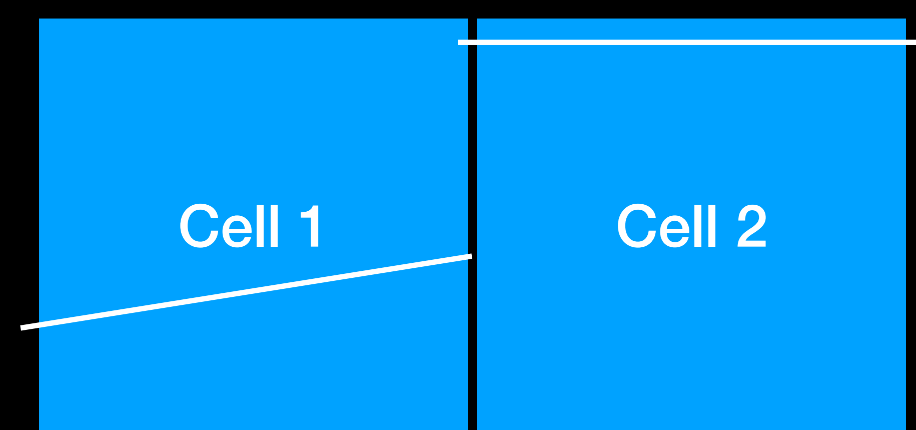
Shock-capturing numerical methods

Overview

- Many numerical methods are unstable / lead to large oscillations in solution when shocks are present

- Shock capturing methods are built to take into account the presence of a discontinuity in the solution in between grid cells

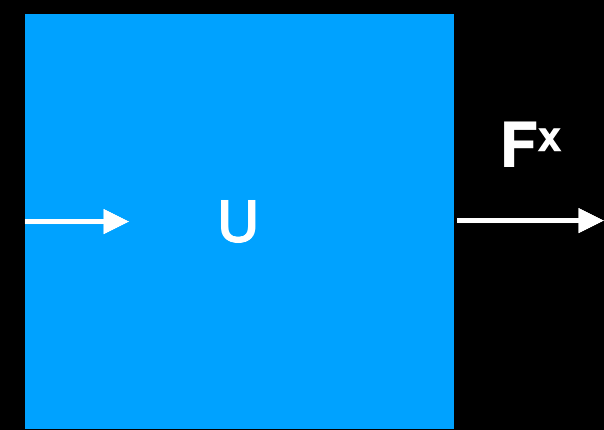


- Solution at cell interface is assumed to be different on the “left” and “right” side of the face

- These two states have to satisfy appropriate “jump” condition (e.g. Rankine-Hugoniot)

Methods

$$\partial_t U + \partial_i F^i(U) = S(U)$$



Compute time derivative of U at cell center from <u>S at cell center</u> and <u>F on cell faces:</u>

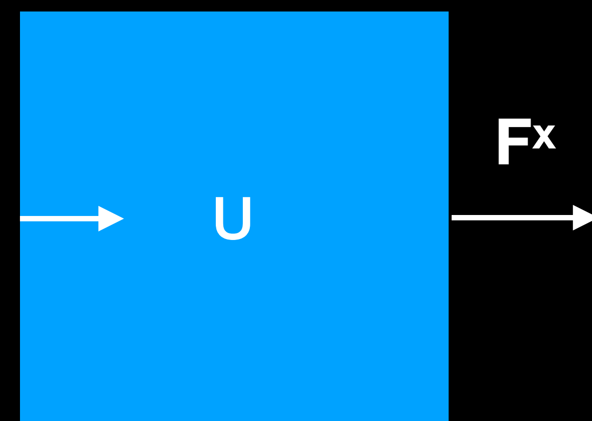
$$\partial_t U(x) = -\frac{F(x+\Delta x/2) - F(x-\Delta x/2)}{\Delta x} + S(x)$$

- “Conservative” scheme: fluxes are estimated on cell faces, and we conserve (up to source terms) :

$$U_{\rm tot} = \int dV U$$

Methods

$$\partial_t U + \partial_i F^i(U) = S(U)$$



Compute time derivative of U at cell center from S at cell center and F on cell faces:

$$\partial_t U(x) = -\frac{F(x+\Delta x/2) - F(x-\Delta x/2)}{\Delta x} + S(x)$$

- Fluxes on faces are computed from the “right” and “left” states $F_{L,R}$, so that if $F_L != F_R$, the flux satisfy the jump conditions = solving the Riemann problem

$$F^i = f(F^i_R, F^i_L)$$

Algorithm

- To compute the time derivative $d_t U$:
  - Compute “Left” and “Right” value of the evolved variables and fluxes **at cell faces**
  - Compute F from $F_{R,L}$ using a **Riemann solver**
  - Compute the divergence $d_i F^i$ from the fluxes at cell faces
  - Compute the source terms at cell centers
- To evolve, we also need to **discretize in time** the equations

Reconstruction

- Simplest method (not very accurate):

$$F_L(x + \Delta x/2) = F(x) \qquad F_R(x + \Delta x/2) = F(x + \Delta x)$$

- Slightly better : minmod reconstruction

$$F_L(x + \Delta x/2) = F(x) + s(x)/2$$
$$F_R(x - \Delta x/2) = F(x) - s(x)/2$$

$$s(x) = \mathrm{minmod}[\mathrm{F(x - \Delta x), F(x), F(x + \Delta x)}]$$

$$\boxed{\mathrm{minmod[a, b, c]} = \begin{cases} 0 & (c-b)(b-a) \leq 0 \\ \mathrm{sign[b-a] \times min[(b-a), (c-b)]} & \text{otherwise} \end{cases}}$$

- Alternative: **reconstruct U on faces, then compute F(U)** [use this for our test problem]

Riemann solver

- Provides F at cell faces from $F_{L,R}$ and $U_{L,R}$ on the same face
- Simplest: Local Lax Friedrich method

$$F = \frac{F_L + F_R}{2} - \frac{c}{2}(U_R - U_L)$$

where c is the largest *characteristic speed* (in absolute value)
of the sytem of equations on the face where we are computing F.
So, if $c_{i,R}$ and $c_{i,L}$ are the characteristic speeds computed from $U_{R,L}$,

$$c = \max\left(|c_{i,L}|, |c_{i,R}|\right)$$

- Note that to compute F, we need $U_{R,L}$ and $F_{R,L}$ on the face
- The second term adds *dissipation* at shocks: if $U_R > U_L$, F becomes more negative (i.e. the flux of U goes from right to left)

Time stepping

- A simple 2nd order method to take a time step dt is the 2nd order Runge-Kutta algorithm :

(1) Take a half-step to estimate U(t+dt/2)

$$U(t+\frac{dt}{2}) = U(t) + \frac{dt}{2}\partial_t U(t)$$

(2) Take a full step, with the time derivative computed using U(t+dt/2)

$$U(t+dt) = U(t) + dt\partial_t U(t+\frac{dt}{2})$$

Example

- Try to implement the following equation in 1D (use e.g. 100 grid cells in [-1,1] with periodic boundaries):

$$\partial_t y + \partial_x (\frac{y^2}{2}) = 0$$

$$y(x, t = 0) = \exp\left(-16x^2\right)$$

- The (only) characteristic speed of this equation is c=y

- You may also try this problem *without* using a Riemann solver, to see the importance of using shock-capturing methods