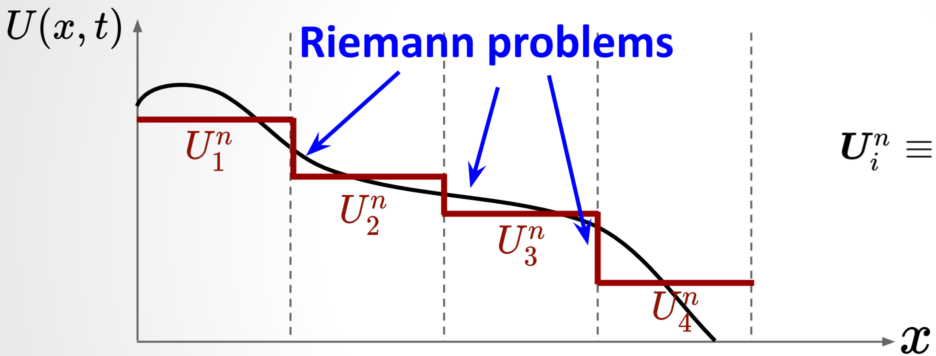
Hydrodynamics-II

Hsi-Yu Schive (薛熙于)
National Taiwan University

High-Resolution Shock-Capturing Methods

- Godunov method
  - Approximate data with a <u>piecewise constant</u> distribution



$$\boldsymbol{U}_i^n \equiv \frac{1}{\Delta x}\int_{x_{i-1/2}}^{x_{i+1/2}} U(x,t^n)dx$$

  - Solve the local Riemann problems
    - Piecewise constant data with a single discontinuity
    - Apply either exact or approximate solutions
  - Update data by averaging the Riemann problem solution over each cell
    - Equivalently, we can solve the intercell fluxes
    - Avoid wave interaction within each cell

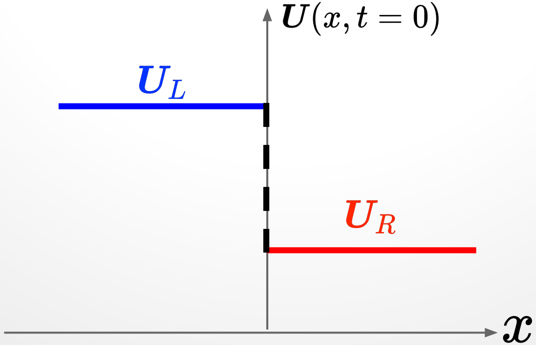
Riemann Problem in 1D Hydro

- **Euler eqs. in 1D:** $$\frac{\partial \boldsymbol{U}}{\partial t} + \frac{\partial \boldsymbol{F}_x(\boldsymbol{U})}{\partial x} = 0,\ \boldsymbol{U} = \begin{bmatrix} \rho \\ \rho v_x \\ E \end{bmatrix},\ \boldsymbol{F}_x = \begin{bmatrix} \rho v_x \\ \rho v_x^2 + P \\ (E+P)v_x \end{bmatrix}$$

- **Riemann problem:** $$\boxed{\boldsymbol{U}(x, t=0) = \begin{cases} \boldsymbol{U}_L = \begin{bmatrix} \rho_L \\ \rho_L v_{xL} \\ E_L \end{bmatrix}, & x \le 0 \\ \boldsymbol{U}_R = \begin{bmatrix} \rho_R \\ \rho_R v_{xR} \\ E_R \end{bmatrix}, & x > 0 \end{cases}}$$

