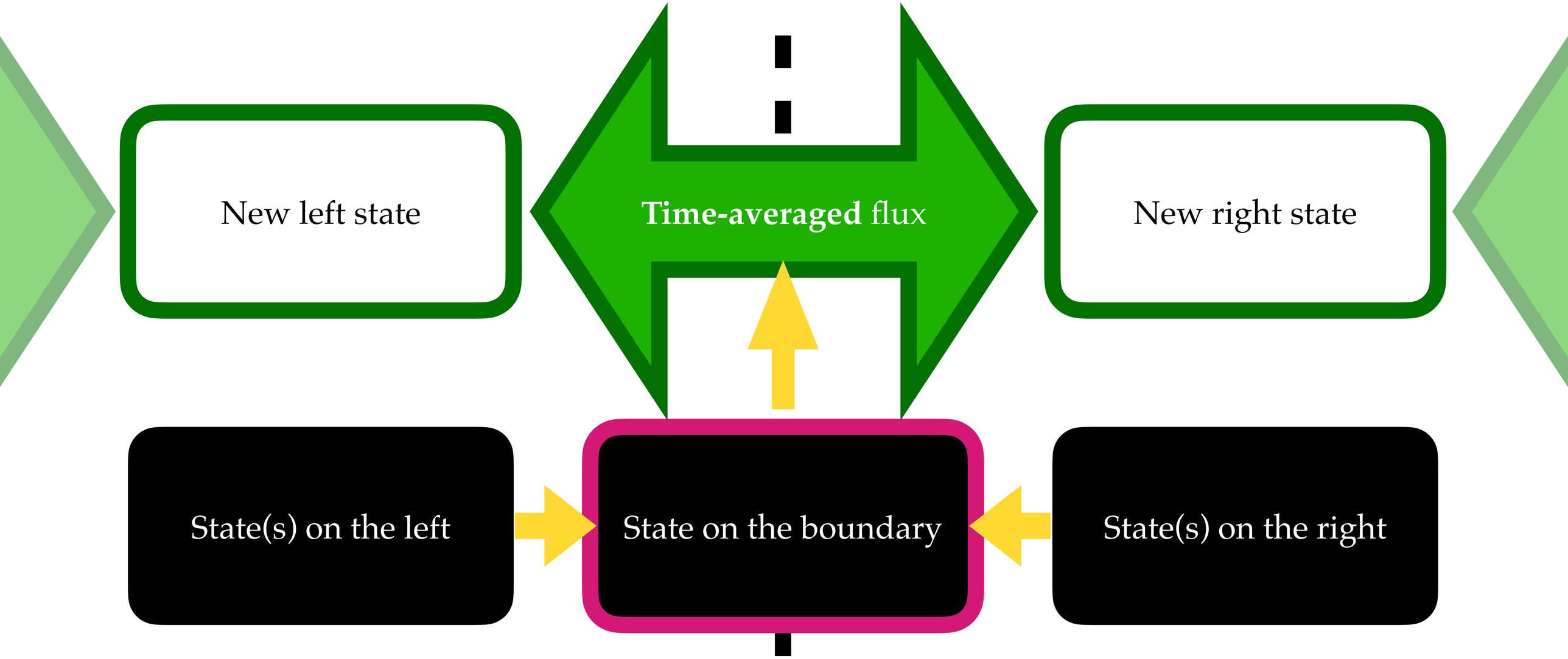


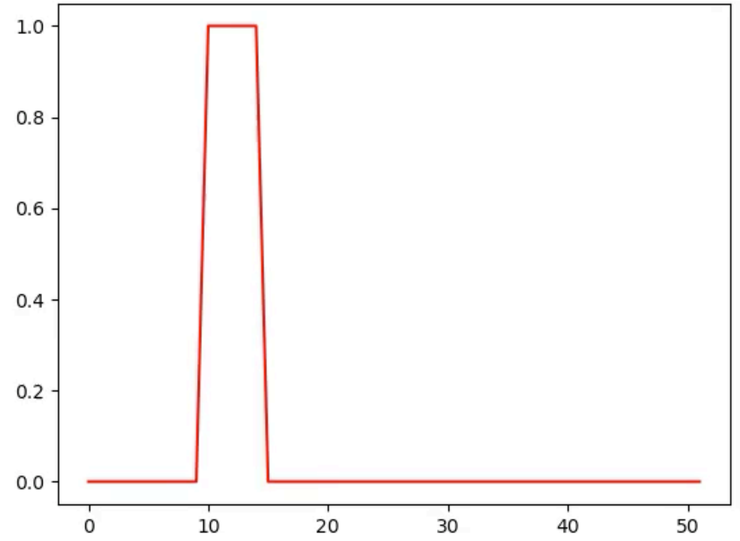
Figure: B. Diemer

Godunov schemes in general



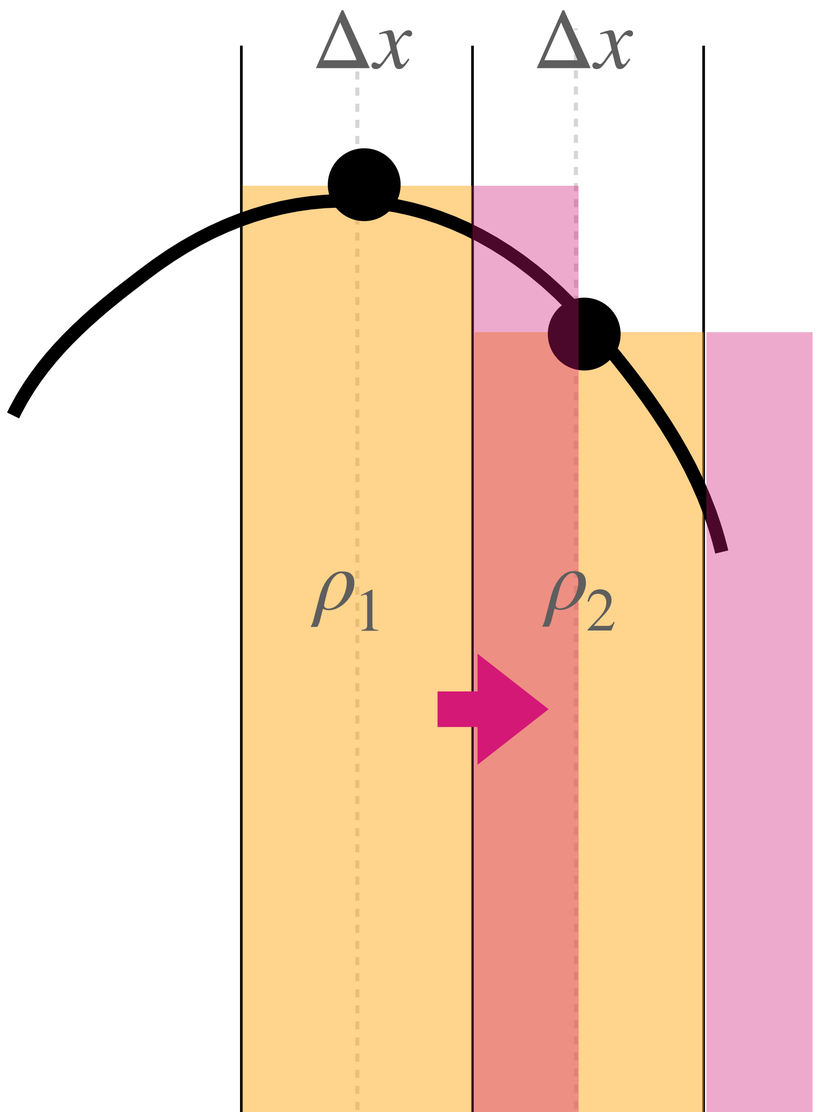
In code:

Similar in many ways to the FTBS finite difference method.




```
# Timestep units
dx     = 1.0
dt     = 1.0

# Advection speed
a = 0.7
# Courant number
C = a*dx/dt

for n in range(1,nsteps):
    # 1. Set boundary conditions (periodic, ghost cells)
    density[n-1, 0] = density[n-1,-2]
    density[n-1,-1] = density[n-1, 1]

    # 2. Calculate fluxes using backwards difference
    #    in space, with densities at previous step.
    for j in range(1,ngrid):
        # Advection: F = a*Q[i-1]
        fluxes[j] = a*(density[n-1,j-1])

    # 3. Update grid using Euler step (forwards difference)
    #    in time, with newly estimated fluxes.
    for i in range(1,ngrid-1):
        density[n,i] = density[n-1,i] + (dt/dx)*(fluxes[i]-fluxes[i+1])
```

Riemann problems



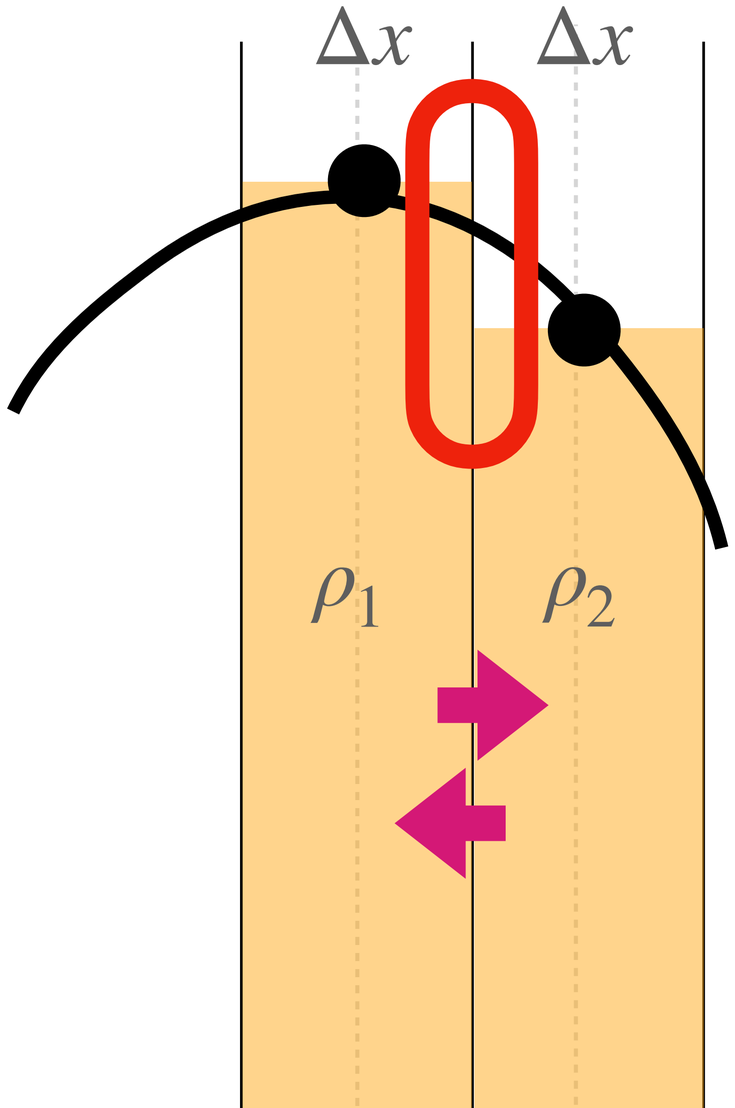
For the simple advection problem, this step is very simple, because we already know the answer (provided the Courant condition is satisfied and we know which way the fluid is moving):

The mass in each cell advects to the right, at constant speed.

