

Magnetohydrodynamic (MHD) simulations

Seiji ZENITANI

Space Research Institute (IWF)
Austrian Academy of Sciences

August 2024, Munich

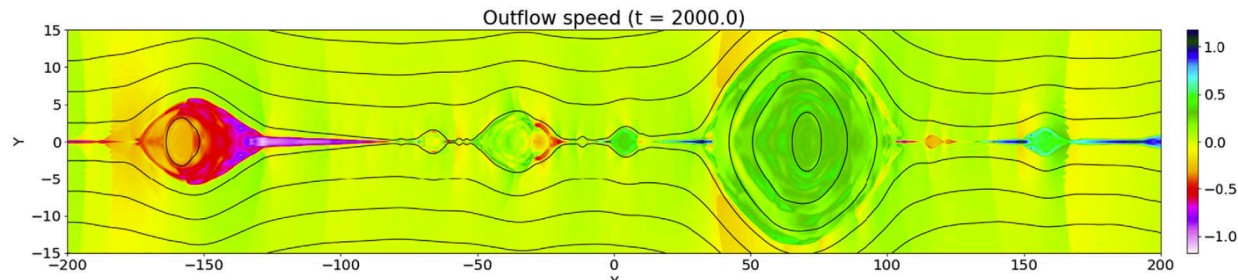
Outline

1. MHD at a glance
2. Basic theory: Advection problem
3. Basic theory: Finite-volume method and Riemann solver
4. MHD simulation with Riemann solver
5. MHD simulation in multi-dimensions
6. Hands on

1. MHD at a glance

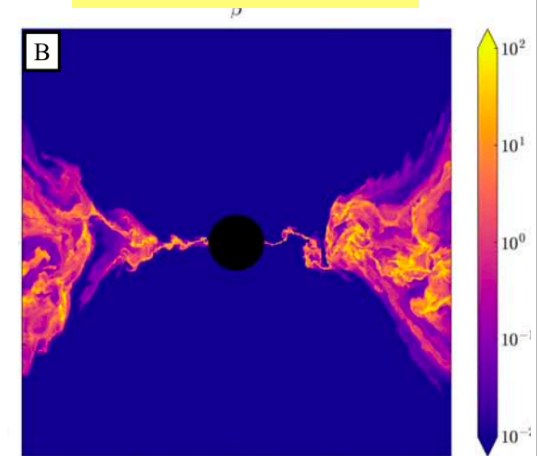
MHD gallery

Basic process (magnetic reconnection)

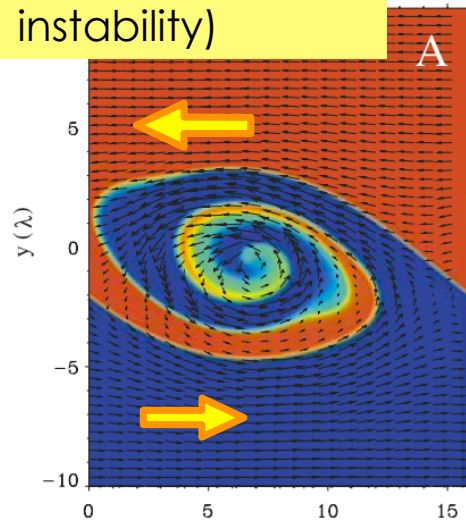


Zenitani+ 2020 ApJ

Black hole accretion disk

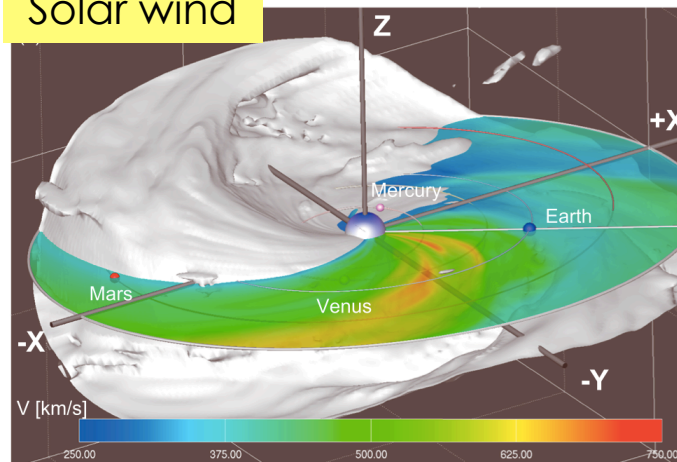


Basic process (Kelvin-Helmholtz instability)

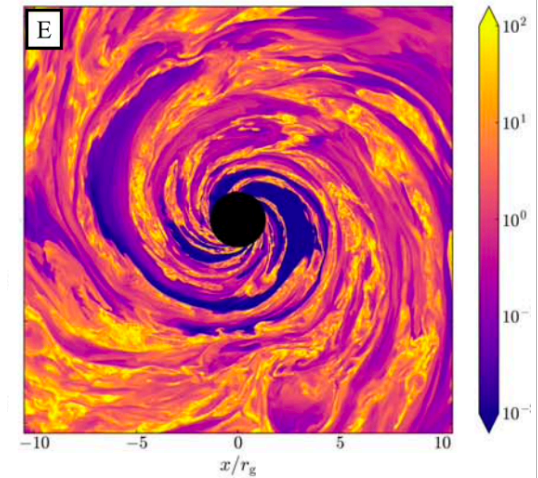


Matsumoto+ 2004 GRL

Solar wind



Shiota+ 2014 Space Weather



Ripperda+ 2022 ApJ

MHD equations = fluid + mag. field + ?

- Continuity $\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{v}) = 0$
- Momentum $\rho \frac{d\mathbf{v}}{dt} = -\nabla p + \frac{\mathbf{j} \times \mathbf{B}}{c}$ ← Ampere's law $\nabla \times \mathbf{B} = \frac{4\pi}{c} \mathbf{j} + \frac{1}{c} \frac{\partial \mathbf{E}}{\partial t}$

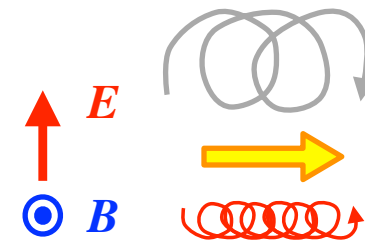
$$= -\nabla p - \nabla \left(\frac{B^2}{8\pi} \right) + \frac{1}{4\pi} (\mathbf{B} \cdot \nabla) \mathbf{B}$$
- Energy* $\frac{d}{dt} \left(\frac{p}{\rho^\gamma} \right) = 0$
- Magnetic field $\frac{\partial \mathbf{B}}{\partial t} = -c(\nabla \times \mathbf{E})$

We have 8 equations, but 11 unknowns...

MHD equations = fluid + mag. field + Ohm's law

- Continuity $\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{v}) = 0$
- Momentum $\rho \frac{d\mathbf{v}}{dt} = -\nabla p + \frac{\mathbf{j} \times \mathbf{B}}{c}$
 $= -\nabla p - \nabla \left(\frac{B^2}{8\pi} \right) + \frac{1}{4\pi} (\mathbf{B} \cdot \nabla) \mathbf{B}$
- Energy* $\frac{d}{dt} \left(\frac{p}{\rho^\gamma} \right) = 0$
- Magnetic field $\frac{\partial \mathbf{B}}{\partial t} = -c(\nabla \times \mathbf{E})$

- Ohm's law $\mathbf{E} + \frac{\mathbf{v}}{c} \times \mathbf{B} = 0$



$$\mathbf{v}_{\mathbf{E} \times \mathbf{B}} = \frac{c \mathbf{E} \times \mathbf{B}}{B^2}$$

MHD equations - conservative form (1/2)

- Continuity

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{v}) = 0$$

- Momentum

$$\frac{\partial \rho \mathbf{v}}{\partial t} + \nabla \cdot \left(\rho \mathbf{v} \mathbf{v} + \left(p + \frac{B^2}{8\pi} \right) \mathbf{I} - \frac{1}{4\pi} \mathbf{B} \mathbf{B} \right) = 0$$

- Energy

$$\frac{\partial \mathcal{E}}{\partial t} + \nabla \cdot \left((\mathcal{E} + p) \mathbf{v} \right) = \mathbf{j} \cdot \mathbf{E} \quad \mathcal{E} = \frac{1}{2} \rho v^2 + \frac{1}{\gamma - 1} p$$

$$\frac{\partial}{\partial t} \left(\frac{B^2}{8\pi} + \frac{E^2}{8\pi} \right) + \nabla \cdot \left(\frac{c}{4\pi} \mathbf{E} \times \mathbf{B} \right) = -\mathbf{j} \cdot \mathbf{E}$$

$$\frac{\partial}{\partial t} \left(\mathcal{E} + \frac{B^2}{8\pi} \right) + \nabla \cdot \left((\mathcal{E} + p) \mathbf{v} + \frac{c}{4\pi} \mathbf{E} \times \mathbf{B} \right) = 0$$

- Mag. field

$$\frac{\partial \mathbf{B}}{\partial t} = -c(\nabla \times \mathbf{E})$$

- Ohm's law

$$\mathbf{E} + \frac{\mathbf{v}}{c} \times \mathbf{B} = 0$$

$$\frac{\partial}{\partial t} \mathbf{B} - \nabla \times (\mathbf{v} \times \mathbf{B}) = 0$$

MHD equations - conservative form (2/2)

Conserved quantities

Numerical flux

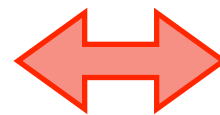
Source term

$$\frac{\partial}{\partial t} \begin{pmatrix} \rho \\ \rho \mathbf{v} \\ \frac{1}{2} \rho v^2 + \frac{1}{\gamma-1} p + \frac{1}{8\pi} B^2 \\ \mathbf{B} \end{pmatrix} + \nabla \cdot \begin{pmatrix} \rho \mathbf{v} \\ \rho \mathbf{v} \mathbf{v} + (p + \frac{B^2}{8\pi}) \mathbf{I} - \frac{1}{4\pi} \mathbf{B} \mathbf{B} \\ (\frac{1}{2} \rho v^2 + \frac{\gamma}{\gamma-1} p) \mathbf{v} + \frac{c}{4\pi} \mathbf{E} \times \mathbf{B} \\ \mathbf{v} \mathbf{B} - \mathbf{B} \mathbf{v} \end{pmatrix} = 0$$

$$\frac{\partial}{\partial t} \mathbf{U} + \nabla \cdot \mathbf{F} = 0$$

Conserved quantities

$$\mathbf{U} \equiv (\rho \quad \rho \mathbf{v} \quad \frac{1}{2} \rho v^2 + \frac{p}{\gamma-1} + \frac{B^2}{8\pi} \quad \mathbf{B})^T$$



Primitive variables

$$\mathbf{V} \equiv (\rho \quad \mathbf{v} \quad p \quad \mathbf{B})^T$$

MHD waves - magnetosonic waves

- **Alfvén** wave

$$\left(\frac{\omega}{k}\right)^2 = c_A^2 \cos^2 \theta$$

Alfvén speed

$$c_A^2 = \frac{B^2}{4\pi\rho}$$

Sound speed

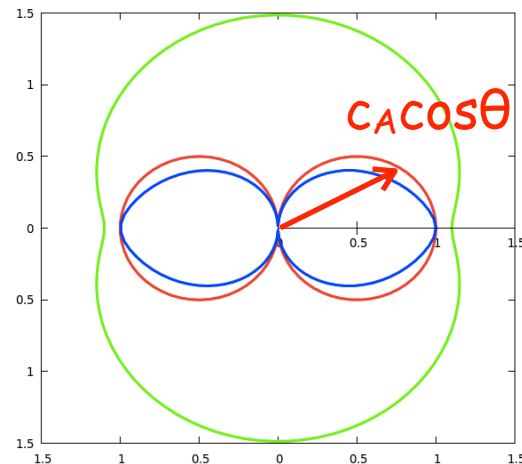
$$c_s^2 = \frac{\gamma p}{\rho}$$

- **Fast** and **slow** magnetosonic waves

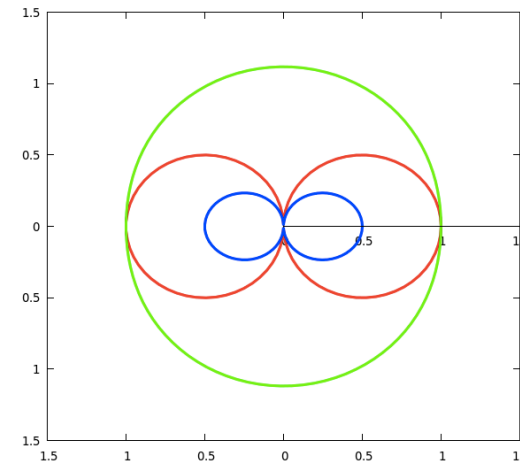
$$\left(\frac{\omega}{k}\right)^2 = \frac{1}{2} \left\{ (c_A^2 + c_s^2) \pm \sqrt{(c_A^2 + c_s^2)^2 - 4c_A^2 c_s^2 \cos^2 \theta} \right\}$$

- Friedrichs diagram (phase-speed)

- $c_s > c_A$



- $c_s < c_A$

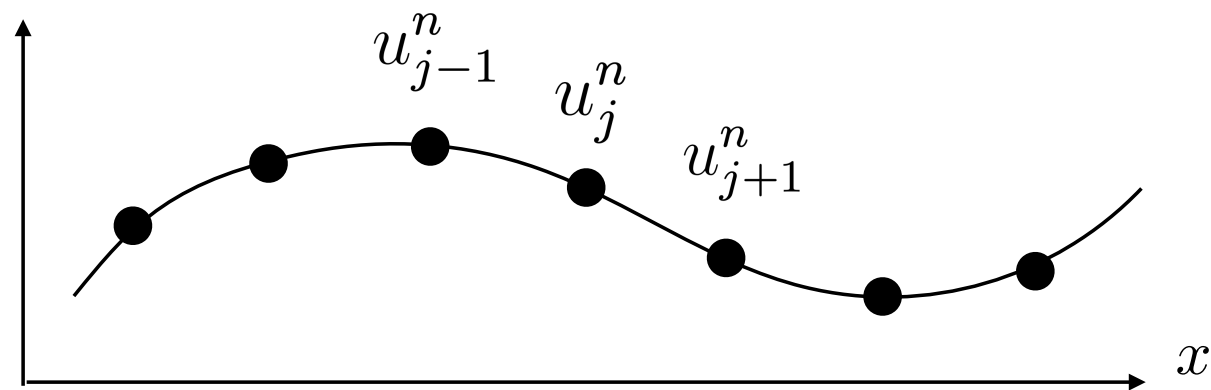


2. Advection problem

Discretization

- We approximate the real system,
by using a finite number of discrete grid points
- Grid points in space (x) and time (t) directions

Some quantity: u



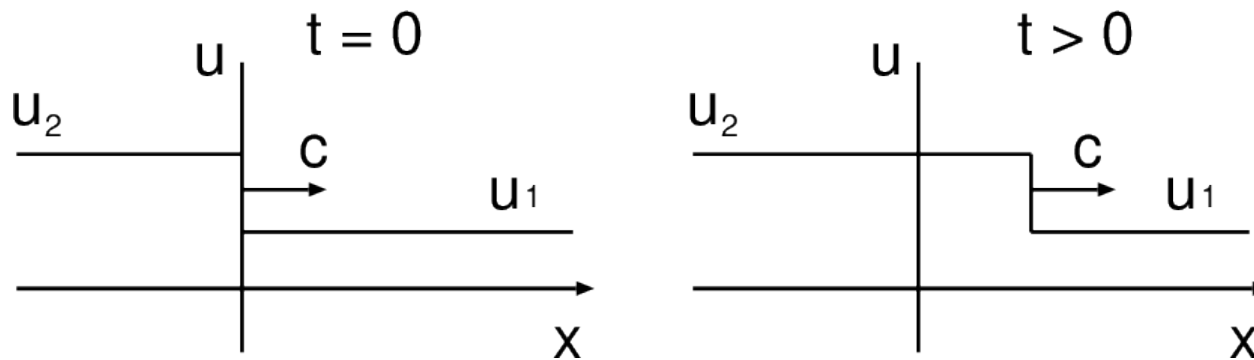
n : time step
 j : spatial step

Linear advection equation

- Linear advection equation (Note: c is a positive constant)

$$\frac{\partial u}{\partial t} + c \frac{\partial u}{\partial x} = 0 \quad u = f(x - ct)$$

- It allows any profiles traveling at the speed of c



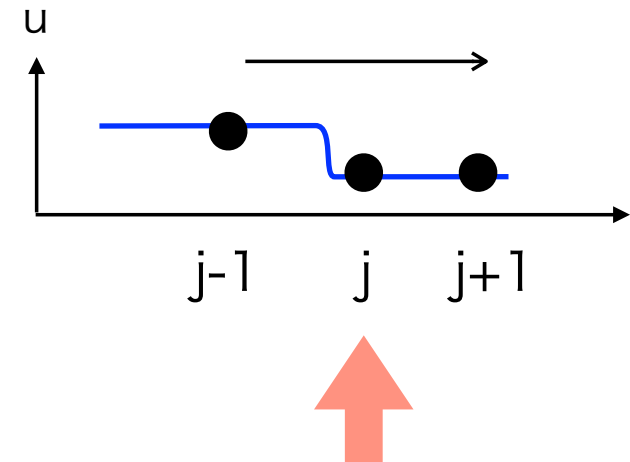
FTCS scheme

(Forward in Time and Centered in Space scheme)

- Equation $\frac{\partial u}{\partial t} + c \frac{\partial u}{\partial x} = 0$
- Time derivative $\frac{\partial u}{\partial t} \approx \frac{u_j^{n+1} - u_j^n}{\Delta t}$ forward difference
explicit method
- Spatial derivative $\frac{\partial u}{\partial x} \approx \frac{u_{j+1}^n - u_{j-1}^n}{2\Delta x}$ central difference
- Discretized equation $u_j^{n+1} = u_j^n - \frac{c\Delta t}{2\Delta x} (u_{j+1}^n - u_{j-1}^n)$
n : time step
j : spatial step

Upwind scheme (1/2)

- Information comes from the left
→ Left information should be used



- Equation

$$\frac{\partial u}{\partial t} + c \frac{\partial u}{\partial x} = 0$$

- Time derivative

$$\frac{\partial u}{\partial t} \approx \frac{u_j^{n+1} - u_j^n}{\Delta t}$$

- Spatial derivative

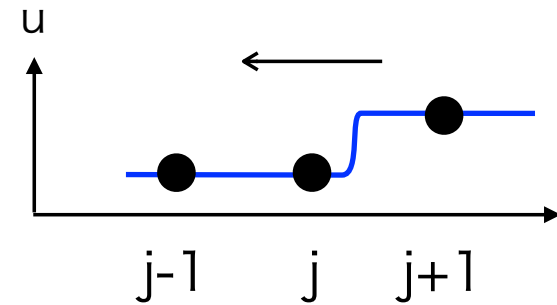
$$\frac{\partial u}{\partial x} \approx \frac{u_j^n - u_{j-1}^n}{\Delta x}$$

- Discretized eq.

$$u_j^{n+1} = u_j^n - c \frac{\Delta t}{\Delta x} (u_j^n - u_{j-1}^n)$$

Upwind scheme (2/2)

- Information from the right ($c < 0$)
→ Right information should be used



- Spatial derivative

$$\frac{\partial u}{\partial x} \approx \frac{u_{j+1}^n - u_j^n}{\Delta x}$$

- Discretized eq.

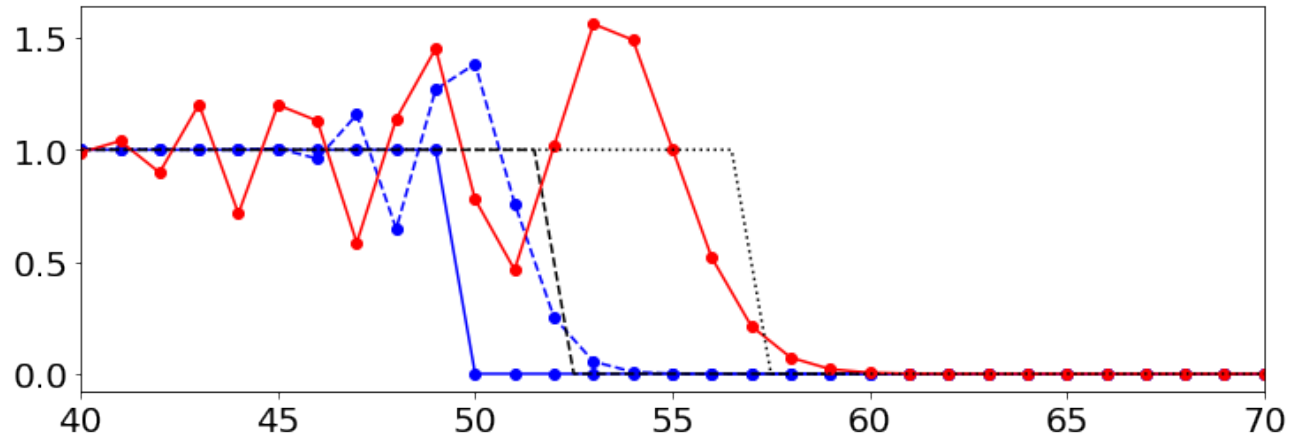
$$u_j^{n+1} = u_j^n - c \frac{\Delta t}{\Delta x} (u_{j+1}^n - u_j^n)$$

- **Combing left and right cases:**

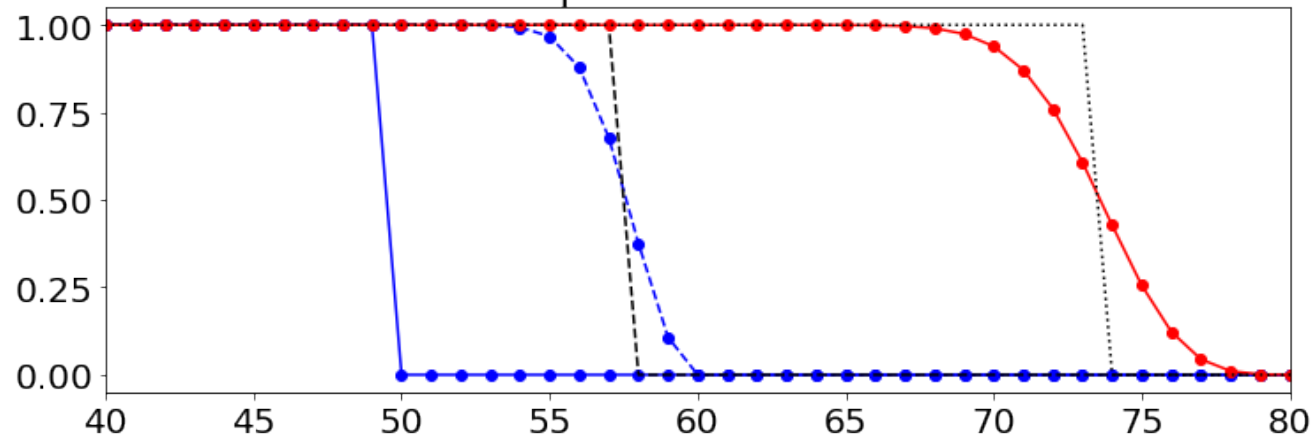
$$u_j^{n+1} = u_j^n - \frac{\Delta t}{\Delta x} \left(\frac{c - |c|}{2} (u_{j+1}^n - u_j^n) + \frac{c + |c|}{2} (u_j^n - u_{j-1}^n) \right)$$

Linear advection problem

FTCS scheme



Upwind scheme



von Neumann analysis

- Fourier component

$$u_j^n = g^n \exp(i(j\theta)) = g^n e^{ij\theta}$$

wavenumber
↓
↑ amplification factor

← $\exp(ix) = \cos x + i \sin x$

Courant number

$$\nu = c \frac{\Delta t}{\Delta x}$$

- FTCS scheme

$$g^{n+1} e^{ij\theta} = g^n \left\{ e^{ij\theta} - \frac{\nu}{2} (e^{i(j+1)\theta} - e^{i(j-1)\theta}) \right\}$$

$$g = g^{n+1}/g^n = 1 - \frac{\nu}{2} (e^{i\theta} - e^{-i\theta})$$

$$= 1 - i\nu \sin \theta$$

$$|g|^2 = 1 + \nu^2 \sin^2 \theta$$

- Upwind scheme

$$g^{n+1} e^{ij\theta} = g^n \left\{ e^{ij\theta} - \nu (e^{ij\theta} - e^{i(j-1)\theta}) \right\}$$

$$g = g^{n+1}/g^n = 1 - \nu(1 - e^{-i\theta})$$

$$= (1 - \nu) + \nu \cos \theta + i\nu \sin \theta$$

$$|g|^2 = 1 + 2(1 - \nu)\nu(1 - \cos \theta) \leq 1$$

The code always amplify waves!!