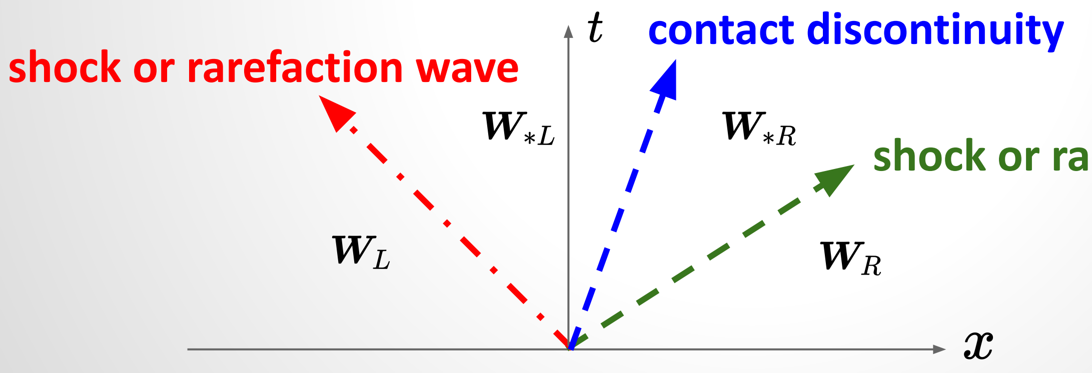
left state

right state



Riemann Problem in 1D Hydro

- Exact solution of the Riemann problem involves three waves
  - Contact discontinuity
  - Shock wave
  - Rarefaction wave
- Decompose the entire domain into four regions W_L, W_{*L}, W_{*R}, W_R

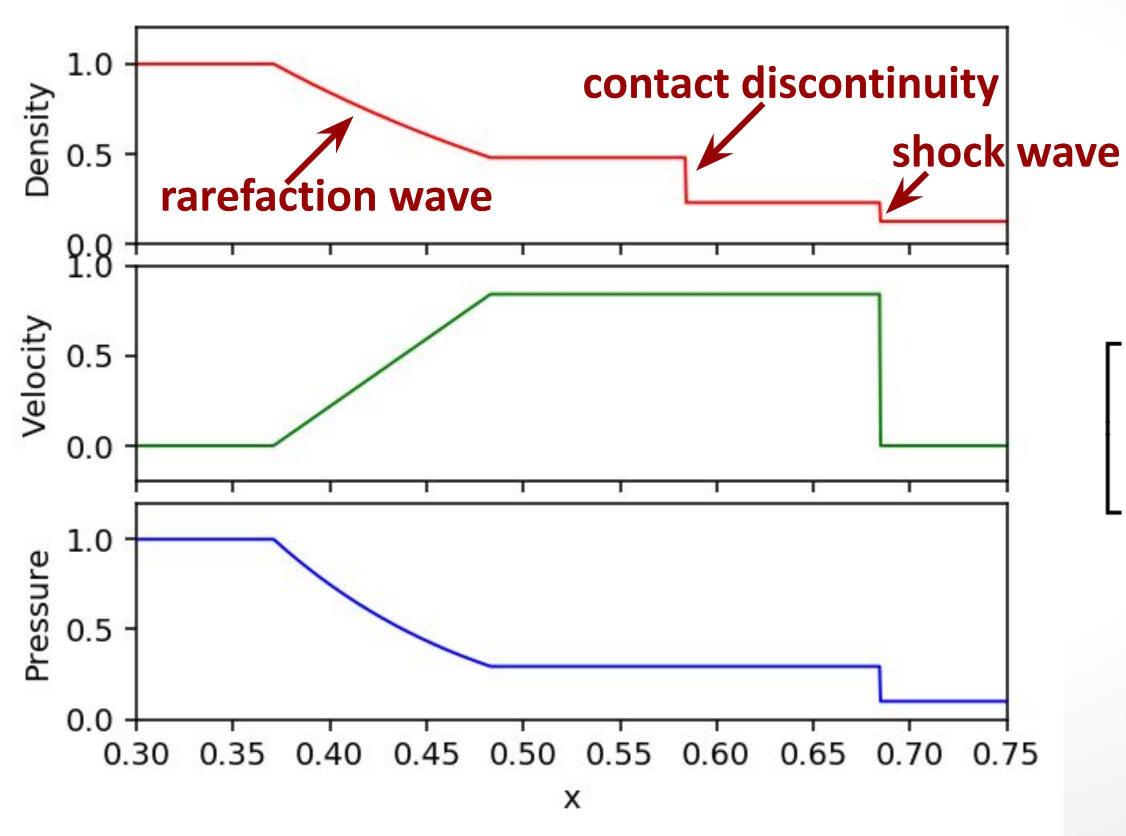


Riemann Problem in 1D Hydro

- **Riemann problem can be solved analytically**
  - **Known:** W_L, W_R
  - **Unknowns:** W_{*L}, W_{*R}
    - **In fact, we always have** $P_{*L} = P_{*R}$ and $v_{x,*L} = v_{x,*R}$ **(because the middle wave is always a contact discontinuity)**
    - **So only 4 unknown variables:** $\rho_{*L}, \rho_{*R}, P_*, v_{x*}$

- **However, exact Riemann solver is very computationally expensive**
  - **Approximate Riemann solvers are usually accurate enough**
    - **All we need is the interface fluxes**
    - **Examples**
      - **<u>Roe solver</u>**
      - **HLLE solver**
      - **HLLC solver**