This is not true in general, so we want Riemann solvers that can be applied to arbitrary conditions, with flux in *either direction*, due to bulk flow *and* diffusion (e.g. sound waves).

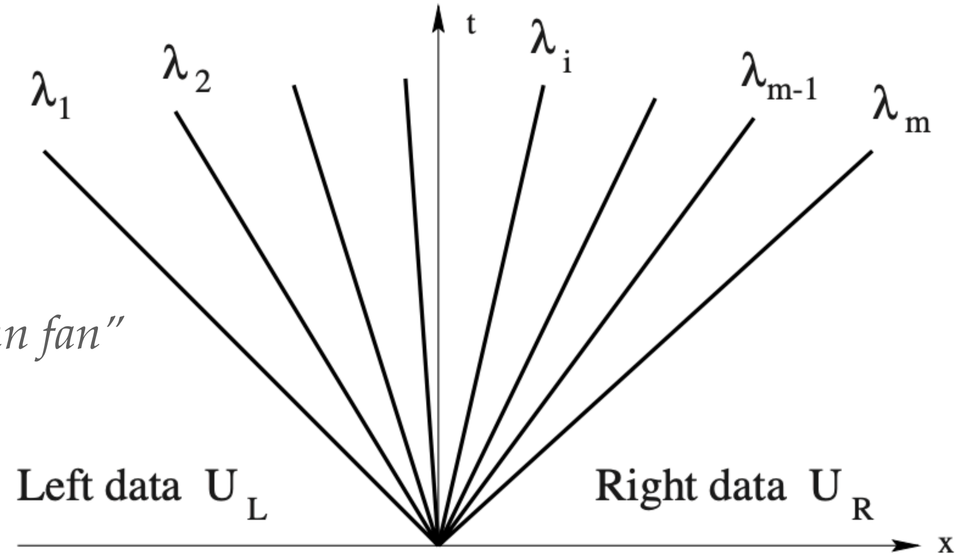
Even though it was simple, the advection solution involved **physics** (a, the advection speed). This is true also for more general Riemann solvers: we **need to know** about the physics that determines the fluxes.

Riemann problems



The states on the left and right are different.

We therefore have multiple characteristic speeds on each side of the boundary (three, for the standard Euler equations).



"Riemann fan"

Figure: Toro via Diemer

Photo: Wikipedia / F. Veronesi

Malayan Night-Heron
黑冠麻鷺

ASTR 660 / Class 12

Finite volume methods for general fluid problems

Fluid equations

Now we will go beyond advection to consider solutions to a complete set of fluid equations .

We can write these in a common form to emphasise their conservative nature, such that the system as a whole obeys a vector constraint:

$$\frac{\partial \vec{U}}{\partial t} + \nabla \cdot \vec{\mathcal{F}}(\vec{U}) = \vec{S}$$

Here $\vec{U}$ is a vector of conserved quantities, $\vec{\mathcal{F}}(\vec{U})$ is the flux of those quantities due to advection/dispersion, and $\vec{S}$ represents the sources or sinks of each quantity (i.e. things that can add or subtract from that quantity, independent of motion).

These vectors have **one row for each conserved quantity** in the problem.

Fluid equations

For example, mass is a conserved quantity, so we immediately write down one row:

$$\overrightarrow{U} = \begin{pmatrix} \rho \\ \vdots \\ \vdots \end{pmatrix}, \quad \overrightarrow{\mathcal{F}}(\overrightarrow{U}) = \begin{pmatrix} \rho\vec{v} \\ \vdots \\ \vdots \end{pmatrix} = \begin{pmatrix} \rho v_x & \rho v_y & \rho v_z \\ \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots \end{pmatrix}, \quad \vec{S} = \begin{pmatrix} 0 \\ \vdots \\ \vdots \end{pmatrix}$$

This corresponds to the continuity equation: $\frac{\partial \rho}{\partial t} + \nabla \cdot \rho\vec{v} = 0$. Notice that $\overrightarrow{\mathcal{F}}$ is a tensor.

The density ρ and velocity field $\vec{v}$ are fundamental (or *primative*) quantities, which are not conserved (the density and velocity at a particular position can change, as the continuity equation is obeyed).

Fluid equations

The other fluid equations also represent conservation laws, for **momentum** and **energy** (see e.g. appendix B1 in Diemer's notes).

$$\vec{U} = \begin{pmatrix} \rho \\ \rho v_x \\ \rho v_y \\ \rho v_z \\ E \end{pmatrix}, \quad \vec{\mathscr{F}}(\vec{U}) = \begin{pmatrix} \rho\vec{v} \\ \rho v_x\vec{v} + \delta_{xj}p \\ \rho v_x\vec{v} + \delta_{yj}p \\ \rho v_x\vec{v} + \delta_{zj}p \\ (E+p)\vec{v} \end{pmatrix}, \quad \vec{S} = \begin{pmatrix} 0 \\ -\rho\partial\Phi/\partial x \\ -\rho\partial\Phi/\partial y \\ -\rho\partial\Phi/\partial z \\ \rho\partial\Phi/\partial t + \Gamma - \Lambda \end{pmatrix}$$

The Euler equation involves a pressure-momentum **tensor**; the δ_{ij} terms represent the fact that the (scalar) pressure only appears in the terms along the diagonal.

The gravitational potential Φ is included: potential gradient $\Longrightarrow$ force $\frac{d}{dt}$(momentum).

Fluid equations

$$\vec{U} = \begin{pmatrix} \rho \\ \rho v_x \\ \rho v_y \\ \rho v_z \\ E \end{pmatrix}, \quad \vec{\mathcal{F}}(\vec{U}) = \begin{pmatrix} \rho\vec{v} \\ \rho v_x\vec{v} + \delta_{xj}p \\ \rho v_x\vec{v} + \delta_{yj}p \\ \rho v_x\vec{v} + \delta_{zj}p \\ (E+p)\vec{v} \end{pmatrix}, \quad \vec{S} = \begin{pmatrix} 0 \\ -\rho\partial\Phi/\partial x \\ -\rho\partial\Phi/\partial y \\ -\rho\partial\Phi/\partial z \\ \rho\partial\Phi/\partial t + \Gamma - \Lambda \end{pmatrix}$$

The energy part comes from $E = \rho\left(\frac{v^2}{2} + \epsilon + \Phi\right)$ with ϵ the specific internal energy.

A conservation equation for E follows from considering the time derivative and using the Euler equation to substitute p (see e.g. appendix B1 in Diemer's notes). $\frac{\partial E}{\partial t} + \nabla \cdot \left(\left[E+p\right]\vec{v}\right) = 0$

The Γ and Λ represent energy sources (heating) and sinks (cooling). In astrophysics these are important! Notice that variations of the potential in *time* change the energy.

Eigenvalues and wave speeds

A linear-algebra approach leads to the idea that the **eigenvalues** of $\vec{\mathcal{F}}(\vec{U})$ correspond to characteristic speeds at which information propagates in the fluid (see e.g. B2 in Diemer's notes).

If we assume no sources or sinks of any of the conserved quantities ($\vec{S} = 0$) and only one dimension:

$$\frac{\partial \vec{U}}{\partial t} + \frac{\partial \vec{\mathcal{F}}(\vec{U})}{dx} = \frac{\partial \vec{U}}{\partial t} + \frac{\partial \vec{\mathcal{F}}}{\partial \vec{U}} \frac{\partial \vec{U}}{\partial x} = \frac{\partial \vec{U}}{\partial t} + \vec{J}(\vec{U}) \frac{\partial \vec{U}}{\partial x} = 0$$

We have re-written the equations in terms of their **Jacobian**, a matrix of the first derivatives $\partial \mathcal{F}_i / \partial U_j$:

$$\vec{J}(\vec{U}) = \begin{pmatrix} \frac{\partial \mathcal{F}_0}{\partial U_0} & \frac{\partial \mathcal{F}_0}{\partial U_1} & \cdots \\ \frac{\partial \mathcal{F}_1}{\partial U_0} & \frac{\partial \mathcal{F}_1}{\partial U_1} & \cdots \\ \vdots & \vdots & \ddots \end{pmatrix}$$

Eigenvalues and wave speeds

$$\frac{\partial \vec{U}}{\partial t} + \vec{J}(\vec{U})\frac{\partial \vec{U}}{\partial x} = 0$$

The k'th eigenvalue of $\vec{J}$ satisfies $\vec{J}\,\vec{U}^{(k)} = \lambda_k \vec{U}^{(k)}$ where $\vec{U}^{(k)}$ is the corresponding eigenvector. Hence:

$$\vec{J}\frac{\partial \vec{U}^{(k)}}{\partial x} = \lambda_k \frac{\partial \vec{U}^{(k)}}{\partial x}\,.$$

And therefore we have:

$$\frac{\partial \vec{U}^{(k)}}{\partial t} + \lambda_k \frac{\partial \vec{U}^{(k)}}{\partial x} = 0$$

This is the advection equation ,with speed λ_k.

For 1-d advection of mass, there is one eigenvalue, the advection speed. When we add more equations to the system, more eigenvalues will appear.

Wave speeds

The basic set of fluid equations (in 1-d) have three physically conserved quantities: mass, momentum and energy, hence three characteristic velocities: the bulk velocity of the fluid, v, and $v \pm c_s$, the sound speed.

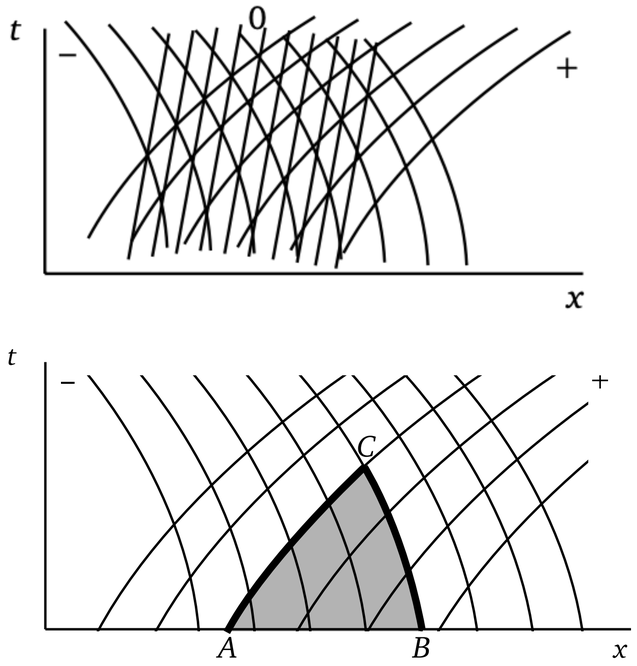
Sound waves can travel in either direction relative to the bulk velocity.

