Numerical Simulations

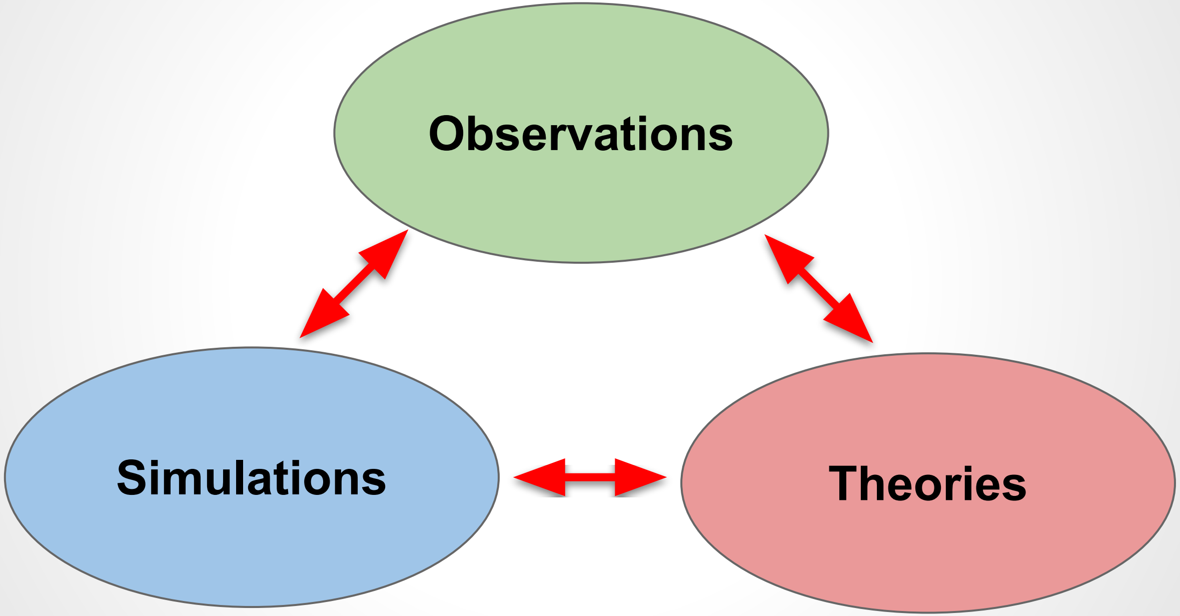
NCTS-TCA Summer Student Program
Mini-Workshop 2024

Hsi-Yu Schive (薛熙于)
National Taiwan University

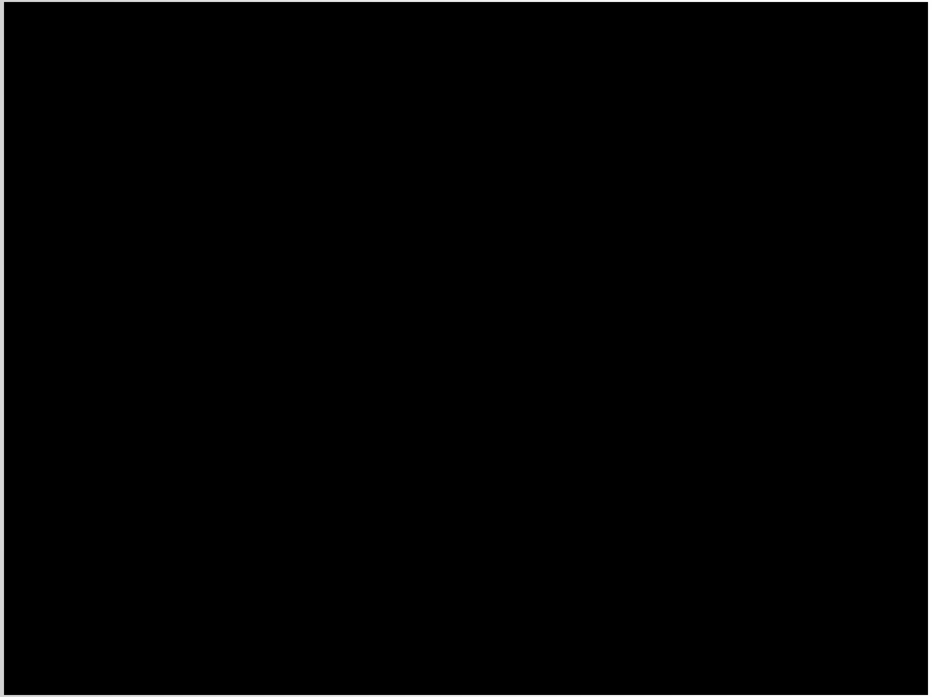
Outline

- Introduction
- (Magneto-)Hydrodynamics
- Self-gravity
- Particles

Why Simulations?



Example: Simulating Cosmic Gas



- Illustris TNG (https://www.tng-project.org)
- Cosmological magnetohydrodynamic simulations of galaxy formation
- Dark matter and gas
- Radiative cooling and heating, chemical enrichment
- Star formation and feedback
- Black hole formation and feedback
- Magnetic field

https://www.tng-project.org/movies/tng/tng50_sb2_gasvel_stars_1080p.mp4

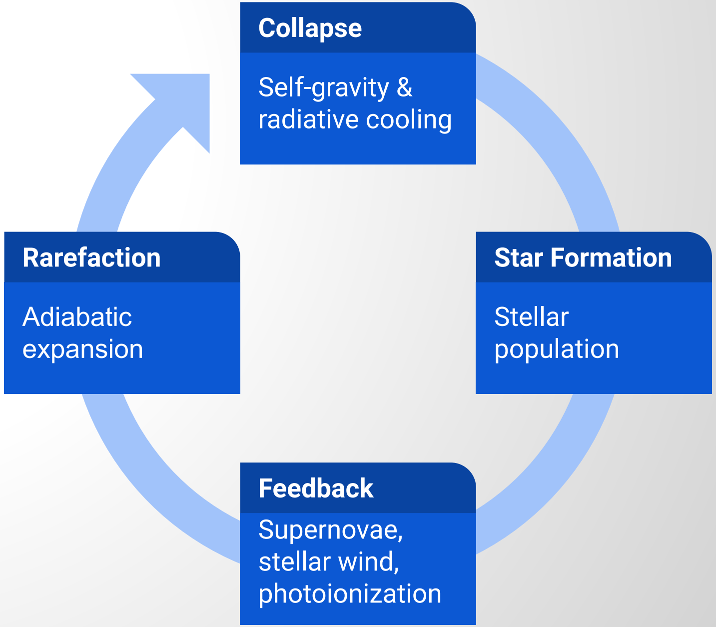
Credit: TNG Collaboration

Example: Simulating Milky Way

https://www.youtube.com/watch?v=52qIVFNJahc

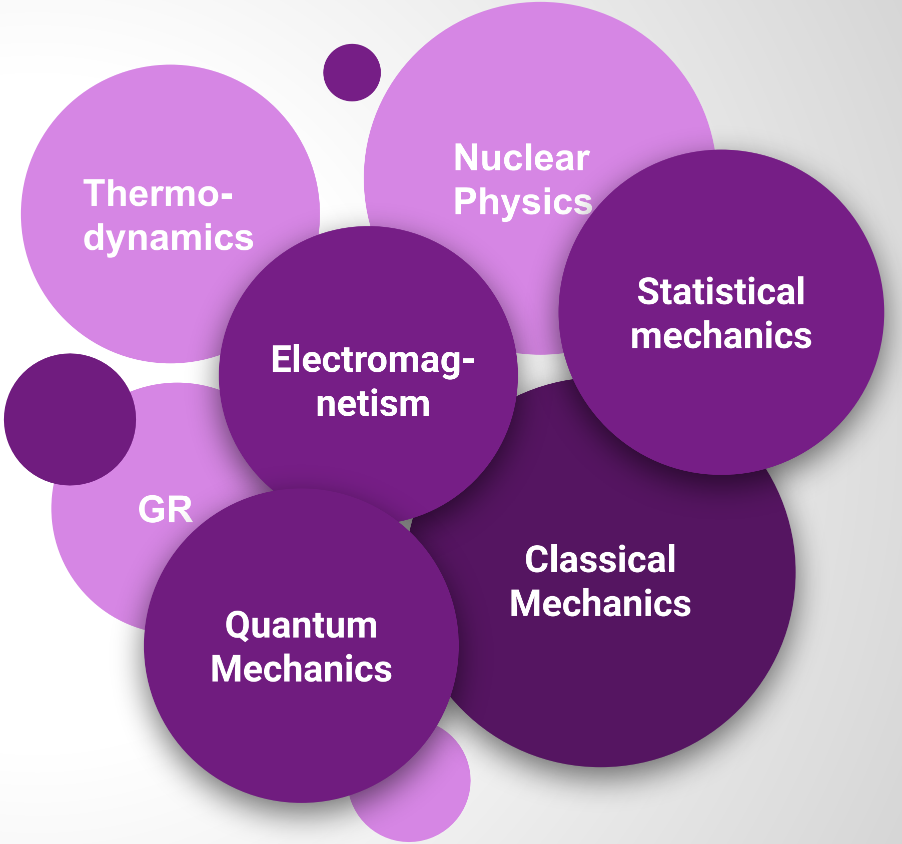
Credit: Advanced Visualization Laboratory at NCSA

- **Isolated disk galaxy simulation**
  - **Similar to our Milky Way**
- **Physics cycle**



Key Physics

- Hydrodynamics
- Magnetic field
- Gravity
- Dark matter
- Chemistry
- Radiative transfer
  - Cooling, ionization, etc
- Star formation and evolution
- Feedback
  - Supernovae explosion
  - Stellar wind
  - SMBH/AGN jets
  - …
- Multimessenger (cosmic rays, neutrinos, gravitational waves, …)