Example: Sod Shock Tube Problem



Initial condition

Left state **Right state**

$$\begin{bmatrix} \rho_L \\ v_L \\ P_L \end{bmatrix} = \begin{bmatrix} 1.0 \\ 0.0 \\ 0.125 \end{bmatrix}, \begin{bmatrix} \rho_R \\ v_R \\ P_R \end{bmatrix} = \begin{bmatrix} 0.125 \\ 0.0 \\ 0.1 \end{bmatrix}$$

Diagonalization of a 1D Linear System

$$\frac{\partial \boldsymbol{U}}{\partial t} + \boldsymbol{A}\frac{\partial \boldsymbol{U}}{\partial x} = 0,$$ where A is a constant

Diagonalize A: $\boldsymbol{\lambda} = \boldsymbol{K}^{-1}\boldsymbol{A}\boldsymbol{K}$

$$\boldsymbol{\lambda} = \begin{bmatrix} \lambda_1 & 0 & 0 & \dots & 0 \\ 0 & \lambda_2 & 0 & \dots & 0 \\ 0 & 0 & \lambda_3 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & \lambda_N \end{bmatrix}, \quad \boldsymbol{A}\boldsymbol{K} = \boldsymbol{K}\boldsymbol{\lambda}, \quad \boldsymbol{K} = \left[\boldsymbol{K}^{(1)}, \dots, \boldsymbol{K}^{(N)}\right]$$

right eigenvectors (constant)

eigenvalues (constant)

Diagonalization of a 1D Linear System

Introduce the characteristic variables: $\boldsymbol{C} \equiv \boldsymbol{K}^{-1}\boldsymbol{U}$

$$\boldsymbol{K}\frac{\partial \boldsymbol{C}}{\partial t} + \boldsymbol{A}\boldsymbol{K}\frac{\partial \boldsymbol{C}}{\partial x} = 0 \rightarrow \frac{\partial \boldsymbol{C}}{\partial t} + \boldsymbol{\lambda}\frac{\partial \boldsymbol{C}}{\partial x} = 0$$

$$\rightarrow \boxed{\frac{\partial C_i}{\partial t} + \lambda_i \frac{\partial C_i}{\partial x} = 0,\ i = 1, \ldots, N}$$

N decoupled linear advection equations with characteristic speed λ_i

$$\boldsymbol{U}(x,t) = \boldsymbol{K}\boldsymbol{C} = \sum_{i=1}^{N} C_i(x,t)\boldsymbol{K}^{(i)}$$

$C_i(x,t)$ is the coefficient in the eigenvector expansion of $U(x,t)$

Riemann Problem for Linearized Hydro

$$\begin{cases} \dfrac{\partial \rho}{\partial t} + \rho_0 \dfrac{\partial v_x}{\partial x} = 0 \\ \dfrac{\partial v_x}{\partial t} + \dfrac{C_s^2}{\rho_0}\dfrac{\partial \rho}{\partial x} = 0 \end{cases}, \begin{bmatrix} \rho \\ v_x \end{bmatrix} = \begin{bmatrix} \rho_L \\ v_{xL} \end{bmatrix} \text{ for } x \le 0,\ \begin{bmatrix} \rho \\ v_x \end{bmatrix} = \begin{bmatrix} \rho_R \\ v_{xR} \end{bmatrix} \text{ for } x > 0$$

$$\Rightarrow \quad \frac{\partial \boldsymbol{U}}{\partial t} + \boldsymbol{A}\frac{\partial \boldsymbol{U}}{\partial x} = 0,\ \boldsymbol{U} = \begin{bmatrix} \rho \\ v_x \end{bmatrix},\ \boldsymbol{A} = \begin{bmatrix} 0 & \rho_0 \\ C_s^2/\rho_0 & 0 \end{bmatrix}$$

$$\Rightarrow \quad \boldsymbol{\lambda} = \begin{bmatrix} -C_s & 0 \\ 0 & C_s \end{bmatrix},\ \boldsymbol{K} = \begin{bmatrix} \rho_0 & \rho_0 \\ -C_s & C_s \end{bmatrix},\ \boldsymbol{K}^{-1} = \frac{1}{2C_s\rho_0}\begin{bmatrix} C_s & -\rho_0 \\ C_s & \rho_0 \end{bmatrix}$$