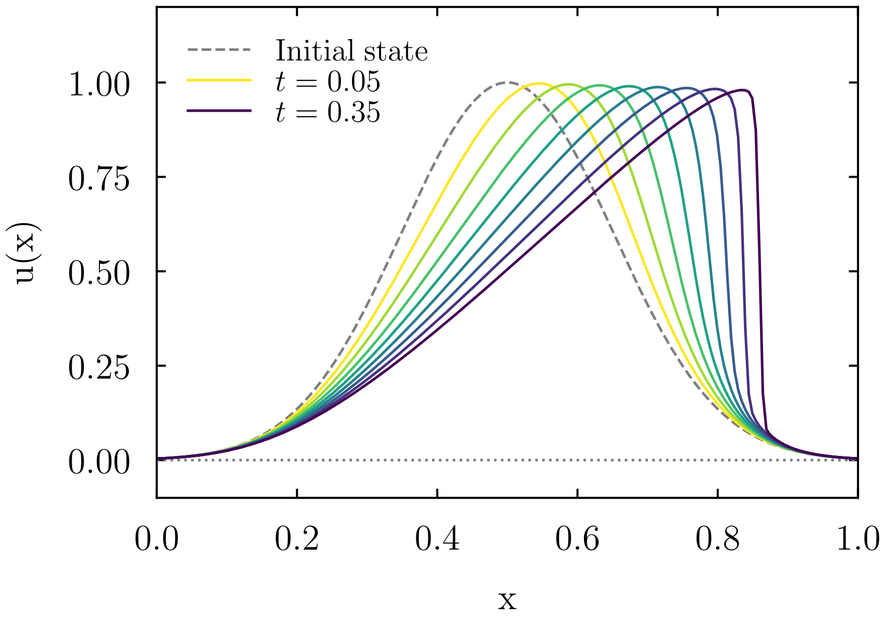
The speed of sound depends on the properties (equation of state) of the fluid. For an ideal gas with adiabatic index γ:

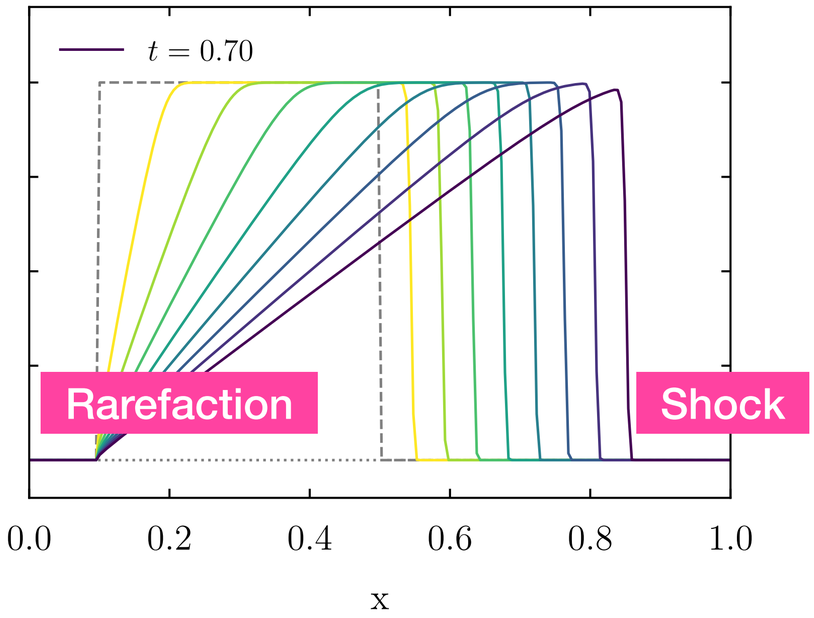
$$p = \rho\epsilon(\gamma - 1), \quad E = \frac{p}{\gamma - 1} + \frac{1}{2}\rho v^2 \text{ (total internal energy + KE)}; \quad c_s = \sqrt{\gamma \frac{p}{\rho}}$$

Shocks and rarefaction waves

These nonlinear features form as a consequence of interactions between advection and dispersion, which correspond to different characteristic solutions to the PDEs.







Fluid equations

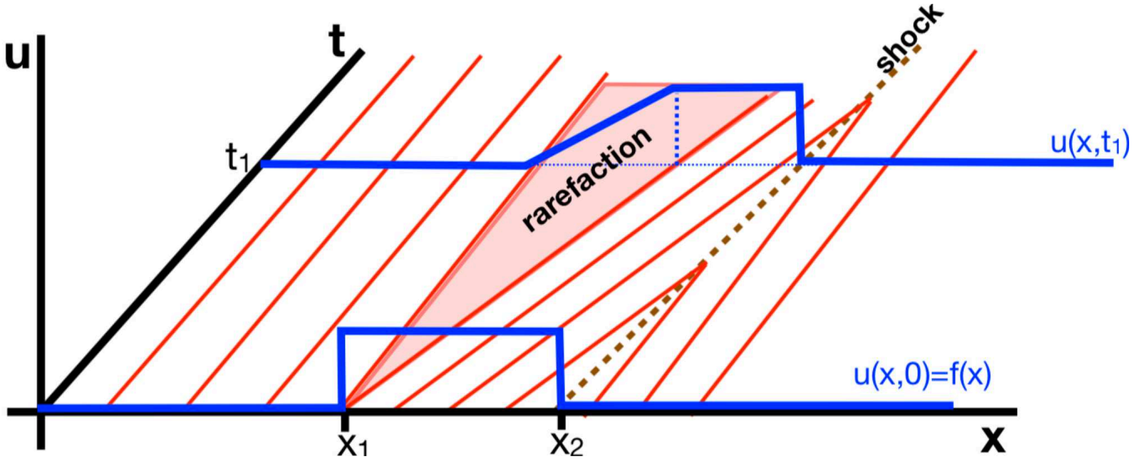
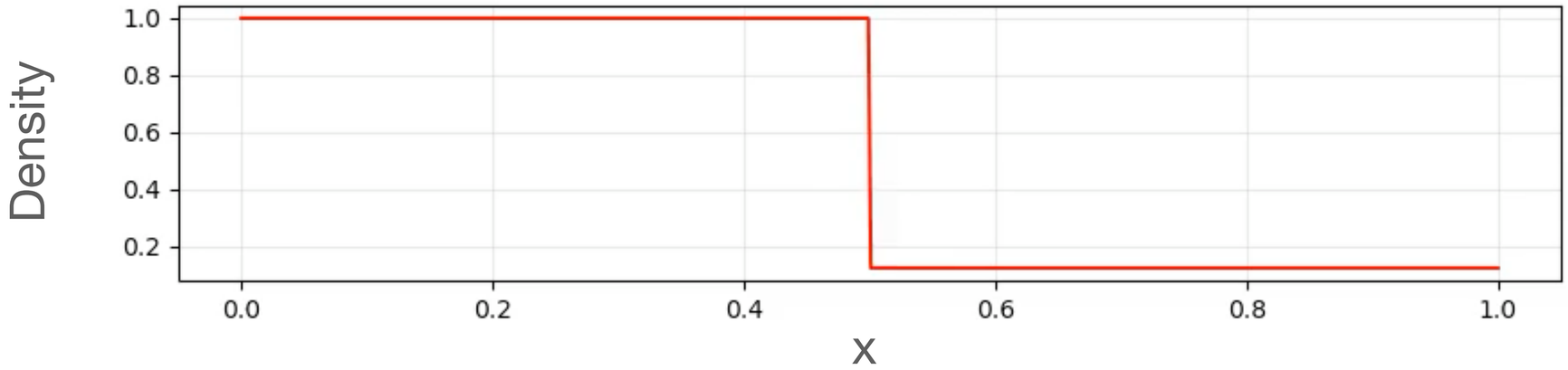


Figure: van den Bosch, via Diemer (caption)

Figure 13: Solution of the Burgers' equation by the method of characteristics. The red lines indicate so-called characteristics where the velocity remains at its initial value. Where multiple characteristics intersect, a shock forms; where they diverge, we get a rarefaction wave. Figure from van den Bosch.

The Sod shock tube problem

A standard test for hydro solvers (not just grid-based ones) is Sod's **shock tube**. This is a 1-d problem for the evolution of a **contact discontinuity** between a region of high density and pressure (left) and a region of low density and pressure (right), with no bulk motion.



We don't have time to go into the details (see e.g. Diemer's notes, chapter 8). The important things to appreciate are what's happening, and why this is a useful test problem.

The Sod shock tube problem

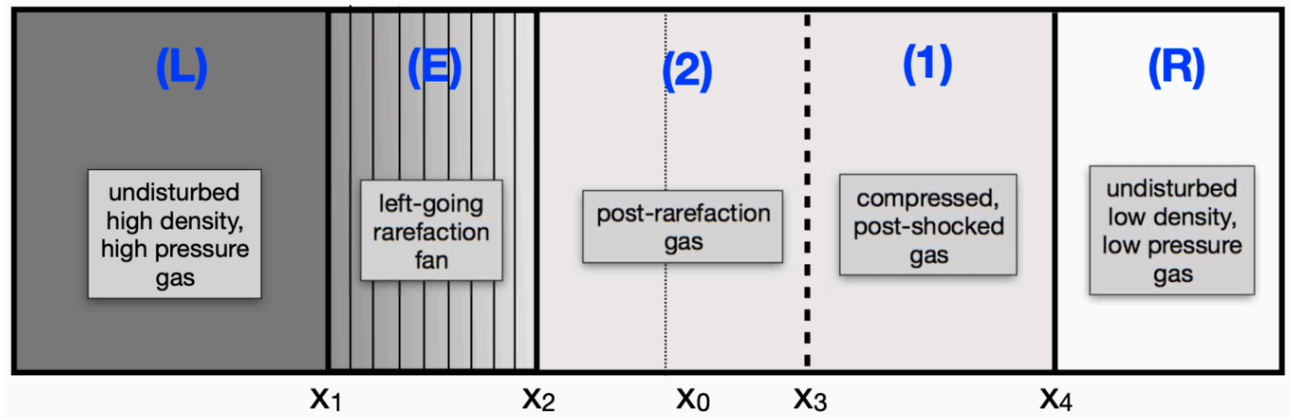
The shock tube is a “macroscopic” physical problem. It is a Riemann problem.