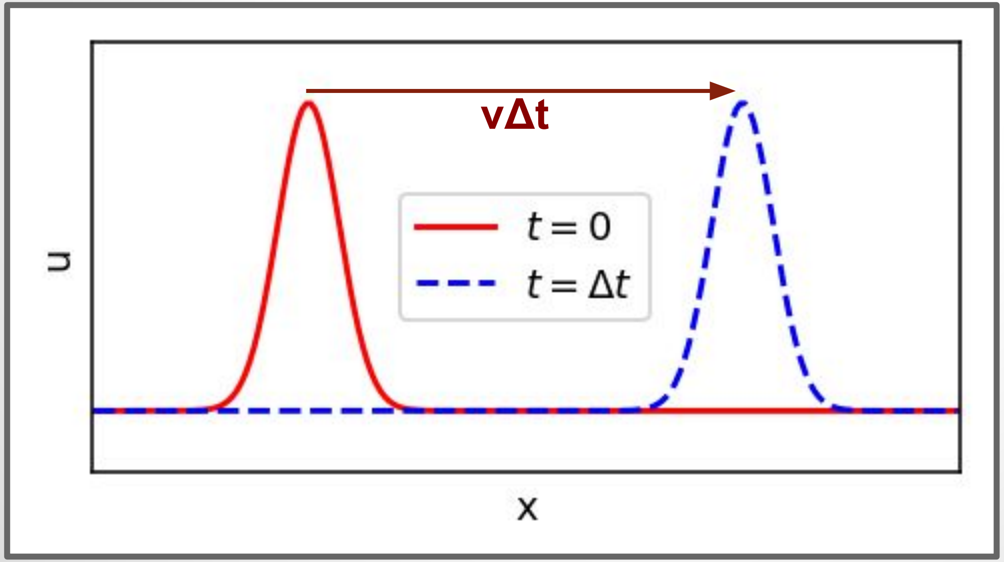


Key Techniques

- Numerical algorithms
- Parallel computing
  - CPU/GPU parallelization
- Code co-development
- Data analysis and visualization
- Debugging
- Reproducibility
  - Data sharing
  - Open source

Advection of a Scalar

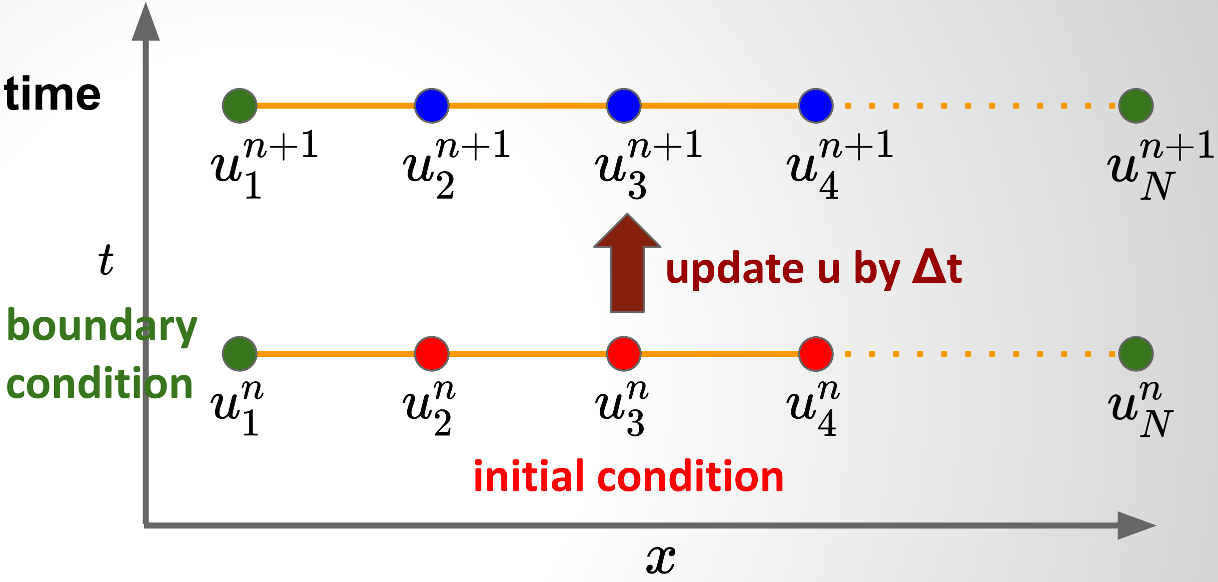
- **Governing eq.** $$\frac{\partial u(x,t)}{\partial t} = -v\frac{\partial u(x,t)}{\partial x}$$
  - **Scalar u is simply transported with a velocity v**
  - **Assuming v is constant**
  - **u is conserved → $\int u(x,t)dx = constant$**



Finite Difference Approximation

- **Discretize space and time**

$$u(x,t) \Rightarrow u_j^n$$
$$x_j = x_0 + j\Delta x$$
$$t_n = t_0 + n\Delta t$$



- **Given u_j^n , solve u_j^{n+1}**
- **Taylor expansion**

$$f(\alpha+\Delta\alpha) = f(\alpha) + f'(\alpha)\Delta\alpha + \frac{1}{2!}f''(\alpha)\Delta\alpha^2 + \frac{1}{3!}f'''(\alpha)\Delta\alpha^3 + \ldots$$

  - **Use it to approximate partial derivatives by discrete u_j^n**
  - **That's what differentiates different numerical schemes**
    - **May NOT be as trivial as you think!**

Forward-Time Central-Space Scheme

- **Advection eq.** $$\frac{\partial u(x,t)}{\partial t} = -v\frac{\partial u(x,t)}{\partial x}$$
- **FTCS scheme:**

$$\frac{\partial u(x_j,t_n)}{\partial t} \rightarrow \frac{u_j^{n+1}-u_j^n}{\Delta t} + \underline{O(\Delta t)}$$

$$\frac{\partial u(x_j,t_n)}{\partial x} \rightarrow \frac{u_{j+1}^n-u_{j-1}^n}{2\Delta x} + \underline{O(\Delta x^2)}$$

↙️forward-time

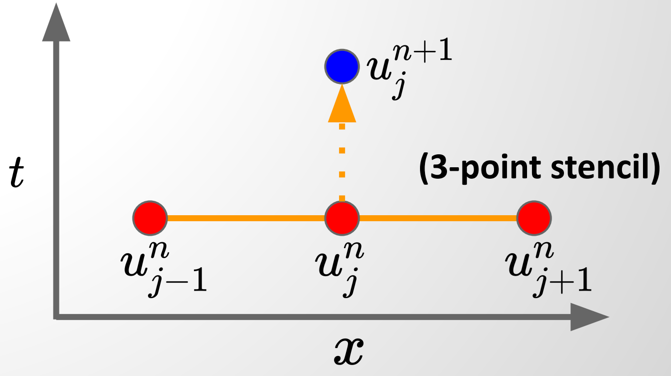
↖️errors↙️

↖️central-space

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

LHS: t=n+1 (unknown)

RHS: t=n (known)

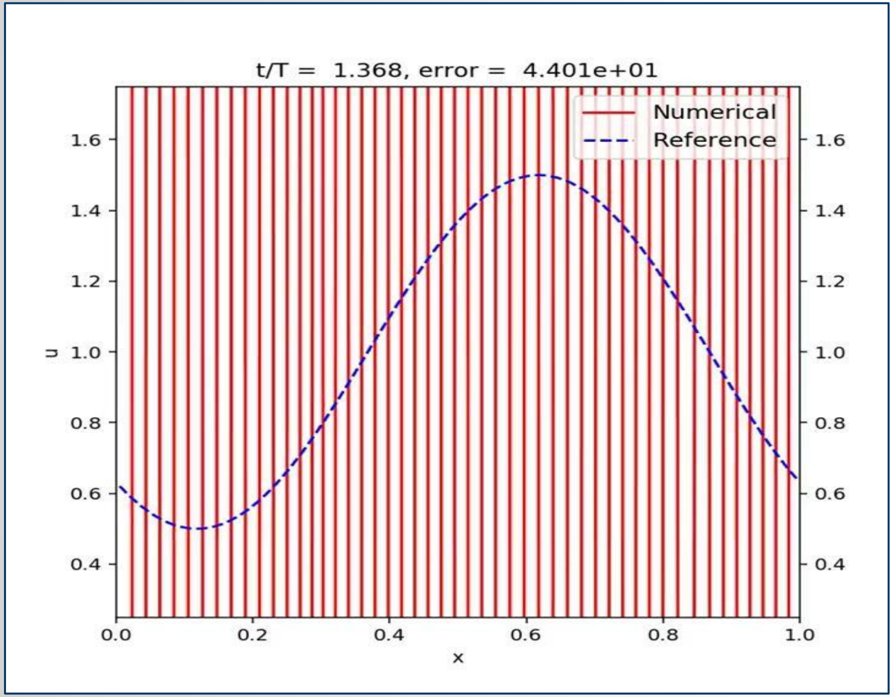


Forward-Time Central-Space Scheme

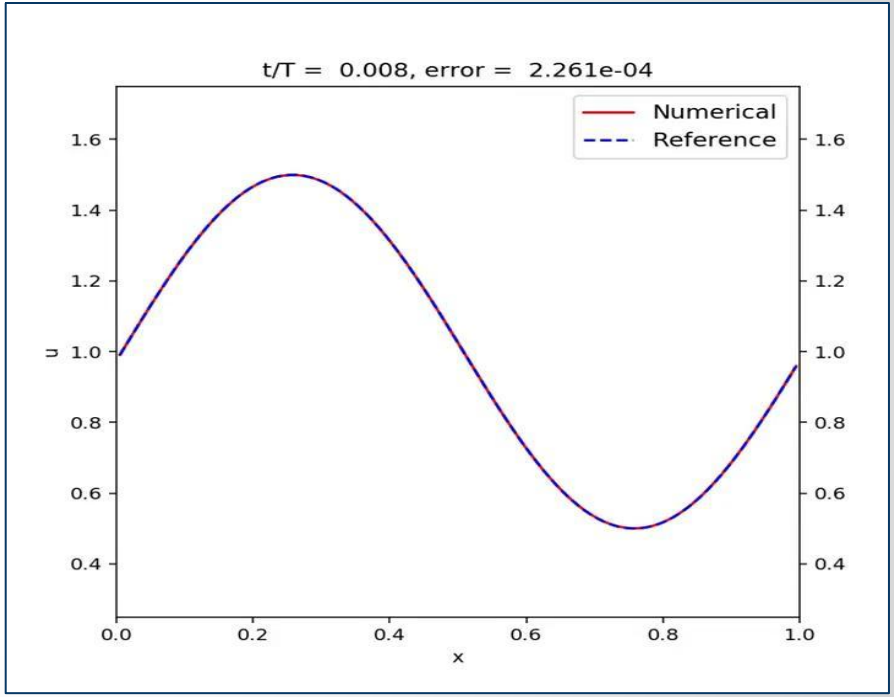
- Explicit schemes
  - u_j^{n+1} of each j can be computed explicitly from values at t_n
  - u_j^{n+1} of different j can be computed independently (i.e., the calculation of different u_j^{n+1} is fully decoupled)
    - Important for parallelization
  - In comparison, implicit schemes solve coupled equations of u_j^{n+1} with different j simultaneously
    - For example, check the Crank–Nicolson method
- FTCS scheme is very simple. But, it is **UNSTABLE** in general for hyperbolic equations!
  - It can be demonstrated using the von Neumann stability analysis
  - See the next demo