Riemann Problem for Linearized Hydro

left-moving component with $-C_s$

$$\boldsymbol{C} = \boldsymbol{K}^{-1}\boldsymbol{U} = \frac{1}{2C_s\rho_0}\begin{bmatrix} C_s\rho - \rho_0 v_x \\ C_s\rho + \rho_0 v_x \end{bmatrix} \equiv \begin{bmatrix} C_1 \\ C_2 \end{bmatrix}$$

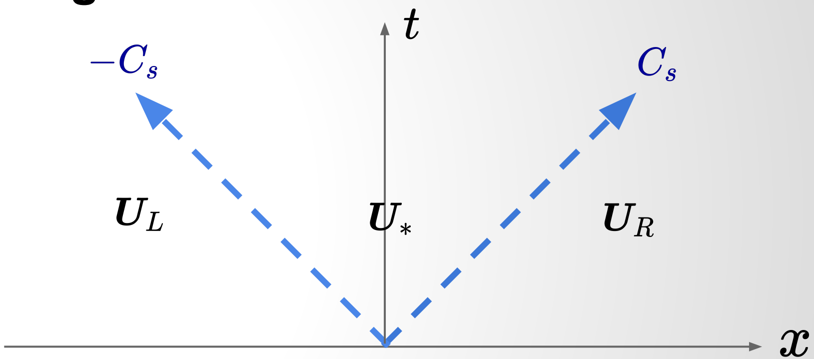
right-moving component with C_s

$$\begin{cases} \boldsymbol{U}_L = \begin{bmatrix} \rho_L \\ v_{xL} \end{bmatrix} = C_{1,L}\boldsymbol{K}^{(1)} + \underline{C_{2,L}\boldsymbol{K}^{(2)}} \\ \boldsymbol{U}_R = \begin{bmatrix} \rho_R \\ v_{xR} \end{bmatrix} = \underline{C_{1,R}\boldsymbol{K}^{(1)}} + C_{2,R}\boldsymbol{K}^{(2)} \end{cases}$$

Riemann Problem for Linearized Hydro

- The two characteristic waves (propagating with speeds $\pm C_s$) decompose the solution into three regions

$$\boldsymbol{U}(x,t) = \begin{cases} \boldsymbol{U}_L, & x < -C_s t \\ \boldsymbol{U}_R, & x > C_s t \\ \boldsymbol{U}_*, & |x| < C_s t \end{cases}$$



$$\boldsymbol{U}_* = C_{1,R}\boldsymbol{K}^{(1)} + C_{2,L}\boldsymbol{K}^{(2)}$$
$$= \begin{bmatrix} \frac{1}{2}(\rho_L + \rho_R) - \frac{\rho_0}{2C_s}(v_{xR} - v_{xL}) \\ \frac{1}{2}(v_{xL} + v_{xR}) - \frac{C_s}{2\rho_0}(\rho_R - \rho_L) \end{bmatrix}$$

solution in the star region in between two characteristic waves

Roe's Riemann Solver

- *Roe, P. L., 1981. JCP, 43, 357*
- **Rewrite the 1D Euler eqs. into a matrix form**

$$\frac{\partial \boldsymbol{U}}{\partial t} + \frac{\partial \boldsymbol{F}_x(\boldsymbol{U})}{\partial x} = 0, \ \boldsymbol{A}(\boldsymbol{U}) \equiv \frac{\partial \boldsymbol{F}_x}{\partial \boldsymbol{U}}$$

$$\rightarrow \frac{\partial \boldsymbol{U}}{\partial t} + \boldsymbol{A}(\boldsymbol{U}) \frac{\partial \boldsymbol{U}}{\partial x} = 0$$

$$\boldsymbol{U} = \begin{bmatrix} \rho \\ \rho v_x \\ \rho v_y \\ \rho v_z \\ E \end{bmatrix}, \quad \boldsymbol{A}(\boldsymbol{U}) = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ -v_x^2 + \hat{\gamma} v^2/2 & (2-\hat{\gamma}) v_x & -\hat{\gamma} v_y & -\hat{\gamma} v_z & \hat{\gamma} \\ -v_x v_y & v_y & v_x & 0 & 0 \\ -v_x v_z & v_z & 0 & v_x & 0 \\ -v_x H + \hat{\gamma} v_x v^2/2 & H - \hat{\gamma} v_x^2 & -\hat{\gamma} v_x v_y & \hat{\gamma} v_x v_z & (\hat{\gamma}+1) v_x \end{bmatrix}$$