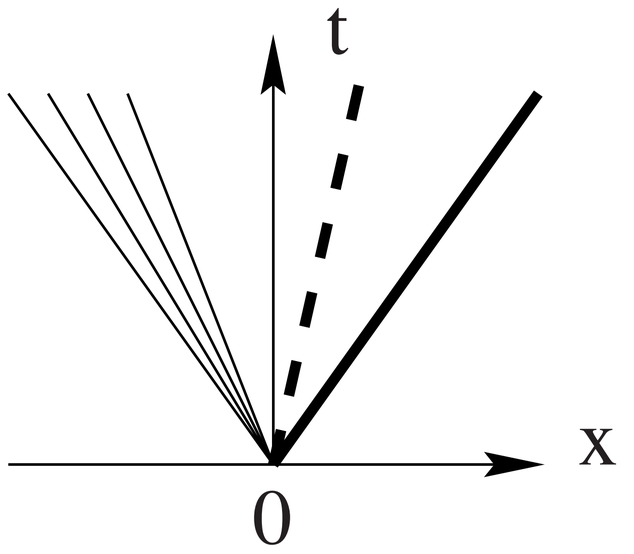
There is one big *physical* discontinuity in the initial conditions (we will see this gives rise to additional physical discontinuities at later times).

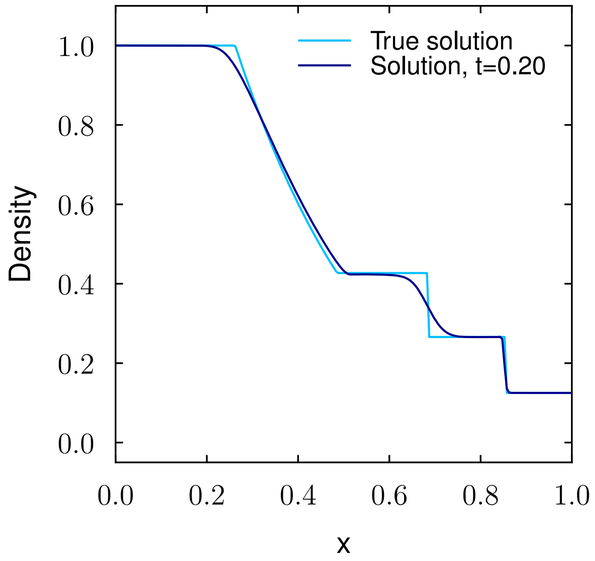
We’re going to model the evolution in time it with a finite volume method, which introduces lots of smaller, artificial Riemann problems for the fluxes across the boundary of every cell.

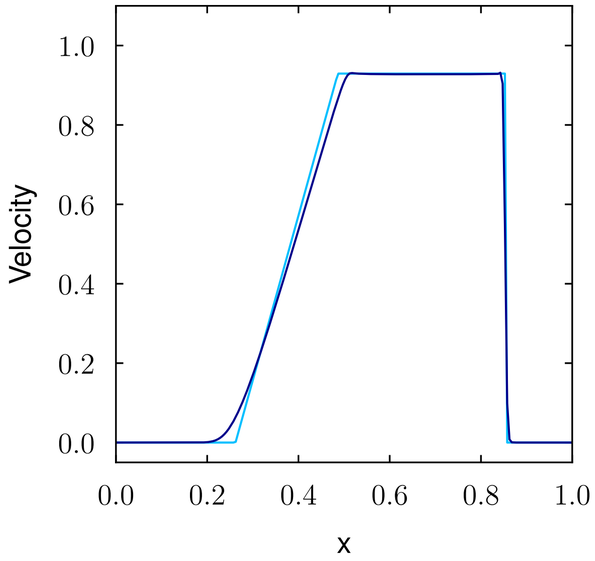
If our method is accurate enough, sharp discontinuities in the true solution should be preserved. No other features should appear due to our approximations in solving the Riemann problem between each cell.

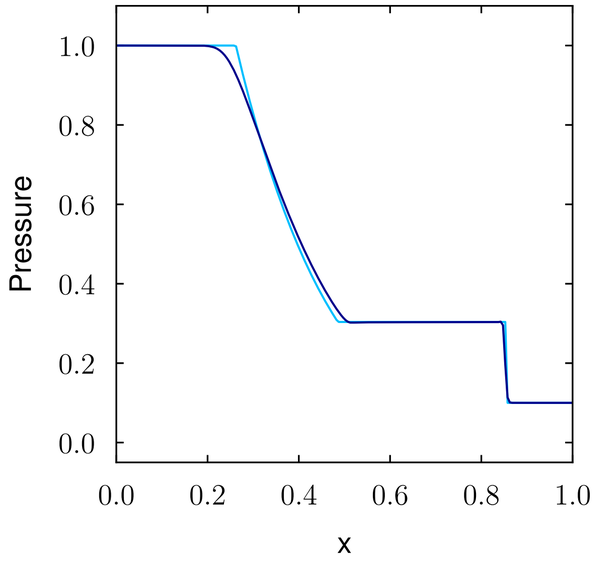
The shock tube

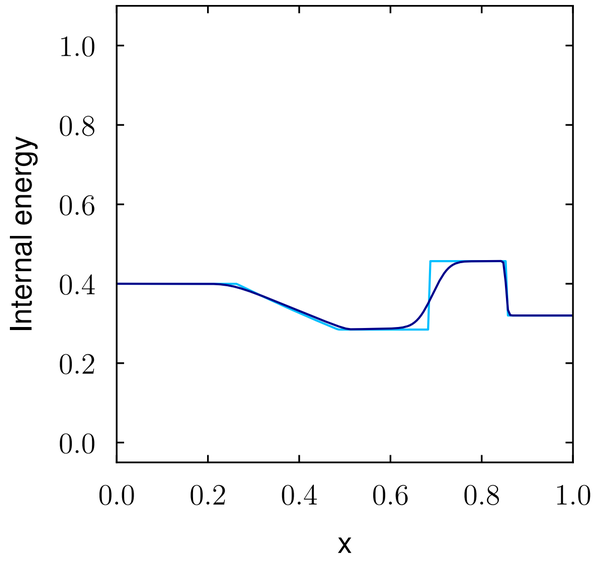








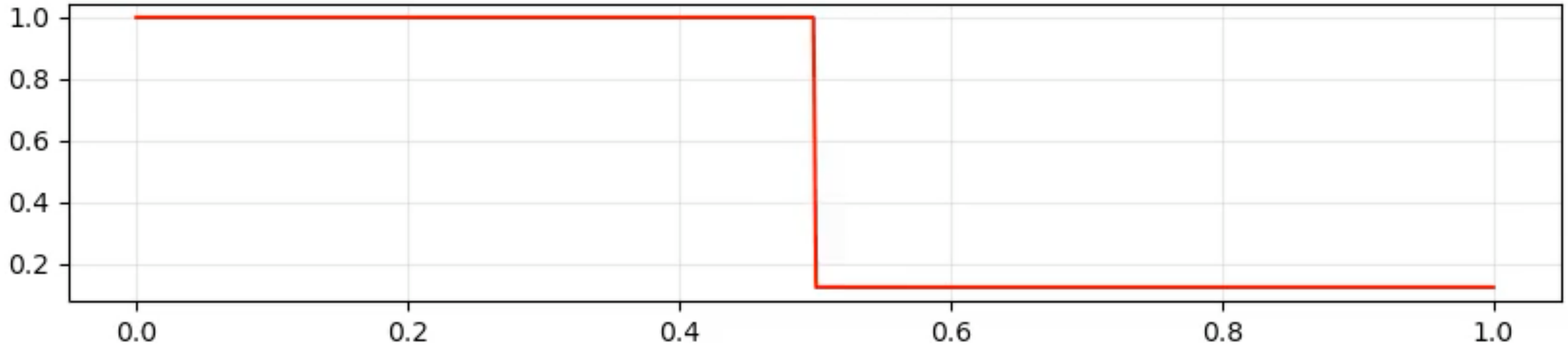




Figures: F. van den Bosch and B. Diemer

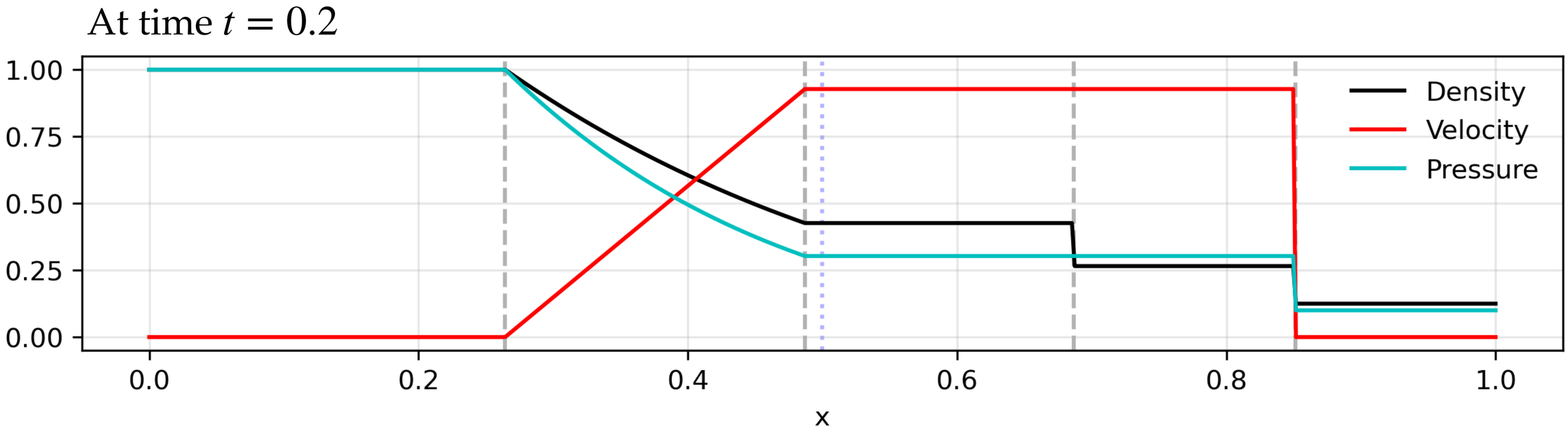
The shock tube: direct solution

$$\rho_{L,0} = 1, p_{L,0} = 1, v_{L,0} = 0 \qquad \rho_{R,0} = \frac{1}{8}, p_{R,0} = \frac{1}{10}, v_{R,0} = 0$$



The code for this direct solution is given on the class Github (`examples/finite_vol/shock_tube.ipynb`). It is not trivial: it involves a lot of algebra and a root-finding step (for which I use the numpy Brent solver).