Demo: Advection

FTCS → unconditionally unstable



Lax → conditionally stable



Complete source code:

- FTCS vs Lax: https://gist.github.com/hyschive/1efd5f8f0b7eb2e6b7c92d2919f6beb7

Lessons Learned from FTCS

- Numerical errors are dominated by amplitude errors
  - Both phase and dispersion errors are negligible
- Amplitude errors increase with time
  - Low-k (long-wavelength) errors dominate first
  - High-k (short-wavelength) errors appear later but grow faster (why?)
  - Amplitude increases instead of decreases → sign of instability
  - Smaller Δt → errors decrease, but still unstable!
- Is mass conserved?

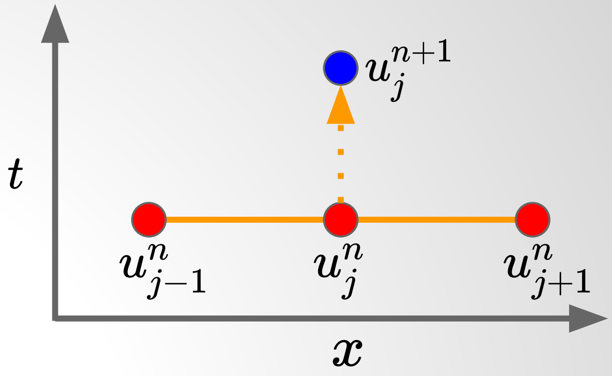
Lax Scheme

- $$u_j^{n+1} = \frac{1}{2}(u_{j+1}^n + u_{j-1}^n) - \frac{v\Delta t}{2\Delta x}(u_{j+1}^n - u_{j-1}^n)$$
- **Stability criterion:** $\Delta t \le \Delta x / v$
  - **Courant-Friedrichs-Lewy (CFL) condition**
  - **CFL number:** $v\Delta t / \Delta x$
- **But why?**
  - **For a time-step** Δt**, the max distance information can propagate is** $v\Delta t$
  - **But our finite difference scheme only collects data from** Δx
  - **If** $v\Delta t > \Delta x$**, the correct update requires information more distant than the finite difference scheme knows**
- **Numerical dissipation: the Lax scheme can be rewritten as**

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

original FTCS scheme

numerical dissipation $\frac{(\Delta x)^2}{2\Delta t}\frac{\partial^2 u}{\partial x^2}$



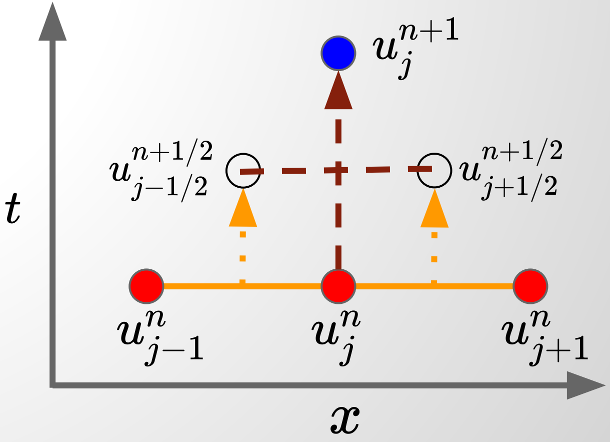
Lax-Wendroff Scheme

- **Two-step approaches (similar to the trapezoidal rule of integration)**
  - **Step 1: evaluate $u_{j+1/2}^{n+1/2}$ defined at the half time-step $n+1/2$ and the cell interface $j+1/2$ with the Lax scheme**

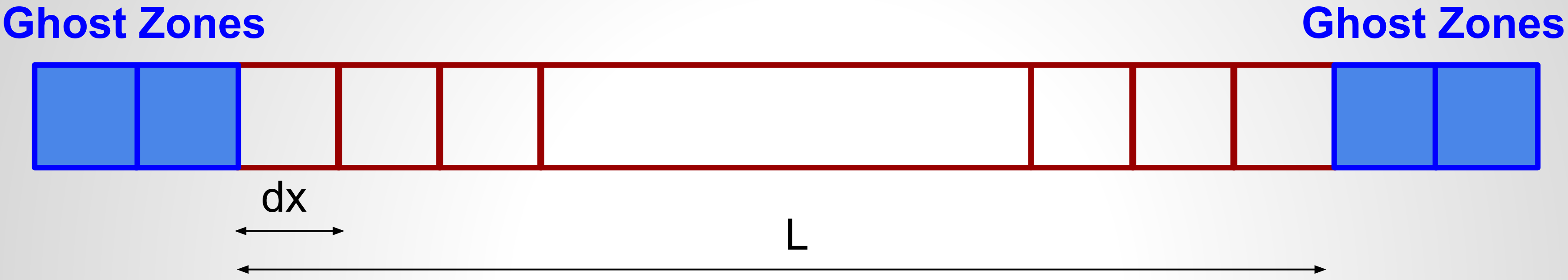
$$u_{j+1/2}^{n+1/2} = \frac{1}{2}(u_{j+1}^n + u_j^n) - \frac{v\Delta t}{2\Delta x}(u_{j+1}^n - u_j^n)$$

  - **Step 2: use $u_{j+1/2}^{n+1/2}$ to evaluate the half-step fluxes for the full-step update**

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



Ghost Zones/Grids/Cells



- Ghost zones are used for setting the <u>boundary conditions</u>
  - Physical boundaries (e.g., periodic, outflow, inflow)
  - Numerical boundaries between different parallel processes
- Number of ghost zones depends on the stencil size
  - Lax-Friedrichs: 1
  - Higher-order schemes in general require more ghost zones
  - Affect parallel scalability

Hydrodynamics: Governing Equations

- **Euler eqs.**

$$\frac{\partial \rho}{\partial t} + \boldsymbol{\nabla} \cdot (\rho \boldsymbol{v}) = 0$$ ← mass conservation

$$\frac{\partial (\rho \boldsymbol{v})}{\partial t} + \boldsymbol{\nabla} \cdot (\rho \boldsymbol{v}\boldsymbol{v} + P\boldsymbol{I}) = 0$$ ← momentum conservation

$$\frac{\partial E}{\partial t} + \boldsymbol{\nabla} \cdot [(E + P)\boldsymbol{v}] = 0$$ ← energy conservation

- ρ: **mass density,** v: **velocity,** P: **pressure,** E: **total energy density,** I: **identity matrix**

  $E = e + \frac{1}{2}\rho v^2$, **where** e **is the internal energy density**

- **6 variables, 5 equations → need <u>equation of state</u> to compute** P
  - **For example, ideal gas:** $e = \frac{P}{\gamma - 1}$, **where** γ **is the ratio of specific heat**

Flux-Conservative Form in 1D

- **Euler eqs. in a compact flux-conservative form:**

$$\frac{\partial \boldsymbol{U}}{\partial t} + \frac{\partial \boldsymbol{F_x}}{\partial x} + \frac{\partial \boldsymbol{F_y}}{\partial y} + \frac{\partial \boldsymbol{F_z}}{\partial z} = 0$$

  - F_x, F_y, F_z **are the fluxes along different directions**

$$\boldsymbol{F_x} = \begin{bmatrix} \rho v_x \\ \rho v_x^2 + P \\ \rho v_x v_y \\ \rho v_x v_z \\ (E+P)v_x \end{bmatrix} \quad \boldsymbol{F_y} = \begin{bmatrix} \rho v_y \\ \rho v_y v_x \\ \rho v_y^2 + P \\ \rho v_y v_z \\ (E+P)v_y \end{bmatrix} \quad \boldsymbol{F_z} = \begin{bmatrix} \rho v_z \\ \rho v_z v_x \\ \rho v_z v_y \\ \rho v_z^2 + P \\ (E+P)v_z \end{bmatrix}$$

Finite-Volume Scheme

- **Divergence theorem:** $\int_V \frac{\partial \boldsymbol{U}}{\partial t} dV = -\int_V (\boldsymbol{\nabla} \cdot \boldsymbol{F}) dV = -\oint_S (\boldsymbol{F} \cdot \boldsymbol{n}) dS$
- **Integrate over the cell volume** $\Delta x \Delta y \Delta z$ **and time interval** $\Delta t = t^{n+1} - t^n$

$$\boldsymbol{U}^n_{i,j,k} \equiv \frac{1}{\Delta x \Delta y \Delta z} \int_{z_{k-1/2}}^{z_{k+1/2}} \int_{y_{j-1/2}}^{y_{j+1/2}} \int_{x_{i-1/2}}^{x_{i+1/2}} U(x, y, z, t^n) dx dy dz$$

$$\boldsymbol{F}^{n+1/2}_{x,i-1/2,j,k} \equiv \frac{1}{\Delta y \Delta z \Delta t} \int_{t^n}^{t^{n+1}} \int_{z_{k-1/2}}^{z_{k+1/2}} \int_{y_{j-1/2}}^{y_{j+1/2}} F(x_{i-1/2}, y, z, t) dy dz dt$$