Unconditionally unstable

when $0 < \nu \leq 1$ Stable

Courant-Friedrich-Lewy (CFL) condition

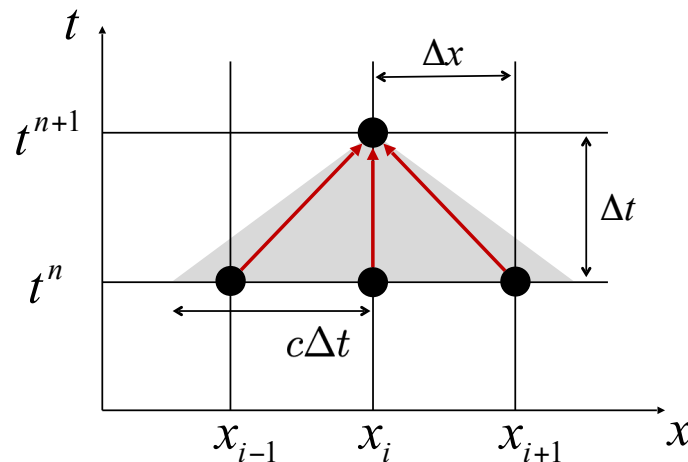
- Courant number

$$\nu = c \frac{\Delta t}{\Delta x} \quad \nu \leq 1$$

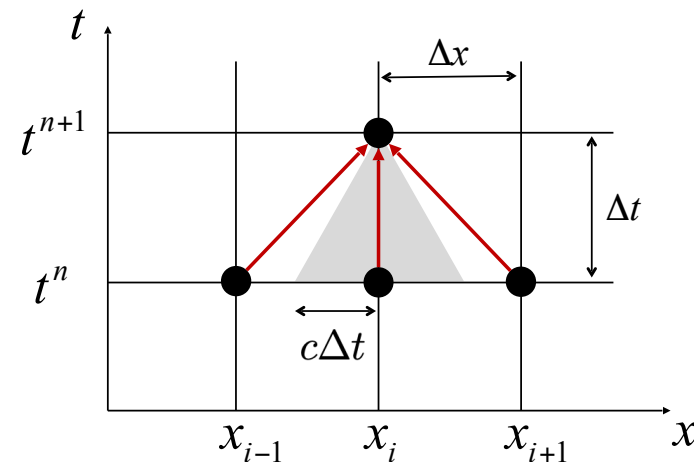
- A "must" condition for explicit schemes

$$\nu > 1 \implies c\Delta t > \Delta x$$

$$\nu \leq 1 \implies c\Delta t \leq \Delta x$$



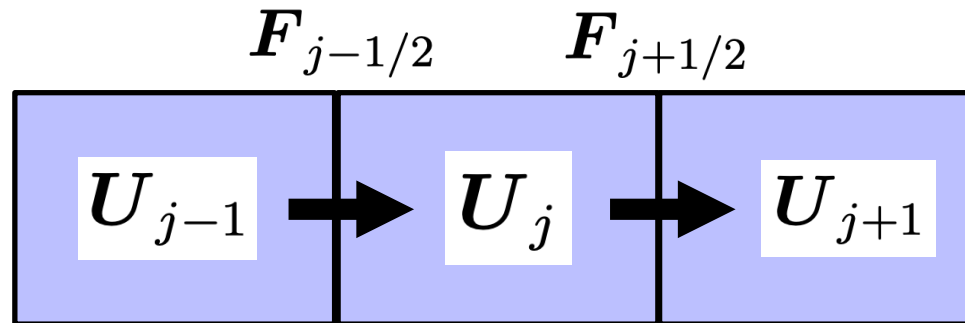
Unstable; breaks causality



Stable

3. Finite-volume method and Riemann solver

Finite volume method

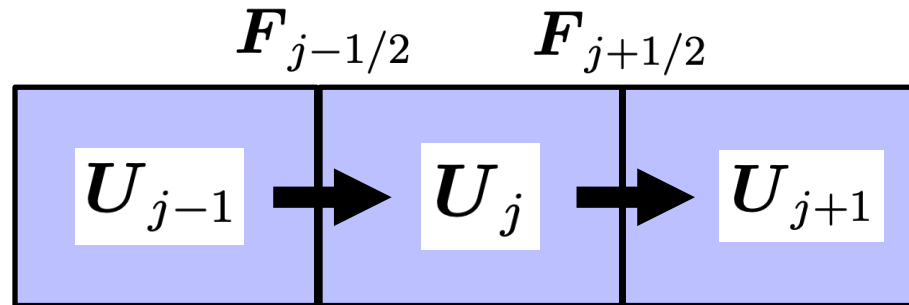


$$U_j(t + \Delta t) = U_j(t) - \frac{\Delta t}{\Delta x} (\mathbf{F}_{j+1/2} - \mathbf{F}_{j-1/2})$$

$$\frac{\partial}{\partial t} \mathbf{U} + \nabla \cdot \mathbf{F} = 0$$

- Physical quantities (U) are defined at the cell center
- Numerical fluxes (F) are defined between the cells

Numerical flux in linear advection problem



$$\frac{\partial u}{\partial t} + c \frac{\partial u}{\partial x} = 0 \quad F_j = cu_j$$

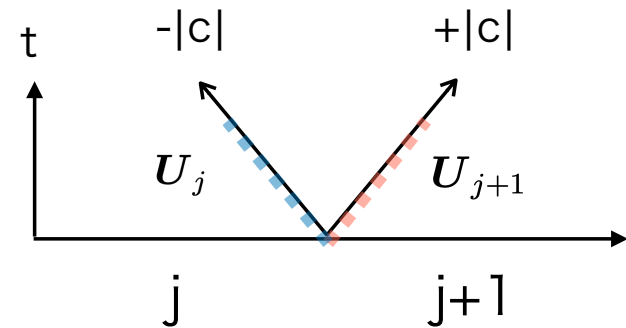
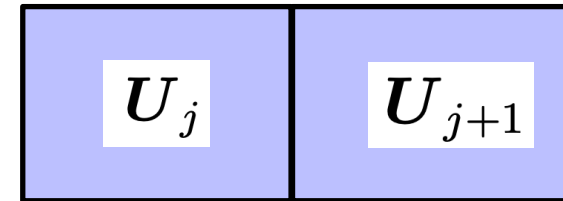
- FTCS scheme $F_{j+1/2}^n = \frac{1}{2} (cu_{j+1}^n + cu_j^n) = \frac{1}{2} (F_{j+1}^n + F_j^n)$

- Upwind scheme $F_{j+1/2}^n = \frac{1}{2} (F_{j+1}^n + F_j^n) - \frac{|c|}{2} (u_{j+1}^n - u_j^n)$

$$F_{j+1/2}^n = \begin{cases} F_j^n & (\text{for } c \geq 0) \\ F_{j+1}^n & (\text{for } c < 0) \end{cases}$$

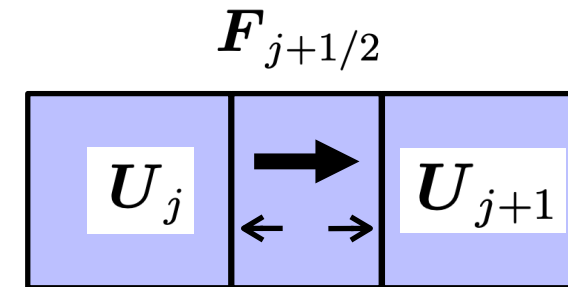
Upwind method - intermediate state

- Let's consider a pair of boundaries, which expand at the speed of $\pm|c|$



Upwind method - intermediate state

- Let's consider a pair of boundaries, which expand at the speed of $\pm|c|$



- Conservation of physical quantities

$$F_{left} - \lambda U_{left} = F_{right} - \lambda U_{right}$$

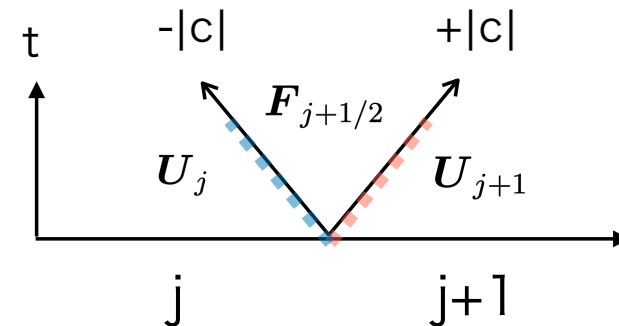
λ : speed of the boundary

- Across the two boundaries

$$F_{j+1} - |c|U_{j+1} = F_{j+1/2} - |c|U_{j+1/2}$$

$$F_j + |c|U_j = F_{j+1/2} + |c|U_{j+1/2}$$

$$F_{j+1/2} = \frac{1}{2} (F_{j+1} + F_j) - \frac{|c|}{2} (U_{j+1} - U_j)$$



Reminder: MHD equations - conservative form

Conserved quantities

Numerical flux

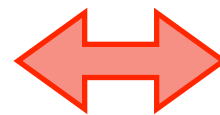
Source term

$$\frac{\partial}{\partial t} \begin{pmatrix} \rho \\ \rho \mathbf{v} \\ \frac{1}{2} \rho v^2 + \frac{1}{\gamma-1} p + \frac{1}{8\pi} B^2 \\ \mathbf{B} \end{pmatrix} + \nabla \cdot \begin{pmatrix} \rho \mathbf{v} \\ \rho \mathbf{v} \mathbf{v} + (p + \frac{B^2}{8\pi}) \mathbf{I} - \frac{1}{4\pi} \mathbf{B} \mathbf{B} \\ (\frac{1}{2} \rho v^2 + \frac{\gamma}{\gamma-1} p) \mathbf{v} + \frac{c}{4\pi} \mathbf{E} \times \mathbf{B} \\ \mathbf{v} \mathbf{B} - \mathbf{B} \mathbf{v} \end{pmatrix} = 0$$

$$\frac{\partial}{\partial t} \mathbf{U} + \nabla \cdot \mathbf{F} = 0$$

Conserved quantities

$$\mathbf{U} \equiv (\rho \quad \rho \mathbf{v} \quad \frac{1}{2} \rho v^2 + \frac{p}{\gamma-1} + \frac{B^2}{8\pi} \quad \mathbf{B})^T$$



Primitive variables

$$\mathbf{V} \equiv (\rho \quad \mathbf{v} \quad p \quad \mathbf{B})^T$$

MHD equations in the X direction (1/3)

$$\frac{\partial}{\partial t} \mathbf{U} + \nabla \cdot \mathbf{F} = 0 \quad B_x = \text{const.} \quad \nabla \cdot \mathbf{B} = 0$$

$$\frac{\partial}{\partial t} \begin{pmatrix} \rho \\ \rho v_x \\ \rho v_y \\ \rho v_z \\ B_y \\ B_z \\ \mathcal{E} \end{pmatrix} + \frac{\partial}{\partial x} \begin{pmatrix} \rho v_x \\ \rho v_x v_x + p \\ \rho v_x v_y \\ \rho v_x v_z \\ B_y v_x - v_y B_x \\ B_z v_x - v_z B_x \\ (\mathcal{E} + p_T) v_x - B_x (v_x B_x + v_y B_y + v_z B_z) \end{pmatrix} = 0$$

$$\mathcal{E} = \frac{1}{2} \rho v^2 + \frac{1}{\gamma - 1} p + \frac{B^2}{2}, \quad p_T = p + \frac{B^2}{2}$$

MHD equations in the X direction (2/3)

$$\frac{\partial \mathbf{U}}{\partial t} + \frac{\partial \mathbf{F}}{\partial x} = 0$$

Jacobean matrix

$$\frac{\partial \mathbf{U}}{\partial t} + \mathbf{A} \frac{\partial \mathbf{U}}{\partial x} = 0, \quad \mathbf{A} = \frac{\partial \mathbf{F}}{\partial \mathbf{U}}$$

Λ Diagonal matrix

\mathbf{R} Right eigenvectors

$$\mathbf{R}^{-1} \frac{\partial \mathbf{U}}{\partial t} + \mathbf{R}^{-1} \mathbf{A} \mathbf{R} \mathbf{R}^{-1} \frac{\partial \mathbf{U}}{\partial x} = 0$$

\mathbf{R}^{-1} Left eigenvectors

$$\frac{\partial \mathbf{W}}{\partial t} + \Lambda \frac{\partial \mathbf{W}}{\partial x} = 0, \quad d\mathbf{W} = \mathbf{R}^{-1} d\mathbf{U}$$

Characteristic variables
(Properties transported by waves)

$$\mathbf{R}^{-1} \mathbf{A} \mathbf{R} = \Lambda = \text{diag} (v_x - c_f, v_x - c_a, \dots, v_x + c_a, v_x + c_f)$$

- See Stone et al. 2008 ApJS for further detail

MHD equations in the X direction (3/3)

$$\frac{\partial \mathbf{W}}{\partial t} + \mathbf{A} \frac{\partial \mathbf{W}}{\partial x} = 0, \quad d\mathbf{W} = \mathbf{R}^{-1} d\mathbf{U}$$

$$\frac{\partial}{\partial t} \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_m \end{pmatrix} + \begin{pmatrix} \lambda_1 & 0 & \dots & 0 \\ 0 & \lambda_2 & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \dots & 0 & \lambda_m \end{pmatrix} \frac{\partial}{\partial x} \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_m \end{pmatrix} = 0$$

Characteristic variables
(Properties transported
by waves)

MHD wavespeeds

$$\mathbf{R}^{-1} \mathbf{A} \mathbf{R} = \mathbf{\Lambda} = \text{diag} (v_x - c_f, v_x - c_a, \dots, v_x + c_a, v_x + c_f)$$

fast
Alfvén
...
Alfvén
fast

An advection problem of various MHD waves

Local Lax-Friedrich (LLF) method

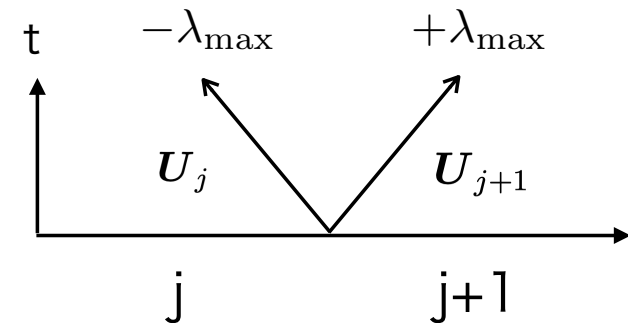
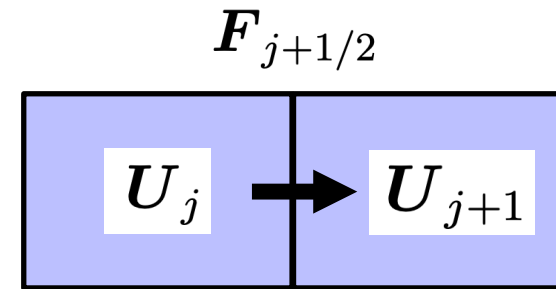
- MHD variant of the upwind scheme
- We employ fastest fast-wave speeds as the signal speeds

$$\lambda_{\max} = \max\left(|v - c_f|_j, |v - c_f|_{j+1}, |v + c_f|_j, |v + c_f|_{j+1}\right)$$

c_f : the fast-mode speed

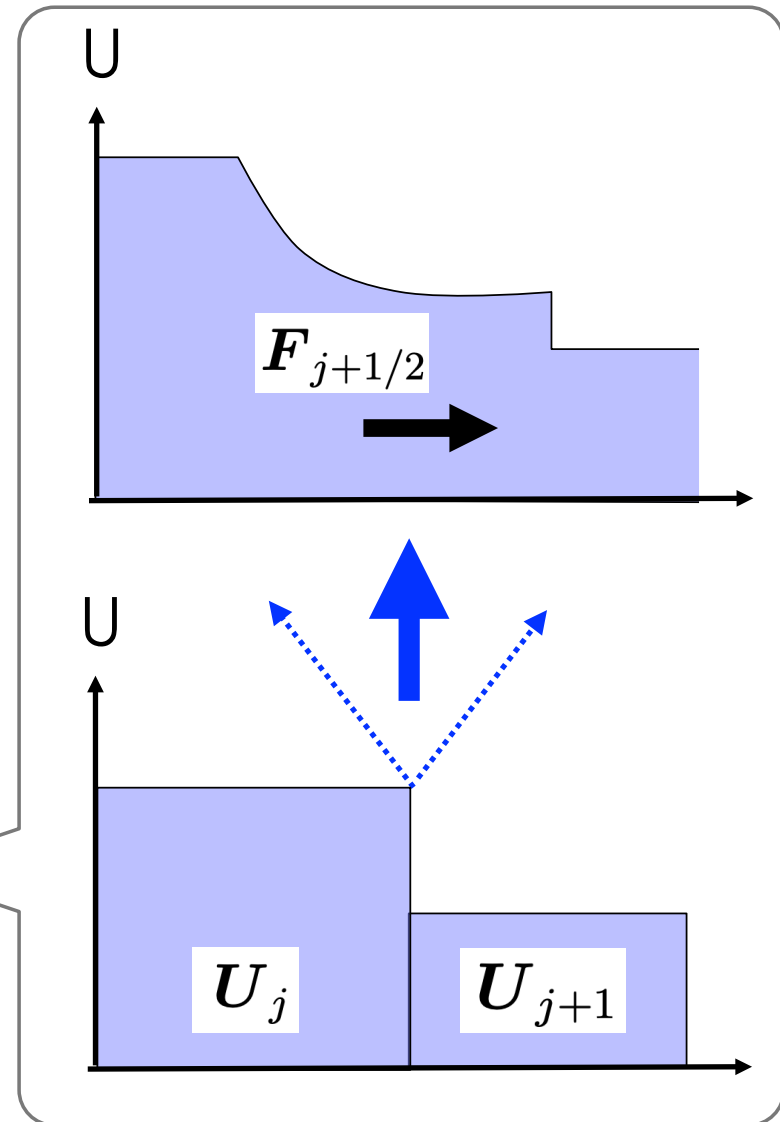
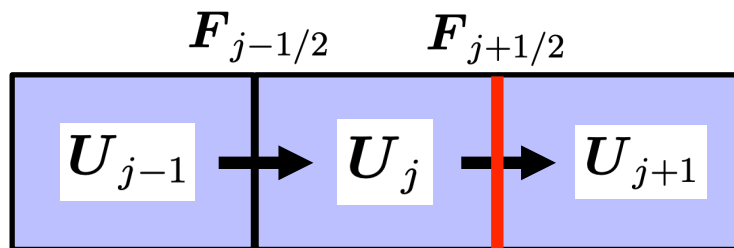
- Numerical flux can be calculated easily

$$\mathbf{F}_{j+1/2} = \frac{1}{2} (\mathbf{F}_{j+1} + \mathbf{F}_j) - \frac{|\lambda_{\max}|}{2} (\mathbf{U}_{j+1} - \mathbf{U}_j)$$



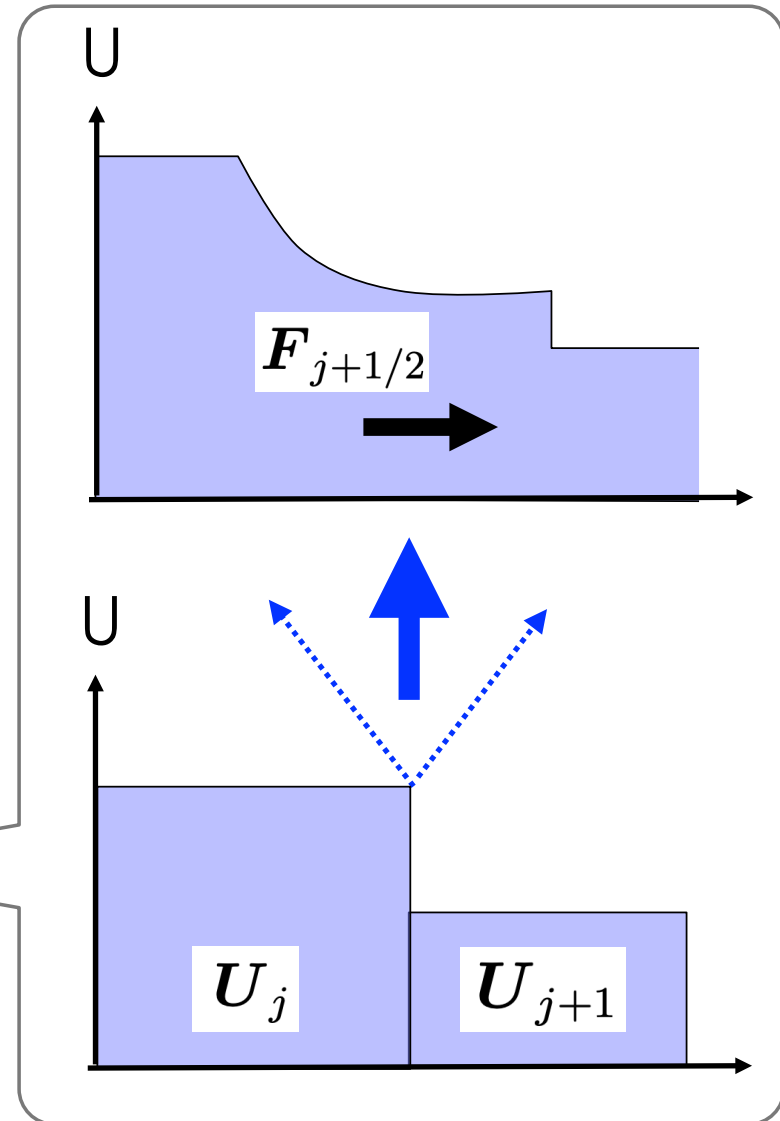
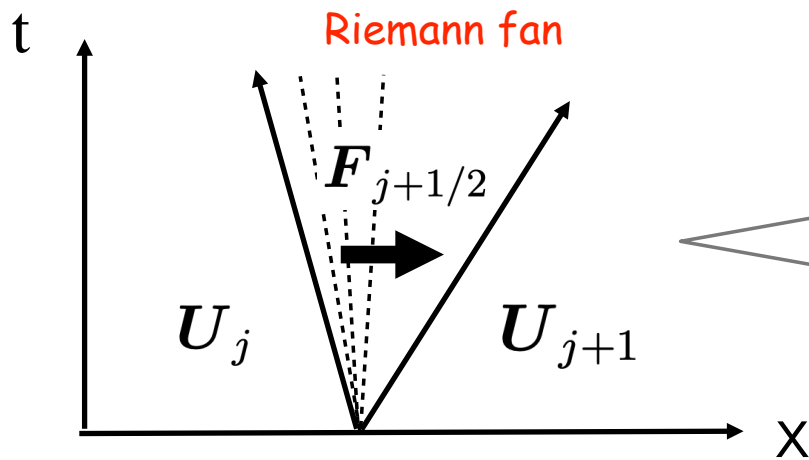
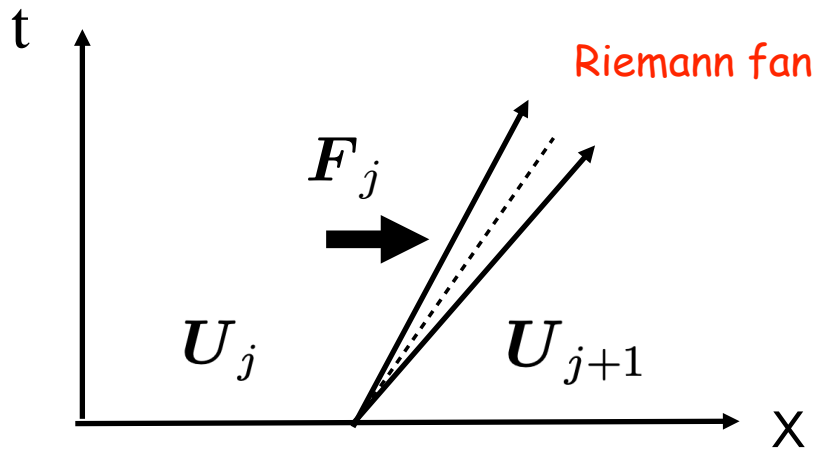
Riemann solver (1/2)