The shock tube: direct solution



The shock tube: numerical solution

In the shock tube, there is an overall direction of travel, but we also have sound waves propagating in both directions.

Although we apply the same general principle as we did for the simple the advection case, we can't take as many shortcuts.

We need an algorithm for the Riemann problem that copes with the different wave speeds (and directions) in each cell.

A common choice is the HLL (Harten, Lax & van Leer) solver.

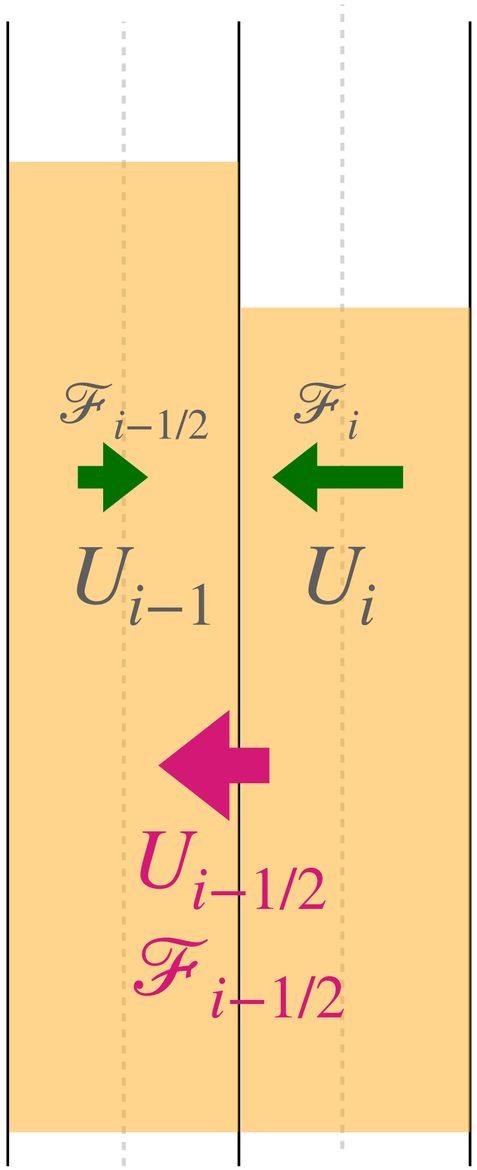
An important general point

Given the **conserved quantities** (density, momentum, energy) we can calculate the **instantaneous flux** at a point using the equations we saw earlier. In one dimension:

$$\overrightarrow{\mathcal{F}}(\overrightarrow{U}) = \begin{pmatrix} \rho v \\ \rho v^2 + p \\ (E + p)v \end{pmatrix}$$

Since we're going to track the conserved quantities, we will need to compute these fluxes via an intermediate calculation of the velocity and the pressure.

The next slide shows code that does this.



Cell midpoints (averages)

Boundary states (reconstructed)

```python
def point_fluxes(U):
    """
    U: [density, momentum, total energy]
    """
    # Bulk velocity = momentum/density
    v = U[1]/U[0]
    # Pressure = (gamma-1)*( E - 0.5 m*v**2)
    pressure = (gamma-1.0)*(U[2] - 0.5*U[0]*v**2)

    # Compute fluxes following the flux matrix for the Euler
    # equations in the class slides.

    fluxes = np.zeros_like(U)

    # Density flux is advected mass, m*v, i.e. the momentum.
    fluxes[0] = U[1] # i.e. v*U[0]
    # Momentum flux: advect momentum, add pressure
    fluxes[1] = v*U[1] + pressure
    # Energy flux: advect (E+P)
    fluxes[2] = v*(U[2] + pressure)

    return fluxes
```

HLL solver

For each boundary, calculate **two** instantaneous fluxes, one on either side of the boundary: $\mathscr{F}_R = \mathscr{F}(U_i^n)$ and $\mathscr{F}_L = \mathscr{F}(U_{i-1}^n)$.

Then consider the maximum left-going signal speed on the left side and maximum right-going signal speed on the right side : $S_L = v_{i-1}^n - c_{i-1}^n$ and $S_R = v_i^n - c_i^n$.

If all the waves are going one way, advect the conserved quantities in that direction: in cells with $S_R \leq 0$, $F_{i+1/2}^n = \mathscr{F}_R$ and in cells with $S_L \geq 0$, $F_{i+1/2}^n = \mathscr{F}_L$.

In general these extreme conditions won't be satisfied. In those cases, HLL finds an "averaged" flux:

$$F_{i+1/2}^n = \frac{S_L \mathscr{F}_L - S_R \mathscr{F}_R + S_L S_R (U_L - U_R)}{S_R - S_L}$$