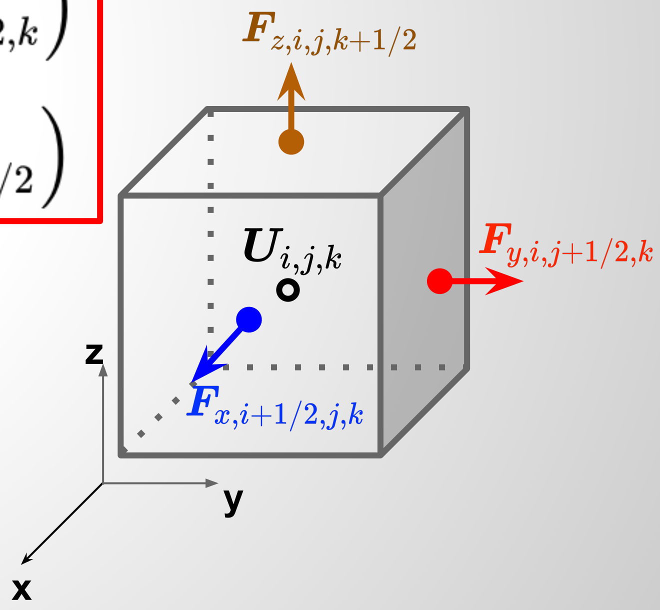
similar for $\boldsymbol{F}^{n+1/2}_{y,i,j-1/2,k}$ and $\boldsymbol{F}^{n+1/2}_{z,i,j,k-1/2}$

Finite-Volume Scheme

- Euler eqs. can be casted into the following form:

$$\begin{aligned}\boldsymbol{U}^{n+1}_{i,j,k} = \boldsymbol{U}^{n}_{i,j,k} &- \frac{\Delta t}{\Delta x}\left(\boldsymbol{F}^{n+1/2}_{x,i+1/2,j,k} - \boldsymbol{F}^{n+1/2}_{x,i-1/2,j,k}\right) \\ &- \frac{\Delta t}{\Delta y}\left(\boldsymbol{F}^{n+1/2}_{y,i,j+1/2,k} - \boldsymbol{F}^{n+1/2}_{y,i,j-1/2,k}\right) \\ &- \frac{\Delta t}{\Delta z}\left(\boldsymbol{F}^{n+1/2}_{z,i,j,k+1/2} - \boldsymbol{F}^{n+1/2}_{z,i,j,k-1/2}\right)\end{aligned}$$

  - Note that this form is EXACT!
    - No approximation has been made
  - $\boldsymbol{U}^{n}_{i,j,k}$: volume-averaged values
  - $\boldsymbol{F}^{n+1/2}_{x,i-1/2,j,k}$: time- and area-averaged values



Finite-Volume Scheme

- **The major task is to compute $\boldsymbol{F}^{n+1/2}_{x,i-1/2,j,k}$ etc**
- **Conservative quantities $\boldsymbol{U}^{n}_{i,j,k}$ (i.e., mass, momentum, energy) are guaranteed to conserve to the machine precision!**
- **It doesn't mean no numerical errors. It just means that numerical errors won't contaminate conservation laws.**

Lax-Friedrichs Scheme for Hydro

- **Lax-Friedrichs scheme can be rewritten into a flux-conservative form**

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

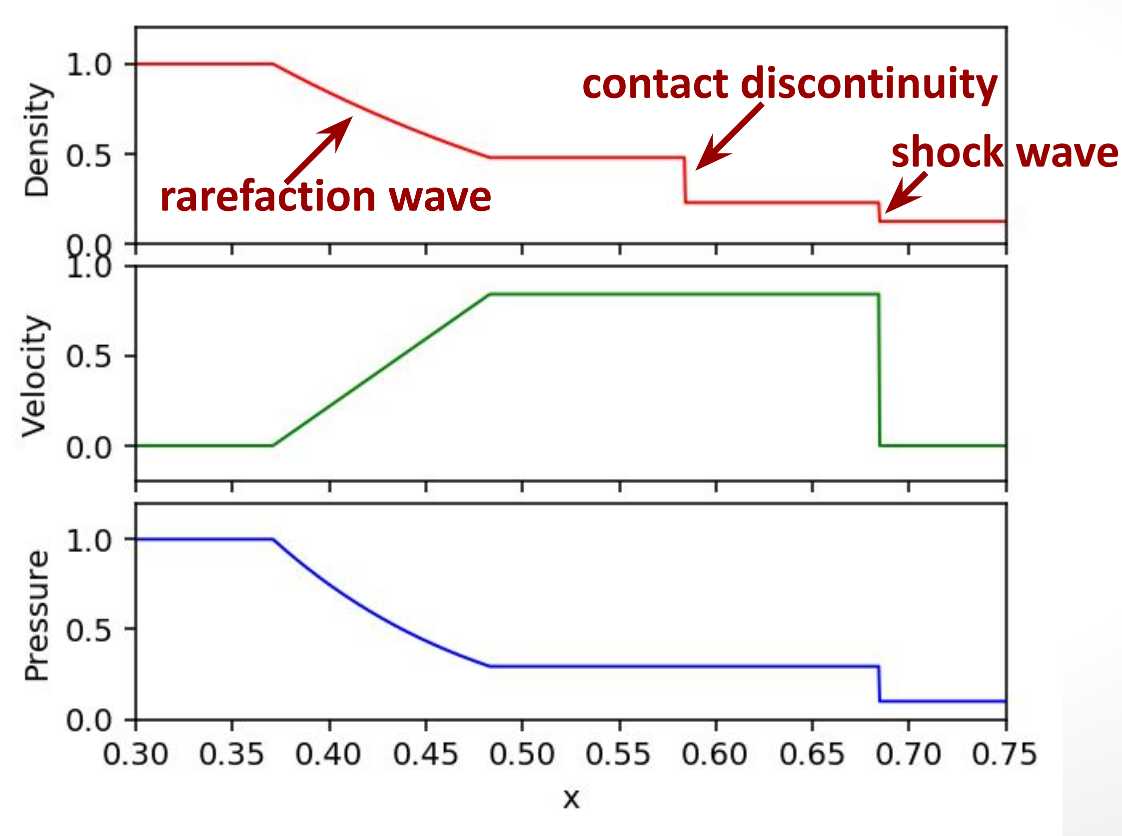
$$u_j^{n+1} = u_j^n - \frac{\Delta t}{\Delta x}(\tilde{F}_{j+1/2}^n - \tilde{F}_{j-1/2}^n)$$

$$\tilde{F}_{j-1/2}^n \equiv \frac{1}{2}\left[(vu_j^n + vu_{j-1}^n) - \frac{\Delta x}{\Delta t}(u_j^n - u_{j-1}^n)\right]$$

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

- **Hydro: simply evaluate F_j with hydrodynamic fluxes**
- **Courant condition:** $\Delta t \leq \frac{\Delta x}{|v_x| + C_s}$ ← sound speed

Sod Shock Tube Problem



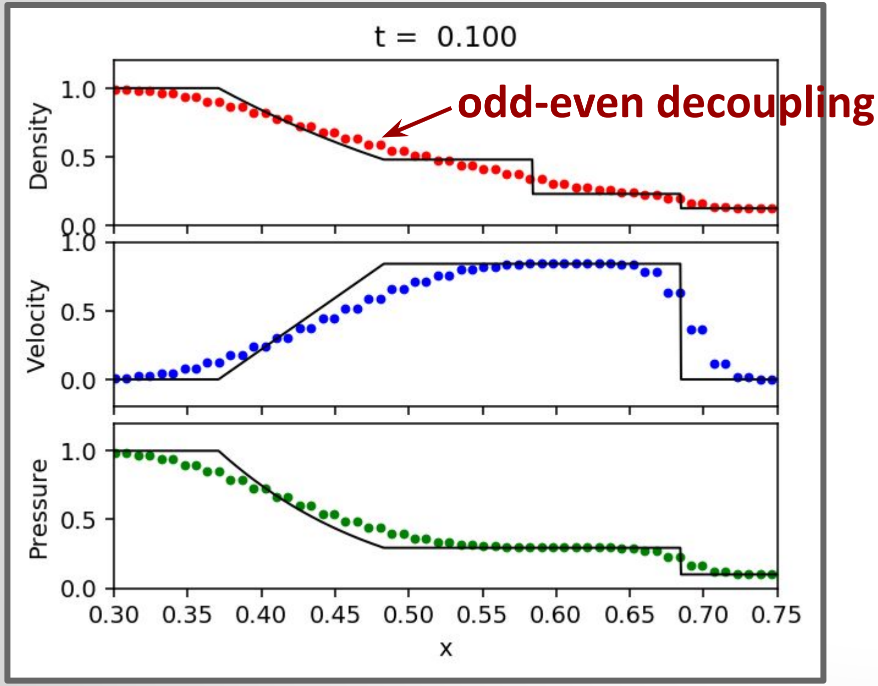
Initial condition

Left state	Right state

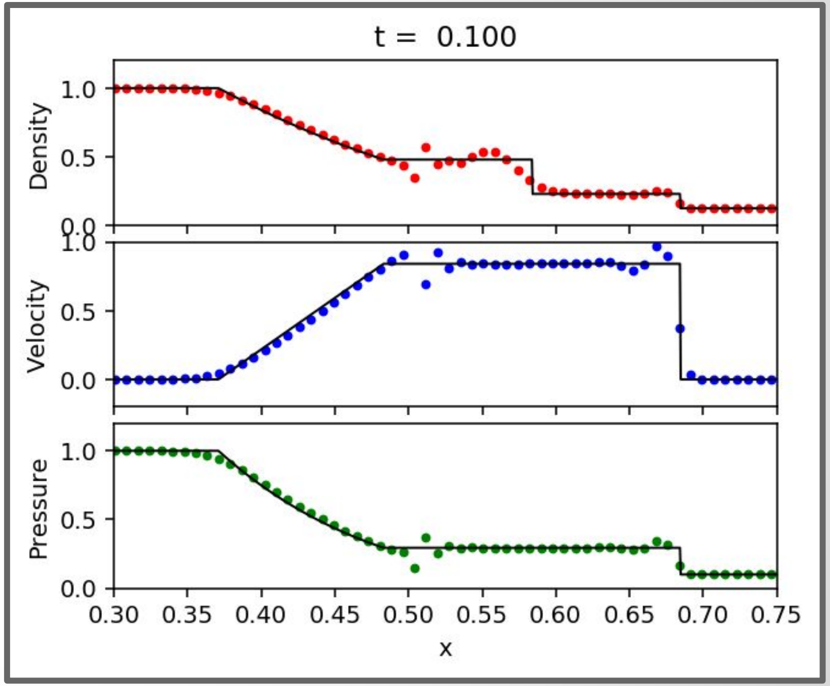
$$\begin{bmatrix} \rho_L \\ v_L \\ P_L \end{bmatrix} = \begin{bmatrix} 1.0 \\ 0.0 \\ 1.0 \end{bmatrix}, \begin{bmatrix} \rho_R \\ v_R \\ P_R \end{bmatrix} = \begin{bmatrix} 0.125 \\ 0.0 \\ 0.1 \end{bmatrix}$$