Roe’s Riemann Solver

- **Diagonalize** *A(U)*

$$\boldsymbol{\lambda} = [v_x - C_s, v_x, v_x, v_x, v_x + C_s]$$

$$\boldsymbol{K} = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 \\ v_x - C_s & 0 & 0 & v_x & v_x + C_s \\ v_y & 1 & 0 & v_y & v_y \\ v_z & 0 & 1 & v_z & v_z \\ H - v_x C_s & v_y & v_z & v^2/2 & H + v_x C_s \end{bmatrix}$$

$$H = (E+P)/\rho,\ \hat{\gamma} = \gamma - 1,\ v^2 = v_x^2 + v_y^2 + v_z^2,\ C_s^2 = \gamma P/\rho$$

Roe's Riemann Solver

- Approximate $A(U)$ by a **constant** Jacobian matrix $A(\bar{U})$, where $\bar{U} = \bar{U}(\boldsymbol{U}_L, \boldsymbol{U}_R)$ is a constant mean state between the left and right states

$$\frac{\partial \boldsymbol{U}}{\partial t} + \tilde{\boldsymbol{A}}(\boldsymbol{U}_L, \boldsymbol{U}_R)\frac{\partial \boldsymbol{U}}{\partial x} = 0 \leftarrow \text{approximate (linearized) equations}$$

- Finding an appropriate form of $\bar{U}(\boldsymbol{U}_L, \boldsymbol{U}_R)$ is non-trivial
  - Roe proposed the following linearization

$$\begin{cases} \bar{\rho} = \sqrt{\rho_L}\sqrt{\rho_R} \\ \bar{\boldsymbol{v}} = \dfrac{\sqrt{\rho_L}\boldsymbol{v}_L + \sqrt{\rho_R}\boldsymbol{v}_R}{\sqrt{\rho_L} + \sqrt{\rho_R}} \\ \bar{H} = \dfrac{\sqrt{\rho_L}H_L + \sqrt{\rho_R}H_R}{\sqrt{\rho_L} + \sqrt{\rho_R}} \end{cases}$$

- Ensure conservation: $\boldsymbol{F}(\boldsymbol{U}_R) - \boldsymbol{F}(\boldsymbol{U}_L) = \tilde{\boldsymbol{A}} \cdot (\boldsymbol{U}_R - \boldsymbol{U}_L)$
- Next, compute the averaged eigenvalues $\bar{\lambda}$ and the averaged right eigenvectors $\bar{\boldsymbol{K}}$

Roe's Riemann Solver

- **Compute the Roe averaged fluxes F_{Roe}**

$$\bar{\boldsymbol{C}} = \bar{\boldsymbol{K}}^{-1}(\boldsymbol{U}_R - \boldsymbol{U}_L)$$

$$\begin{aligned}\boldsymbol{F}_{\text{Roe}} &= \boldsymbol{F}_L + \sum_{\bar{\lambda}_i \le 0} \bar{C}_i \bar{\lambda}_i \bar{K}^{(i)} \\ &= \boldsymbol{F}_R - \sum_{\bar{\lambda}_i \ge 0} \bar{C}_i \bar{\lambda}_i \bar{K}^{(i)} \\ &= \boxed{\frac{1}{2}(\boldsymbol{F}_L + \boldsymbol{F}_R) - \frac{1}{2}\sum_{\bar{\lambda}_i} \bar{C}_i |\bar{\lambda}_i| \bar{K}^{(i)}}\end{aligned}$$