Godunov scheme with HLL Solver

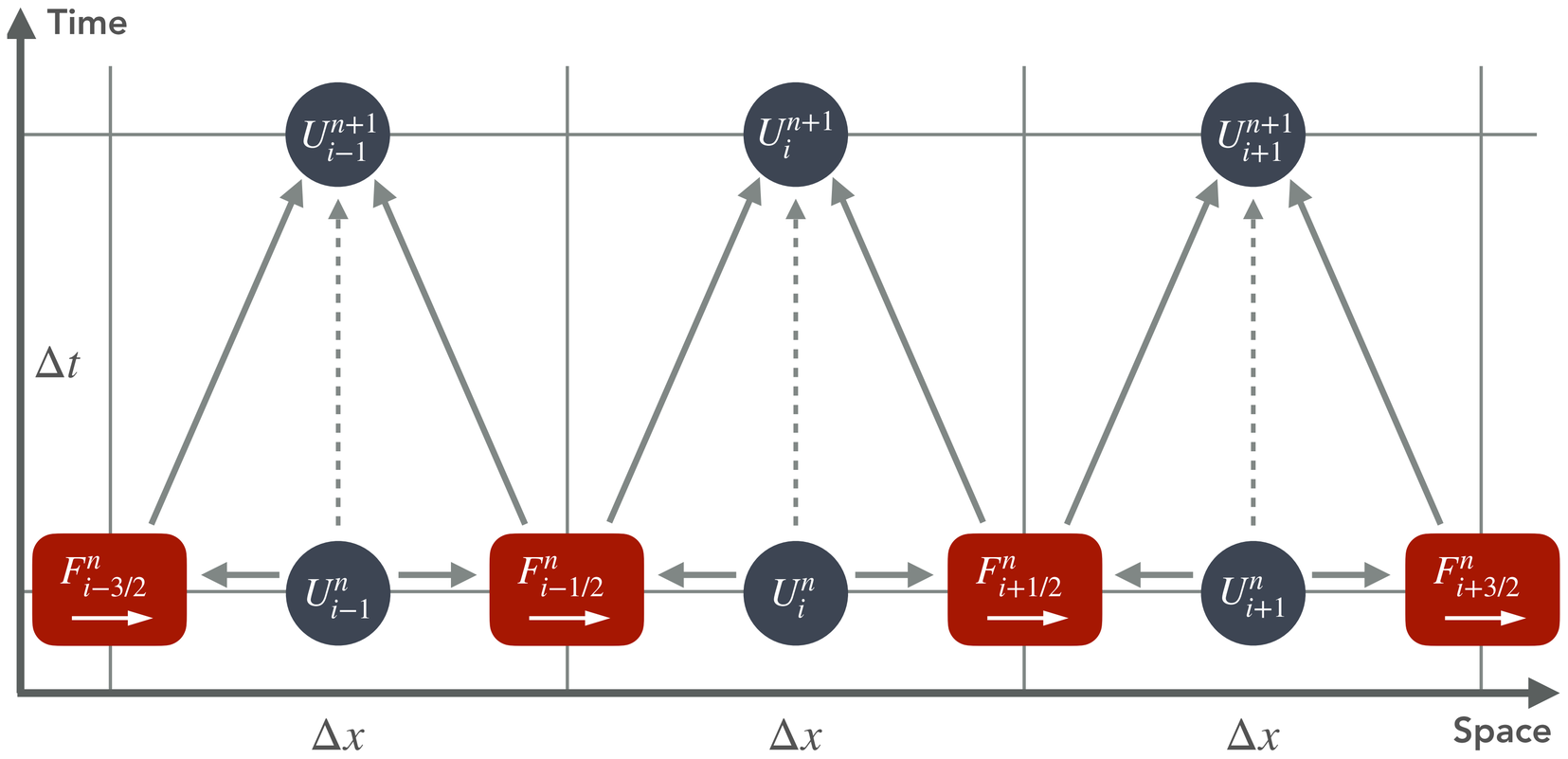


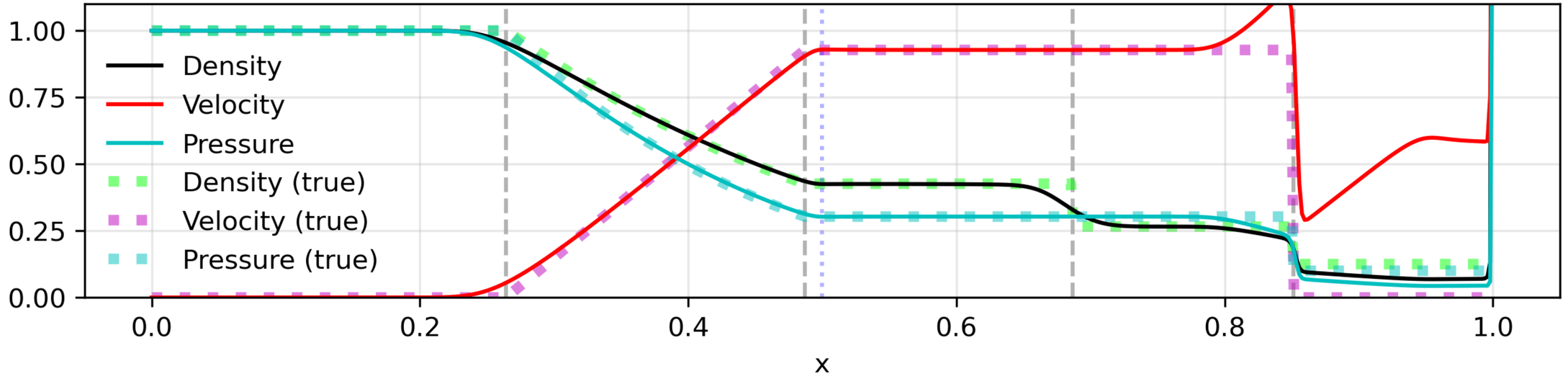
Figure: B. Diemer

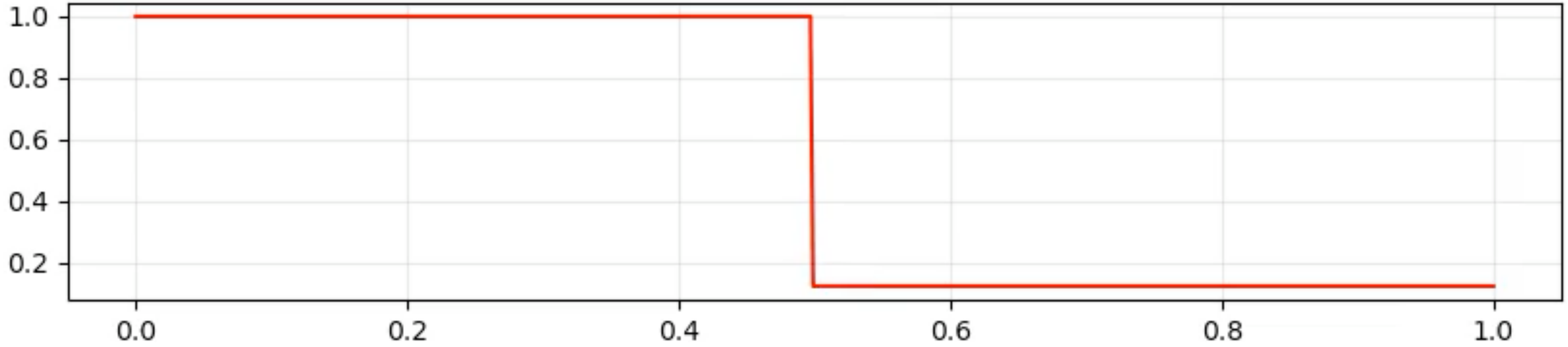
Godunov scheme with HLL Solver

```
# 1. Compute conserved quantities in each cell at previous timestep
U_old = np.array([density[n-1],
                  density[n-1]*velocity[n-1],
                  energy])

# 2. Calculate fluxes of conserved quantities
fluxes = calculate_fluxes_hll(U_old)

# 3. Update grid of conserved quantities using Euler step in time, with newly estimated fluxes.
U_new = np.zeros_like(U_old)
for i in range(1,ngrid-1):
    U_new[0,i]  = U_old[0,i] + (dt/dx)*(fluxes[0,i]-fluxes[0,i+1])
    U_new[1,i]  = U_old[1,i] + (dt/dx)*(fluxes[1,i]-fluxes[1,i+1])
    U_new[2,i]  = U_old[2,i] + (dt/dx)*(fluxes[2,i]-fluxes[2,i+1])

# 4. Recompute the primitive variables from the conserved quantities and save.
density[n,:]  = U_new[0,:]
velocity[n,:] = U_new[1,:]/U_new[0,:]
pressure[n,:] = (gamma-1.0)*(U_new[2,:] - 0.5*density[n,:]*velocity[n,:]**2)
```

An HLL solution to the shock tube

`examples/finite_vol/shock_tube.ipynb`

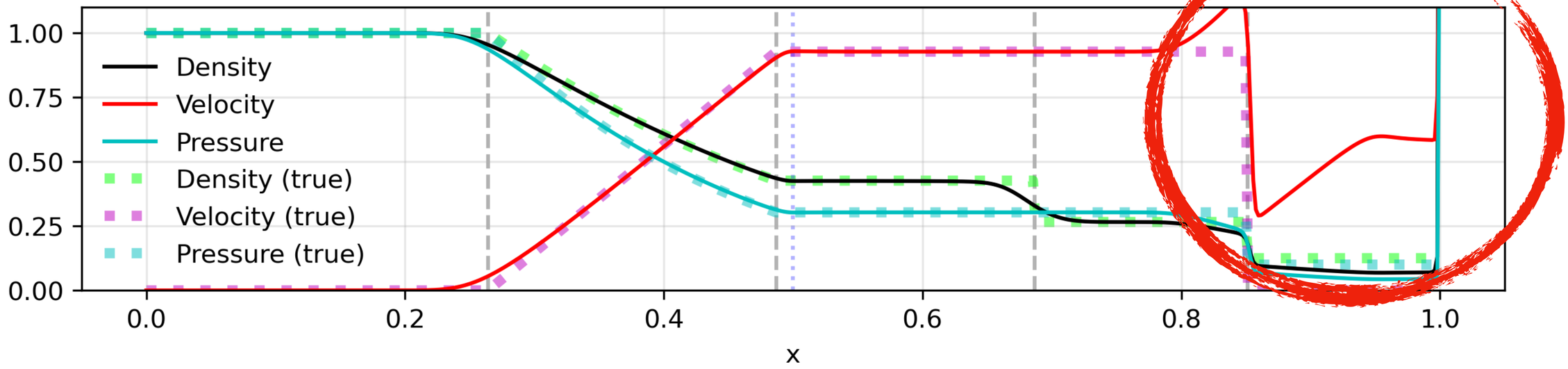


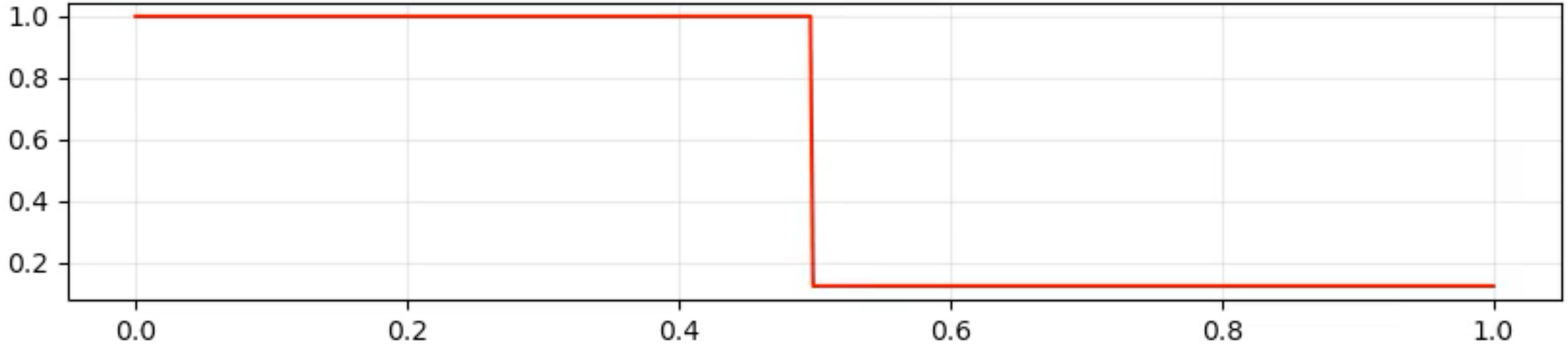


An HLL solution to the shock tube

BUG!