Test on Sod Shock Tube Problem

Lax-Friedrichs → too diffusive



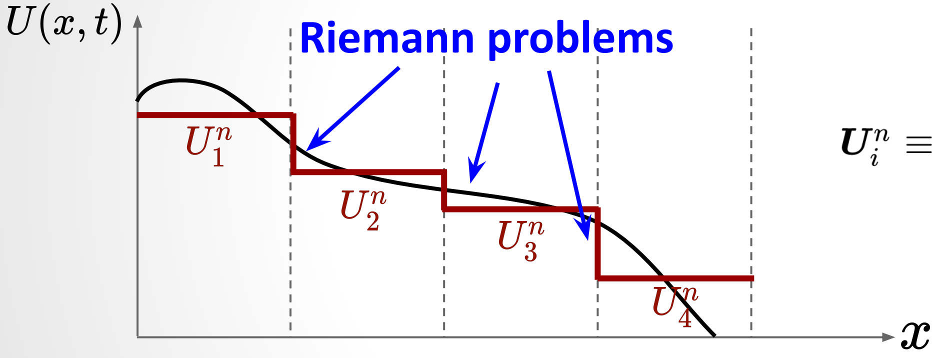
Lax-Wendroff → unphysical oscillations



- Motivate <u>high-resolution shock-capturing</u> schemes

High-Resolution Shock-Capturing Methods

- Godunov method
  - Approximate data with a <u>piecewise constant</u> distribution (in practice, higher-order approximations like <u>piecewise linear/parabolic</u> are adopted)



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

  - 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

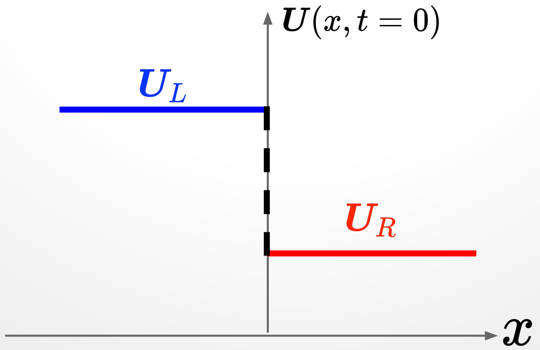
Riemann Problem in 1D Hydro

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

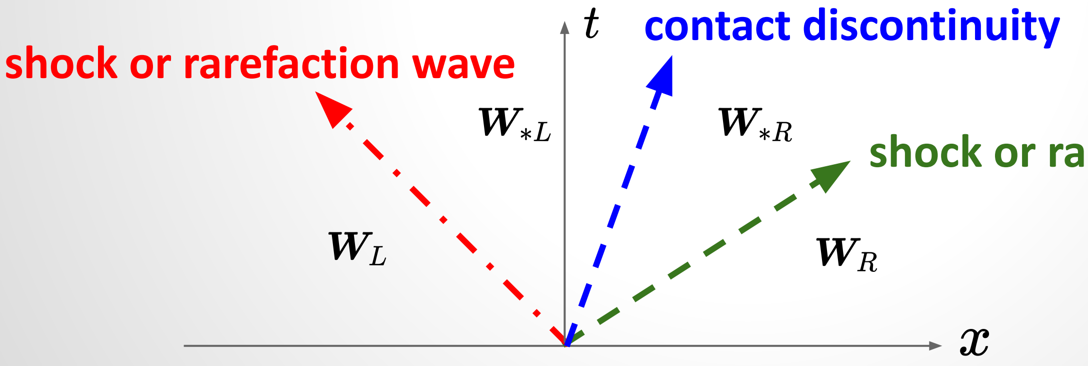
left state

right state



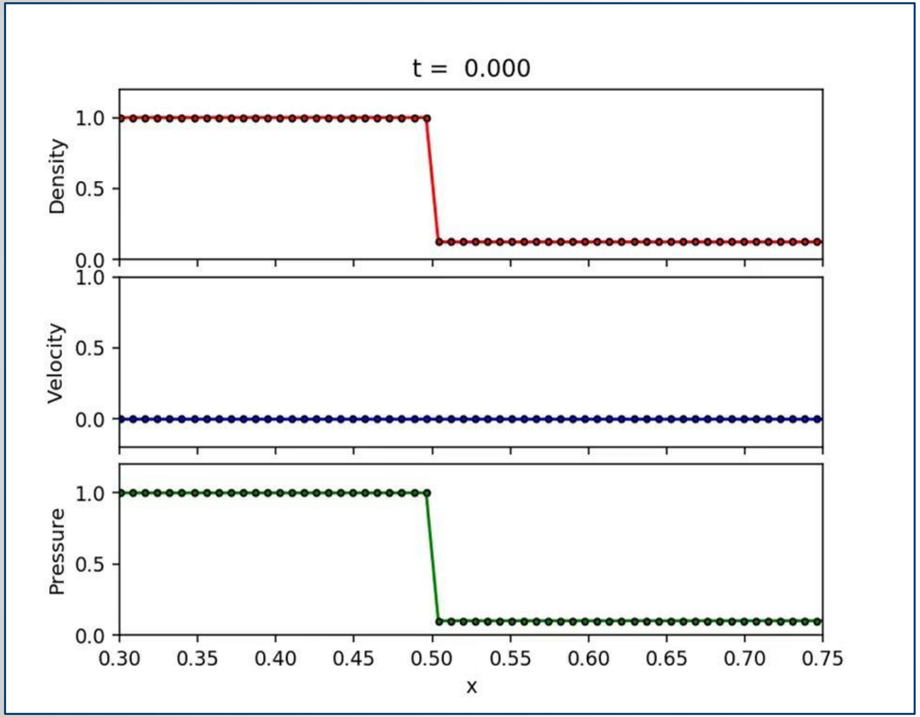
Riemann Problem in 1D Hydro

- 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



Demo: Sod Shock Tube

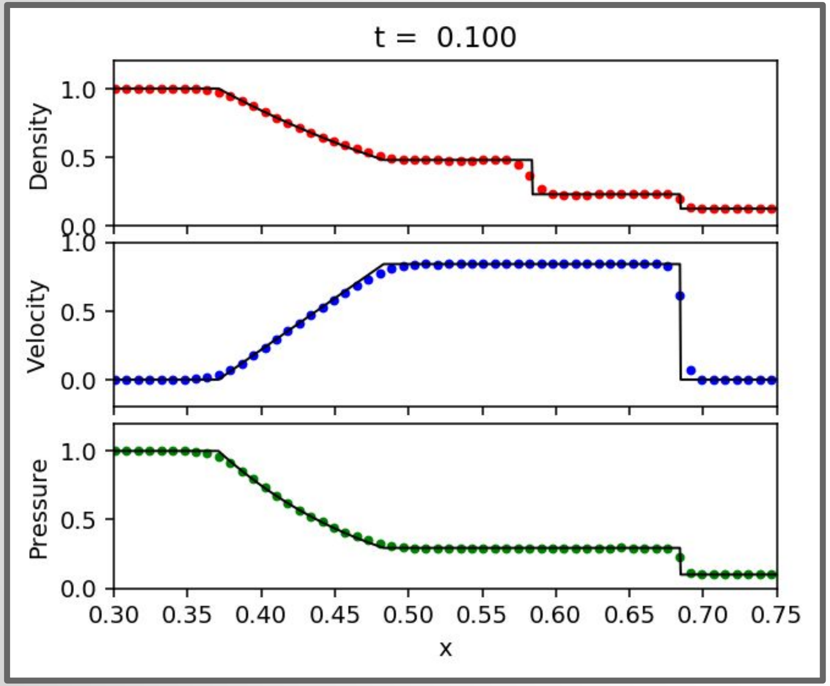
MUSCL-Hancock



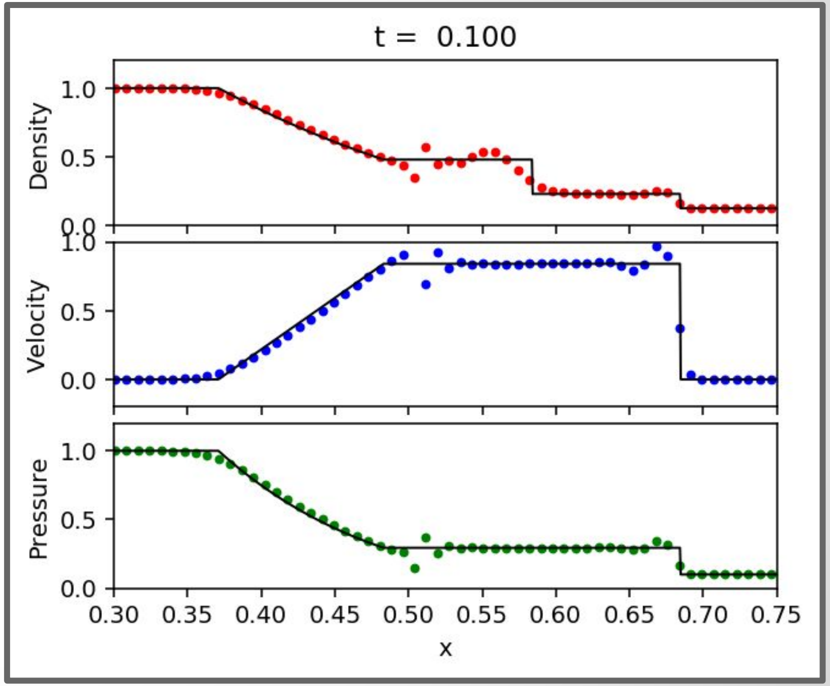
Lax-Wendroff

Demo: Sod Shock Tube

MUSCL-Hancock → much better!