- Riemann problem
 - Time evolution from two flat states
 - Basic problem in hydrodynamics
- Riemann solver
 - Solve Riemann problem at each cell interface

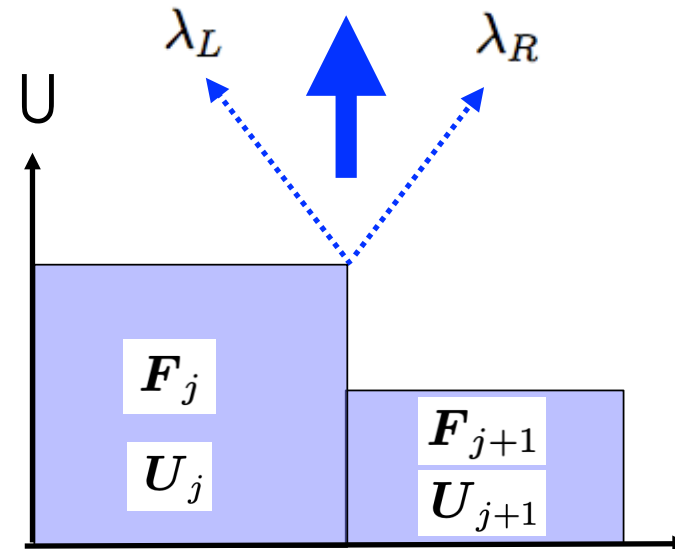
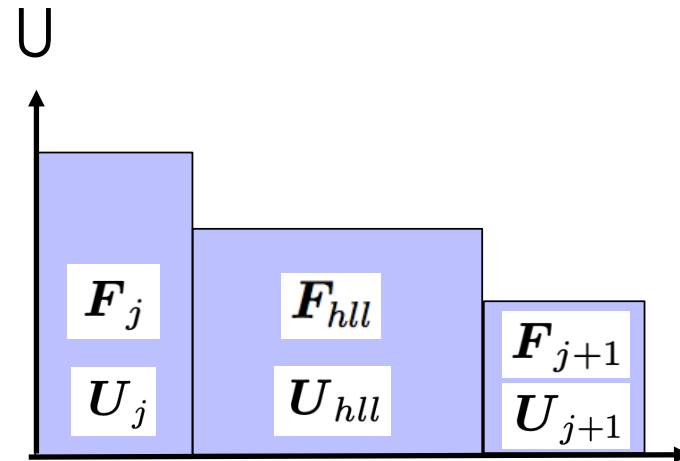
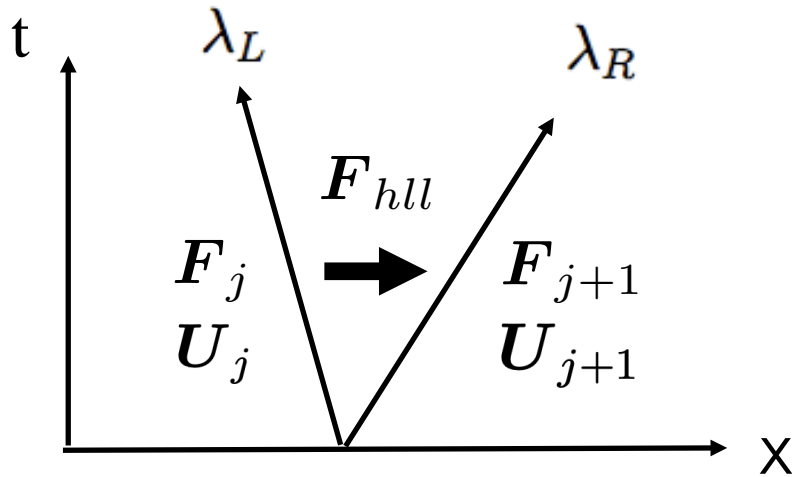


Riemann solver (2/2)

- It acts as an upwind scheme, when necessary.



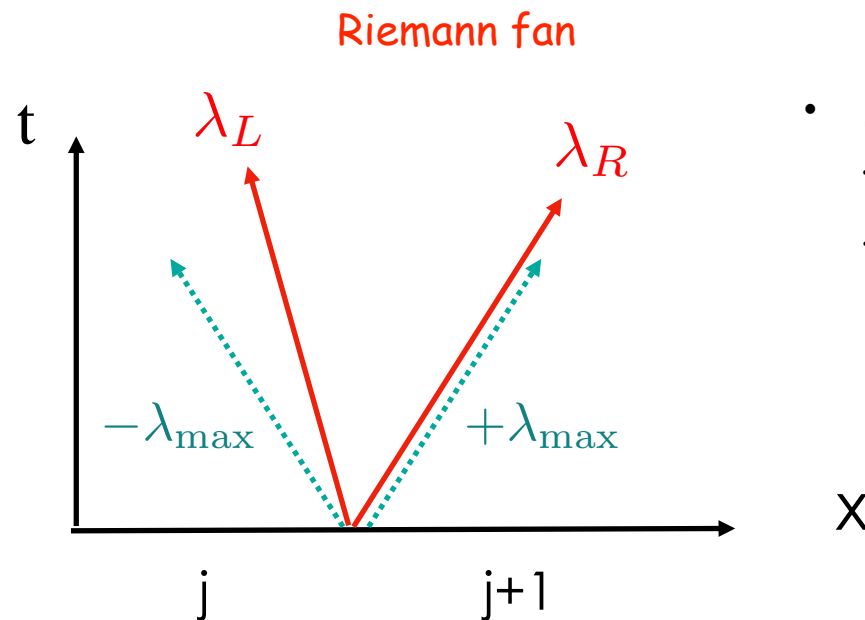
Approximate Riemann solver - HLL solver (1/2)



$$\mathbf{F} = \begin{cases} \mathbf{F}_j & (\lambda_L > 0) \\ \mathbf{F}_{hll} & (\lambda_L \leq 0 \leq \lambda_R) \\ \mathbf{F}_{j+1} & (\lambda_R < 0) \end{cases}$$

$$\mathbf{F}_{hll} = \frac{\lambda_R \mathbf{F}_j - \lambda_L \mathbf{F}_{j+1} + \lambda_R \lambda_L (\mathbf{U}_{j+1} - \mathbf{U}_j)}{\lambda_R - \lambda_L}$$

Approximate Riemann solver - HLL solver (2/2)



- **HLL solver** (or Riemann solvers) uses the fastest left-going signals and the fastest right-going signals

$$\lambda_L = \min\left((v - c_f)_j, (v - c_f)_{j+1}\right)$$

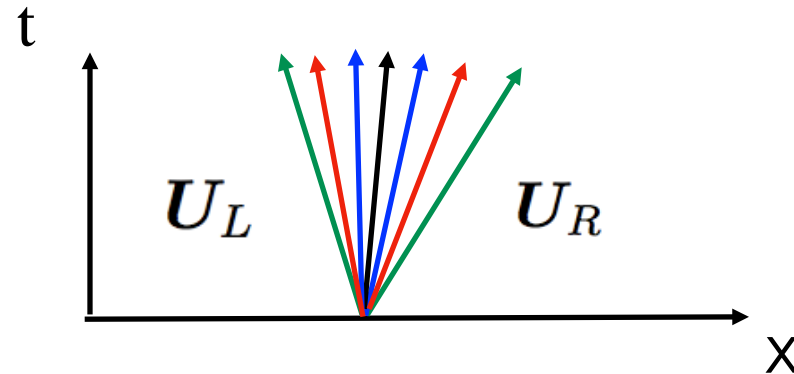
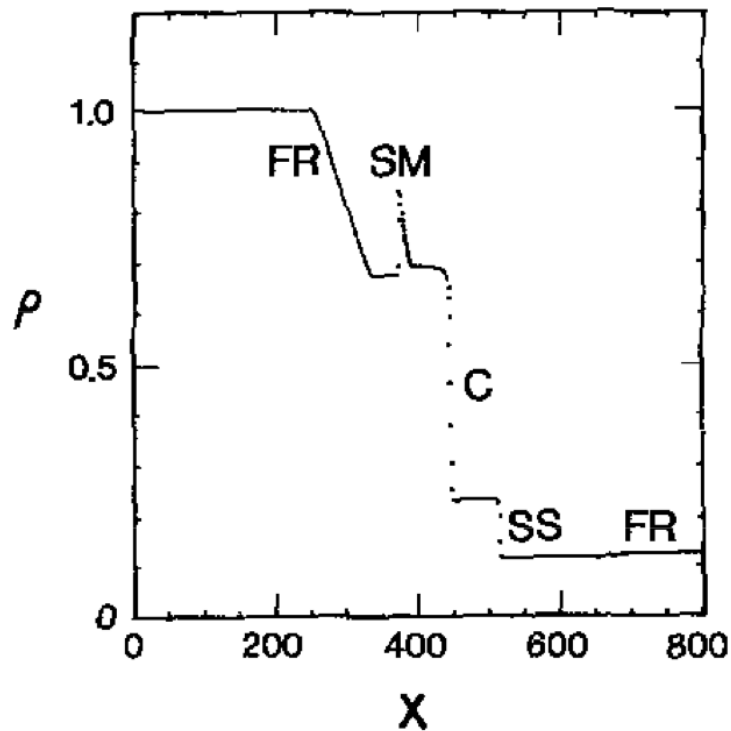
$$\lambda_R = \max\left((v + c_f)_j, (v + c_f)_{j+1}\right)$$

- **Local Lax-Friedrich solver** is more diffusive than HLL solver, because it spreads physical quantities wider

$$\begin{aligned} \lambda_{\max} &= \max\left(|v - c_f|_j, |v - c_f|_{j+1}, |v + c_f|_j, |v + c_f|_{j+1}\right) \\ &\equiv \max(|\lambda_R|, |\lambda_L|) \end{aligned}$$

MHD Riemann problem

- There are 6 intermediate states, corresponding to 6 MHD waves
- Ex. Brio=Wu problem

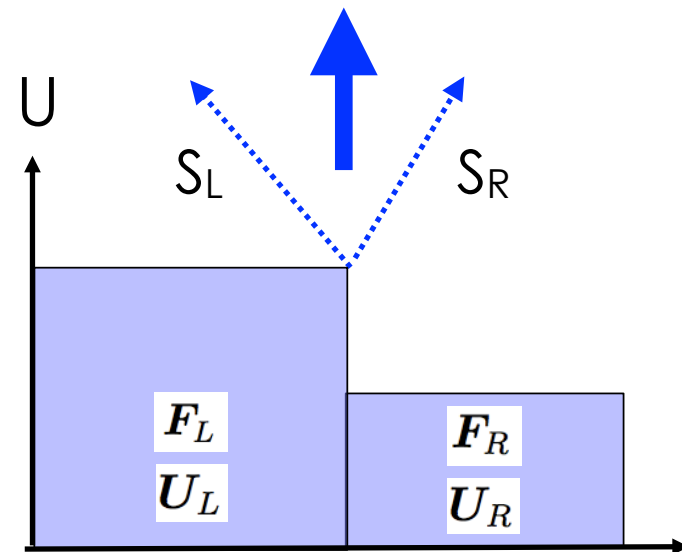
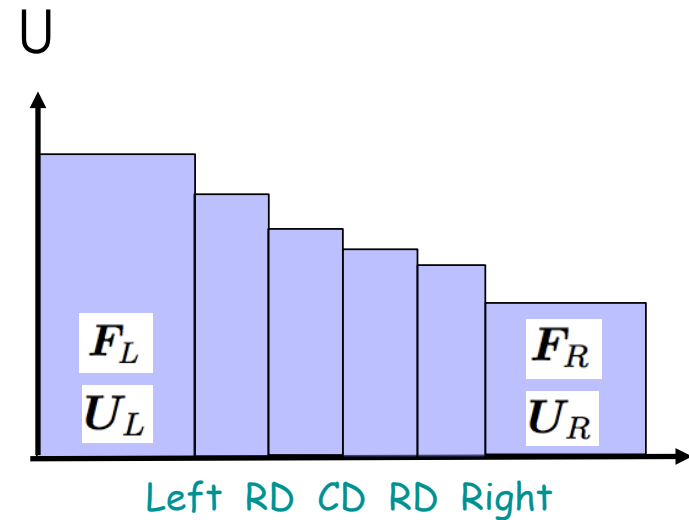
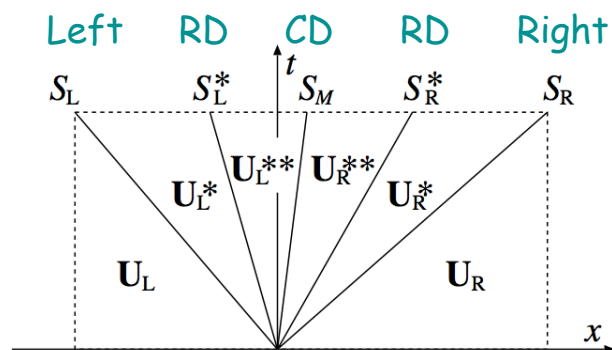


- Alfvén wave
- Fast and slow magnetosonic waves

Approximate Riemann solver - HLLD solver (1/3)

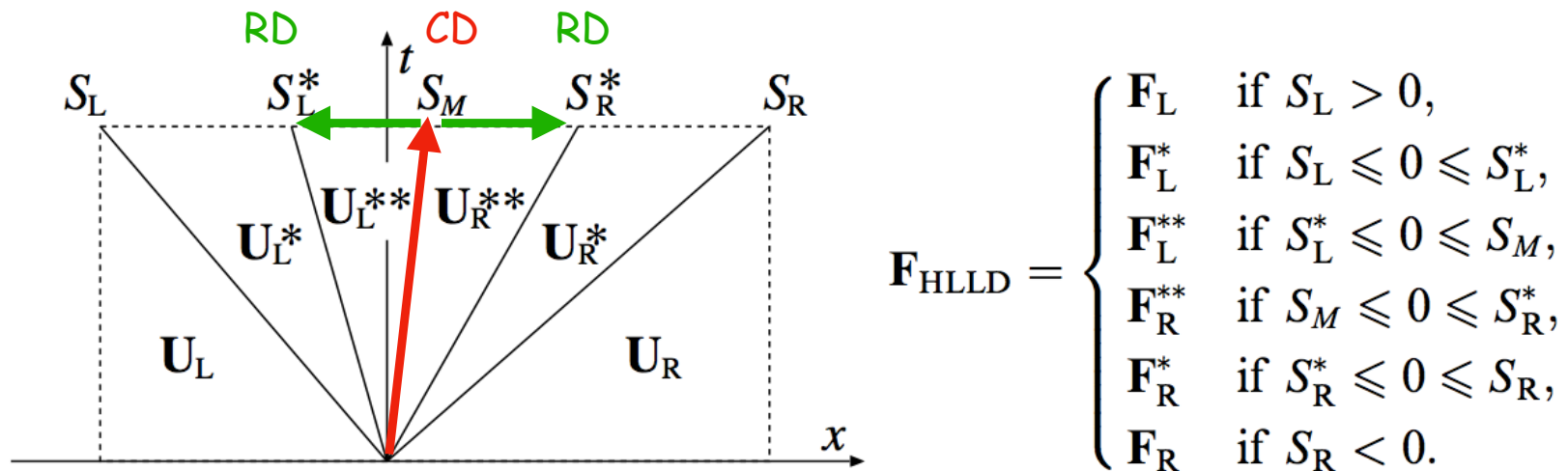
- Four intermediate states, separated by MHD discontinuities
- Note: there can be six states in the MHD

$$\mathbf{F}_{\text{HLLD}} = \begin{cases} \mathbf{F}_L & \text{if } S_L > 0, \\ \mathbf{F}_L^* & \text{if } S_L \leq 0 \leq S_L^*, \\ \mathbf{F}_L^{**} & \text{if } S_L^* \leq 0 \leq S_M, \\ \mathbf{F}_R^{**} & \text{if } S_M \leq 0 \leq S_R^*, \\ \mathbf{F}_R^* & \text{if } S_R^* \leq 0 \leq S_R, \\ \mathbf{F}_R & \text{if } S_R < 0. \end{cases}$$



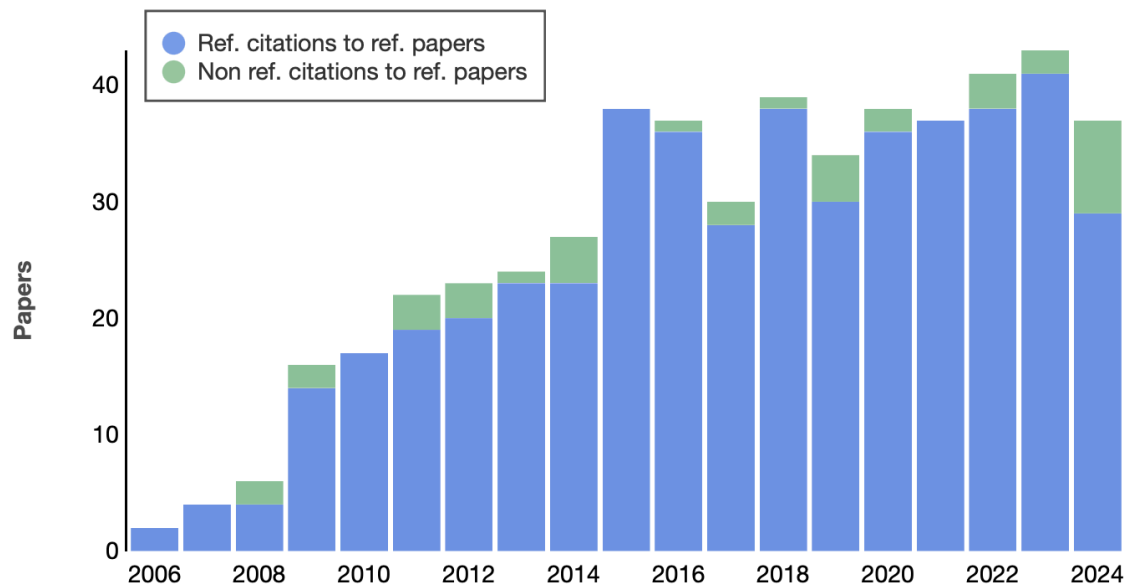
HLLD solver (2/3) - key points

- 1. We first derive **the entropy wave speed (S_M)**
- 2. Then we assume two rotational discontinuities (**RDs**), which propagate outwards **at local Alfvén speeds**
- 3. We consider the conservation laws across all the boundaries
- 4. We calculate an appropriate numerical flux F_{HLLD}



HLLD solver (3/3) - some more

- HLLD solver is a de-facto standard MHD solver
- If you want to be an MHD expert, I highly recommend you to read the original paper (Miyoshi & Kusano 2005, JCP).
- Fortran/C/Python source files are available.



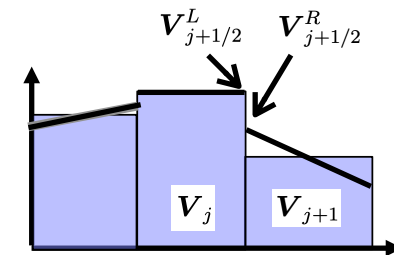
<https://ui.adsabs.harvard.edu/abs/2005JCoPh.208..315M>

4. MHD simulation with Riemann solver

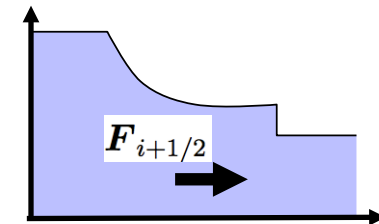
Simulation cycle

- 1. Spatial interpolation

Compute left and right states: V_L, V_R



- 2. Calculate numerical flux F ,
considering a Riemann problem



- 3. Update conservative variable U

$$U_i(t + \Delta t) = U_i(t) - \frac{\Delta t}{\Delta x} (F_{i+1/2} - F_{i-1/2})$$

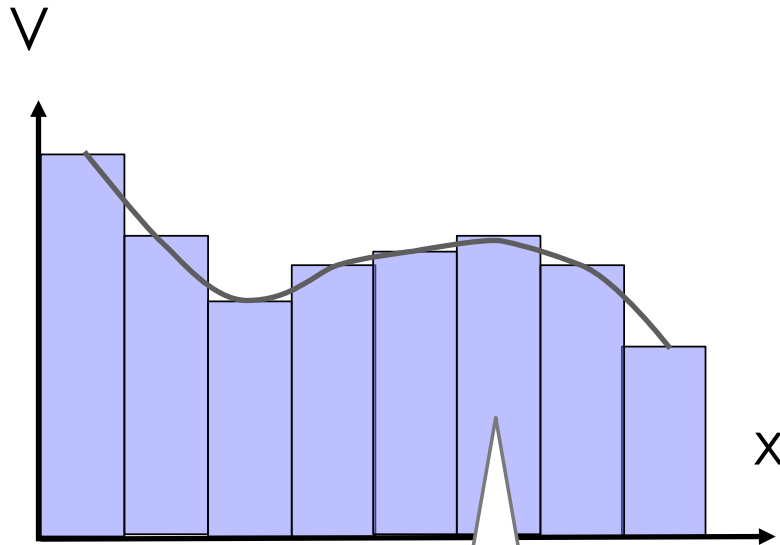
- 4. Recovery of primitive variables

Convert U to V

$$U \equiv \left(\rho \quad \rho v \quad \frac{1}{2} \rho v^2 + \frac{p}{\gamma - 1} + \frac{B^2}{8\pi} \quad B \right)^T$$

➔ $V \equiv (\rho \quad v \quad p \quad B)^T$

STEP 1/4: Spatial interpolation



- Left and right states of the cell boundary

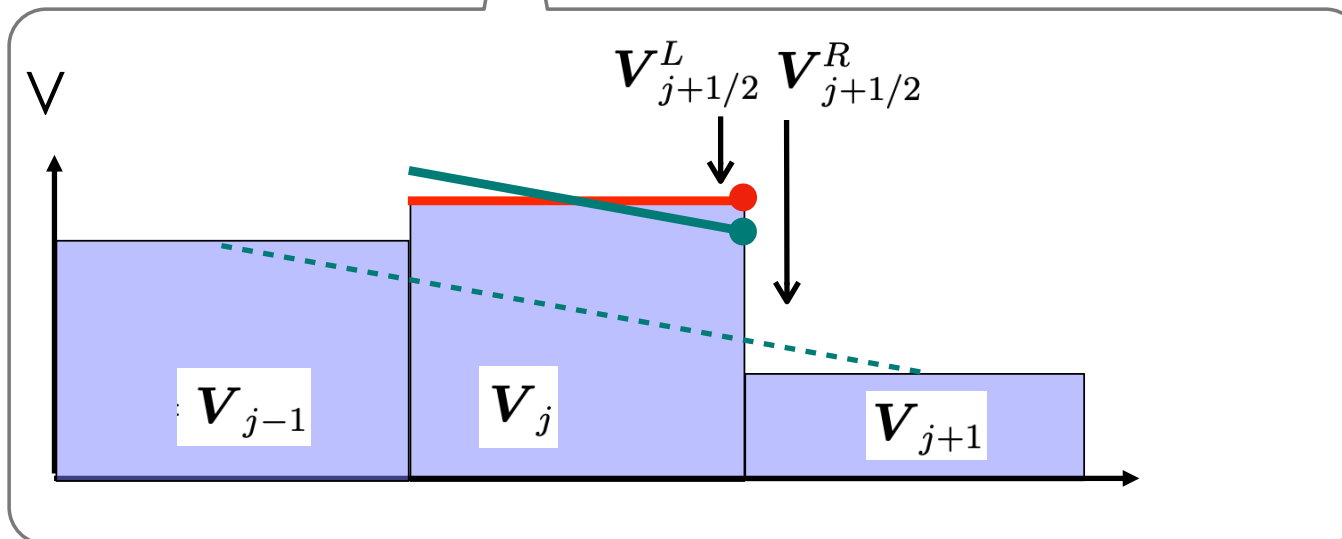
$$V_{j+1/2}^L \quad V_{j+1/2}^R$$

- They are interpolated by...