examples/finite_vol/shock_tube.ipynb

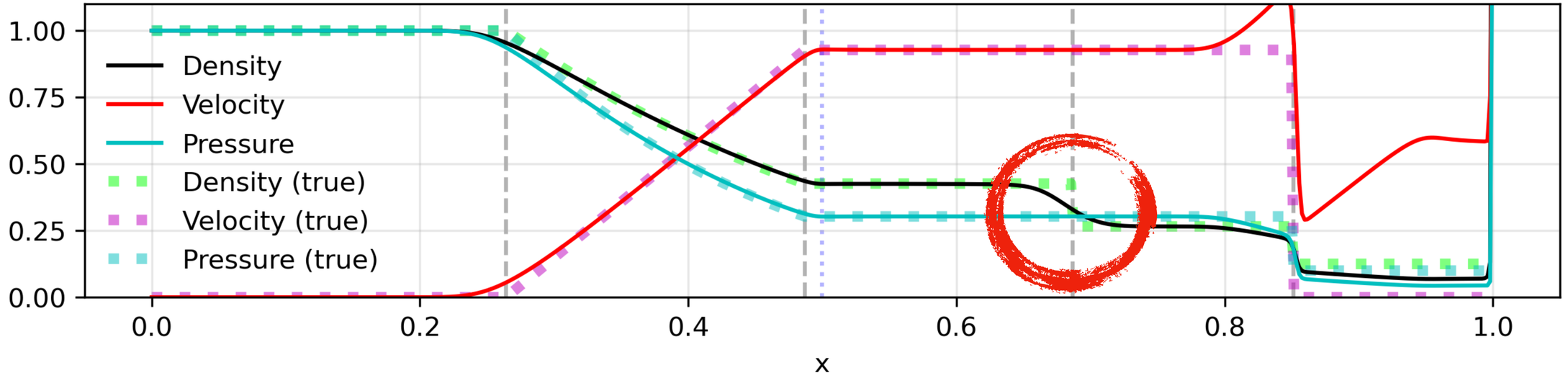


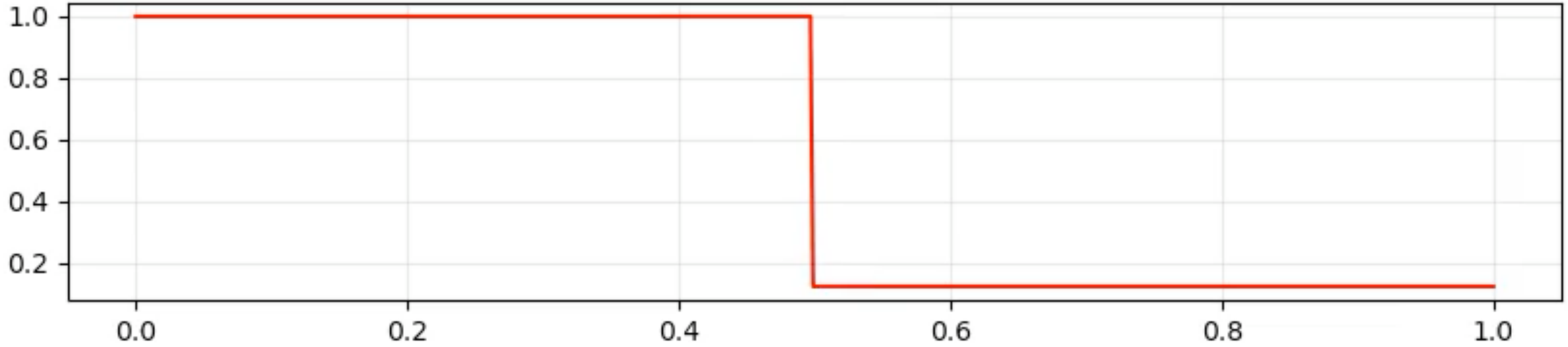


An HLL solution to the shock tube

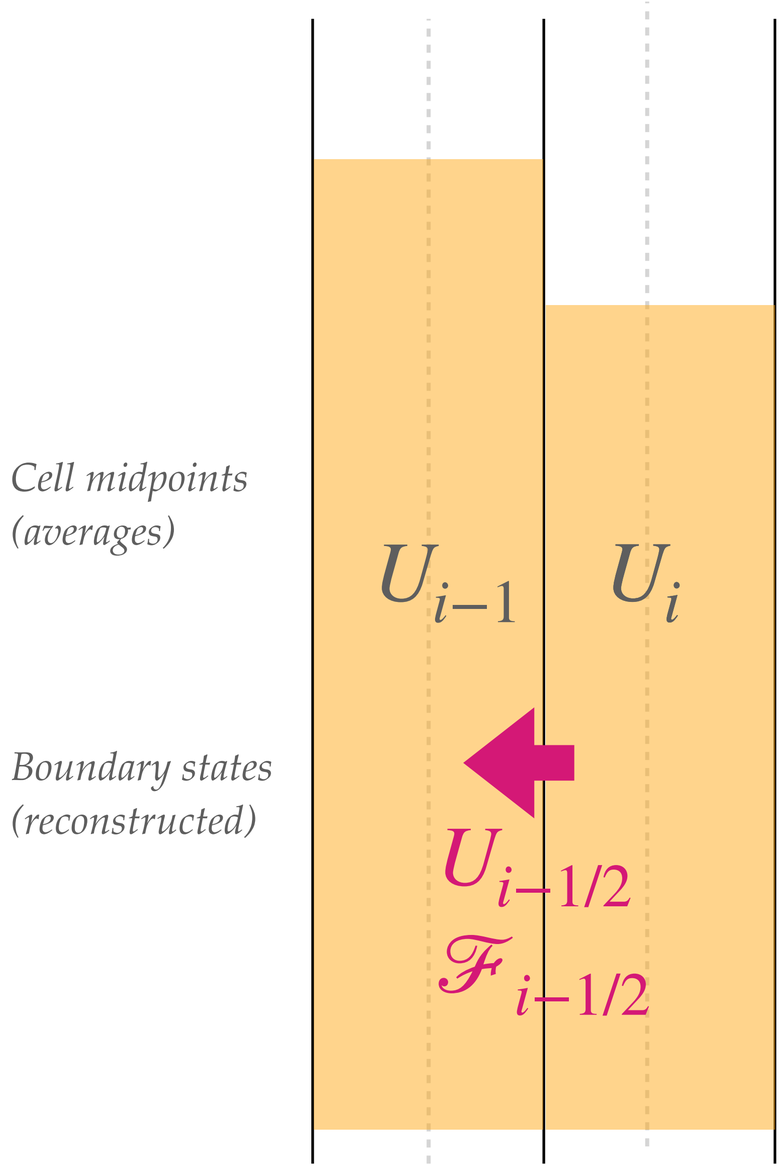
`examples/finite_vol/shock_tube.ipynb`

Smoothing over contact discontinuity





Other approaches: Lax-Wendroff



The HLL solver calculates a flux at the boundary by separately computing the fluxes "predicted" by the state on each side, and then doing some "averaging".

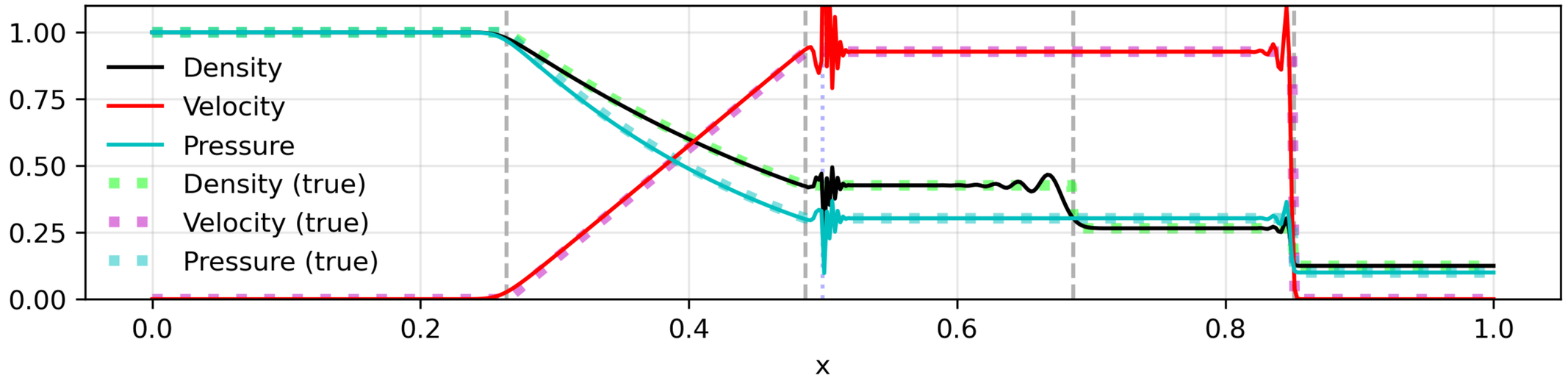
Instead, we could try first to **reconstruct** the state at the boundary, using the states on either side, and **then** compute a single flux from that.

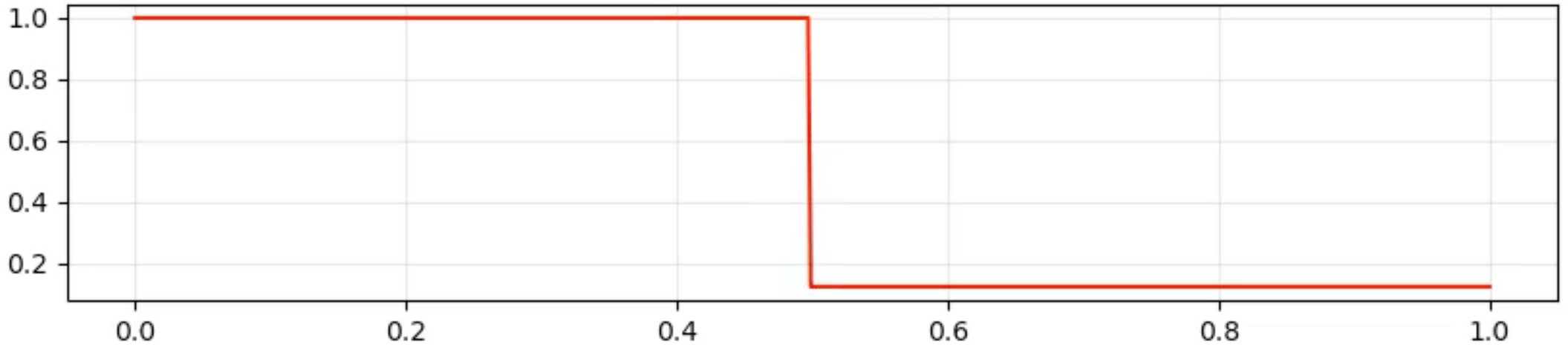
This is similar to centred finite-differencing. As we saw last week, that doesn't work well unless (a) the method accounts for artificial diffusion and (b) the method is more than first-order accurate in time.

$$U^{n+1/2}_{i-1/2} = \frac{1}{2}\left(U^n_{i-1} + U^n_i\right) - \frac{\Delta x}{2\Delta t}\left[\mathscr{F}(U^n_i) - \mathscr{F}(U^n_{i-1})\right];$$

$$\mathscr{F}^{n+1/2}_{i+1/2} = \mathscr{F}\left(U^{n+1/2}_{i+1/2}\right)$$

Lax-Wendroff for finite volumes





Reconstruction

HLL “reconstructs” the boundary states by assuming the distribution of the fluid is “piecewise constant”. In principle, we can do better. For example, we can make the distribution piecewise linear. The boundaries will still be discontinuous.

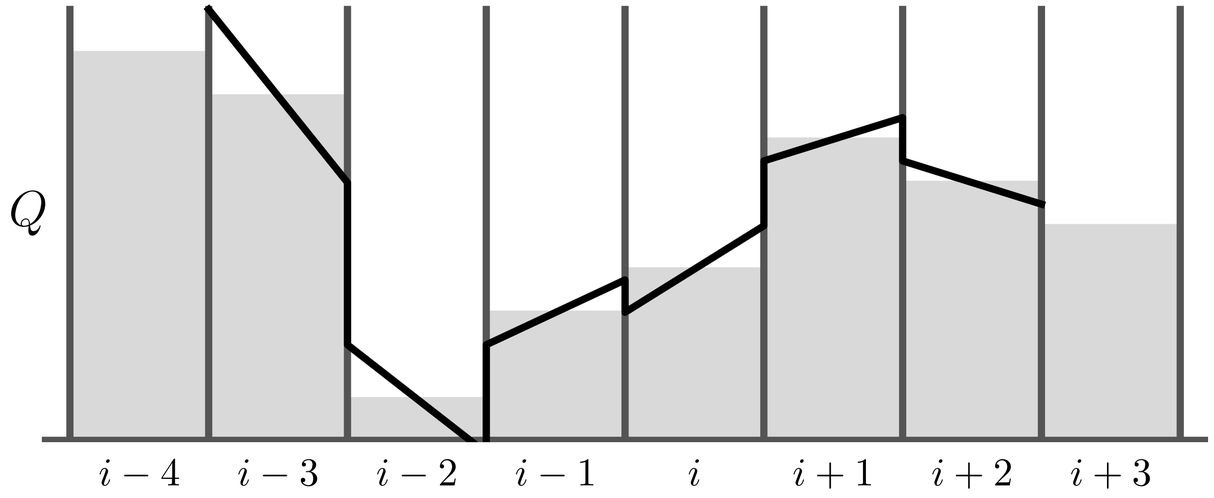


Figure adapted from B. Diemer

Reconstruction

HLL “reconstructs” the boundary states by assuming the distribution of the fluid is “piecewise constant”. In principle, we can do better. For example, we can make the distribution piecewise linear. The boundaries will still be discontinuous.