Lax-Wendroff → unphysical oscillations...



Complete source codes:

- MUSCL-Hancock: https://gist.github.com/hyschive/0e3472c48df1e7eb0b2018a59bc2c111
- Lax-Wendroff: https://gist.github.com/hyschive/46bab6434f1b9b9aee23aeaeb71b90b6

Magnetohydrodynamics (MHD)

- **Ideal MHD:**

$$\frac{\partial \rho}{\partial t} + \boldsymbol{\nabla} \cdot (\rho \boldsymbol{v}) = 0$$ ← mass conservation

$$\frac{\partial (\rho \boldsymbol{v})}{\partial t} + \boldsymbol{\nabla} \cdot (\rho \boldsymbol{v}\boldsymbol{v} - \boldsymbol{B}\boldsymbol{B} + P^* \boldsymbol{I}) = 0$$ ← momentum conservation

$$\frac{\partial E}{\partial t} + \boldsymbol{\nabla} \cdot [(E + P^*)\boldsymbol{v} - \boldsymbol{B}(\boldsymbol{B} \cdot \boldsymbol{v})] = 0$$ ← energy conservation

$$\frac{\partial \boldsymbol{B}}{\partial t} - \boldsymbol{\nabla} \times (\boldsymbol{v} \times \boldsymbol{B}) = 0$$ ← induction eq. + ideal Ohm's law $\boldsymbol{E} + \boldsymbol{v} \times \boldsymbol{B} = 0$

- $E = e + \frac{1}{2}\rho v^2 + \frac{B^2}{2}, \quad P^* = P + \frac{B^2}{2}$
- **9 variables to be solved by the 8 equations above + equation of state**
- **Divergence-free constraint on the magnetic field:** $\boldsymbol{\nabla} \cdot \boldsymbol{B} = 0$

Flux-conservative Form for MHD

- $$\frac{\partial \boldsymbol{U}}{\partial t} + \frac{\partial \boldsymbol{F_x}}{\partial x} + \frac{\partial \boldsymbol{F_y}}{\partial y} + \frac{\partial \boldsymbol{F_z}}{\partial z} = 0,$$

$$\boldsymbol{U} = \begin{bmatrix} \rho \\ \rho v_x \\ \rho v_y \\ \rho v_z \\ E \\ B_x \\ B_y \\ B_z \end{bmatrix}, \quad \boldsymbol{F_x} = \begin{bmatrix} \rho v_x \\ \rho v_x^2 + P^* - B_x^2 \\ \rho v_x v_y - B_x B_y \\ \rho v_x v_z - B_x B_z \\ (E + P^*) v_x - B_x (\boldsymbol{B} \cdot \boldsymbol{v}) \\ 0 \\ v_x B_y - v_y B_x \\ v_x B_z - v_z B_x \end{bmatrix}$$
, similarly for $\boldsymbol{F_y}, \boldsymbol{F_z}$

- **Fluid conserved variables can be updated similarly using the finite-volume scheme for pure hydro**
- **Key question: how to <u>ensure the divergence-free constraint</u> when updating the magnetic field?**

Constrained Transport (CT) Method

- **Stokes' theorem:** $\int_A \frac{\partial \boldsymbol{B}}{\partial t} \cdot d\boldsymbol{A} = \int_A [\boldsymbol{\nabla} \times (\boldsymbol{v} \times \boldsymbol{B})] \cdot d\boldsymbol{A} = \oint_{\partial A} \boldsymbol{v} \times \boldsymbol{B} \cdot d\boldsymbol{l}$
  - **Electromotive force (EMF):** $\boldsymbol{\varepsilon} = -\boldsymbol{v} \times \boldsymbol{B}$
- **Integrate over cell area (e.g., $\Delta y \Delta z$) and time interval $\Delta t = t^{n+1}$ - t^n**

$$B^n_{x,i-1/2,j,k} \equiv \frac{1}{\Delta y \Delta z} \int_{z_{k-1/2}}^{z_{k+1/2}} \int_{y_{j-1/2}}^{y_{j+1/2}} B_x(x_{i-1/2}, y, z, t^n) dydz$$

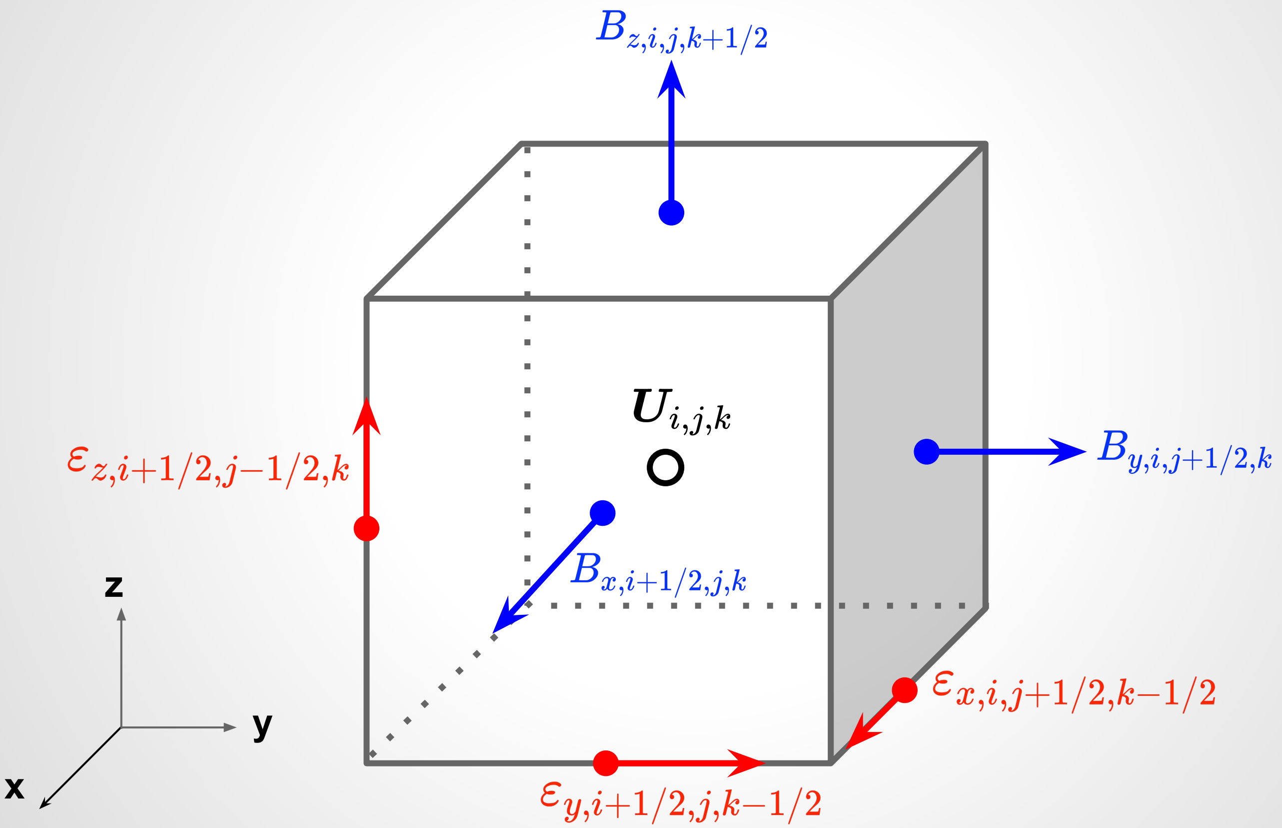
$$\varepsilon^{n+1/2}_{y,i-1/2,j,k-1/2} \equiv \frac{1}{\Delta y \Delta t} \int_{t^n}^{t^{n+1}} \int_{y_{j-1/2}}^{y_{j+1/2}} \varepsilon_y(x_{i-1/2}, y, z_{k-1/2}, t) dydt$$

$$\varepsilon^{n+1/2}_{z,i-1/2,j-1/2,k} \equiv \frac{1}{\Delta z \Delta t} \int_{t^n}^{t^{n+1}} \int_{z_{k-1/2}}^{z_{k+1/2}} \varepsilon_z(x_{i-1/2}, y_{j-1/2}, z, t) dzdt$$

Constrained Transport (CT) Method

- $$B^{n+1}_{x,i-1/2,j,k} = B^{n}_{x,i-1/2,j,k} - \frac{\Delta t}{\Delta y}\left(\varepsilon^{n+1/2}_{z,i-1/2,j+1/2,k} - \varepsilon^{n+1/2}_{z,i-1/2,j-1/2,k}\right) + \frac{\Delta t}{\Delta z}\left(\varepsilon^{n+1/2}_{y,i-1/2,j,k+1/2} - \varepsilon^{n+1/2}_{y,i-1/2,j,k-1/2}\right)$$
  - This form is again exact → similar to the finite-volume formulation
  - $B^{n}_{x,i-1/2,j,k}$: area-averaged magnetic field
  - $\varepsilon^{n+1/2}_{z,i-1/2,j\pm1/2,k}$, $\varepsilon^{n+1/2}_{y,i-1/2,j,k\pm1/2}$: time- and line-averaged EMF
- Similar expressions can be derived for $B^{n+1}_{y,i,j-1/2,k}$ & $B^{n+1}_{z,i,j,k-1/2}$
- Area-averaged magnetic field are located at the cell faces instead of centers → staggered grid

Staggered Grid in CT



Divergence Free in CT

- **Finite-volume representation of the divergence-free constraint:**

$$\frac{1}{\Delta x \Delta y \Delta z}\int_{V_{i,j,k}} (\nabla \cdot \boldsymbol{B}^n) dV = 0$$

