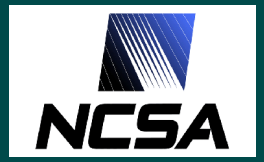
Computational Astrophysics

Lecture 2: Gasdynamics

Paul Ricker

University of Illinois at Urbana-Champaign
National Center for Supercomputing Applications
Urbana, Illinois USA



Navier-Stokes equations for gasdynamics

Mass

$$\frac{\partial}{\partial t}\rho + \nabla\cdot\left[\rho\,\mathbf{v}\right] = \nabla\cdot\left(D_\rho\nabla\rho\right)$$

Momentum

$$\frac{\partial}{\partial t}\left[\rho\,\mathbf{v}\right] + \nabla\cdot\left[\rho\,\mathbf{v}\,\mathbf{v}\right] + \nabla p + \rho\nabla\phi = \nabla\cdot\left(D_v\nabla\rho\,\mathbf{v}\right)$$

Energy

$$\frac{\partial}{\partial t}\left[\rho E\right] + \nabla\cdot\left[\left(\rho E + p\right)\mathbf{v}\right] - \rho\frac{\partial\phi}{\partial t} = \nabla\cdot\left(D_\epsilon\nabla\rho\epsilon\right) + \Lambda$$

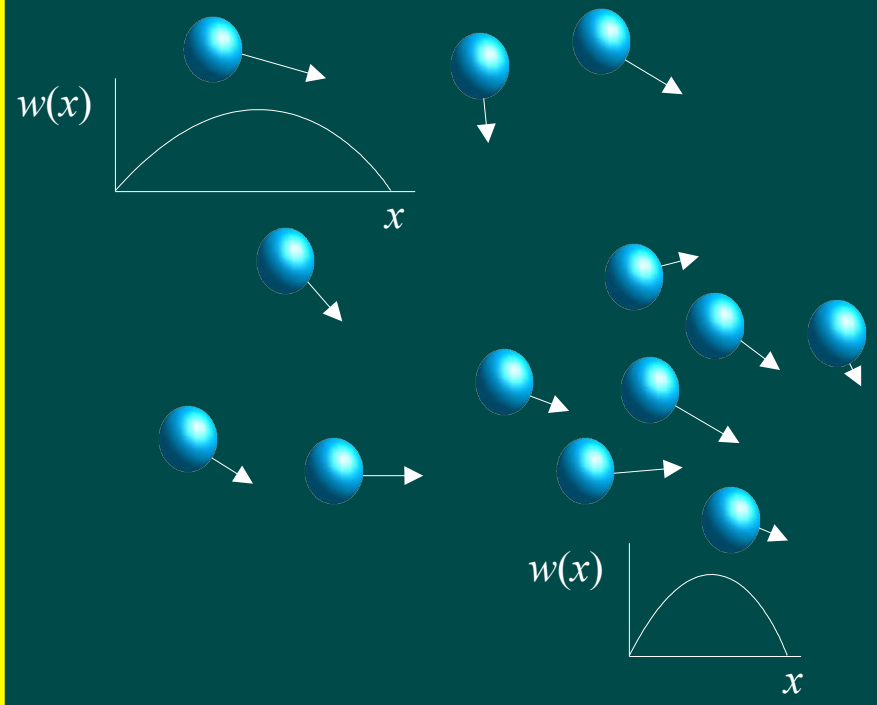
$$\frac{\partial}{\partial t}\left[\rho\epsilon\right] + \nabla\cdot\left[\left(\rho\epsilon + p\right)\mathbf{v}\right] - \mathbf{v}\cdot\nabla p = \nabla\cdot\left(D_\epsilon\nabla\rho\epsilon\right) + \Lambda$$