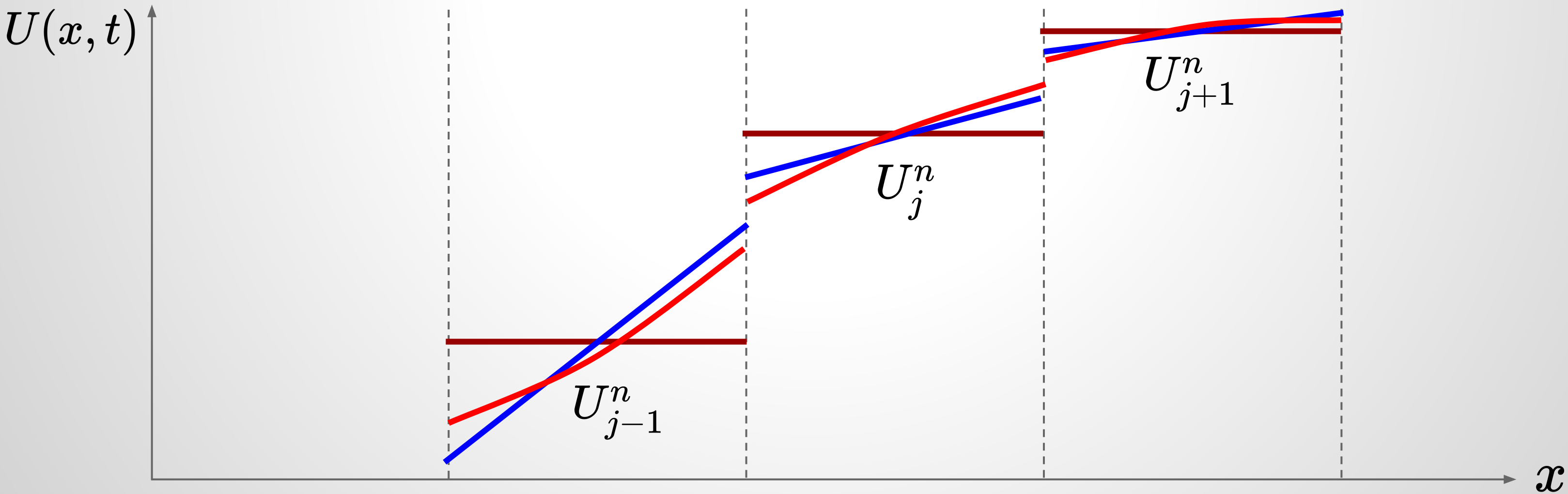
- **Procedure summary:**
  - **Compute the Roe average values of primitive variables:** $\bar{\rho}, \bar{\boldsymbol{v}}, \bar{H}$
  - **Compute the averaged eigenvalues and eigenvectors:** $\bar{\lambda}, \bar{\boldsymbol{K}}$
  - **Compute the coefficients in the eigenvector expansion:** $\bar{C}$
  - **Compute the Roe flux:** $\bar{\boldsymbol{F}}_{\text{Roe}}$

Other Riemann Solvers

- **HLL-like solvers solve <u>approximate</u> solutions to the original <u>nonlinear</u> equations**
  - **HLLE:** *Einfeldt, et al., 1991. Comp. Phys., 92, 273*
  - **HLLC:** *Toro, et al., 1994. Shock Waves, 4:25–34*
- **In comparison, Roe solver solves <u>exact</u> solutions to the approximate (<u>linearized</u>) equations**

Higher-Order Godunov Methods

- **MUSCL** (*Monotone Upstream–centred Scheme for Conservation Laws*)
- **Data reconstruction within each cell**
  - Original Godunov's scheme: piecewise constant method (**PCM**)
  - Piecewise linear method (**PLM**)
  - Piecewise parabolic method (**PPM**)



Higher-Order Godunov Methods

- Avoid introducing new local extrema during data reconstruction
  - Reduce spurious (i.e., unphysical) oscillations
  - Avoid unphysical values such as negative density/pressure
- Slope limiters
  - $\boldsymbol{U}_j(x) = \boldsymbol{U}_j + \dfrac{(x - x_j)}{\Delta x}\bar{\boldsymbol{\delta}}_i, \quad |x - x_j| \le \Delta x/2$

    where $\bar{\boldsymbol{\delta}}_i = \bar{\boldsymbol{\delta}}_i(\boldsymbol{\delta}_{i-1/2}, \boldsymbol{\delta}_{i+1/2}), \quad \boldsymbol{\delta}_{i-1/2} \equiv \boldsymbol{U}_i - \boldsymbol{U}_{i-1}$

    limited slope satisfying the TVD (Total Variation Diminishing) condition

  - **Examples:** van Leer: $\bar{\boldsymbol{\delta}}_i = \begin{cases} \dfrac{2\boldsymbol{\delta}_{i-1/2}\boldsymbol{\delta}_{i+1/2}}{\boldsymbol{\delta}_{i-1/2} + \boldsymbol{\delta}_{i+1/2}}, & \boldsymbol{\delta}_{i-1/2}\boldsymbol{\delta}_{i+1/2} \ge 0 \\ 0 \quad , & \boldsymbol{\delta}_{i-1/2}\boldsymbol{\delta}_{i+1/2} < 0 \end{cases}$