$$V_{j+1/2}^L = V_j \quad \text{No interpolation}$$

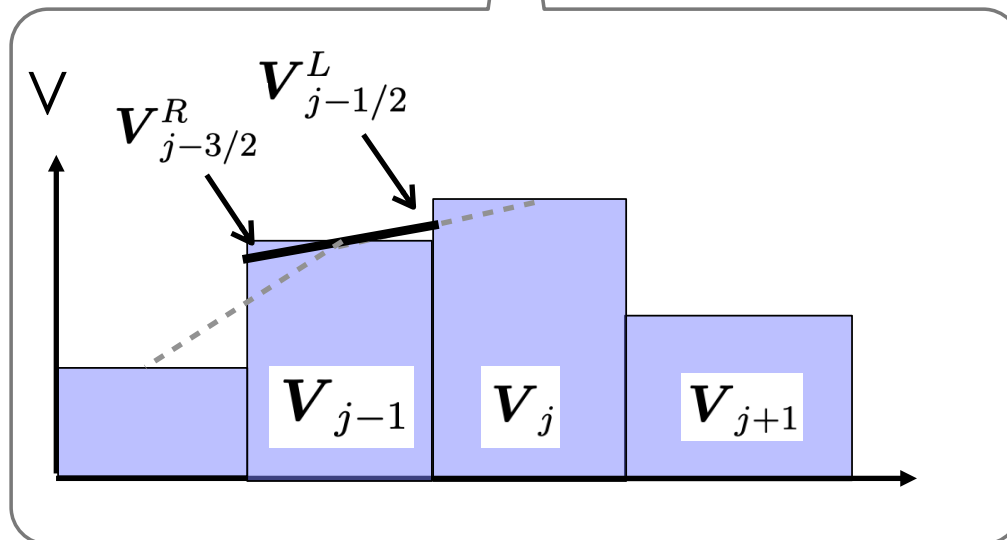
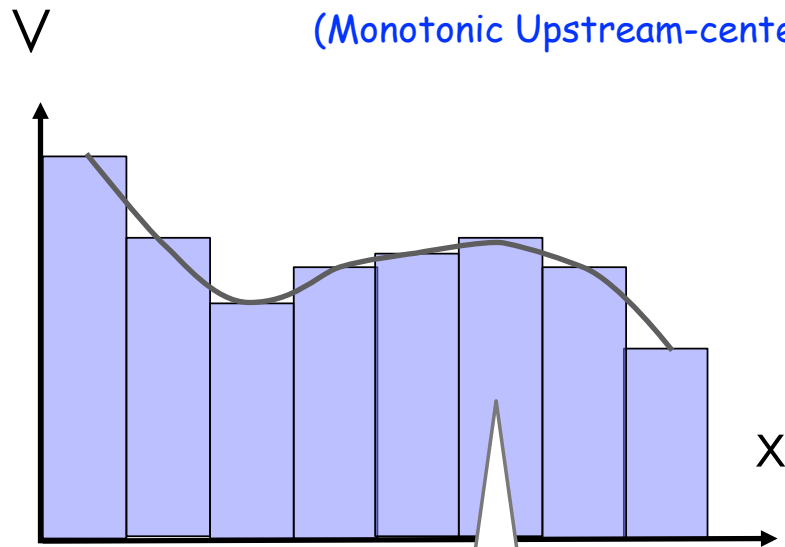
$$V_{j+1/2}^R = V_j + \frac{1}{4}(V_{j+1} - V_{j-1})$$

2nd-order
interpolation



MUSCL interpolation

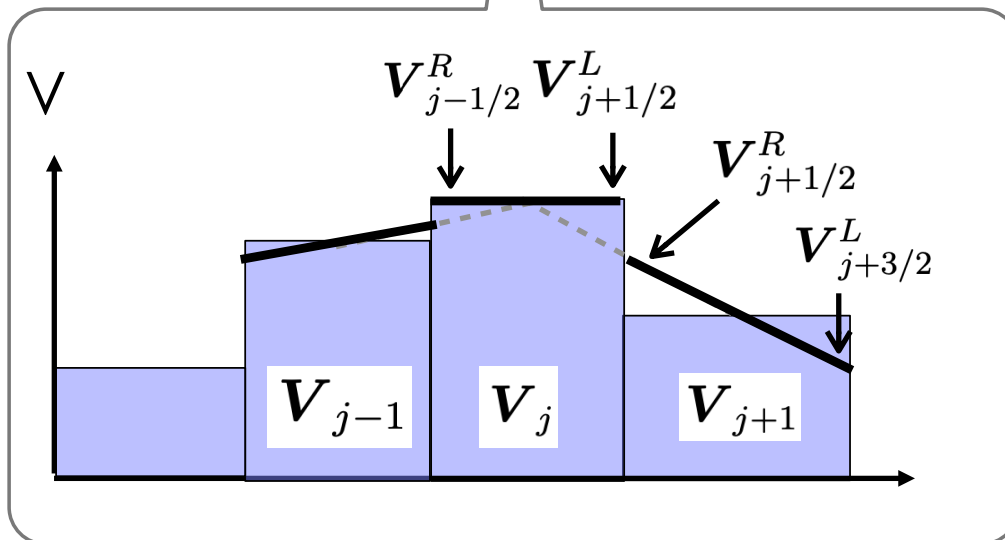
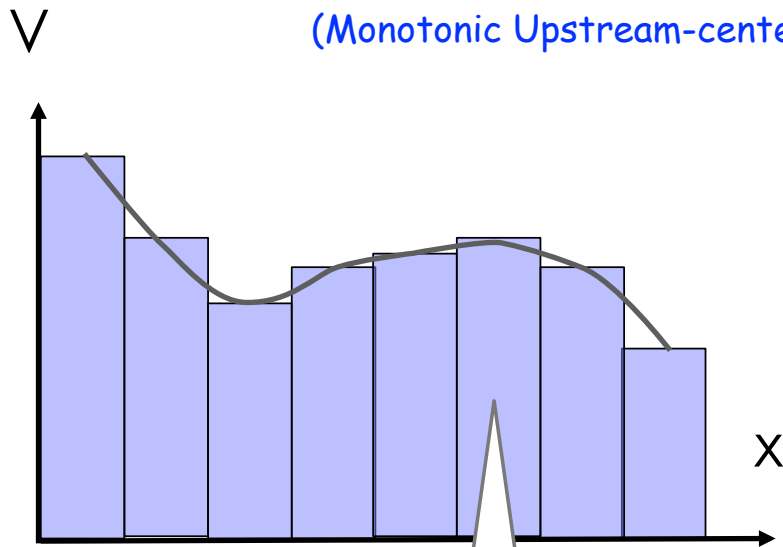
(Monotonic Upstream-centered Scheme for Conservation Laws)



- TVD (total variation diminishing) method
- minmod limiter
 - employ a smaller slope
 - employ a zero slope

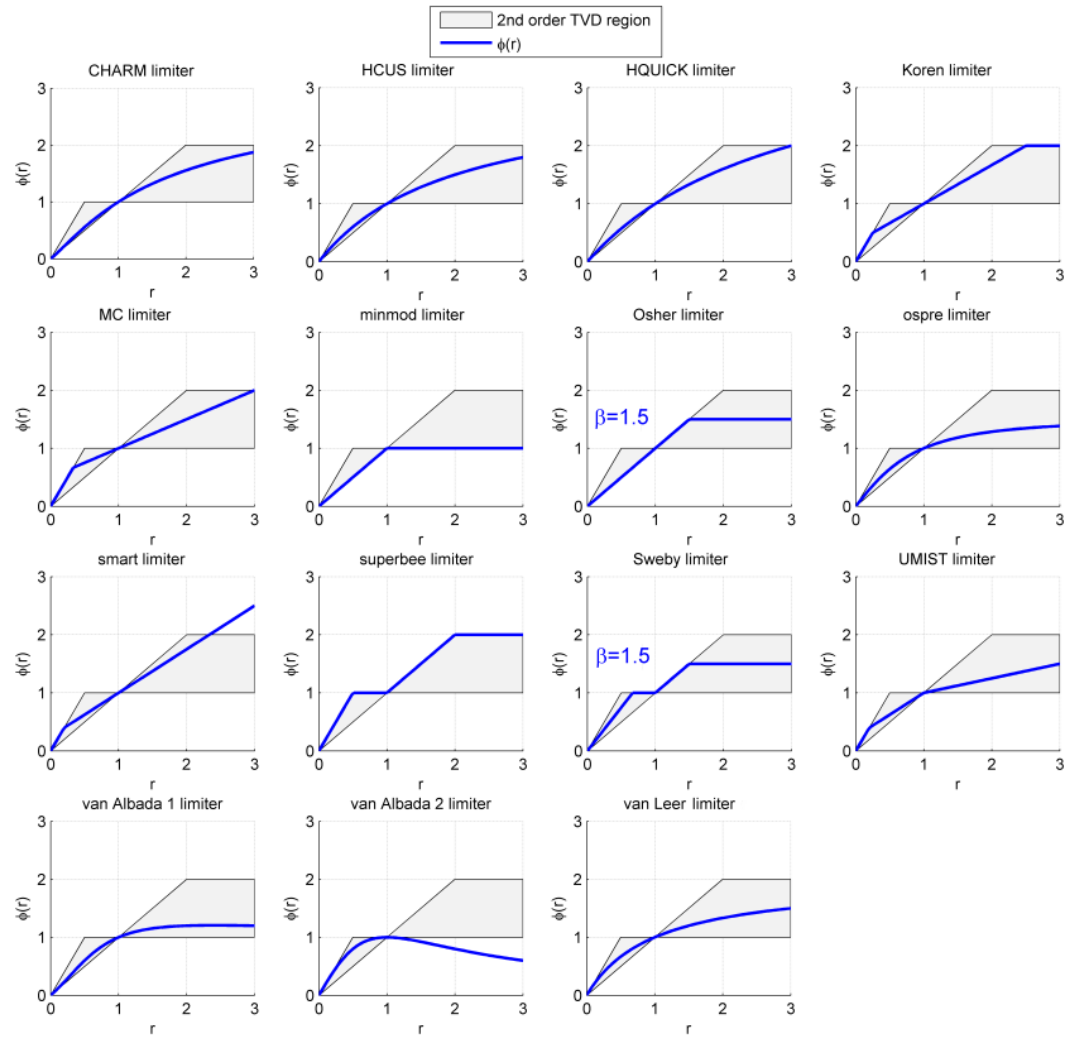
MUSCL interpolation

(Monotonic Upstream-centered Scheme for Conservation Laws)



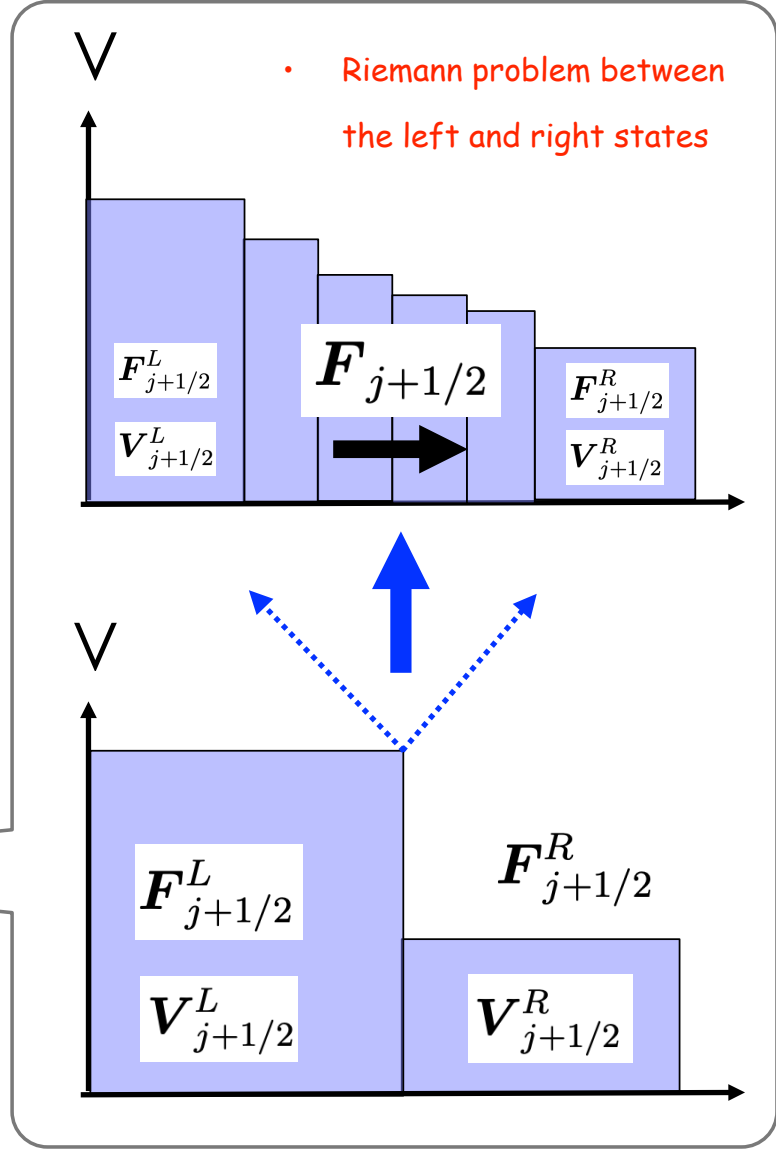
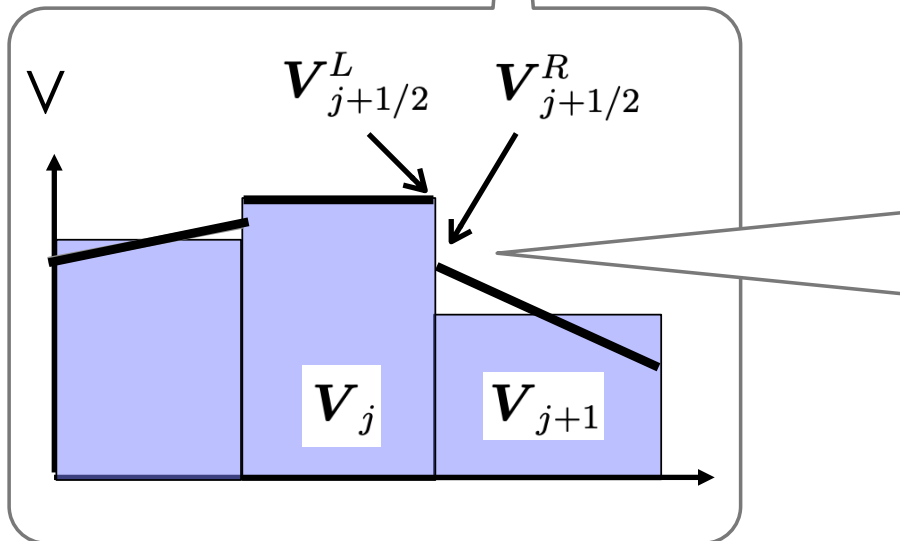
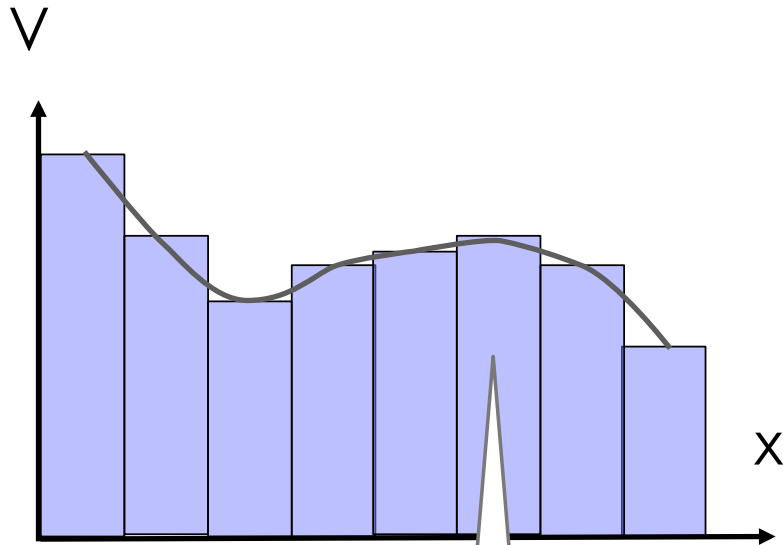
- TVD (total variation diminishing) method
- minmod limiter
 - employ a smaller slope
 - employ a zero slope

Various slope limiters

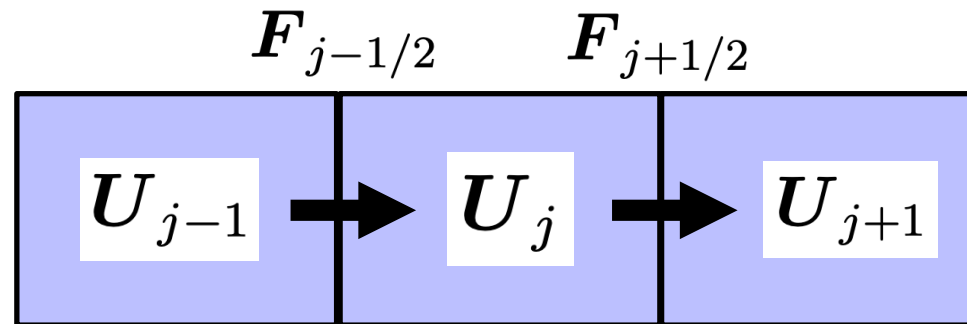


https://en.wikipedia.org/wiki/Flux_limiter

STEP 2/4: Numerical flux



STEP 3/4: Update U (1st order in time)



$$U_j(t + \Delta t) = U_j(t) - \frac{\Delta t}{\Delta x} (F_{j+1/2} - F_{j-1/2})$$

Higher-order evolution in time

- Time evolution (1st order)

$$\begin{aligned}U_j^{n+1} &= U_j^n - \frac{\Delta t}{\Delta x} (\mathbf{F}_{j+1/2}^n - \mathbf{F}_{j-1/2}^n) \\ &\equiv U_j^n - \Delta t \mathcal{L}(U_j^n)\end{aligned}$$

- 2nd-order SSP Runge-Kutta (also known as TVD Runge-Kutta)

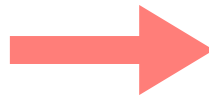
$$\begin{aligned}U_j^* &= U_j^n - \Delta t \mathcal{L}(U_j^n) \\ U_j^{n+1} &= \frac{1}{2} (U_j^n) + \frac{1}{2} \{U_j^* - \Delta t \mathcal{L}(U_j^*)\}\end{aligned}$$

- 3rd-order SSP Runge-Kutta is also popular

STEP 4/4: Recovery of primitive variables

Conserved variables

$$\mathbf{U} \equiv (\rho \quad \rho \mathbf{v} \quad \frac{1}{2} \rho v^2 + \frac{p}{\gamma - 1} + \frac{B^2}{8\pi} \quad \mathbf{B})^T$$



Primitive variables

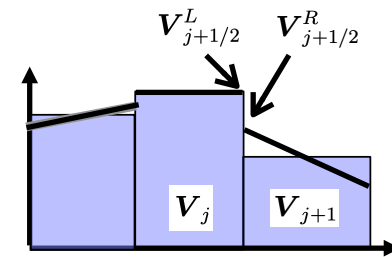
$$\mathbf{V} \equiv (\rho \quad \mathbf{v} \quad p \quad \mathbf{B})^T$$

- When the pressure becomes negative ($p < 0$), we have to stop the simulation.
- Serious problem for finite-volume MHD codes in a low-beta plasma ($\beta \ll 0.1$).

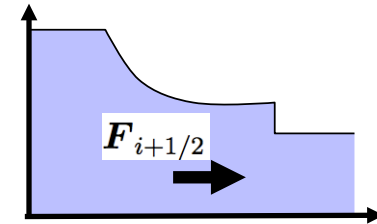
Simulation cycle, again

- 1. Spatial interpolation

Compute left and right states: V_L, V_R



- 2. Calculate numerical flux F ,
considering a Riemann problem



- 3. Update conservative variable U

$$U_i(t + \Delta t) = U_i(t) - \frac{\Delta t}{\Delta x} (F_{i+1/2} - F_{i-1/2})$$

- 4. Recovery of primitive variables

Convert U to V

$$U \equiv \left(\rho \quad \rho v \quad \frac{1}{2} \rho v^2 + \frac{p}{\gamma - 1} + \frac{B^2}{8\pi} \quad B \right)^T$$

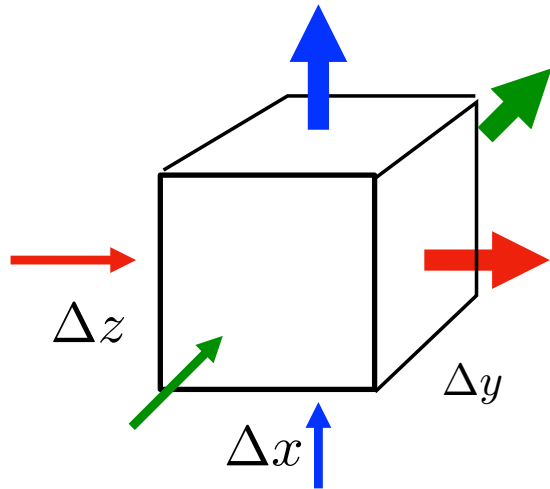
➔ $V \equiv (\rho \quad v \quad p \quad B)^T$

5. MHD simulation in multi-dimensions

Finite volume method in multi-dimensions

$$\frac{\partial}{\partial t} \mathbf{U} + \frac{\partial}{\partial x} \mathbf{F} + \frac{\partial}{\partial y} \mathbf{G} + \frac{\partial}{\partial z} \mathbf{H} = 0$$

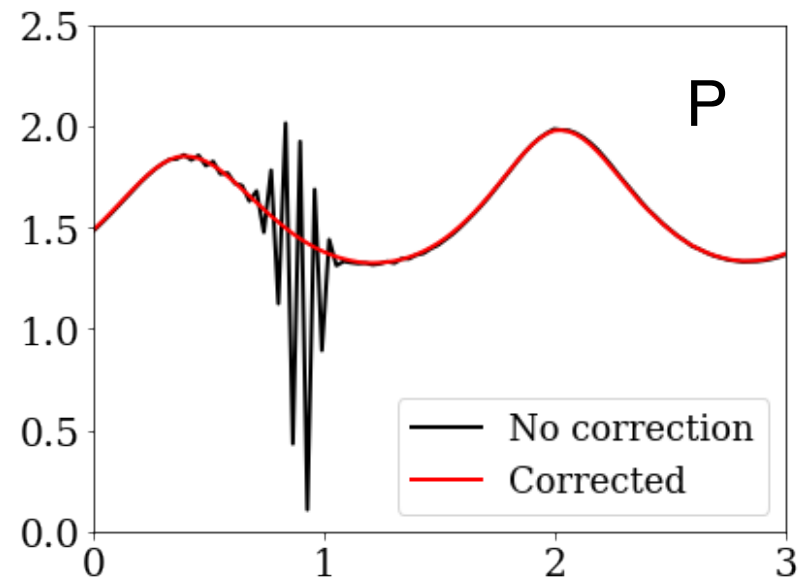
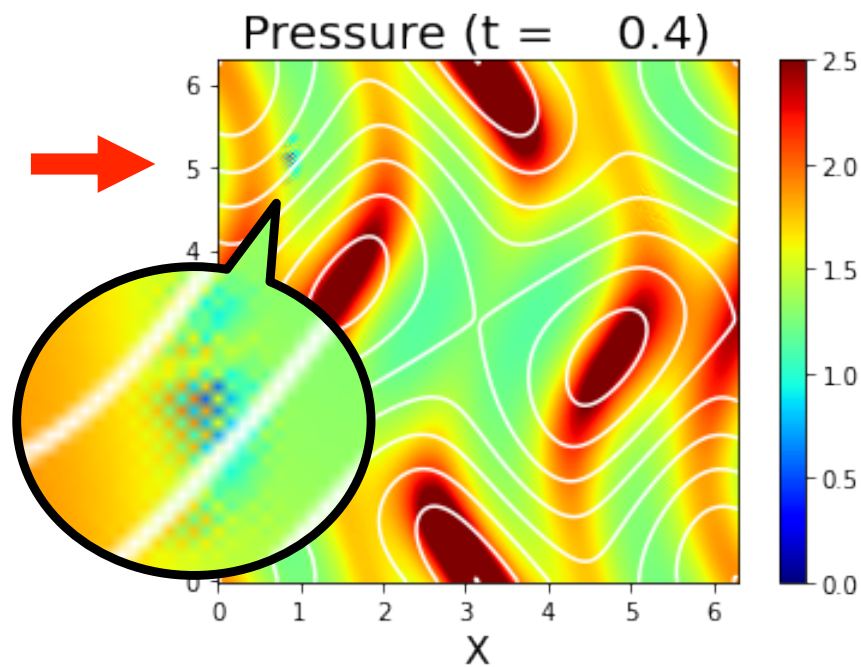
$$\begin{aligned} U_{i,j,k}(t + \Delta t) = & U_{i,j,k}(t) - \frac{\Delta t}{\Delta x} (F_{i+1/2,j,k} - F_{i-1/2,j,k}) \\ & - \frac{\Delta t}{\Delta y} (G_{i,j+1/2,k} - G_{i,j-1/2,k}) \\ & - \frac{\Delta t}{\Delta z} (H_{i,j,k+1/2} - H_{i,j,k-1/2}) \end{aligned}$$