$$\rho E \equiv \rho\epsilon + \frac{1}{2}\rho v^2 + \rho\phi$$

$$\rho\epsilon = \frac{p}{\gamma - 1} = \frac{\rho k T}{(\gamma - 1)\mu}$$

Diffusive terms → 0 ⇒ Euler equations

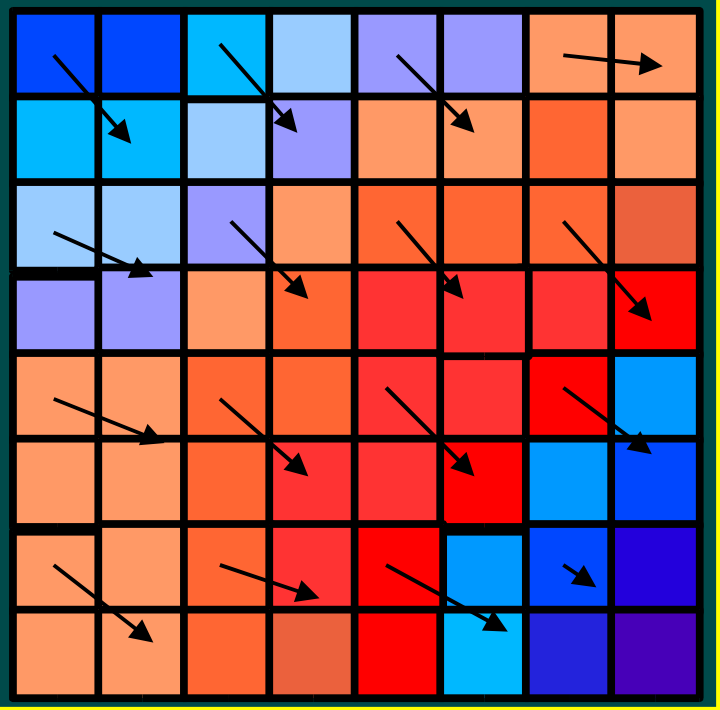
Classes of gasdynamics solvers



Smoothed particle hydrodynamics (SPH)

Uses particles as moving interpolation centers

$\rho(x,y)$



Eulerian grid-based hydrodynamics

Fluid quantities defined on a mesh

Finite difference (staggered mesh) or finite volume representation

Classification of partial differential equations

Consider the most general linear, second-order PDE with constant coefficients:

$$a\frac{\partial^2 q}{\partial x^2}+b\frac{\partial^2 q}{\partial xy}+c\frac{\partial^2 q}{\partial y^2}+d\frac{\partial q}{\partial x}+e\frac{\partial q}{\partial y}+fq=g$$

Such equations are classified in analogy with conic sections:

$$b^2-4ac\begin{cases}<0 & \Rightarrow \text{elliptic equation (two complex characteristic speeds)}\\ =0 & \Rightarrow \text{parabolic equation (one real characteristic speed)}\\ >0 & \Rightarrow \text{hyperbolic equation (two real characteristic speeds)}\end{cases}$$

The methods used to solve each class are quite different.