    MinMod: $\bar{\boldsymbol{\delta}}_i = \begin{cases} sign(\boldsymbol{\delta}_{i-1/2})\, min(|\boldsymbol{\delta}_{i-1/2}|, |\boldsymbol{\delta}_{i+1/2}|), & \boldsymbol{\delta}_{i-1/2}\boldsymbol{\delta}_{i+1/2} \ge 0 \\ 0 \quad , & \boldsymbol{\delta}_{i-1/2}\boldsymbol{\delta}_{i+1/2} < 0 \end{cases}$

Higher-Order Godunov Methods

- **Effects of various slope limiters**
  - **Resolution (diffusiveness) vs. robustness**
- **Left and right states are not equal unless the flow is smooth**
  - **Define Riemann problems**
- **Data reconstruction on the <u>primitive variables</u> usually results in better results (less oscillatory) than on the <u>conserved variables</u>**
  - **It may be even better to reconstruct the <u>characteristic variables</u>**
    - **Diagonalize the linearized eqs. of motion in the primitive variables**
    - **Determine eigenvectors**
    - **Perform eigen-decomposition on $\delta_{i-1/2}$ and $\delta_{i+1/2}$ to get the characteristic variables**
    - **Compute limited slopes on these characteristic variables**

Second-Order Accuracy in Time

Example: MUSCL-Hancock scheme

1. **Data reconstruction → obtain the face-centered data (i.e., data on the left and right edges of each cell) at t^n**

$$\boldsymbol{U}_{i,L}^n = \boldsymbol{U}_i^n - \frac{1}{2}\bar{\boldsymbol{\delta}}_i, \quad \boldsymbol{U}_{i,R}^n = \boldsymbol{U}_i^n + \frac{1}{2}\bar{\boldsymbol{\delta}}_i$$

2. **Evolve the face-centered data by $\Delta t/2$ using**

$$\boldsymbol{U}_{i,L}^{n+1/2} = \boldsymbol{U}_{i,L}^n - \frac{\Delta t}{2\Delta x}\left[\boldsymbol{F}_x(\boldsymbol{U}_{i,R}^n) - \boldsymbol{F}_x(\boldsymbol{U}_{i,L}^n)\right]$$

$$\boldsymbol{U}_{i,R}^{n+1/2} = \boldsymbol{U}_{i,R}^n - \frac{\Delta t}{2\Delta x}\left[\boldsymbol{F}_x(\boldsymbol{U}_{i,R}^n) - \boldsymbol{F}_x(\boldsymbol{U}_{i,L}^n)\right]$$

exactly the same fluxes;
no ghost zones are required

3. **Riemann solver → compute the inter-cell fluxes**

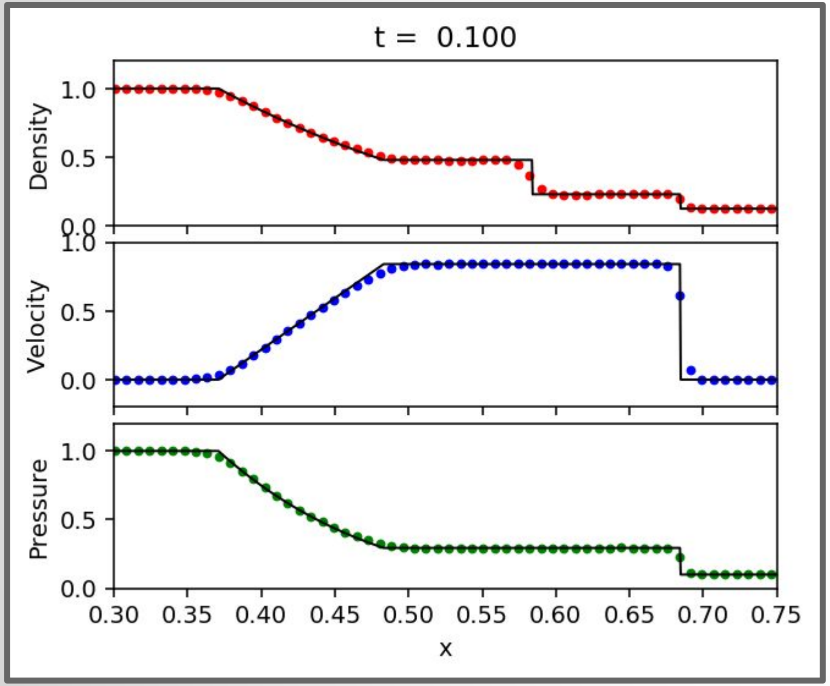
$$\boldsymbol{F}_{x,i-1/2}^{n+1/2} = Riemann(\boldsymbol{U}_L, \boldsymbol{U}_R), \text{ where } \boldsymbol{U}_L = \boldsymbol{U}_{i-1,R}^{n+1/2} \text{ and } \boldsymbol{U}_R = \boldsymbol{U}_{i,L}^{n+1/2}$$