• Is that all? NO

Numerical oscillation in 2D/3D MHD

- MHD simulation often suffers from numerical oscillations, caused by divergence B
- Some correction is necessary



Unphysical force by div.B

- Gauss's law

$$\nabla \cdot \mathbf{E} = 4\pi\rho_c$$

- Newton-Lorentz force

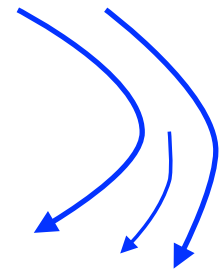
$$\rho_c \left(\mathbf{E} + \frac{\mathbf{v}}{c} \times \mathbf{B} \right) = \frac{\nabla \cdot \mathbf{E}}{4\pi} \left(\mathbf{E} + \frac{\mathbf{v}}{c} \times \mathbf{B} \right)$$

- Numerical magnetic charge

$$\nabla \cdot \mathbf{B} \neq 0$$

- Unphysical force starts to work

$$\frac{\nabla \cdot \mathbf{B}}{4\pi} \left(\mathbf{B} - \frac{\mathbf{v}}{c} \times \mathbf{E} \right)$$



- "Divergence cleaning": we need to keep div B to small or zero

Workaround 1: Projection method

- Helmholtz decomposition for arbitrary vector field

$$\mathbf{u} = \nabla\phi + \nabla \times \mathbf{A}$$

- We assume $\mathbf{B} = \nabla\phi_{\text{err}} + \nabla \times \mathbf{A}$

- Solve a Poisson equation (SOR method, FFT ...) to obtain ϕ_{err}

$$\nabla \cdot \mathbf{B}_{\text{sim}} = \nabla \cdot \nabla\phi_{\text{err}} + \nabla \cdot (\nabla \times \mathbf{A}) = \Delta\phi_{\text{err}}$$

- Correct the magnetic field

$$\mathbf{B}_{\text{new}} = \mathbf{B}_{\text{sim}} - \nabla\phi_{\text{err}}$$

- Then the new magnetic field satisfies

$$\nabla \cdot \mathbf{B}_{\text{new}} = 0$$

Workaround 2: Constraint Transport (CT)

- Assign the magnetic field on a cell center
- There are several variants

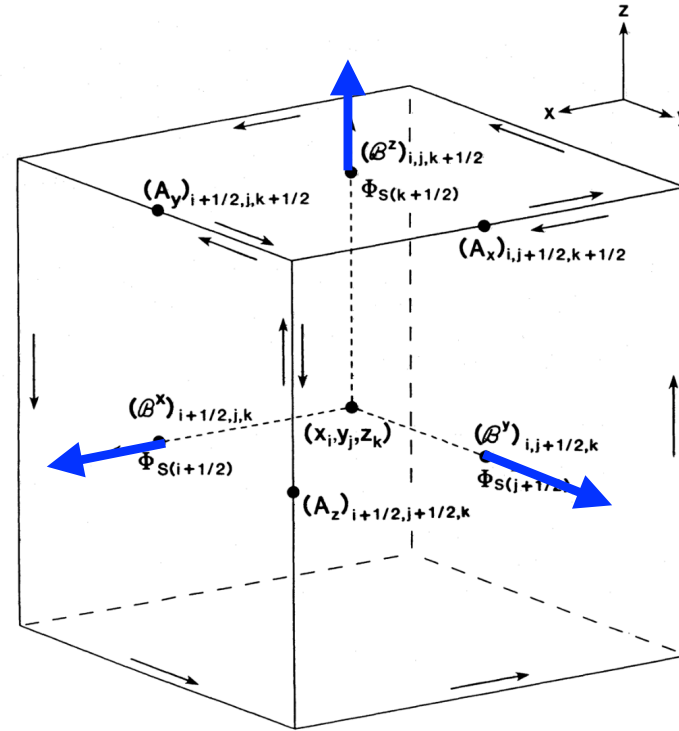
$$B_{x,i-1/2,j,k}^{n+1} = B_{x,i-1/2,j,k}^n - \frac{\Delta t}{\Delta y} \left(E_{z,i-1/2,j+1/2,k}^{n+1/2} - E_{z,i-1/2,j,k}^{n+1/2} \right) + \frac{\Delta t}{\Delta z} \left(E_{y,i-1/2,j,k+1/2}^{n+1/2} - E_{y,i-1/2,j,k-1/2}^{n+1/2} \right)$$

$$B_{y,i,j-1/2,k}^{n+1} = \dots$$

$$B_{z,i,j,k-1/2}^{n+1} = \dots$$

$$(\nabla \cdot \mathbf{B})_{i,j,k}^{n+1} = \frac{B_{x,i+1/2,j,k}^{n+1} - B_{x,i-1/2,j,k}^{n+1}}{\Delta x} + \frac{B_{x,i,j+1/2,k}^{n+1} - B_{x,i,j-1/2,k}^{n+1}}{\Delta y} + \frac{B_{x,i,j,k+1/2}^{n+1} - B_{x,i,j,k-1/2}^{n+1}}{\Delta z}$$

$$= (\nabla \cdot \mathbf{B})_{i,j,k}^n$$



Workaround 3: Hyperbolic divergence cleaning

- A virtual potential ψ is introduced

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{v}) = 0$$

$$\frac{\partial \rho \mathbf{v}}{\partial t} + \nabla \cdot (\rho \mathbf{v} \mathbf{v} + p_t \mathbb{I} - \mathbf{B} \mathbf{B}) = 0$$

$$\frac{\partial \mathcal{E}}{\partial t} + \nabla \cdot \left((\mathcal{E} + p_t) \mathbf{v} - (\mathbf{v} \cdot \mathbf{B}) \mathbf{B} \right) = 0$$

$$\frac{\partial \mathbf{B}}{\partial t} + \nabla \cdot (\mathbf{v} \mathbf{B} - \mathbf{B} \mathbf{v}) + \nabla \psi = 0$$

$$\frac{\partial \psi}{\partial t} + c_h^2 \nabla \cdot \mathbf{B} = - \left(\frac{c_h^2}{c_p^2} \right) \psi$$

- Div \mathbf{B} is temporally stored to ψ .
- Then ψ tries to adjust \mathbf{B} , diffuse itself, and decay.

Almost done!

1. MHD at a glance
2. Basic theory: Advection problem
3. Basic theory: Finite-volume method and Riemann solver
4. MHD simulation with Riemann solver
5. MHD simulation in multi-dimensions
6. Hands on

Further reading

- Plasma Physics for Astrophysics, R. M. Kulsrud (2004)
- Magnetohydrodynamics of the Sun, E. R. Priest (2014)
- A multi-state HLL approximate Riemann solver for ideal magnetohydrodynamics, Miyoshi & Kusano, J. Comput. Phys. (2005)
- The $\text{div } \mathbf{B} = 0$ Constraint in Shock-Capturing Magnetohydrodynamics Codes, Toth, J. Comput. Phys. (2000)
- Riemann Solvers and Numerical Methods for Fluid Dynamics: A Practical Introduction, E. F. Toro (2010)

Public MHD codes

- **Athena++ (Princeton)**
 - <https://www.athena-astro.app/>
 - C++, MPI+OpenMP, General relativistic MHD
 - **Pluto (A. Mignone)**
 - <https://plutocode.ph.unito.it/>
 - C, MPI, Relativistic MHD
 - **MURaM (Max Planck, U. Chicago)**
 - https://www2.mps.mpg.de/projects/solar-mhd/muram_site/
 - Fortran 90, MPI+OpenACC, Radiative MHD
 - **CANS+ (Y. Matsumoto)**
 - <http://www.astro.phys.s.chiba-u.ac.jp/cans/doc/> (in Japanese)
 - Fortran 90, MPI+OpenMP, Python, IDL visualization
 - **OpenMHD**
 - <https://sci.nao.ac.jp/MEMBER/zenitani/openmhd-e.html>
 - Fortran 90 and CUDA Fortran, MPI+OpenMP, Python, IDL visualization
- I recommend you to find a well-tested MHD code, written by your favorite language (C/C++ or Fortran).

Backup slides

NOTE: Units

Ohm's law

Alfvén speed

- cgs units

$$\mathbf{E} + \frac{\mathbf{v}}{c} \times \mathbf{B} = 0,$$

$$c_A = \frac{B}{\sqrt{4\pi\rho}}$$

- Simulation units

$$\mathbf{E} + \mathbf{v} \times \mathbf{B} = 0,$$

$$c_A = \frac{B}{\sqrt{\rho}}$$

- MKS units

$$\mathbf{E} + \mathbf{v} \times \mathbf{B} = 0,$$

$$c_A = \frac{B}{\sqrt{\mu_0\rho}}$$

Lorentz force = magnetic pressure + magnetic tension

$$\rho \frac{d\mathbf{v}}{dt} = -\nabla p + \frac{\mathbf{j} \times \mathbf{B}}{c}$$

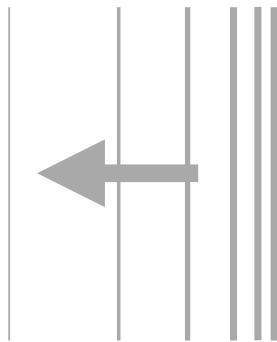
Ampere's law

$$\nabla \times \mathbf{B} = \frac{4\pi}{c} \mathbf{j} + \cancel{\frac{1}{c} \frac{\partial \mathbf{E}}{\partial t}}$$

$$= -\nabla p - \nabla \left(\frac{B^2}{8\pi} \right) + \frac{1}{4\pi} (\mathbf{B} \cdot \nabla) \mathbf{B}$$

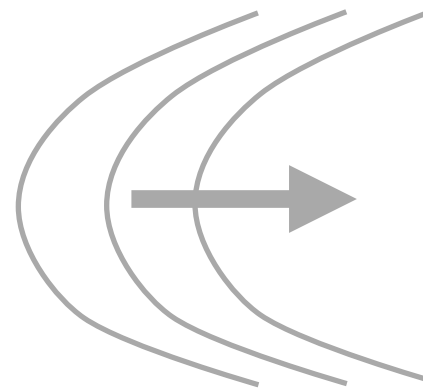
• Examples:

• $\mathbf{B} = (0, x, 0)$



"magnetic pressure"

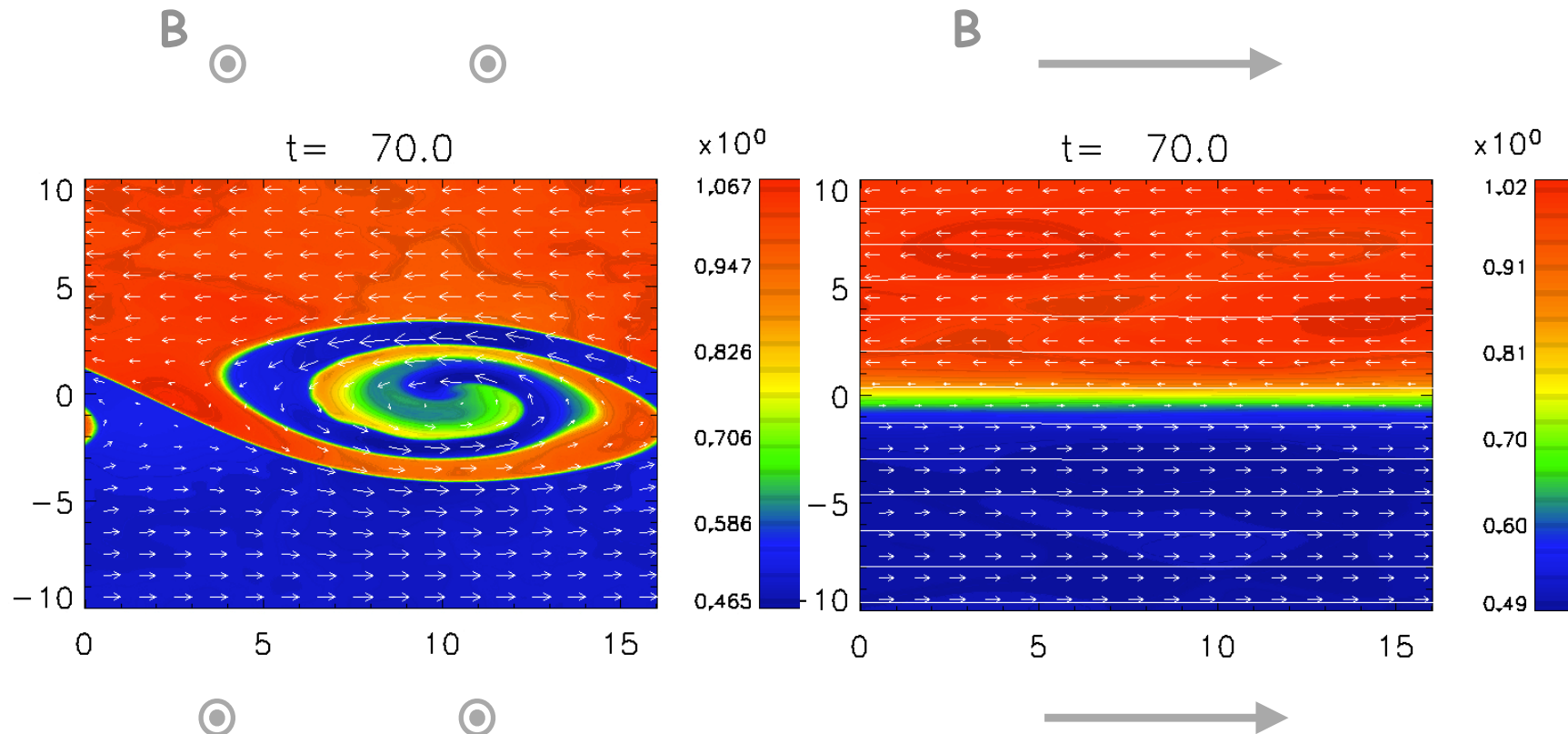
$\mathbf{B} = (y, 1, 0)$



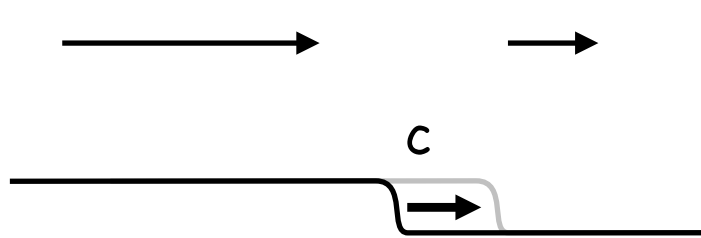
"magnetic tension"

How magnetic tension works ...

- Kelvin-Helmholtz instability in a flow-shear region
- Magnetic field lines tend to be straight



FTCS method vs Upwind method



$$\frac{\partial u}{\partial t} + c \frac{\partial u}{\partial x} = 0$$

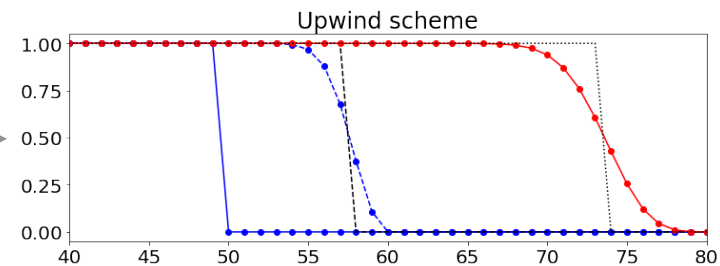
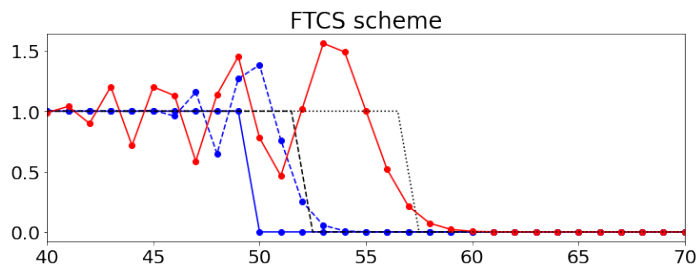
$$f = cu$$

- Equivalent to adding a second-order derivative (diffusion term)

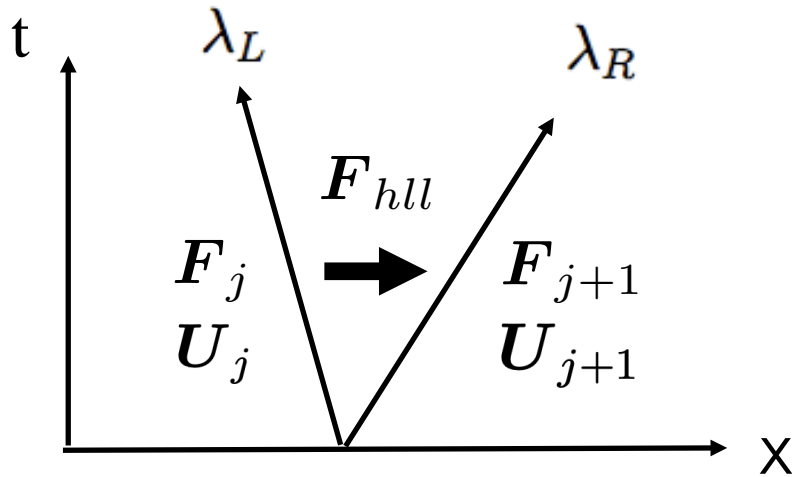
$$u_j^{n+1} = u_j^n - \frac{c\Delta t}{2\Delta x} (u_{j+1}^n - u_{j-1}^n) + \frac{|c|\Delta t\Delta x}{2} \frac{(u_{j+1}^n - 2u_j^n + u_{j-1}^n)}{\Delta x^2}$$

FTCS solver
Diffusion term

 $\approx \frac{\partial^2 u}{\partial x^2}$

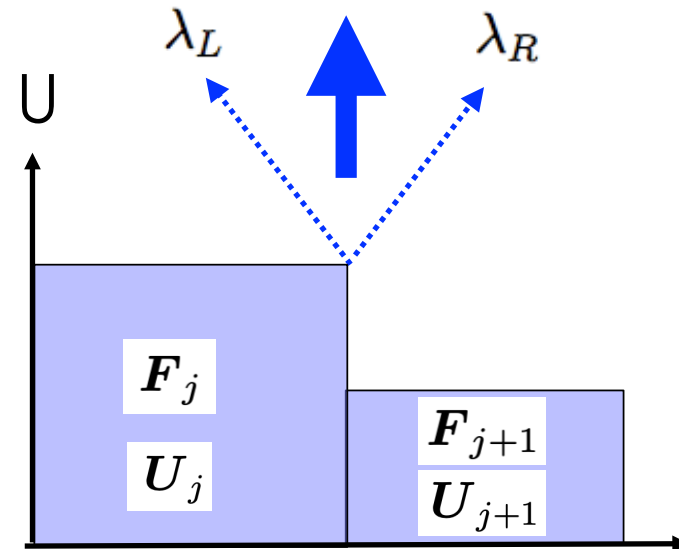
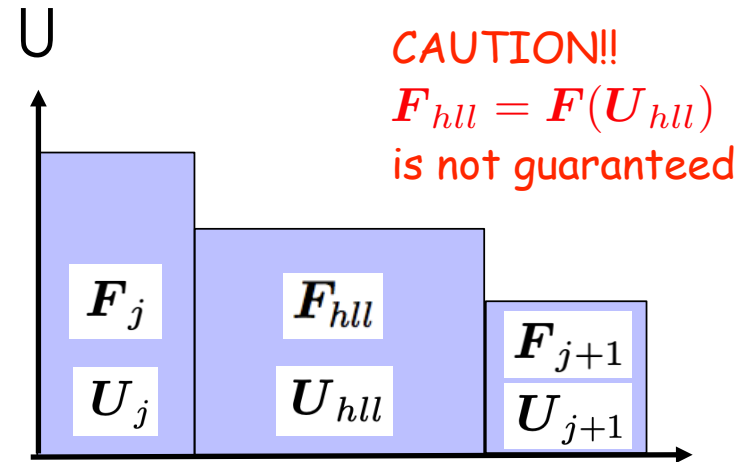


Approximate Riemann solver - HLL solver

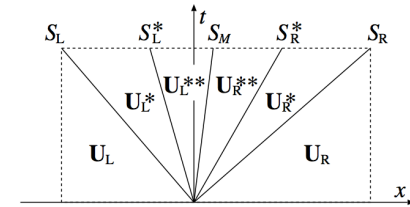


$$\mathbf{F} = \begin{cases} \mathbf{F}_j & (\lambda_L > 0) \\ \mathbf{F}_{hll} & (\lambda_L \leq 0 \leq \lambda_R) \\ \mathbf{F}_{j+1} & (\lambda_R < 0) \end{cases}$$

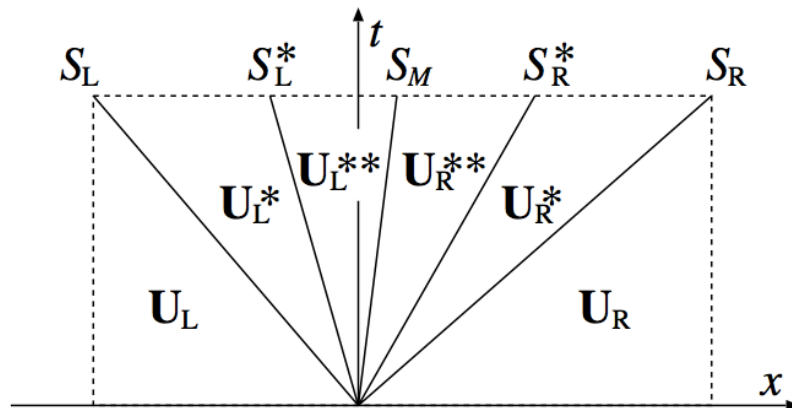
$$\mathbf{F}_{hll} = \frac{\lambda_R \mathbf{F}_j - \lambda_L \mathbf{F}_{j+1} + \lambda_R \lambda_L (\mathbf{U}_{j+1} - \mathbf{U}_j)}{\lambda_R - \lambda_L}$$



HLLD solver in detail [1/24]



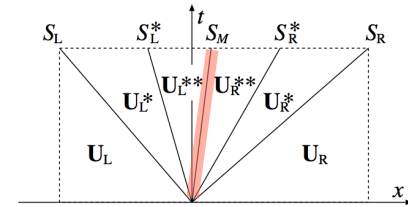
- In the next 24 slides, I show the derivation of the HLLD solution.
- The presentation materials are provided by Dr. Miyoshi, who developed the HLLD scheme.



$$\frac{\partial U}{\partial t} + \frac{\partial F}{\partial x} = 0, \quad B_x = \text{const.},$$

$$U = \begin{pmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ B_y \\ B_z \\ e \end{pmatrix}, \quad F = \begin{pmatrix} \rho u \\ \rho u u + p_T - B_x^2 \\ \rho v u - B_x B_y \\ \rho w u - B_x B_z \\ B_y u - B_x v \\ B_z u - B_x w \\ (e + p_T)u - B_x (u B_x + v B_y + w B_z) \end{pmatrix}$$

HLLD solver [2/24]



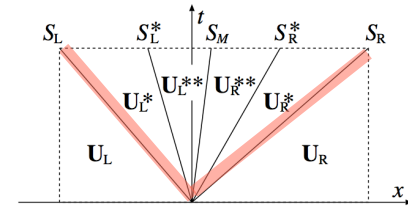
- Evaluating an entropy wave speed (Batterson+ 1997)

$$S_M = \frac{(\rho u)^*}{\rho^*} = \frac{(S_R - u_R)\rho_R u_R - (S_L - u_L)\rho_L u_L - p_{TR} + p_{TL}}{(S_R - u_R)\rho_R - (S_L - u_L)\rho_L}$$