Elliptic equations

Example: Poisson equation $\nabla^2\phi=4\pi G\rho$

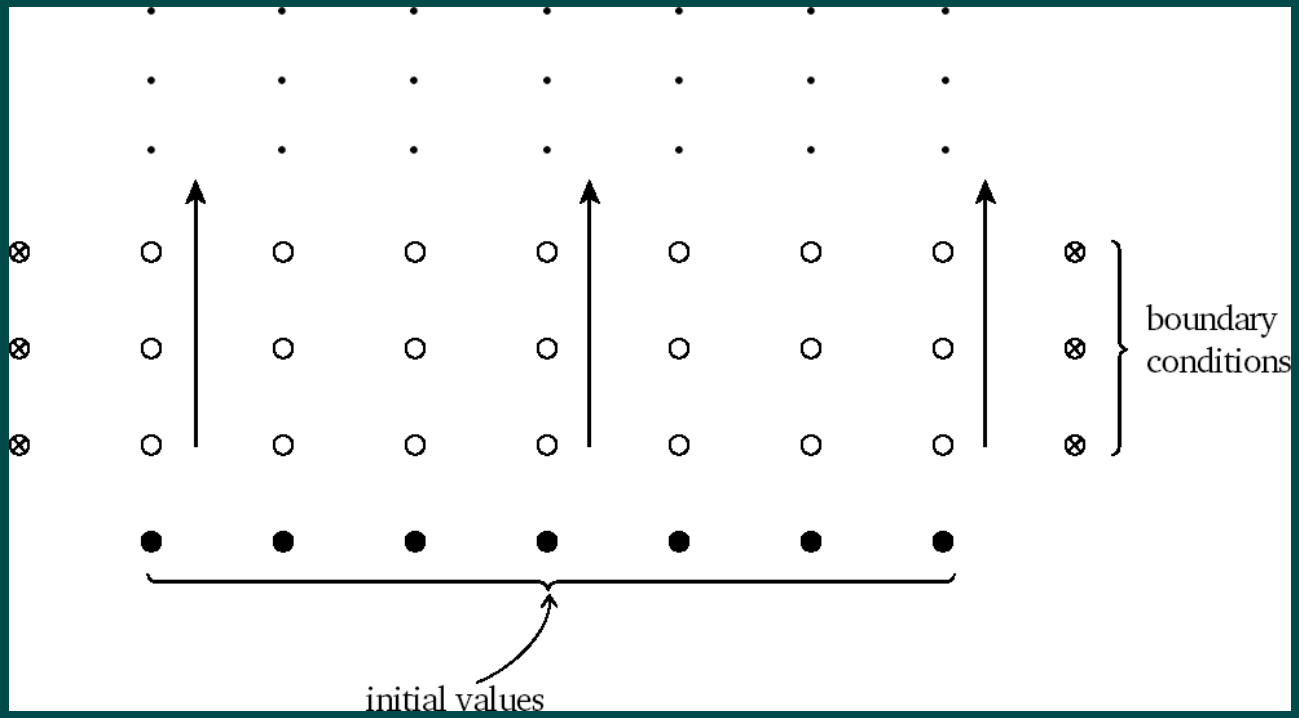
Parabolic equations

Example: Diffusion equation $\frac{\partial T}{\partial t}=\kappa\nabla^2 T$

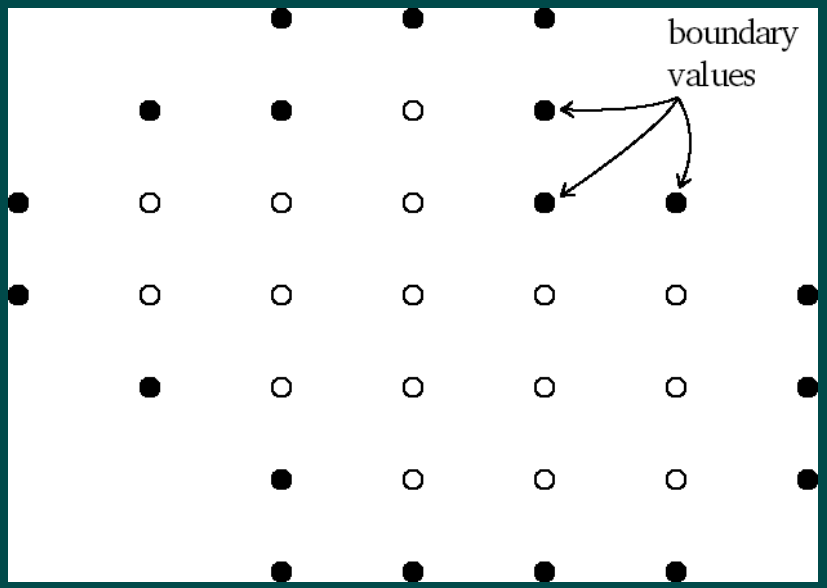
Hyperbolic equations

Example: Wave equation $\frac{\partial^2 \rho}{\partial t^2}-u^2\frac{\partial^2 \rho}{\partial x^2}=0$