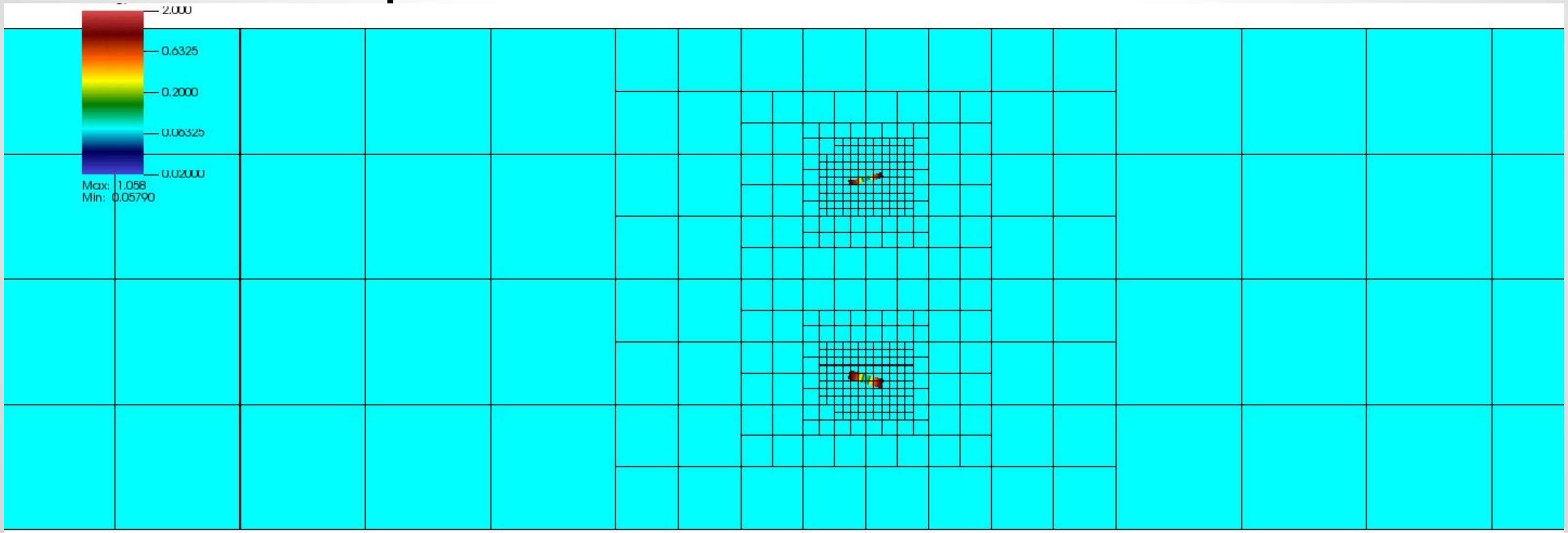
$$\rightarrow (\nabla \cdot \boldsymbol{B}^n)_{i,j,k} = \frac{B^n_{x,i+1/2,j,k} - B^n_{x,i-1/2,j,k}}{\Delta x} + \frac{B^n_{y,i,j+1/2,k} - B^n_{y,i,j-1/2,k}}{\Delta y} + \frac{B^n_{z,i,j,k+1/2} - B^n_{z,i,j,k-1/2}}{\Delta z} = 0$$

exact form

- **CT update guarantees** $\nabla \cdot B^{n+1} = \nabla \cdot B^n$
  - **Divergence-free constraint is preserved to the machine precision**
    - **But it must be satisfied in the initial condition**
  - **The exact way to compute EMF varies from scheme to scheme**

Adaptive Mesh Refinement (AMR)

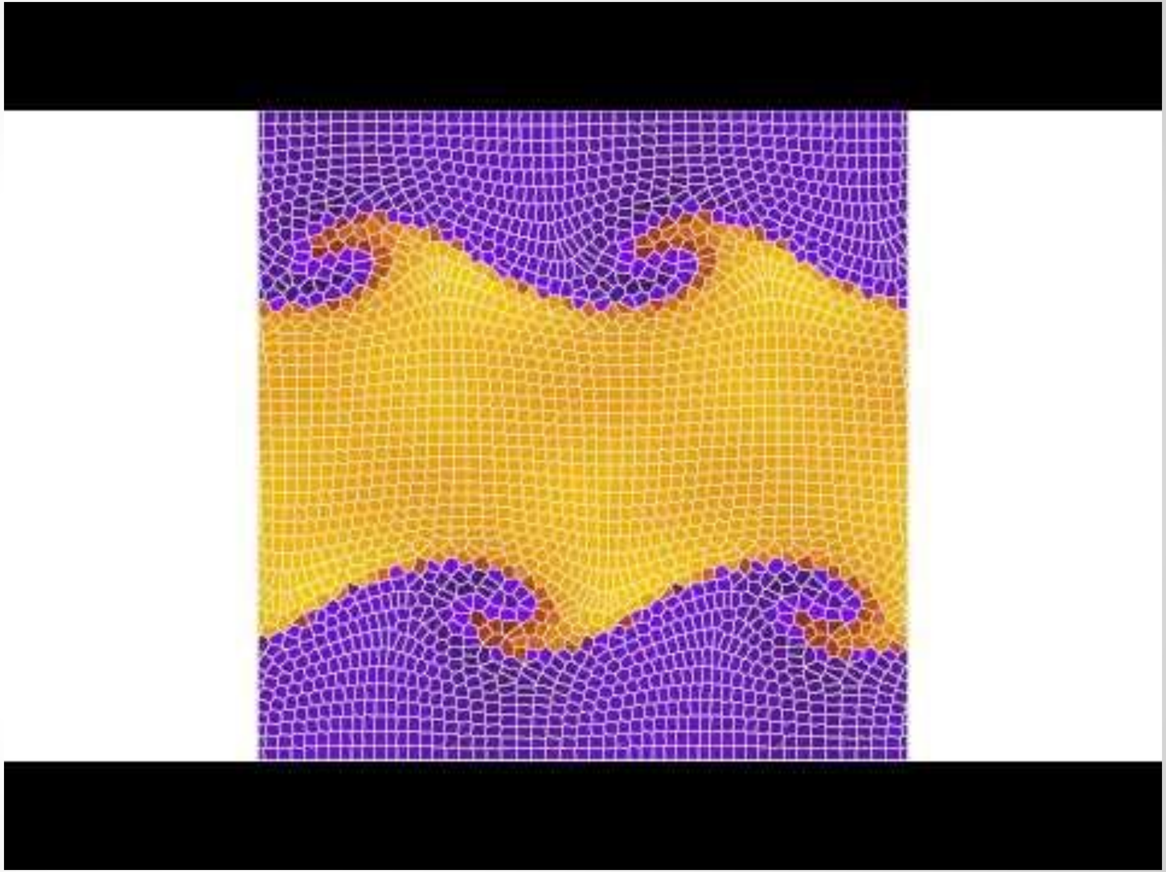
- Astrophysical simulations require a large dynamic range
  - $10^4 - 10^9$ spatial scales
  - Uniform-resolution simulations become impractical
- AMR: allow resolution to adjust locally and automatically
  - Problem-specific refinement criteria



Colliding active galactic nucleus jets using the GAMER code (Sandor, Schive, et al. 2017, ApJ)

Moving Mesh

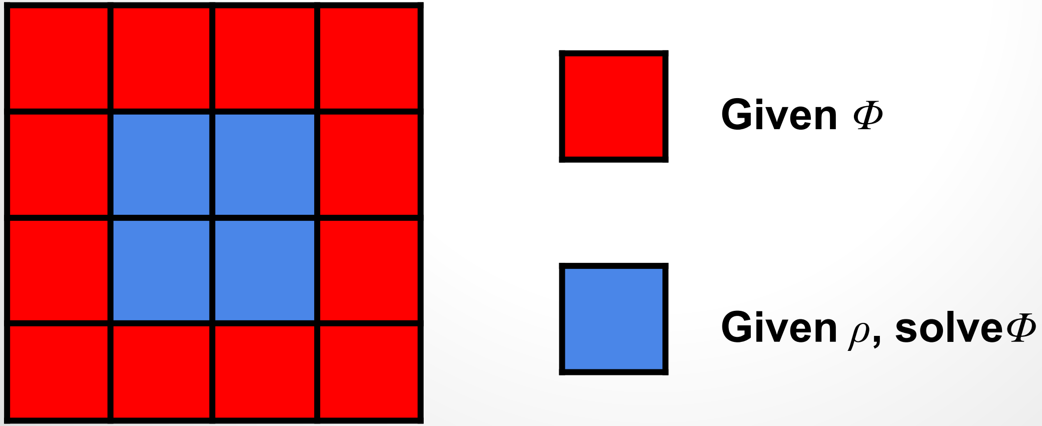
- Lagrangian instead of Eulerian coordinates
- Galilean invariant
- Unstructured mesh
- Finite-volume scheme



Kelvin-Helmholtz instability simulated with the Arepo code

Self-gravity

- Poisson equation: $\nabla^2 \phi(\boldsymbol{r}) = \rho(\boldsymbol{r})$
  - ρ: mass density, Φ: gravitational potential, assuming $4\pi G=1$
- Task: given ρ in V and Φ at ∂V, where V is the computational domain of interest and ∂V is the boundary $\rightarrow$ solve Φ in V



Self-gravity: Relaxation Methods

- $\nabla^2 \phi = \rho \rightarrow \boxed{\dfrac{\partial \phi}{\partial t} = \nabla^2 \phi - \rho}$ ← Diffusion eq. with source -*ρ*
  - Let the system relax until equilibrium is established $\dfrac{\partial \phi}{\partial t} = 0 \rightarrow \nabla^2 \phi = \rho$
  - 2D discrete form using a FTCS scheme (assuming $\Delta x = \Delta y = \Delta$):

$$\frac{\phi^{n+1}_{i,j} - \phi^n_{i,j}}{\Delta t} = \frac{1}{\Delta^2}\left(\phi^n_{i+1,j} + \phi^n_{i-1,j} + \phi^n_{i,j+1} + \phi^n_{i,j-1} - 4\phi^n_{i,j}\right) - \rho_{i,j}$$

  - CFL stability: $\Delta t \leq \Delta^2/4 \rightarrow$ let $\Delta t = \Delta^2/4$

$$\boxed{\phi^{n+1}_{i,j} = \frac{1}{4}\left(\phi^n_{i+1,j} + \phi^n_{i-1,j} + \phi^n_{i,j+1} + \phi^n_{i,j-1} - \Delta^2 \rho_{i,j}\right)}$$

Jacobi's method

→ Iterate until relaxed (convergence)

Self-gravity: Discrete Fourier Transform

- **Poisson eq. in 1D:** $\frac{\partial^2 \phi}{\partial x^2} = \rho$
- **Fourier transform:** $\partial/\partial x \rightarrow ik, \phi(x) \rightarrow \Phi(k), \rho(x) \rightarrow D(k)$

$$\Phi(k) = -\frac{D(k)}{k^2} \quad \rightarrow \quad \phi(x) = FT^{-1}(\Phi(k))$$

  - **Assuming periodic boundary conditions above**
  - **For isolated (vacuum) boundary conditions, it requires convolution of ρ(r) (with zero padding) and the Green's function r^{-1}**

Particles: What Do They Represent?