- Evaluating the total pressure

$$\begin{aligned} p_T^* &= p_{TL} + \rho_L (S_L - u_L)(S_M - u_L) \\ &= p_{TR} + \rho_R (S_R - u_R)(S_M - u_R) \\ &= \frac{(S_R - u_R)\rho_R p_{TL} - (S_L - u_L)\rho_L p_{TR} + \rho_L \rho_R (S_R - u_R)(u_R - u_L)}{(S_R - u_R)\rho_R - (S_L - u_L)\rho_L} \end{aligned}$$

HLLD solver [3/24]

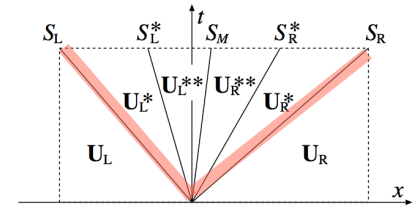


- Jump conditions across the left/right boundaries

$$\begin{aligned}
 & \rightarrow S_\alpha \begin{pmatrix} \rho_\alpha^* \\ \rho_\alpha^* S_M \\ \rho_\alpha^* v_\alpha^* \\ \rho_\alpha^* w_\alpha^* \\ B_{y\alpha}^* \\ B_{z\alpha}^* \\ e_\alpha^* \end{pmatrix} - \begin{pmatrix} \rho_\alpha^* S_M \\ \rho_\alpha^* S_M^2 + p_T^* - B_x^2 \\ \rho_\alpha^* v_\alpha^* S_M - B_x B_{y\alpha}^* \\ \rho_\alpha^* w_\alpha^* S_M - B_x B_{z\alpha}^* \\ B_{y\alpha}^* S_M - B_x v_\alpha^* \\ B_{z\alpha}^* S_M - B_x w_\alpha^* \\ (e_\alpha^* + p_T^*) S_M - B_x (v_\alpha^* \cdot B_\alpha^*) \end{pmatrix} = S_\alpha \begin{pmatrix} \rho_\alpha \\ \rho_\alpha u_\alpha \\ \rho_\alpha v_\alpha \\ \rho_\alpha w_\alpha \\ B_{y\alpha} \\ B_{z\alpha} \\ e_\alpha \end{pmatrix} - \begin{pmatrix} \rho_\alpha u_\alpha \\ \rho_\alpha u_\alpha + p_{T\alpha} - B_x^2 \\ \rho_\alpha v_\alpha u_\alpha - B_x B_{y\alpha} \\ \rho_\alpha w_\alpha u_\alpha - B_x B_{z\alpha} \\ B_{y\alpha} u_\alpha - B_x v_\alpha \\ B_{z\alpha} u_\alpha - B_x w_\alpha \\ (e_\alpha + p_{T\alpha}) u_\alpha - B_x (v_\alpha \cdot B_\alpha) \end{pmatrix}
 \end{aligned}$$

$$\mathbf{v}_\alpha^* = (S_M, v_\alpha^*, w_\alpha^*), \mathbf{B}_\alpha^* = (B_x, B_{y\alpha}^*, B_{z\alpha}^*), \alpha = R, L$$

HLLD solver [4/24]

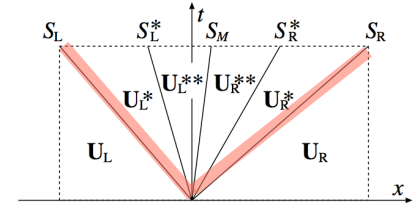


- Solution:

$$U_\alpha^*$$

$$\rho_\alpha^* = \rho_\alpha \frac{S_\alpha - u_\alpha}{S_\alpha - S_M}$$

HLLD solver [5/24]

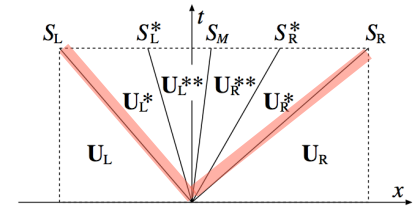


- Jump conditions across the left/right boundaries

$$\begin{array}{c} \rightarrow \\ \rightarrow \\ \rightarrow \\ \rightarrow \end{array}
 \begin{pmatrix} \rho_\alpha^* \\ \rho_\alpha^* S_M \\ \rho_\alpha^* v_\alpha^* \\ \rho_\alpha^* w_\alpha^* \\ B_{y\alpha}^* \\ B_{z\alpha}^* \\ e_\alpha^* \end{pmatrix}
 -
 \begin{pmatrix} \rho_\alpha^* S_M \\ \rho_\alpha^* S_M^2 + p_T^* - B_x^2 \\ \rho_\alpha^* v_\alpha^* S_M - B_x B_{y\alpha}^* \\ \rho_\alpha^* w_\alpha^* S_M - B_x B_{z\alpha}^* \\ B_{y\alpha}^* S_M - B_x v_\alpha^* \\ B_{z\alpha}^* S_M - B_x w_\alpha^* \\ (e_\alpha^* + p_T^*) S_M - B_x (v_\alpha^* \cdot B_\alpha^*) \end{pmatrix}
 = S_\alpha
 \begin{pmatrix} \rho_\alpha \\ \rho_\alpha u_\alpha \\ \rho_\alpha v_\alpha \\ \rho_\alpha w_\alpha \\ B_{y\alpha} \\ B_{z\alpha} \\ e_\alpha \end{pmatrix}
 -
 \begin{pmatrix} \rho_\alpha u_\alpha \\ \rho_\alpha u_\alpha + p_{T\alpha} - B_x^2 \\ \rho_\alpha v_\alpha u_\alpha - B_x B_{y\alpha} \\ \rho_\alpha w_\alpha u_\alpha - B_x B_{z\alpha} \\ B_{y\alpha} u_\alpha - B_x v_\alpha \\ B_{z\alpha} u_\alpha - B_x w_\alpha \\ (e_\alpha + p_{T\alpha}) u_\alpha - B_x (v_\alpha \cdot B_\alpha) \end{pmatrix}$$

$$\mathbf{v}_\alpha^* = (S_M, v_\alpha^*, w_\alpha^*), \mathbf{B}_\alpha^* = (B_x, B_{y\alpha}^*, B_{z\alpha}^*), \alpha = R, L$$

HLLD solver [6/24]



- Solution:

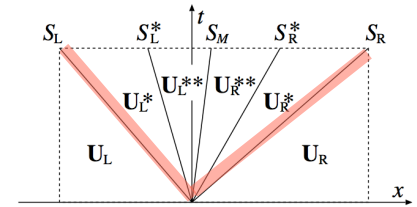
$$U_{\alpha}^*$$

$$\rho_{\alpha}^* = \rho_{\alpha} \frac{S_{\alpha} - u_{\alpha}}{S_{\alpha} - S_M}$$

$$\left\{ \begin{array}{l} \mathbf{v}_{t\alpha}^* = \mathbf{v}_{t\alpha} - B_x \mathbf{B}_{t\alpha} \frac{S_M - u_{\alpha}}{\rho_{\alpha} (S_{\alpha} - u_{\alpha})(S_{\alpha} - S_M) - B_x^2} \\ \mathbf{B}_{t\alpha}^* = \mathbf{B}_{t\alpha} \frac{\rho_{\alpha} (S_{\alpha} - u_{\alpha})^2 - B_x^2}{\rho_{\alpha} (S_{\alpha} - u_{\alpha})(S_{\alpha} - S_M) - B_x^2} \end{array} \right.$$

$$\mathbf{v}_t = (0, v, w), \quad \mathbf{B}_t = (0, B_y, B_z)$$

HLLD solver [7/24]

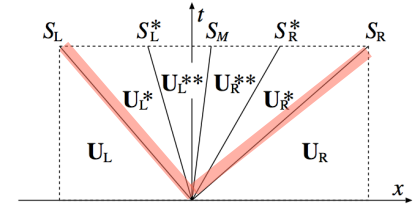


- Jump conditions across the left/right boundaries

$$\begin{array}{c}
 S_\alpha \\
 \left(\begin{array}{c}
 \rho_\alpha^* \\
 \rho_\alpha^* S_M \\
 \rho_\alpha^* v_\alpha^* \\
 \rho_\alpha^* w_\alpha^* \\
 B_{y\alpha}^* \\
 B_{z\alpha}^* \\
 e_\alpha^*
 \end{array} \right) - \left(\begin{array}{c}
 \rho_\alpha^* S_M \\
 \rho_\alpha^* S_M^2 + p_T^* - B_x^2 \\
 \rho_\alpha^* v_\alpha^* S_M - B_x B_{y\alpha}^* \\
 \rho_\alpha^* w_\alpha^* S_M - B_x B_{z\alpha}^* \\
 B_{y\alpha}^* S_M - B_x v_\alpha^* \\
 B_{z\alpha}^* S_M - B_x w_\alpha^* \\
 (e_\alpha^* + p_T^*) S_M - B_x (v_\alpha^* \cdot B_\alpha^*)
 \end{array} \right) = S_\alpha \left(\begin{array}{c}
 \rho_\alpha \\
 \rho_\alpha u_\alpha \\
 \rho_\alpha v_\alpha \\
 \rho_\alpha w_\alpha \\
 B_{y\alpha} \\
 B_{z\alpha} \\
 e_\alpha
 \end{array} \right) - \left(\begin{array}{c}
 \rho_\alpha u_\alpha \\
 \rho_\alpha u_\alpha + p_{T\alpha} - B_x^2 \\
 \rho_\alpha v_\alpha u_\alpha - B_x B_{y\alpha} \\
 \rho_\alpha w_\alpha u_\alpha - B_x B_{z\alpha} \\
 B_{y\alpha} u_\alpha - B_x v_\alpha \\
 B_{z\alpha} u_\alpha - B_x w_\alpha \\
 (e_\alpha + p_{T\alpha}) u_\alpha - B_x (v_\alpha \cdot B_\alpha)
 \end{array} \right)
 \end{array}$$

$$\mathbf{v}_\alpha^* = (S_M, v_\alpha^*, w_\alpha^*), \mathbf{B}_\alpha^* = (B_x, B_{y\alpha}^*, B_{z\alpha}^*) \alpha = R, L$$

HLLD solver [8/24]



- Solution:

$$U_{\alpha}^*$$

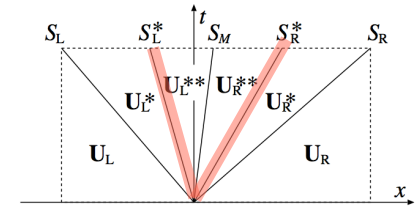
$$\rho_{\alpha}^* = \rho_{\alpha} \frac{S_{\alpha} - u_{\alpha}}{S_{\alpha} - S_M}$$

$$\left\{ \begin{array}{l} \mathbf{v}_{t\alpha}^* = \mathbf{v}_{t\alpha} - B_x \mathbf{B}_{t\alpha} \frac{S_M - u_{\alpha}}{\rho_{\alpha} (S_{\alpha} - u_{\alpha})(S_{\alpha} - S_M) - B_x^2} \\ \mathbf{B}_{t\alpha}^* = \mathbf{B}_{t\alpha} \frac{\rho_{\alpha} (S_{\alpha} - u_{\alpha})^2 - B_x^2}{\rho_{\alpha} (S_{\alpha} - u_{\alpha})(S_{\alpha} - S_M) - B_x^2} \end{array} \right.$$

$$\mathbf{v}_t = (0, v, w), \quad \mathbf{B}_t = (0, B_y, B_z)$$

$$e_{\alpha}^* = \frac{(S_{\alpha} - u_{\alpha}) e_{\alpha} - p_{T\alpha} + p_T^* + B_x (\mathbf{v}_{\alpha} \cdot \mathbf{B}_{\alpha} - \mathbf{v}_{\alpha}^* \cdot \mathbf{B}_{\alpha}^*)}{S_{\alpha} - S_M}$$

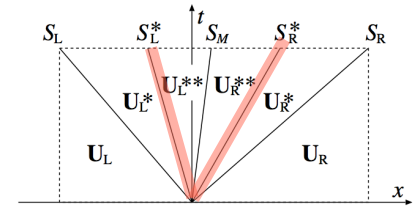
HLLD solver [9/24]



- Jump conditions across the rotational discontinuities

$$\begin{aligned}
 & \left(S_\alpha^* \begin{pmatrix} \rho_\alpha^{**} \\ \rho_\alpha^{**} S_M \\ \rho_\alpha^{**} v_\alpha \\ \rho_\alpha^{**} w_\alpha \\ B_{y\alpha}^{**} \\ B_{z\alpha}^{**} \\ e_\alpha^{**} \end{pmatrix} - \begin{pmatrix} \rho_\alpha^{**} S_M \\ \rho_\alpha^{**} S_M^2 + p_T^* - B_x^2 \\ \rho_\alpha^{**} v_\alpha S_M - B_x B_{y\alpha}^{**} \\ \rho_\alpha^{**} w_\alpha S_M - B_x B_{z\alpha}^{**} \\ B_{y\alpha}^{**} S_M - B_x v_\alpha^{**} \\ B_{z\alpha}^{**} S_M - B_x w_\alpha^{**} \\ (e_\alpha^{**} + p_T^*) S_M - B_x (v_\alpha^{**} \cdot B_\alpha^{**}) \end{pmatrix} \right) = S_\alpha^* \left(\begin{pmatrix} \rho_\alpha^* \\ \rho_\alpha^* S_M \\ \rho_\alpha^* v_\alpha \\ \rho_\alpha^* w_\alpha \\ B_{y\alpha}^* \\ B_{z\alpha}^* \\ e_\alpha^* \end{pmatrix} - \begin{pmatrix} \rho_\alpha^* S_M \\ \rho_\alpha^* S_M^2 + p_T^* - B_x^2 \\ \rho_\alpha^* v_\alpha S_M - B_x B_{y\alpha}^* \\ \rho_\alpha^* w_\alpha S_M - B_x B_{z\alpha}^* \\ B_{y\alpha}^* S_M - B_x v_\alpha^* \\ B_{z\alpha}^* S_M - B_x w_\alpha^* \\ (e_\alpha^* + p_T^*) S_M - B_x (v_\alpha^* \cdot B_\alpha^*) \end{pmatrix} \right)
 \end{aligned}$$

HLLD solver [10/24]

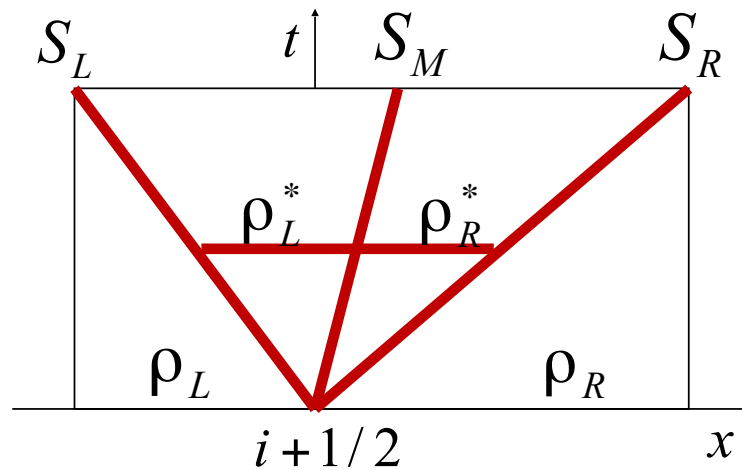


- Solution:

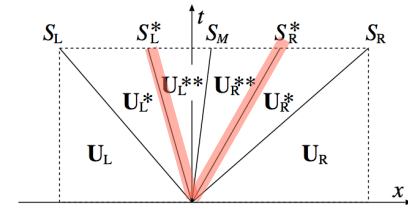
$$U_{\alpha}^{**}$$

$$\rho_{\alpha}^{**} = \rho_{\alpha}^{*}$$

$$S_R^* = S_M + \frac{|B_x|}{\sqrt{\rho_R^*}}, S_L^* = S_M - \frac{|B_x|}{\sqrt{\rho_L^*}}$$



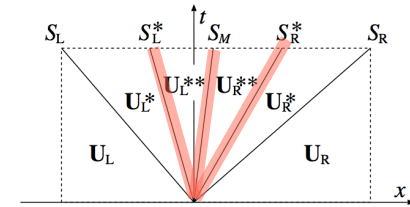
HLLD solver [11/24]



- Jump conditions across the rotational discontinuities

$$\begin{array}{c} \rightarrow \\ \rightarrow \\ \rightarrow \\ \rightarrow \end{array}
 \begin{pmatrix} \rho_\alpha^{**} \\ \rho_\alpha^{**} S_M \\ \rho_\alpha^{**} v_\alpha^{**} \\ \rho_\alpha^{**} w_\alpha^{**} \\ B_{y\alpha}^{**} \\ B_{z\alpha}^{**} \\ e_\alpha^{**} \end{pmatrix}
 - \begin{pmatrix} \rho_\alpha^{**} S_M \\ \rho_\alpha^{**} S_M^2 + p_T^* - B_x^2 \\ \rho_\alpha^{**} v_\alpha^{**} S_M - B_x B_{y\alpha}^{**} \\ \rho_\alpha^{**} w_\alpha^{**} S_M - B_x B_{z\alpha}^{**} \\ B_{y\alpha}^{**} S_M - B_x v_\alpha^{**} \\ B_{z\alpha}^{**} S_M - B_x w_\alpha^{**} \\ (e_\alpha^{**} + p_T^*) S_M - B_x (v_\alpha^{**} \cdot B_\alpha^{**}) \end{pmatrix}
 = S_\alpha^* \begin{pmatrix} \rho_\alpha^* \\ \rho_\alpha^* S_M \\ \rho_\alpha^* v_\alpha^* \\ \rho_\alpha^* w_\alpha^* \\ B_{y\alpha}^* \\ B_{z\alpha}^* \\ e_\alpha^* \end{pmatrix}
 - \begin{pmatrix} \rho_\alpha^* S_M \\ \rho_\alpha^* S_M^2 + p_T^* - B_x^2 \\ \rho_\alpha^* v_\alpha^* S_M - B_x B_{y\alpha}^* \\ \rho_\alpha^* w_\alpha^* S_M - B_x B_{z\alpha}^* \\ B_{y\alpha}^* S_M - B_x v_\alpha^* \\ B_{z\alpha}^* S_M - B_x w_\alpha^* \\ (e_\alpha^* + p_T^*) S_M - B_x (v_\alpha^* \cdot B_\alpha^*) \end{pmatrix}$$

HLLD solver [12/24]



- Jump conditions across the CD (entropy wave)

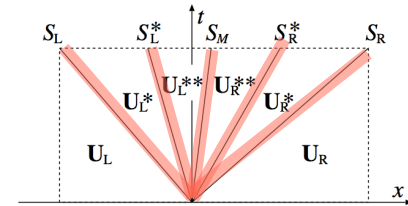
$$\det \left(M \left(\mathbf{v}_{t\alpha}^{**}, \mathbf{B}_{t\alpha}^{**} \right) \right) = 0$$

- Jump conditions across the RDs (Alfvén wave)

$$S_M \begin{pmatrix} \rho_L^* \mathbf{v}_{tL}^{**} \\ \mathbf{B}_{tL}^{**} \end{pmatrix} - \begin{pmatrix} \rho_L^* \mathbf{v}_{tL}^{**} S_M - B_x \mathbf{B}_{tL}^{**} \\ \mathbf{B}_{tL}^{**} S_M - B_x \mathbf{v}_{tL}^{**} \end{pmatrix} = S_M \begin{pmatrix} \rho_R^* \mathbf{v}_{tR}^{**} \\ \mathbf{B}_{tR}^{**} \end{pmatrix} - \begin{pmatrix} \rho_R^* \mathbf{v}_{tR}^{**} S_M - B_x \mathbf{B}_{tR}^{**} \\ \mathbf{B}_{tR}^{**} S_M - B_x \mathbf{v}_{tR}^{**} \end{pmatrix}$$

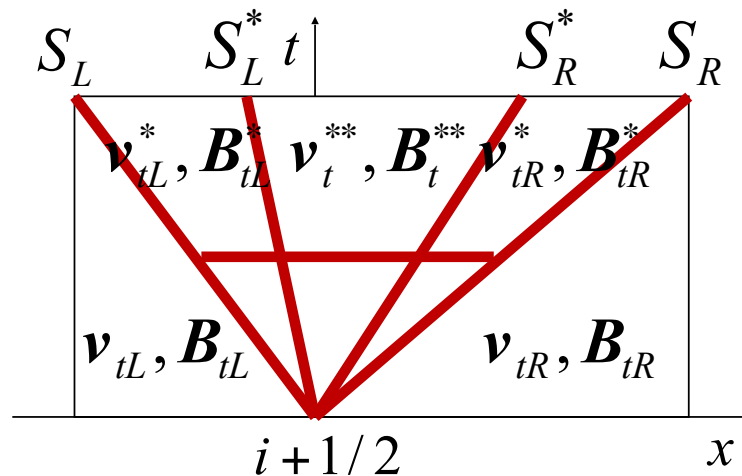
$$\mathbf{v}_{tL}^{**} = \mathbf{v}_{tR}^{**} = \mathbf{v}_t^{**}, \mathbf{B}_{tL}^{**} = \mathbf{B}_{tR}^{**} = \mathbf{B}_t^{**} \quad \text{for } B_x \neq 0$$

HLLD solver [13/24]

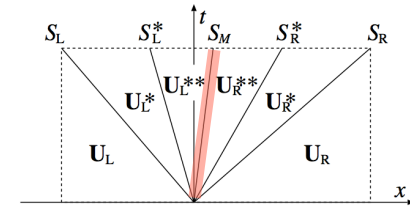


- Jump conditions across the entire intermediate states

$$\begin{aligned}
 & \left(S_R - S_R^* \right) \begin{pmatrix} \rho_R^* \mathbf{v}_{tR}^* \\ \mathbf{B}_{tR}^* \end{pmatrix} + \left(S_R^* - S_M \right) \begin{pmatrix} \rho_R^* \mathbf{v}_t^{**} \\ \mathbf{B}_t^{**} \end{pmatrix} + \left(S_M - S_L^* \right) \begin{pmatrix} \rho_L^* \mathbf{v}_t^{**} \\ \mathbf{B}_t^{**} \end{pmatrix} + \left(S_L^* - S_{RL} \right) \begin{pmatrix} \rho_L^* \mathbf{v}_{tL}^* \\ \mathbf{B}_{tL}^* \end{pmatrix} \\
 & + S_R \begin{pmatrix} \rho_R \mathbf{v}_{tR} \\ \mathbf{B}_{tR} \end{pmatrix} - S_L \begin{pmatrix} \rho_L \mathbf{v}_{tL} \\ \mathbf{B}_{tL} \end{pmatrix} + \begin{pmatrix} \rho_R \mathbf{v}_{tR} u_R - B_x \mathbf{B}_{tR} \\ \mathbf{B}_{tR} u_R - B_x \mathbf{v}_{tR} \end{pmatrix} - \begin{pmatrix} \rho_L \mathbf{v}_{tL} u_L - B_x \mathbf{B}_{tL} \\ \mathbf{B}_{tL} u_L - B_x \mathbf{v}_{tL} \end{pmatrix} = 0
 \end{aligned}$$



HLLD solver [14/24]



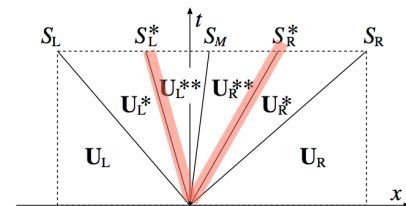
- Solution:

$$U_{\alpha}^{**}$$

$$\rho_{\alpha}^{**} = \rho_{\alpha}^*$$

$$\left\{ \begin{array}{l} \mathbf{v}_t^{**} = \frac{\sqrt{\rho_L^*} \mathbf{v}_{tL}^* + \sqrt{\rho_R^*} \mathbf{v}_{tR}^* + (\mathbf{B}_{tR}^* - \mathbf{B}_{tL}^*) \text{sgn}(B_x)}{\sqrt{\rho_L^*} + \sqrt{\rho_R^*}} \\ \mathbf{B}_t^{**} = \frac{\sqrt{\rho_L^*} \mathbf{B}_{tR}^* + \sqrt{\rho_R^*} \mathbf{B}_{tL}^* + \sqrt{\rho_L^* \rho_R^*} (\mathbf{v}_{tR}^* - \mathbf{v}_{tL}^*) \text{sgn}(B_x)}{\sqrt{\rho_L^*} + \sqrt{\rho_R^*}} \end{array} \right.$$