Initial and boundary conditions



Initial-value problems
Parabolic
Hyperbolic



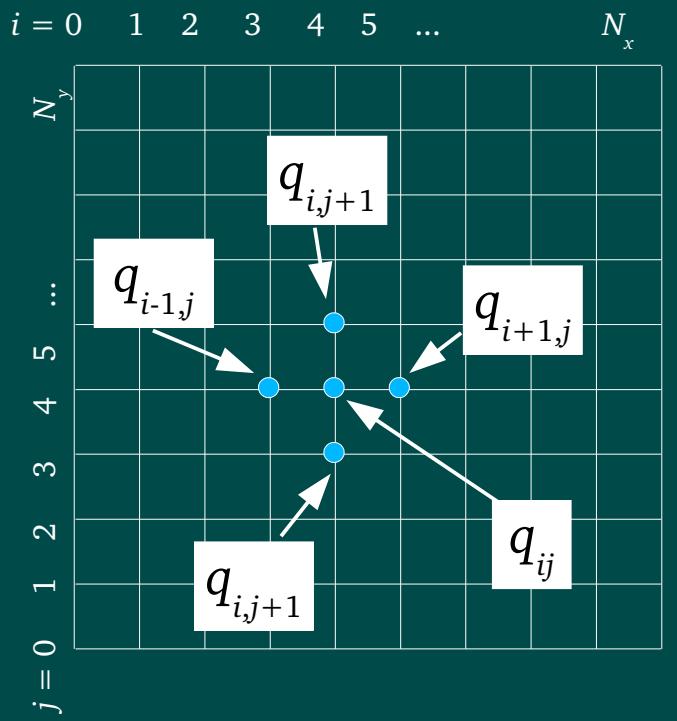
Boundary-value problems
Elliptic

Discretization of PDEs

Finite-difference methods

Work with values of the solution at a number of specified points:

$$q_{ij} \equiv q(x_i, y_j)$$
$$x_i = i\,\Delta x \qquad i = 0 \ldots N_x$$
$$y_j = j\,\Delta y \qquad j = 0 \ldots N_y$$



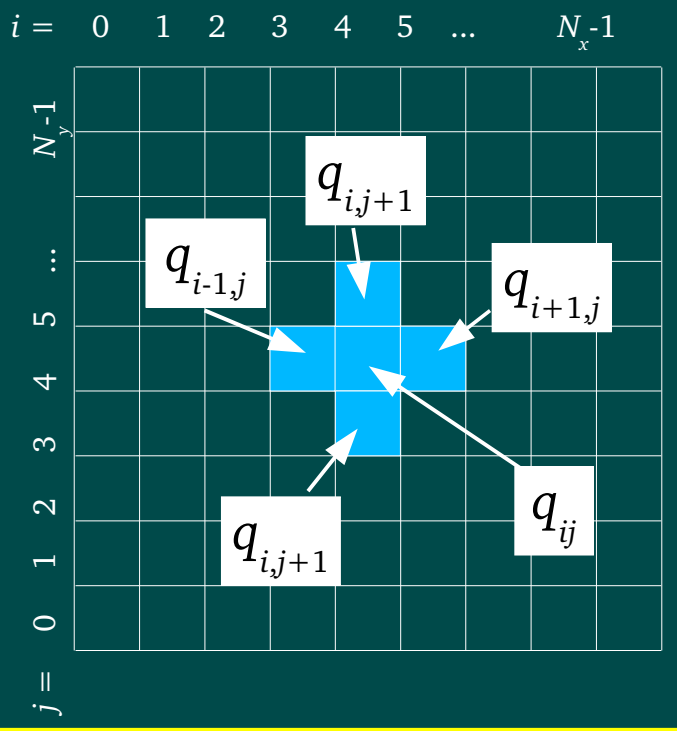
Finite-volume methods

Work with cell averages of the solution:

$$q_{ij} \equiv \frac{1}{\Delta x\,\Delta y} \int_{x_{i-1/2}}^{x_{i+1/2}} \int_{y_{j-1/2}}^{y_{j+1/2}} q(x, y)\,dx\,dy$$