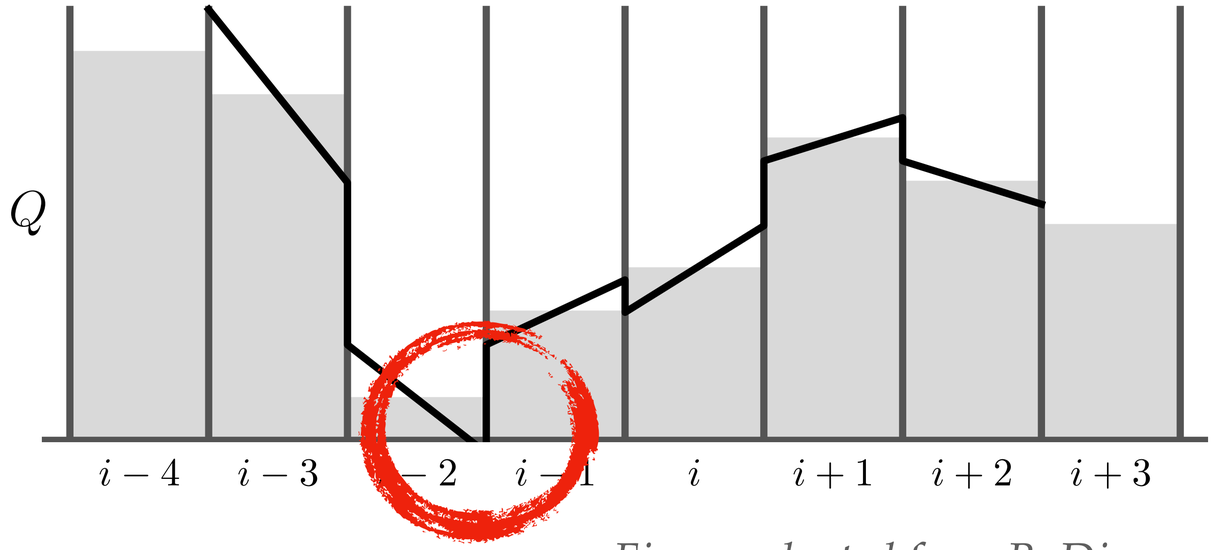


Figure adapted from B. Diemer

Reconstruction and slope-limiting

The risk of unphysical slopes can be mitigated to some extent by imposing various conditions (such as not being too steep).

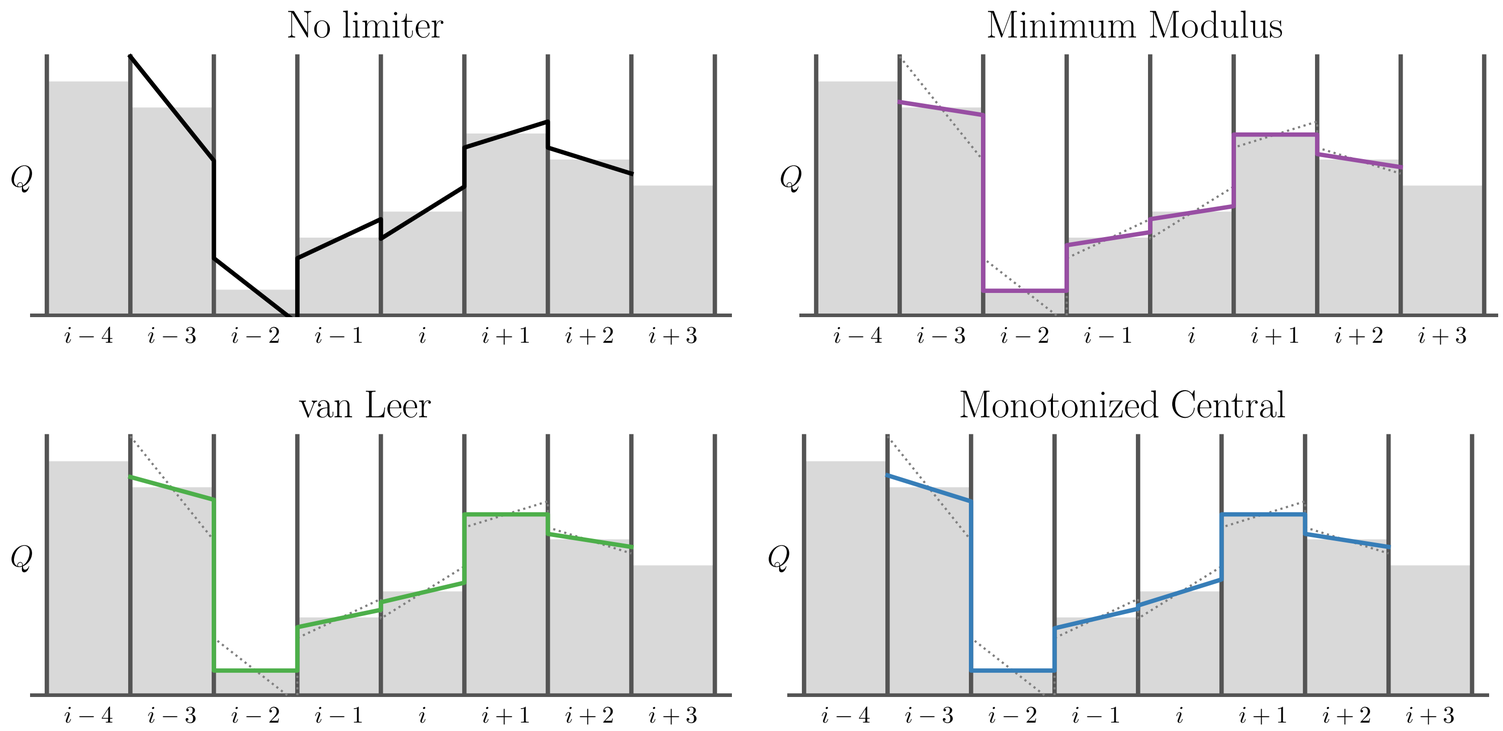


Figure adapted from B. Diemer

MUSCL-Hancock

Clearly, there are a lot of design choices involved in building a finite volume fluid solver!

This is not so surprising — even simple fluid problems are quite complicated, and the field is relatively young.

A "minimum practical" solution combines **(1)** reconstruction of the boundary states, **(2)** time-averaging by advancing these states to the midpoint of the timestep, and finally **(3)** a Riemann solver to compute the fluxes.

Such a scheme can be at least 2nd-order accurate in space and time, while avoiding the artefacts of a Lax-Wendroff-like approach (by treating the boundaries as Riemann problems).

A common implementation is the **MUSCL-Hancock** scheme (Monotonic Upstream Centred Conservation Law; Hancock refers to the timestep part).

MUSCL-Hancock

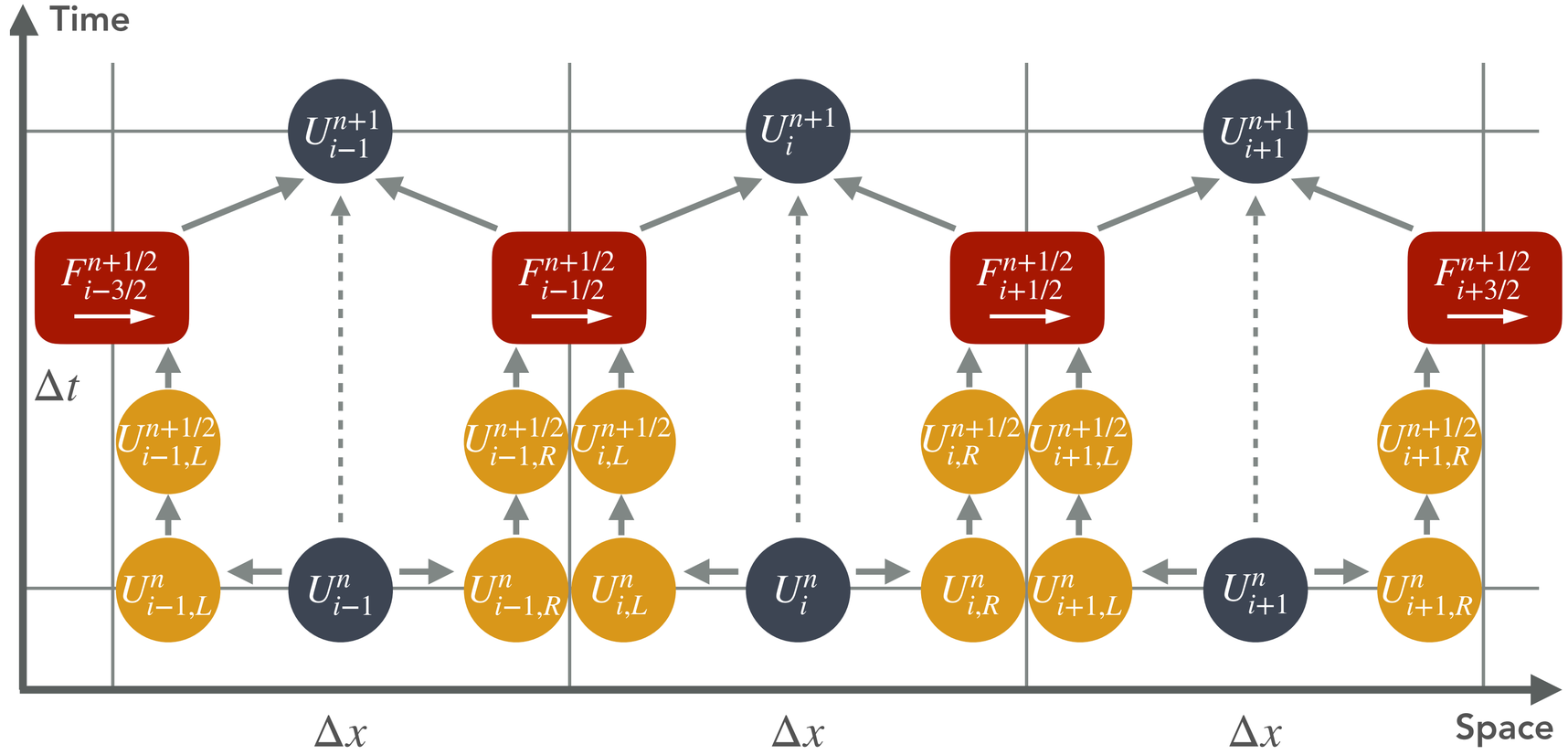


Figure: B. Diemer

Challenges

It is pretty hard to get a good feeling for these topics without trying to code them yourself!

1. Fix the boundary condition bug in my implementation of the Godunov scheme with the HLL solver.

2. Using Ch. 9 of Diemer's notes for reference, extend the example notebook to implement the MUSCL-Hancock scheme for the shock tube problem.

3. Write a neater, self-contained code that solves general 1-d initial conditions.

Probably the simplest astrophysical interesting problem to implement with a self-written code is the Sedov blast wave (again, see Diemer and Zigale's notes).

Sources for today's slides

These slides build on and adapt previous lecture notes for this course, by Kuo-Chuan Pan, Karen Yang, and Hsi-Yu Schive (NTU), and the online notes by Diemer and Zigale (see links on wiki).

Frank van den Bosch's fluid dynamics lecture notes have a good computational section (which Diemer also relies on). See wiki for link.

Hsi-Yu Schive has a nice clean example of finite volume Lax-Wendroff, which was very useful:

https://gist.github.com/hyschive/46bab6434f1b9b9aee23aeaeb71b90b6