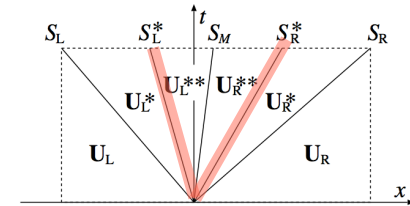
HLLD solver [15/24]



- Jump conditions across the RDs (Alfvén wave)

$$\begin{array}{c}
 S_\alpha^* \\
 \left(\begin{array}{c}
 \rho_\alpha^{**} \\
 \rho_\alpha^{**} S_M \\
 \rho_\alpha^{**} v_\alpha^{**} \\
 \rho_\alpha^{**} w_\alpha^{**} \\
 B_{y\alpha}^{**} \\
 B_{z\alpha}^{**} \\
 e_\alpha^{**}
 \end{array} \right) - \left(\begin{array}{c}
 \rho_\alpha^{**} S_M \\
 \rho_\alpha^{**} S_M^2 + p_T^* - B_x^2 \\
 \rho_\alpha^{**} v_\alpha^{**} S_M - B_x B_{y\alpha}^{**} \\
 \rho_\alpha^{**} w_\alpha^{**} S_M - B_x B_{z\alpha}^{**} \\
 B_{y\alpha}^{**} S_M - B_x v_\alpha^{**} \\
 B_{z\alpha}^{**} S_M - B_x w_\alpha^{**} \\
 (e_\alpha^{**} + p_T^*) S_M - B_x (v_\alpha^{**} \cdot B_\alpha^{**})
 \end{array} \right) = S_\alpha^* \left(\begin{array}{c}
 \rho_\alpha^* \\
 \rho_\alpha^* S_M \\
 \rho_\alpha^* v_\alpha^* \\
 \rho_\alpha^* w_\alpha^* \\
 B_{y\alpha}^* \\
 B_{z\alpha}^* \\
 e_\alpha^*
 \end{array} \right) - \left(\begin{array}{c}
 \rho_\alpha^* S_M \\
 \rho_\alpha^* S_M^2 + p_T^* - B_x^2 \\
 \rho_\alpha^* v_\alpha^* S_M - B_x B_{y\alpha}^* \\
 \rho_\alpha^* w_\alpha^* S_M - B_x B_{z\alpha}^* \\
 B_{y\alpha}^* S_M - B_x v_\alpha^* \\
 B_{z\alpha}^* S_M - B_x w_\alpha^* \\
 (e_\alpha^* + p_T^*) S_M - B_x (v_\alpha^* \cdot B_\alpha^*)
 \end{array} \right)
 \end{array}$$

HLLD solver [16/24]



- Solution:

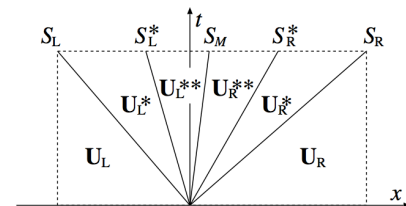
$$U_{\alpha}^{**}$$

$$\rho_{\alpha}^{**} = \rho_{\alpha}^*$$

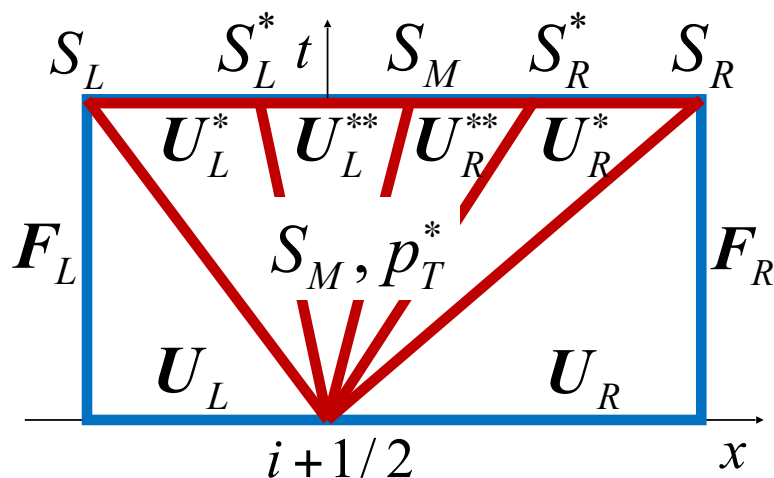
$$\left\{ \begin{array}{l} \mathbf{v}_t^{**} = \frac{\sqrt{\rho_L^*} \mathbf{v}_{tL}^* + \sqrt{\rho_R^*} \mathbf{v}_{tR}^* + (\mathbf{B}_{tR}^* - \mathbf{B}_{tL}^*) \text{sgn}(B_x)}{\sqrt{\rho_L^*} + \sqrt{\rho_R^*}} \\ \mathbf{B}_t^{**} = \frac{\sqrt{\rho_L^*} \mathbf{B}_{tR}^* + \sqrt{\rho_R^*} \mathbf{B}_{tL}^* + \sqrt{\rho_L^* \rho_R^*} (\mathbf{v}_{tR}^* - \mathbf{v}_{tL}^*) \text{sgn}(B_x)}{\sqrt{\rho_L^*} + \sqrt{\rho_R^*}} \end{array} \right.$$

$$e_{\alpha}^{**} = e_{\alpha}^* \mp \sqrt{\rho_{\alpha}^*} (\mathbf{v}_{\alpha}^* \cdot \mathbf{B}_{\alpha}^* - \mathbf{v}^{**} \cdot \mathbf{B}^{**}) \text{sgn}(B_x) \quad (-: R, +: L)$$

HLLD solver [17/24]



- 5-wave approximation

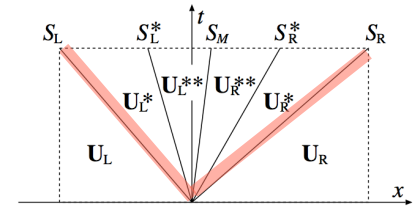


$S_{R,L}$: fast waves
 S_M : entropy wave
 $S_{R,L}^*$: Alfvén waves

$$S_{R,L} (\mathbf{U}_{R,L}^* - \mathbf{U}_{R,L}) = \mathbf{F}_{R,L}^* - \mathbf{F}_{R,L}, \quad S_{R,L}^* (\mathbf{U}_{R,L}^{**} - \mathbf{U}_{R,L}^*) = \mathbf{F}_{R,L}^{**} - \mathbf{F}_{R,L}^*,$$

$$S_M (\mathbf{U}_R^{**} - \mathbf{U}_L^{**}) = \mathbf{F}_R^{**} - \mathbf{F}_L^{**}, \quad \frac{1}{\Delta t} \int_{S_L \Delta t}^{S_R \Delta t} \mathbf{U}(x, t^{n+1}) dx + S_R \mathbf{U}_R - S_L \mathbf{U}_L + \mathbf{F}_R - \mathbf{F}_L = 0$$

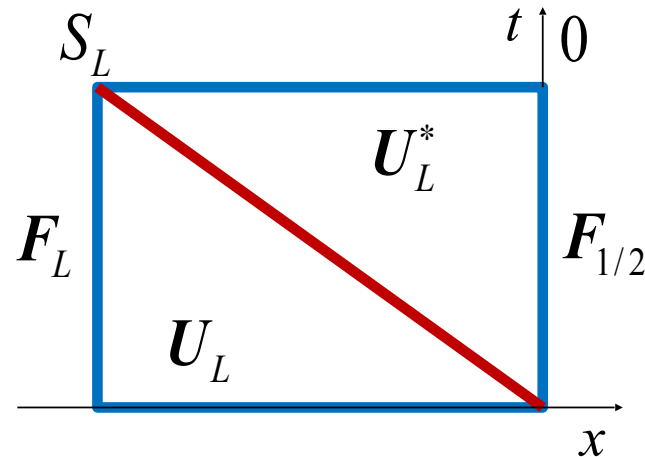
HLLD solver [18/24]



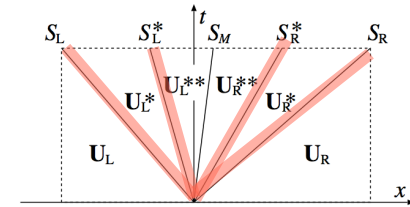
- Numerical flux

$$F_{1/2} = F_L \quad \text{if } S_L \geq 0$$

$$F_{1/2} = F_L + S_L U_L^* - S_L U_L = F_L^* \quad \text{if } S_L \leq 0 \leq S_L^*$$



HLLD solver [19/24]

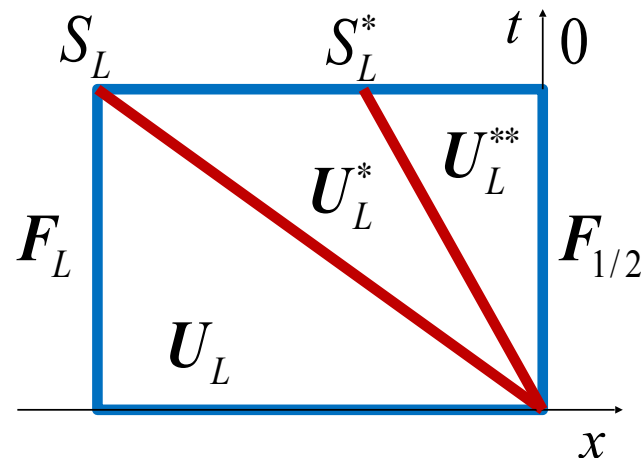


- Numerical flux

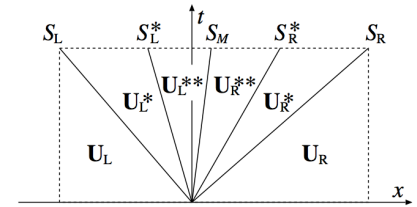
$$F_{1/2} = F_L \quad \text{if } S_L \geq 0$$

$$F_{1/2} = F_L + S_L U_L^* - S_L U_L = F_L^* \quad \text{if } S_L \leq 0 \leq S_L^*$$

$$\begin{aligned} F_{1/2} &= F_L + S_L^* U_L^{**} - (S_L^* - S_L) U_L^* - S_L U_L \\ &= F_L^* + S_L^* U_L^{**} - S_L^* U_L^* = F_L^{**} \quad \text{if } S_L^* \leq 0 \leq S_M \end{aligned}$$



HLLD solver [20/24]



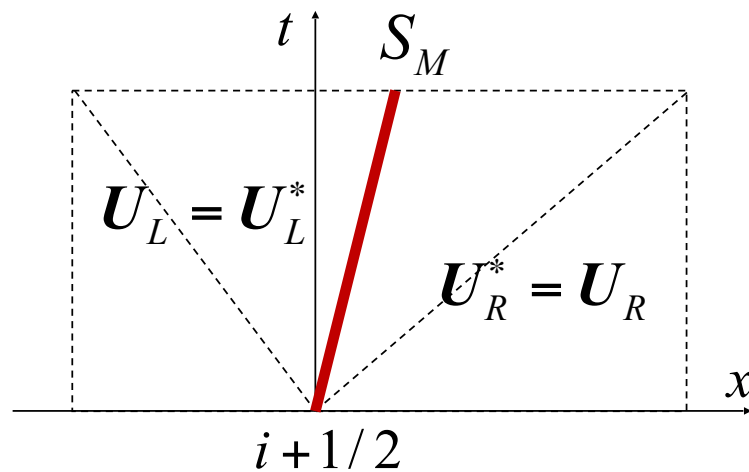
- Numerical flux

$$F_{1/2} = \begin{cases} F_L & \text{if } S_L \geq 0 \\ F_L^* & \text{if } S_L \leq 0 \leq S_L^* \\ F_L^{**} & \text{if } S_L^* \leq 0 \leq S_M \\ F_R^{**} & \text{if } S_M \leq 0 \leq S_R^* \\ F_R^* & \text{if } S_R^* \leq 0 \leq S_R \\ F_R & \text{if } S_R \leq 0 \end{cases}$$

$$F_\alpha^{*/**} = F(\rho_\alpha^{*/**}, S_M, v_{t\alpha}^{*/**}, B_x, B_{t\alpha}^{*/**}, e_\alpha^{*/**}, p_T^*)$$

HLLD solver [21/24]: HLLD can deal with..

- An isolated tangential discontinuity (TD)

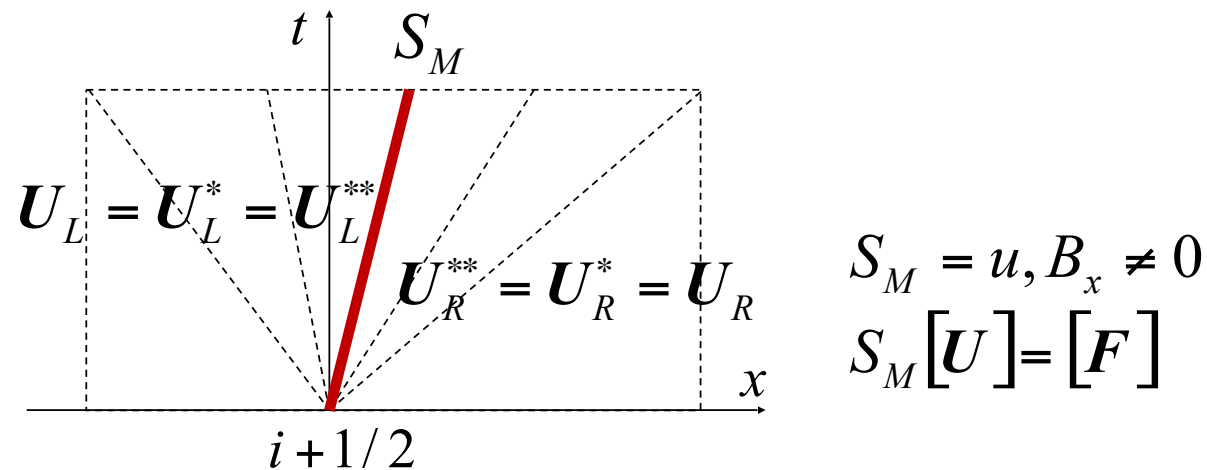


$$S_M = u, B_x = 0$$

$$S_M [U] = [F]$$

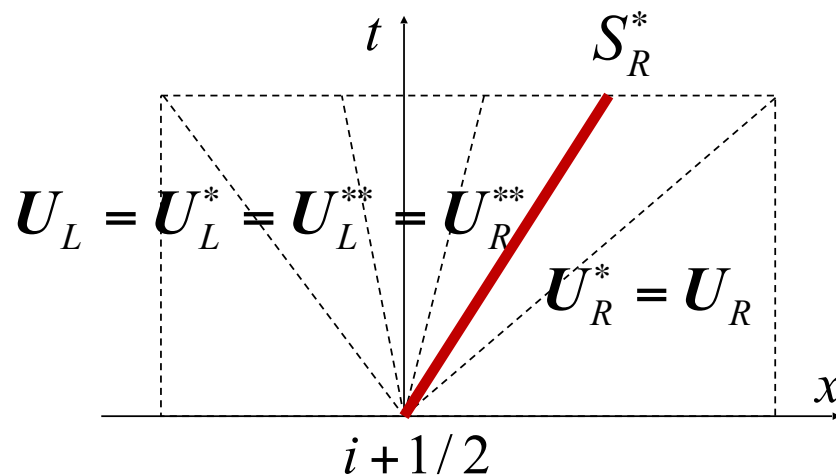
HLLD solver [22/24]: HLLD can deal with..

- An isolated tangential discontinuity (TD)
- An isolated contact discontinuity (CD)



HLLD solver [23/24]: HLLD can deal with..

- An isolated tangential discontinuity (TD)
- An isolated contact discontinuity (CD)
- An isolated rotational discontinuity (RD)

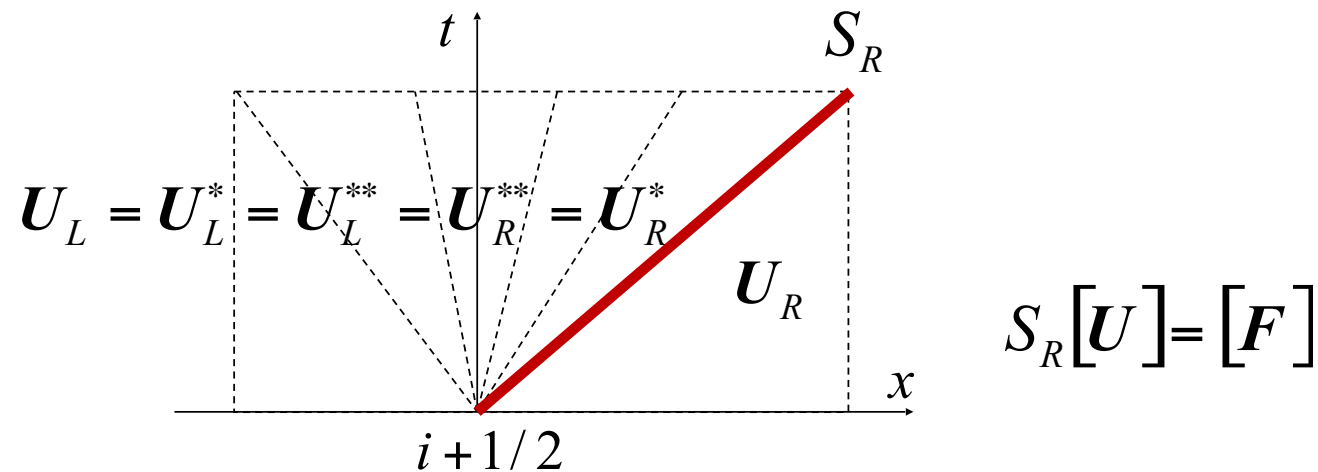


$$S_R^* = u + \frac{B_x}{\sqrt{\rho}}, B_x > 0$$

$$S_R^*[U] = [F]$$

HLLD solver [24/24]: HLLD can deal with..

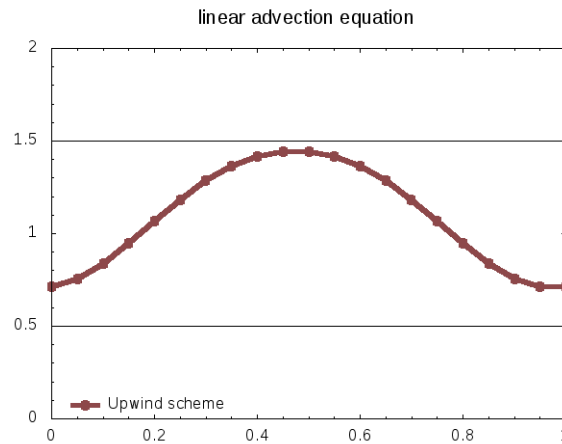
- An isolated tangential discontinuity (TD)
- An isolated contact discontinuity (CD)
- An isolated rotational discontinuity (RD)
- An isolated fast shock (FS)



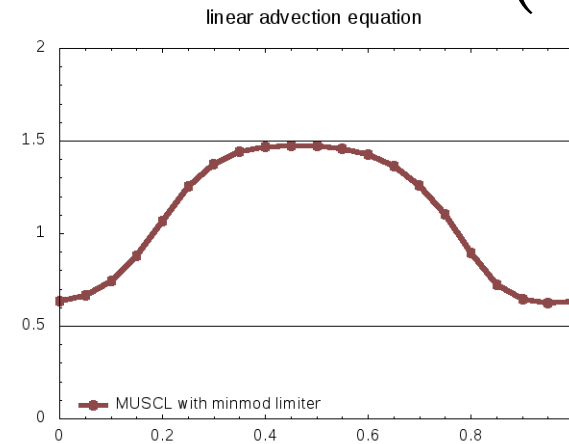
Comparisons of slope limiters (movie)

Courtesy of Dr. Miyoshi (Hiroshima University)

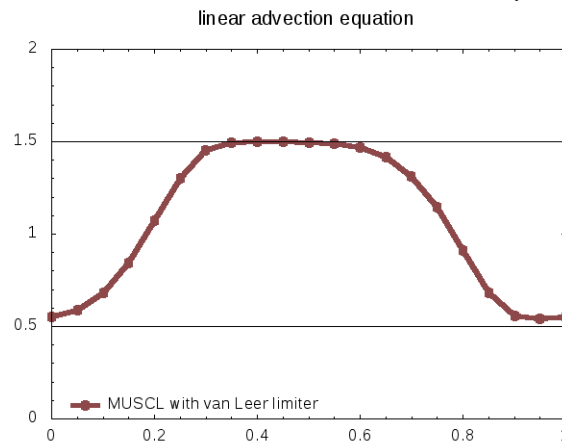
($\nu = 0.5$)



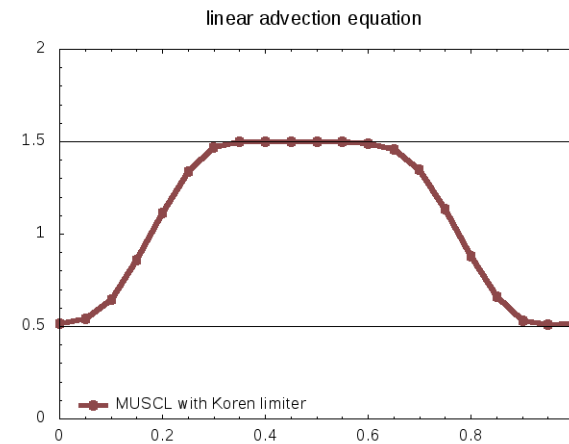
Upwind



MUSCL (minmod)



MUSCL (van Leer)

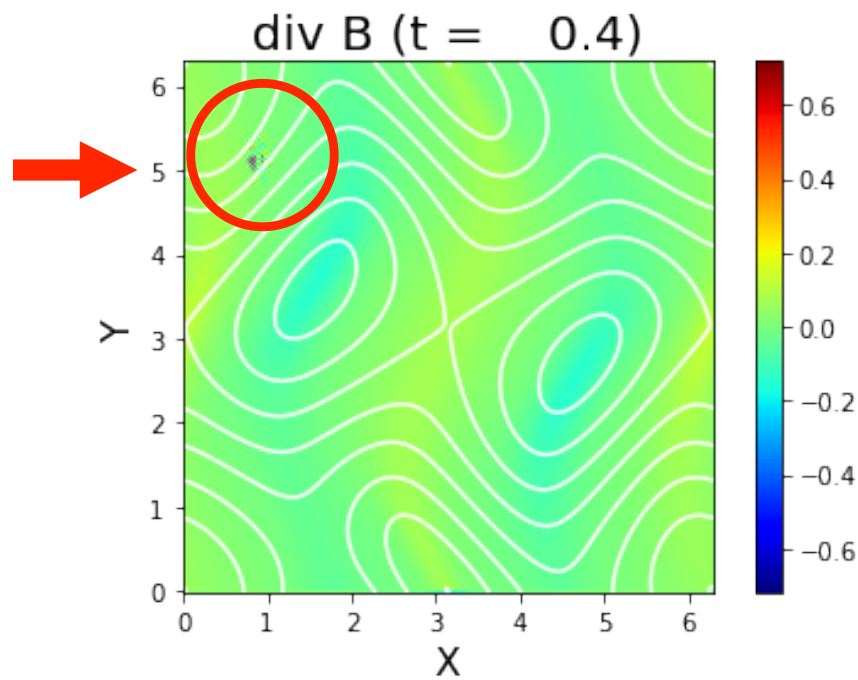


MUSCL (Koren)

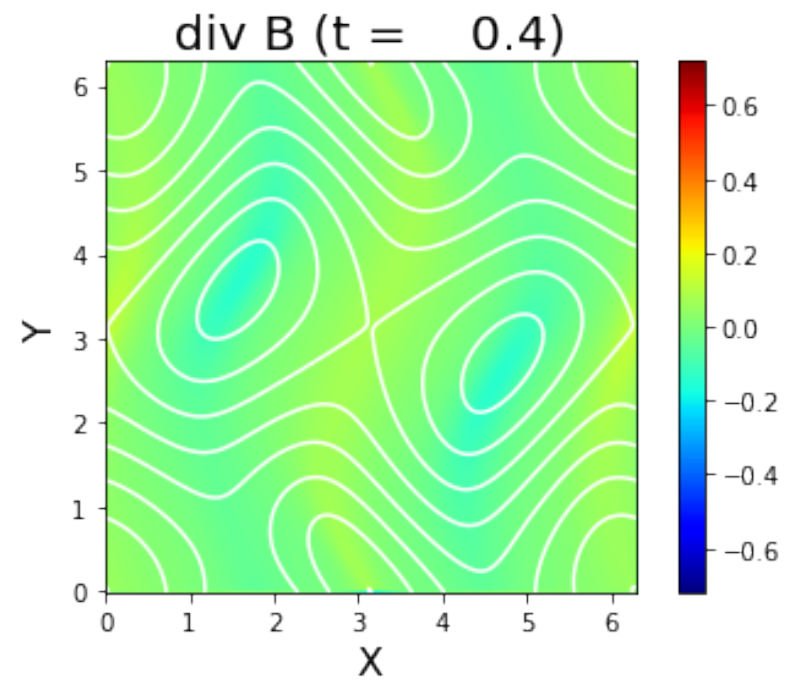
Divergence B

- The oscillation is related to divergence B
- With adequate correction, we can proceed