$$x_i = \left(i + \frac{1}{2}\right)\Delta x \qquad i = 0 \ldots N_x - 1$$

$$y_j = \left(j + \frac{1}{2}\right)\Delta y \qquad j = 0 \ldots N_y - 1$$



Finite volume methods – conservation form

We can write a general conservation law in the form

$$\frac{\partial q}{\partial t}+\nabla\cdot\boldsymbol{F}(q,t)=0$$

For example, for the Euler equations we have

$$q=\begin{Bmatrix}\rho\\ \rho u_x\\ \rho u_y\\ \rho u_z\\ \rho E\end{Bmatrix}\qquad \boldsymbol{F}(q)=\begin{Bmatrix}\rho u_x & \rho u_y & \rho u_z\\ \rho u_x^2+P & \rho u_x u_y & \rho u_x u_z\\ \rho u_x u_y & \rho u_y^2+P & \rho u_y u_z\\ \rho u_x u_z & \rho u_y u_z & \rho u_z^2+P\\ (\rho E+P)u_x & (\rho E+P)u_y & (\rho E+P)u_z\end{Bmatrix}$$

We can integrate any one of these equations over a cell volume to obtain

$$\frac{\partial q_{ijk}}{\partial t}+\frac{1}{\Delta x}\left[\boldsymbol{F}_{i+1/2,jk}-\boldsymbol{F}_{i-1/2,jk}\right]+\frac{1}{\Delta y}\left[\boldsymbol{F}_{i,j+1/2,k}-\boldsymbol{F}_{i,j-1/2,k}\right]+\frac{1}{\Delta z}\left[\boldsymbol{F}_{i,j,k+1/2}-\boldsymbol{F}_{i,j,k-1/2}\right]=0$$

where q_{ijk} is a finite-volume quantity. The function $\boldsymbol{F}_{i+1/2,jk}$ is the average of $\boldsymbol{F}(q)$ over the face between cells ijk and $i+1,jk$:

$$\boldsymbol{F}_{i+1/2,jk}(t)=\frac{1}{\Delta y\,\Delta z}\int_{y_{j-1/2}}^{y_{j+1/2}}\int_{z_{k-1/2}}^{z_{k+1/2}}\boldsymbol{F}(q,t)\,dy\,dz$$

Properties of numerical methods

Consistency

Define the local truncation error of a method by substituting a Taylor expansion for each of the terms in a difference equation. If the lowest-order error terms are of order $O(\Delta x^p + \Delta t^p)$, the difference operator is said to be locally *p*th-order.

If the local truncation error goes to zero as $\Delta x, \Delta t \to 0$, the method is consistent.

Stability

A method is stable if, as Δx and Δt are reduced, the method produces a result that tends toward some finite limit.

Convergence

If the global error of a solution (defined in some fashion) goes to zero as fast as $\Delta x^p + \Delta t^p$, the method is *p*th-order convergent.

Lax Equivalence Theorem: if and only if a method is consistent and stable, it is convergent.

Explicit vs. implicit methods

How to handle time integration? Two first-order methods:

$$\frac{q_{ijk}^{n+1}-q_{ijk}^{n}}{\Delta t}=-\frac{1}{\Delta x}\left[\boldsymbol{F}_{i+1/2,\,jk}^{n}-\boldsymbol{F}_{i-1/2,\,jk}^{n}\right]$$ Explicit (FTCS) – solve directly

$$\frac{q_{ijk}^{n+1}-q_{ijk}^{n}}{\Delta t}=-\frac{1}{\Delta x}\left[\boldsymbol{F}_{i+1/2,\,jk}^{n+1}-\boldsymbol{F}_{i-1/2,\,jk}^{n+1}\right]$$ Implicit (BTCS) – solve by matrix inversion