1. Planets, stars, supernovae, black holes
   a. Each particle represents a single point mass
2. Star clusters
   a. Each particle represents a bunch of stars
3. Dark matter
   a. Finite sampling of the phase space distribution function
   b. Can be either collisionless (CDM) or collisional (SIDM)
4. Gas → Smooth Particle Hydrodynamics (SPH)
   a. Lagrangian nature → adaptive resolution
   b. Mesh-free
   c. Self-gravity can be computed in the same way as other types of particles
5. Tracers
   a. Trace the trajectory of gas elements
6. Photons
   a. Radiation transfer

Particle Properties

1. Point-mass objects
   a. Two-body relaxation may be essential → collisional system
   b. Gravity diverges at the center → numerically challenging
   c. Binaries
2. Finite-sized objects
   a. Star clusters, dark matter
   b. Avoid two-body relaxation and binary formation → smooth out gravity in the short range (smoothing/softening length)
3. Particles can be created, destroyed, or scattered on-the-fly
4. Particle properties may change on-the-fly
   a. Mass, age, metalicity, spin, stellar composition, ...
5. Feedback
   a. Stellar wind, AGN jets, SN explosion, ...

Computing Self-gravity

- **Direct N-body:** $\boldsymbol{a_i} = G \sum_{j \neq i} m_j \frac{\boldsymbol{r_j} - \boldsymbol{r_i}}{|\boldsymbol{r_j} - \boldsymbol{r_i}|^3}$
  - **Computational complexity $O(N^2)$ → extremely expensive**
  - **Mostly used when particles represent point masses where very high accuracy is essential**
- **Particle Mesh (PM)**
  - **Deposit particle mass onto grids → grid-base Poisson solver → interpolate gravity back to particles**
- **Tree / Fast Multipole Method**
  - **Multipole expansion → Group distant particles into a single large particle (higher-order corrections such as quadrupole can be included)**
- **Hybrid Method: P^3M, TreePM**
  - **Long range: PM**
  - **Short range: direct N-body (P^3M) or tree (TreePM)**
  - **Be careful about connecting long- and short-range forces**

Orbit Integration

- **Kick operator K: update velocity while fixing position**

$$K(\Delta t)\begin{bmatrix}\boldsymbol{r}(t)\\ \boldsymbol{v}(t)\end{bmatrix}=\begin{bmatrix}\boldsymbol{r}(t)\\ \boldsymbol{v}(t)+\boldsymbol{a}\Delta t\end{bmatrix}$$

- **Drift operator D: update position while fixing velocity**

$$D(\Delta t)\begin{bmatrix}\boldsymbol{r}(t)\\ \boldsymbol{v}(t)\end{bmatrix}=\begin{bmatrix}\boldsymbol{r}(t)+\boldsymbol{v}(t)\Delta t\\ \boldsymbol{v}(t)\end{bmatrix}$$

- **KDK scheme:** $K(\Delta t/2)\ D(\Delta t)\ K(\Delta t/2)$

$$\begin{aligned}\boldsymbol{v}(t+\Delta t/2)&=\boldsymbol{v}(t)+\boldsymbol{a}(t)\Delta t/2\\ \boldsymbol{x}(t+\Delta t)&=\boldsymbol{x}(t)+\boldsymbol{v}(t+\Delta t/2)\Delta t\\ \boldsymbol{v}(t+\Delta t)&=\boldsymbol{v}(t+\Delta t/2)+\boldsymbol{a}(t+\Delta t)\Delta t/2\end{aligned}$$

⟺

Euler's scheme (1st order)

$$\begin{aligned}\boldsymbol{x}(t+\Delta t)&=\boldsymbol{x}(t)+\boldsymbol{v}(t)\Delta t\\ \boldsymbol{v}(t+\Delta t)&=\boldsymbol{v}(t)+\boldsymbol{a}(t)\Delta t\end{aligned}$$

  - **Equivalent to the Leapfrog scheme (2nd order)**
  - **Time reversibility**
  - **Symplectic nature → preserve a slightly perturbed Hamiltonian → good for long-term evolution**
  - **One force evaluation per time-step**

Code Snippets

Euler

```
#     calculate a(t)
      r     = ( x*x + y*y )**0.5
      a_abs = G*M/(r*r)
      ax    = -a_abs*x/r
      ay    = -a_abs*y/r

#     use v(t) and a(t) to update position
#     and velocity by dt
      x   = x + vx*dt
      y   = y + vy*dt
      vx  = vx + ax*dt
      vy  = vy + ay*dt
```

⇐ Be careful about the order of update

DKD

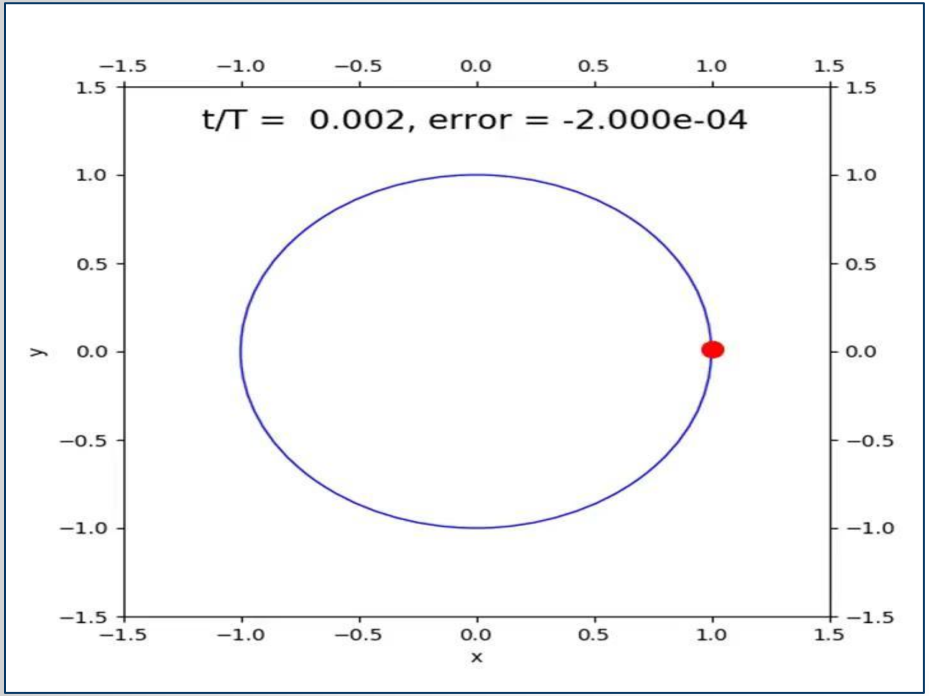
```
#     drift: update position by 0.5*dt
      x = x + vx*0.5*dt
      y = y + vy*0.5*dt

#     kick: calculate a(t+0.5*dt) and use that
#     to update velocity by dt
      r     = ( x*x + y*y )**0.5
      a_abs = G*M/(r*r)
      ax    = -a_abs*x/r
      ay    = -a_abs*y/r
      vx    = vx + ax*dt
      vy    = vy + ay*dt

#     drift: use v(t+dt) to update position
#     by another 0.5*dt
      x = x + vx*0.5*dt
      y = y + vy*0.5*dt
```

Complete source codes:

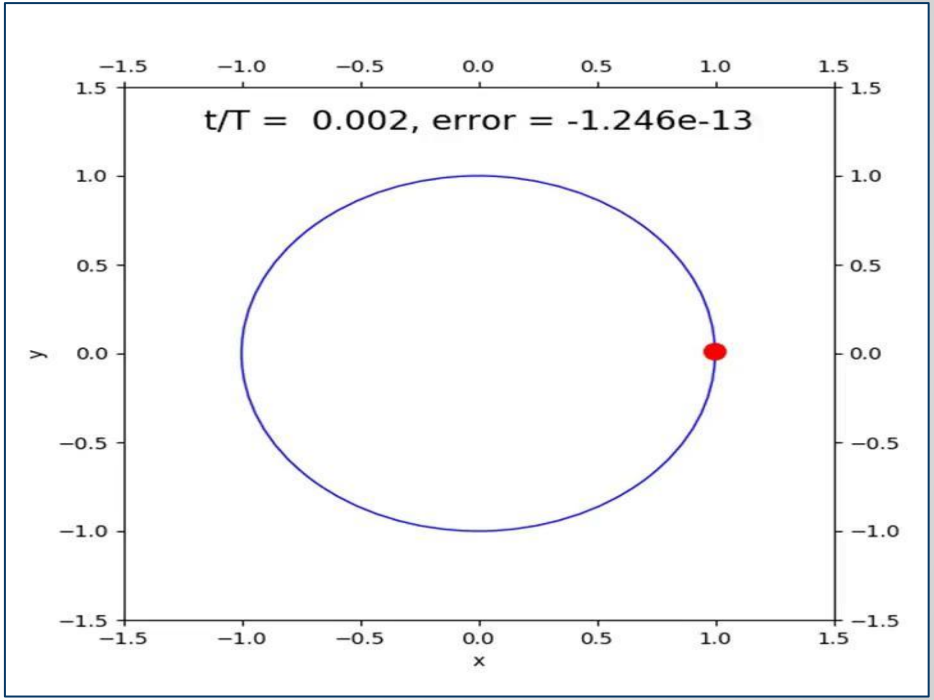
- Euler: https://gist.github.com/hyschive/5db0f4235f7ccabf5567e30a2dacca07
- DKD: https://gist.github.com/hyschive/b59143f14ee89d188a06a1ae29c9cfe7

Demo

Euler



DKD



Questions!