Without correction



With correction



Hyperbolic divergence cleaning (1/3)

- Additional equation & term for a new variable ψ

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{v}) = 0$$

$$\frac{\partial \rho \mathbf{v}}{\partial t} + \nabla \cdot (\rho \mathbf{v} \mathbf{v} + p_t \mathbb{I} - \mathbf{B} \mathbf{B}) = 0$$

$$\frac{\partial \mathcal{E}}{\partial t} + \nabla \cdot \left((\mathcal{E} + p_t) \mathbf{v} - (\mathbf{v} \cdot \mathbf{B}) \mathbf{B} \right) = 0$$

$$\frac{\partial \mathbf{B}}{\partial t} + \nabla \cdot (\mathbf{v} \mathbf{B} - \mathbf{B} \mathbf{v}) + \nabla \psi = 0$$

$$\frac{\partial \psi}{\partial t} + c_h^2 \nabla \cdot \mathbf{B} = - \left(\frac{c_h^2}{c_p^2} \right) \psi$$

Many different names:
1. Virtual potential, 2. Divergence-cleaning potential, 3. GLM parameter

- Equation system is sometimes called GLM (Generalized Lagrangian multiplier) MHD

Hyperbolic divergence cleaning (2/3)

- GLM terms

$$\frac{\partial \mathbf{B}}{\partial t} + \nabla \times (\mathbf{v} \times \mathbf{B}) + \nabla \psi = 0,$$

$$\frac{\partial \psi}{\partial t} + c_h^2 \nabla \cdot \mathbf{B} = - \left(\frac{c_h^2}{c_p^2} \right) \psi,$$

- Some math

$$\frac{\partial(\nabla \cdot \mathbf{B})}{\partial t} + \nabla \cdot [\nabla \times (\mathbf{v} \times \mathbf{B})] + \Delta \psi = 0,$$

$$\frac{\partial^2 \psi}{\partial t^2} + c_h^2 \nabla \cdot \left(\frac{\partial \mathbf{B}}{\partial t} \right) = - \left(\frac{c_h^2}{c_p^2} \right) \frac{\partial \psi}{\partial t},$$

- We obtain two telegraph equations

$$\frac{\partial^2(\nabla \cdot \mathbf{B})}{\partial t^2} + \left(\frac{c_h^2}{c_p^2} \right) \frac{\partial(\nabla \cdot \mathbf{B})}{\partial t} - c_h^2 \Delta(\nabla \cdot \mathbf{B}) = 0,$$

$$\frac{\partial^2 \psi}{\partial t^2} + \left(\frac{c_h^2}{c_p^2} \right) \frac{\partial \psi}{\partial t} - c_h^2 \Delta \psi = 0,$$

- Telegraph equations scatter and dump the variables

Hyperbolic divergence cleaning (3/3)

- GLM terms in 1-D Riemann problem

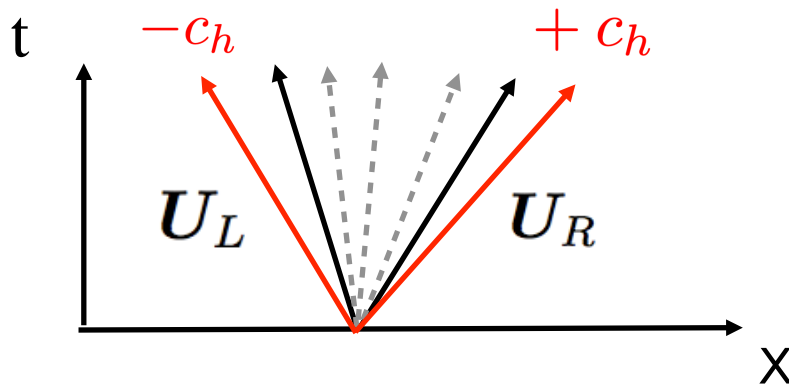
$$\frac{\partial}{\partial y} = \frac{\partial}{\partial z} = 0$$

↓ Wave equation

$$\frac{\partial}{\partial t} \begin{pmatrix} B_x \\ \psi \end{pmatrix} + \begin{pmatrix} 0 & 1 \\ c_h^2 & 0 \end{pmatrix} \frac{\partial}{\partial x} \begin{pmatrix} B_x \\ \psi \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

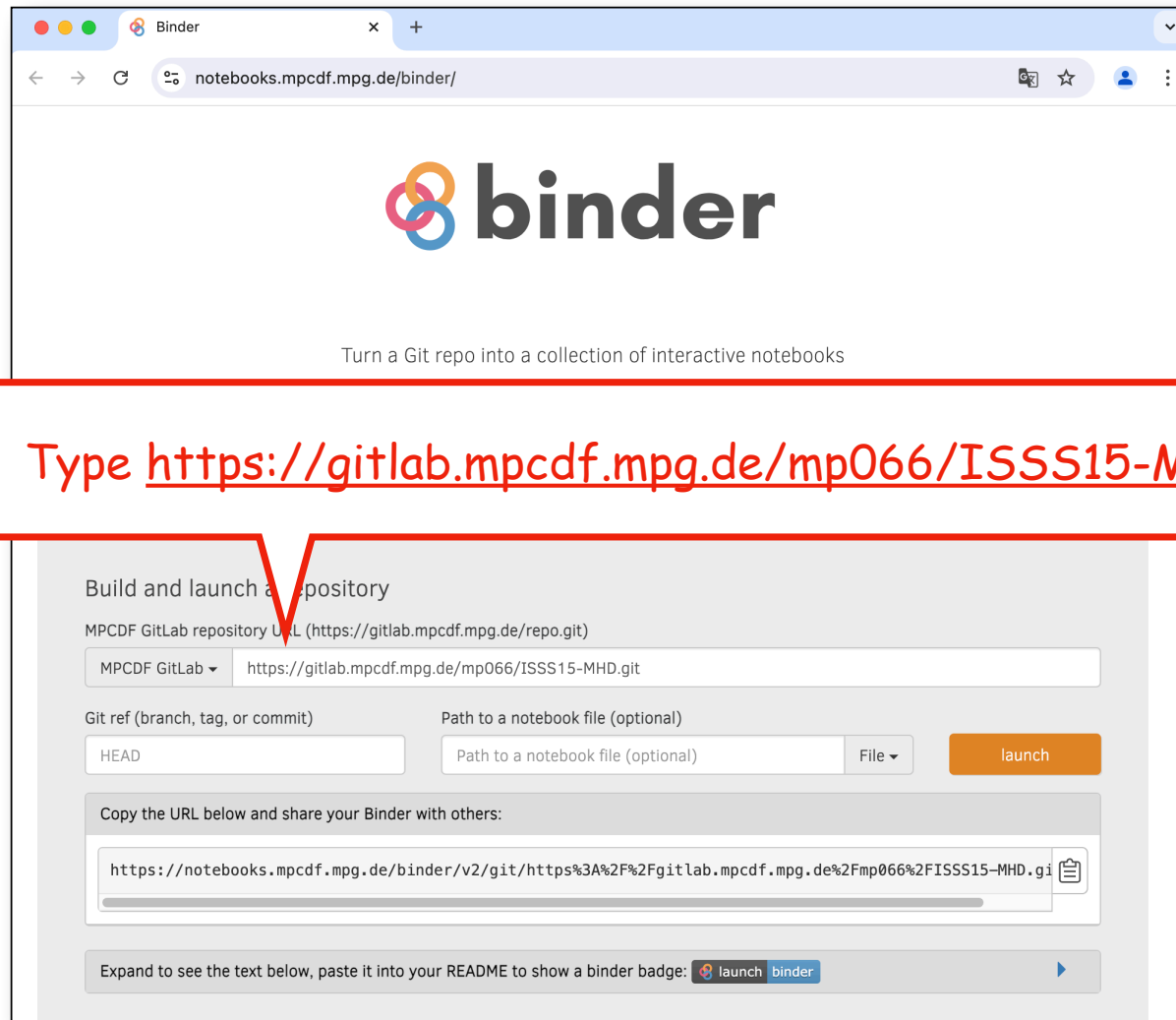
$$\frac{\partial \psi}{\partial t} = - \left(\frac{c_h^2}{c_p^2} \right) \psi \quad \leftarrow \text{Exponential decay}$$

- Riemann problem can be modified to



- Typically, c_h is set to the fastest c_f in the simulation domain
- $(c_p^2/c_h) = 0.18$ is the optimum, according to numerical tests (Dedner+ 2002)

Binder



The screenshot shows the Binder website interface. At the top, the Binder logo is displayed with the tagline "Turn a Git repo into a collection of interactive notebooks". Below this, a red callout box contains the instruction: "Type <https://gitlab.mpcdf.mpg.de/mp066/ISSS15-MHD.git>". The main form is titled "Build and launch a repository" and includes the following fields and buttons:

- MPCDF GitLab repository URL (https://gitlab.mpcdf.mpg.de/repo.git): A dropdown menu set to "MPCDF GitLab" and a text input field containing "https://gitlab.mpcdf.mpg.de/mp066/ISSS15-MHD.git".
- Git ref (branch, tag, or commit): A text input field containing "HEAD".
- Path to a notebook file (optional): A text input field containing "Path to a notebook file (optional)" and a "File" dropdown menu.
- A "launch" button.
- A section titled "Copy the URL below and share your Binder with others:" containing a text area with the URL "https://notebooks.mpcdf.mpg.de/binder/v2/git/https%3A%2F%2Fgitlab.mpcdf.mpg.de%2Fmp066%2FISSS15-MHD.git" and a copy icon.
- A section titled "Expand to see the text below, paste it into your README to show a binder badge:" containing a "launch binder" badge and a right-pointing arrow.

Using Binder (1/5) - Start page

Double click the "notebook.ipynb".

The screenshot shows the Binder start page interface. On the left, a file explorer sidebar is visible with a search bar and a list of files. The file 'notebook.ipynb' is highlighted in blue and circled in red. The main area of the interface is divided into sections: 'Notebook' with a Python 3 (ipykernel) icon, 'Console' with another Python 3 (ipykernel) icon, and 'Other' with icons for Terminal, Text File, Markdown File, Python File, and Show Contextual Help. The bottom status bar shows 'Simple' mode, memory usage 'Mem: 141.11 / 6144.00 MB', and 'Launcher 1'.

Using Binder (2/5) - Jupyter notebook

notebooks.mpcdf.mpg.de/binder/jupyter/user/mp066-iss15-mhd-ir8jr5u3/lab/tree

File Edit View Run Kernel Tabs Settings Help

Launcher notebook.ipynb Python 3 (ipykernel)

Filter files by name

Name	Last Modified
OpenMHD	14 minutes ago
apt.txt	14 minutes ago
notebook.ipynb	2 minutes ago
README.md	14 minutes ago
requirements.txt	14 minutes ago

MHD code for ISSS-15 MHD sessions

One can run cells by hitting Shift+Enter (or Shift+Return) keys.

The original MHD code is available at <https://sci.nao.ac.jp/MEMBER/zenitani/openmhd-e.html>.

1D Riemann problem (1D_basic)

```
[ ]: # Make sure to go back to the home directory
%cd

[ ]: # Changing to a directory for the 1-D basic proje
%cd OpenMHD/1D_basic/

[ ]: # Running the make command.
! make

[ ]: # Running the program.
! ./a.out

[ ]: # Plotting the result.
%run -i plot.py

[ ]: %matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
import glob
from ipywidgets import interact

labels={"x":0,"y":1,"Density":2,"Pressure":3,"vx":4,"vy":5,"vz":6,"Bx":7,"By":8,"Bz":9}
colors=['r','g','b']
markers=['o','x','x']
```

Select a cell, and then hit Shift+Enter (Shift+return)

メニューを表示 0 Python 3 (ipykernel) | Idle Mem: 141.34 / 6144.00 MB Mode: Command Ln 1, Col 1 notebook.ipynb 1

Some more on Jupyter

- Shift + enter (return) will run the command in a cell
- 1) Python codes
- 2) Magic commands
 - `%cd` # Changing the directory
 - `%run -i plot.py` # Run the python file
 - `%%time` # Measure the execution time of the cell
- 3) UNIX shell commands
 - `! pwd`
 - `! ./a.out` # Use "./" to explicitly specify your program

Using Binder (3/5) - File editor

By double-clicking a file in the left panel, you can edit the file in the right. Your modifications are automatically saved. (If not, press ctrl+S [command+S]).

You can navigate the filesystem in the left panel.

```
1 prog
2 !-----
3 !
4 !-----
5 !
6 !-----
7 im
8 in
9 in
10 integer, parameter :: jx = 200 + 2
11 integer, parameter :: loop_max = 200000
12 real(8), parameter :: tend = 100.0d0
13 real(8), parameter :: dtout = 5.0d0 ! output interval
14 real(8), parameter :: cfl = 0.4d0 ! time step
15 ! Slope limiter (0: flat, 1: minmod, 2: MC, 3: van Leer, 4: Koren)
16 integer, parameter :: lm_type = 1
17 ! Numerical flux (0: LLF, 1: HLL, 2: HLLC, 3: HLLD)
18 integer, parameter :: flux_type = 3
19 ! Time marching (0: TVD RK2, 1: RK2)
20 integer, parameter :: time_type = 0
21 !-----
22 ! See also model.f90
23 !-----
24 integer :: n_loop, n_output
25 real(8) :: t, dt, t_output
26 real(8) :: ch
27 character*256 :: filename
28 !-----
29
30 conserved variables (U)
31 conserved variables: medium state (U*)
32 primitive variables (V)
33 (ix,jx,var1) ! interpolated states
34 (ix,jx,var1) ! numerical flux (F,G)
35 !-----
36
37
38
39
40 call bc_for_U(U,ix,jx)
41 call set_dt(U,V,ch,dt,dx,cfl,ix,jx)
42 t_output = -dt/3.d0
43
```

Ln 10, Col 37 Spaces: 4 main.f90 1

Using Binder (4/5) - Terminal

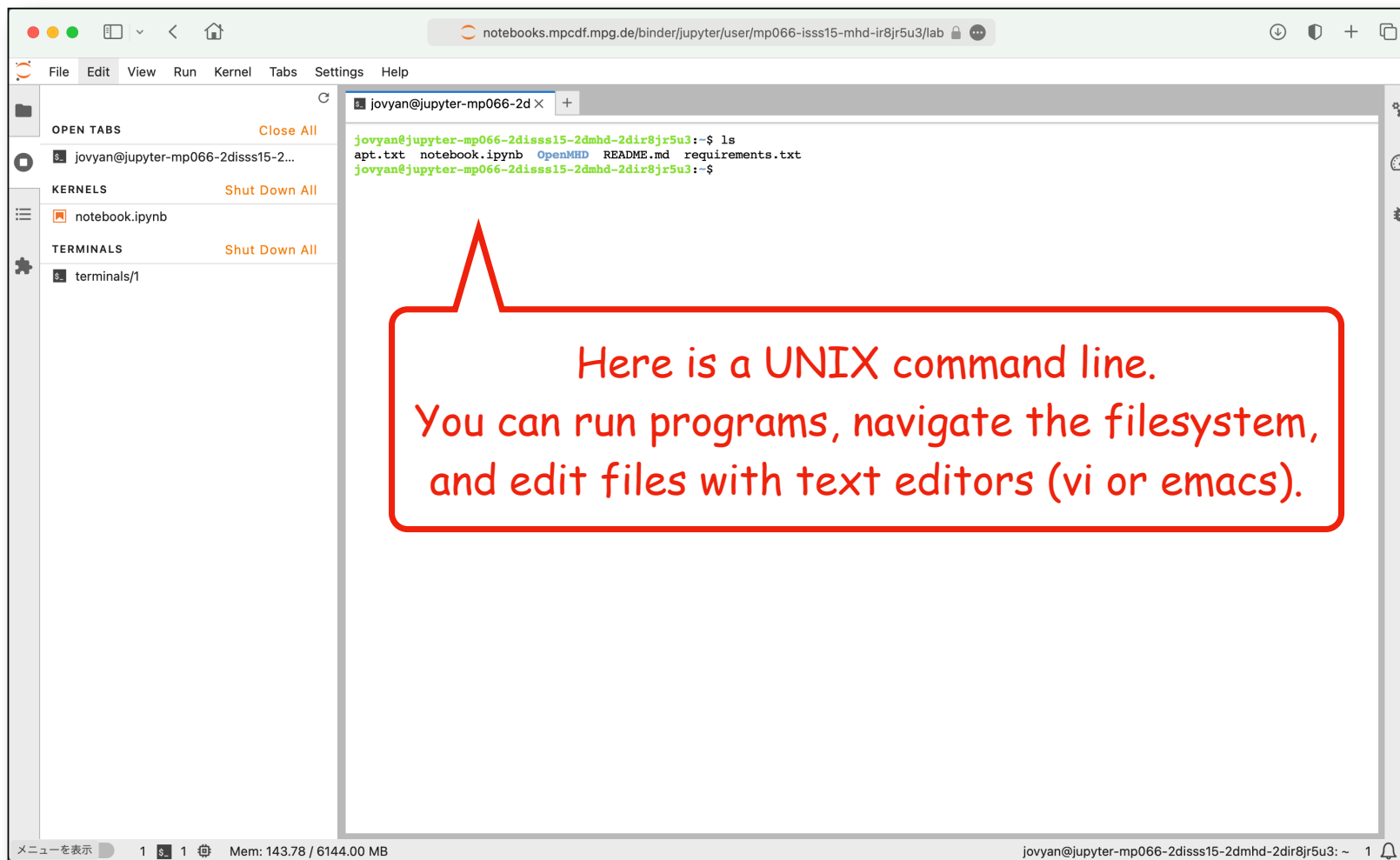
The screenshot shows the Binder Jupyter Launcher interface. On the left is a file browser with a search bar and a table of files. The main area displays a 'Launcher' view with a 'Notebook' section containing a Python icon and an 'Other' section containing icons for Terminal, Text File, Markdown File, Python File, and Show Contextual Help. A red callout box with a speech bubble points to the Terminal icon, containing the text: "If you are a UNIX expert, you can launch the Terminal from here." The Terminal icon is also circled in red.

Name	Last Modified
OpenMHD	13 minutes ago
apt.txt	13 minutes ago
notebook.ipynb	seconds ago
README.md	13 minutes ago
requirements.txt	13 minutes ago

Launcher

Simple 0 1 Mem: 141.11 / 6144.00 MB Launcher 1

Using Binder (5/5) - Terminal



The screenshot shows a JupyterLab interface with a terminal window open. The terminal displays the following output:

```
jovyan@jupyter-mp066-2diss15-2dmhd-2dir8jr5u3:~$ ls
apt.txt  notebook.ipynb  OpenMHD  README.md  requirements.txt
jovyan@jupyter-mp066-2diss15-2dmhd-2dir8jr5u3:~$
```

A red callout box with a pointer to the terminal output contains the following text:

Here is a UNIX command line.
You can run programs, navigate the filesystem,
and edit files with text editors (vi or emacs).

The interface also shows a sidebar with sections for OPEN TABS, KERNELS, and TERMINALS. The status bar at the bottom indicates memory usage: Mem: 143.78 / 6144.00 MB.

Using your own computer (1/3)

- You need:
 - Fortran compiler, (MPI), python, git, make, etc.
- Downloading OpenMHD from the website
 - `$ tar zxvf openmhd-20240130.tar.gz`
 - `$ cd openmhd-20240130/1D_basic/`
- Obtaining OpenMHD from GitHub
 - `$ git clone https://github.com/zenitani/OpenMHD.git`
 - `$ cd OpenMHD/1D_basic/`

Using your own computer (2/3)

- Edit the Makefile
 - set the F90 variable to your compiler command
- Then compile the source code according to the Makefile
 - `$ make` # compiling the serial and parallel codes
 - `$ make run` # compiling the serial code
- Running the program
 - `$./a.out` # Use "./" to explicitly specify your program
- Deleting object files and data files
 - `$ make clean`

Using your own computer (3/3) - Visualization

- Python (iPython)

- `$ ipython3 --pylab`

- In[1]: `%run plot.py`

- Python (Jupyter notebook)

- `$ jupyter-notebook plot.ipynb`

- You can run the code by hitting Shift + Return (Enter) key

- Installing python libraries

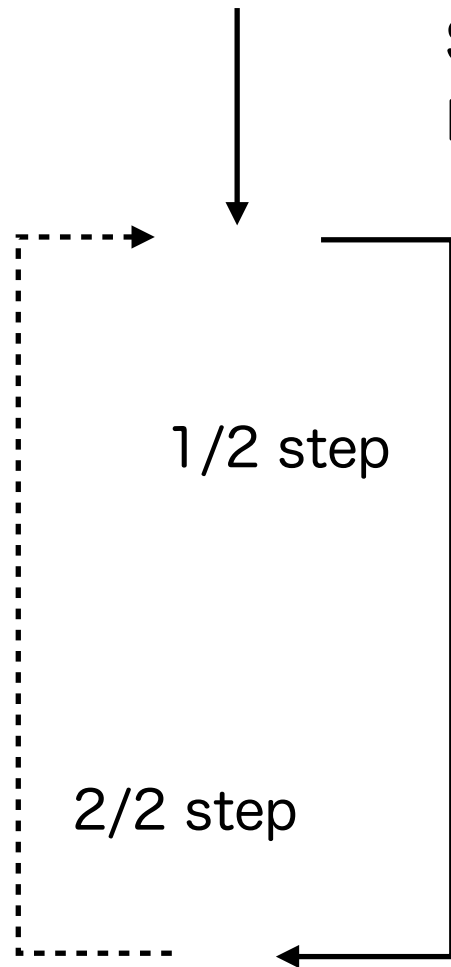
- `$ pip3 install ipython matplotlib`

- `$ pip3 install jupyter ipywidgets`

- (If you have a problem with ipywidgets 8, try ipywidgets 7.7.1 instead.)

Appendix: Logical flow of the program

Simulation settings: ix, jx (main.f90)
Initial configuration (model.f90)



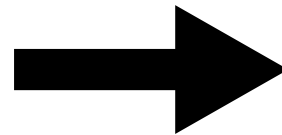
- Convert U to V (u2v.f90)
 - Output the data file (output.f90)
 - Calculate the timestep Δt (set_dt.f90)
- Interpolate U and V in the X direction (limiter.f90)
- Boundary condition for interpolated values
- Calculate numerical flux F in X (flux_solver.f90)
- Interpolate U and V in the Y direction (limiter.f90)
- Boundary condition for interpolated values
- Calculate numerical flux G in Y (flux_solver.f90)
- Advance U by using F and G (rk.f90)
- Boundary condition for U

Appendix: Reading data in Python 3

data files

data/fields-00000.dat
fields-00001.dat
...
fields-00100.dat

`openmhd.data_read()`



3-D array `data[ix,jx,9]` contains the following nine variables

$\rho, \mathbf{v}, p, \mathbf{B}, \psi$

`data[ix,jx,vx]` stands for V_x
`data[ix,jx,bx]` stands for B_x

...

```
import matplotlib.pyplot as plt
import numpy as np
import openmhd
# dummy index
vx=0;vy=1;vz=2;pr=3;ro=4;bx=5;by=6;bz=7;ps=8

# reading the data ...
x,y,t,data = openmhd.data_read("data/field-00020.dat")
# reading the data (partial domain: [ix1,ix2] x [jx1,jx2])
# x,y,t,data = openmhd.data_read("data/field-00020.dat", ix1=0, ix2=100, jx1=11)

# clearing the current figure, if any
plt.clf()
# extent: [left, right, bottom, top]
extent=[x[0],x[-1],y[0],y[-1]]
# 2D plot (vmin/mymin: minimum value, vmax/mymax: max value)
# Note: ().T is necessary for 2-D plot routines (imshow/pcolormesh...)
tmp = np.ndarray((x.size,y.size),np.double)
tmp[:,1] = data[:,1]
```