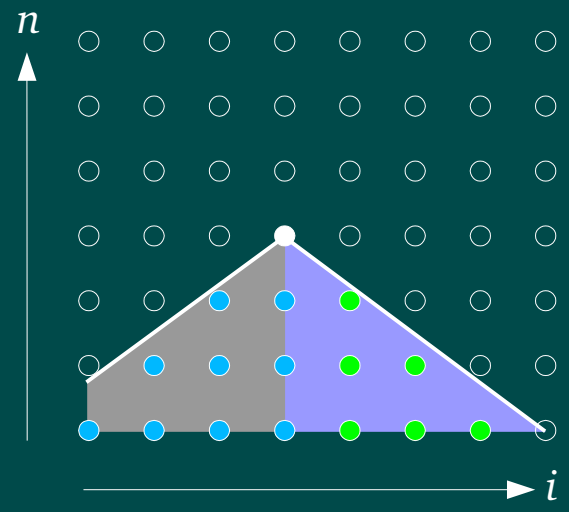
For numerical stability, explicit methods require CFL criterion:

$$\left|\frac{F_{ijk}^{n}\,\Delta t}{q_{ijk}^{n}\,\Delta x}\right|<\sigma \qquad \text{for advection:} \quad \left|\frac{u\,\Delta t}{\Delta x}\right|<\sigma$$

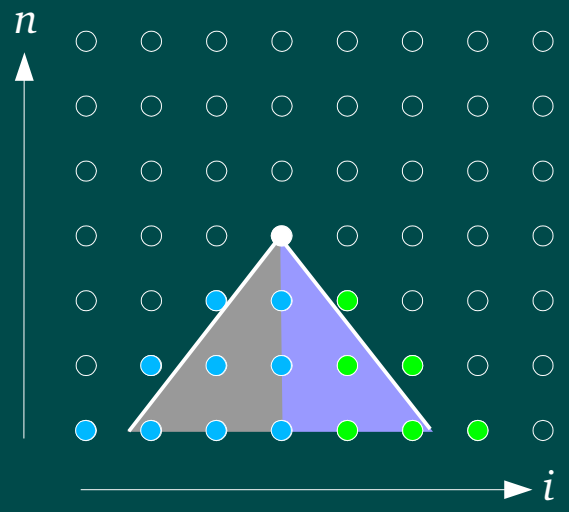
where $\sigma \sim 1$. Implicit methods are unconditionally stable.

Domains of dependence

	$u>0$	$u<0$
Numerical (difference eqn)	●	●
Continuum (PDE)	◢	◣



$|u\Delta t/\Delta x| > 1$ unstable



$|u\Delta t/\Delta x| < 1$ stable

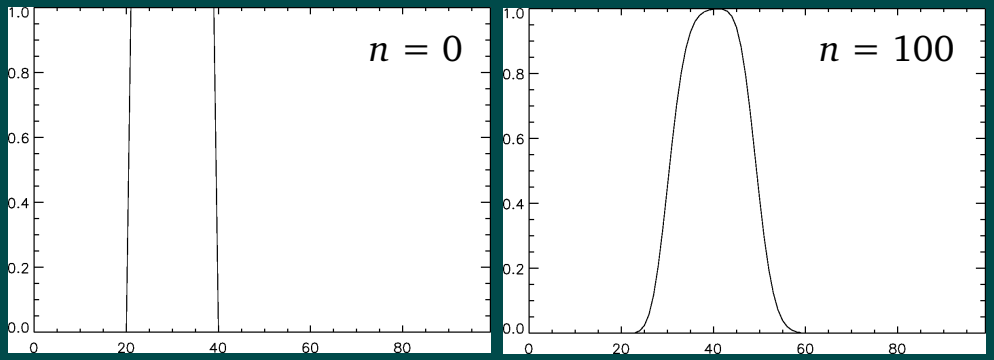
"Classical" methods

Simplest differencing schemes do poorly at flow discontinuities. Consider scalar advection equation

$$\frac{\partial \rho}{\partial t} + u\frac{\partial \rho}{\partial x} = 0, \; u > 0$$