4. **Evolve the volume-averaged data by Δt**

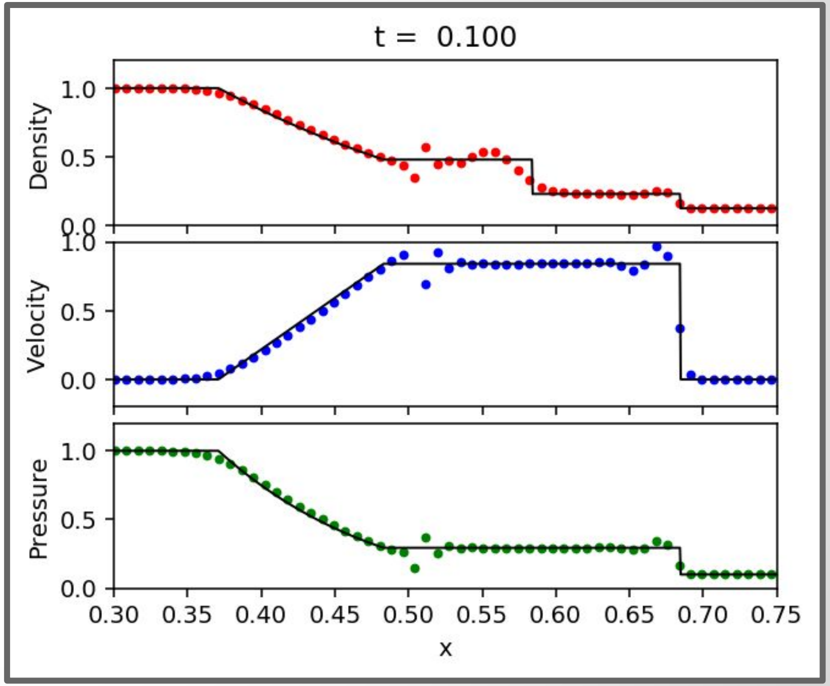
$$\boldsymbol{U}_i^{n+1} = \boldsymbol{U}_i^n - \frac{\Delta t}{\Delta x}\left[\boldsymbol{F}_{x,i+1/2}^{n+1/2} - \boldsymbol{F}_{x,i-1/2}^{n+1/2}\right]$$

Sod Shock Tube with MUSCL-Hancock

MUSCL-Hancock → much better!



Lax-Wendroff → unphysical oscillations...



Demo: MUSCL-Hancock Scheme

```python
def ComputePressure( d, px, py, pz, e ):

def ComputeTimestep( U ):

def ComputeLimitedSlope( L, C, R ):

def Conserved2Primitive( U ):

def Primitive2Conserved( W ):

def DataReconstruction_PLM( U ):

def Conserved2Flux( U ):

def Roe( L, R ):
```

lec03-demo03/04.py

Demo: MUSCL-Hancock Scheme

```
def update( frame ):
#   set the boundary conditions
    BoundaryCondition( U )

#   estimate time-step from the CFL condition
    dt = ComputeTimestep( U )

#   data reconstruction
    L, R = DataReconstruction_PLM( U )

#   update the face-centered variables by 0.5*dt
        for j in range( 1, N-1 ):
            flux_L, flux_R = Conserved2Flux( L[j] ), Conserved2Flux( R[j] )
            dflux = 0.5*dt/dx*( flux_R - flux_L )
            L[j] -= dflux
            R[j] -= dflux
```

lec03-demo03/04.py

Demo: MUSCL-Hancock Scheme

```
def update( frame ):
   ...

#  compute fluxes
   flux = np.empty( (N,5) )
   for j in range( nghost, N-nghost+1 ):
      flux[j] = Roe( R[j-1], L[j] )

#  update the cell-centered input variables by dt
   U[nghost:N-nghost] -= dt/dx*( flux[nghost+1:N-nghost+1] - \
                                 flux[nghost:N-nghost] )
```

lec03-demo03/04.py

Run lec03-demo03/04.py

- Compare with demo01 & 02
- Find a Riemann problem that crashes the code!