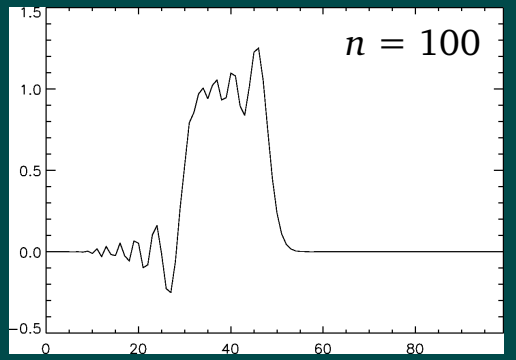
Upwind ($O(\Delta t)$):

$$\rho_i^{n+1} = \rho_i^n - \frac{u\,\Delta t}{\Delta x}\left(\rho_i^n - \rho_{i-1}^n\right)$$



Lax-Wendroff ($O(\Delta t^2)$):

$$\rho_i^{n+1} = \rho_i^n - \frac{u\,\Delta t}{2\,\Delta x}\left(\rho_{i+1}^n - \rho_{i-1}^n\right) + \frac{u^2\,\Delta t^2}{2\,\Delta x^2}\left(\rho_{i+1}^n - 2\,\rho_i^n + \rho_{i-1}^n\right)$$

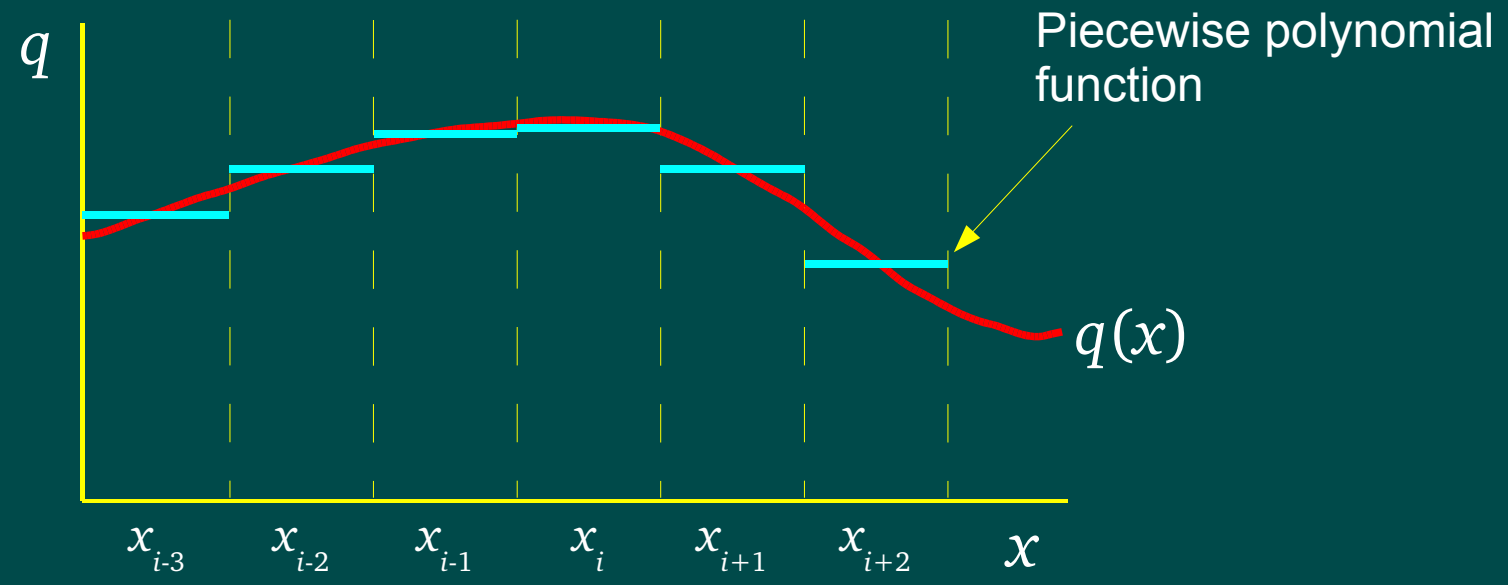


Odd-order methods → diffusive errors
Even-order methods → dispersive errors

Godunov methods

Strategy:

- Treat the solution as piecewise polynomial:



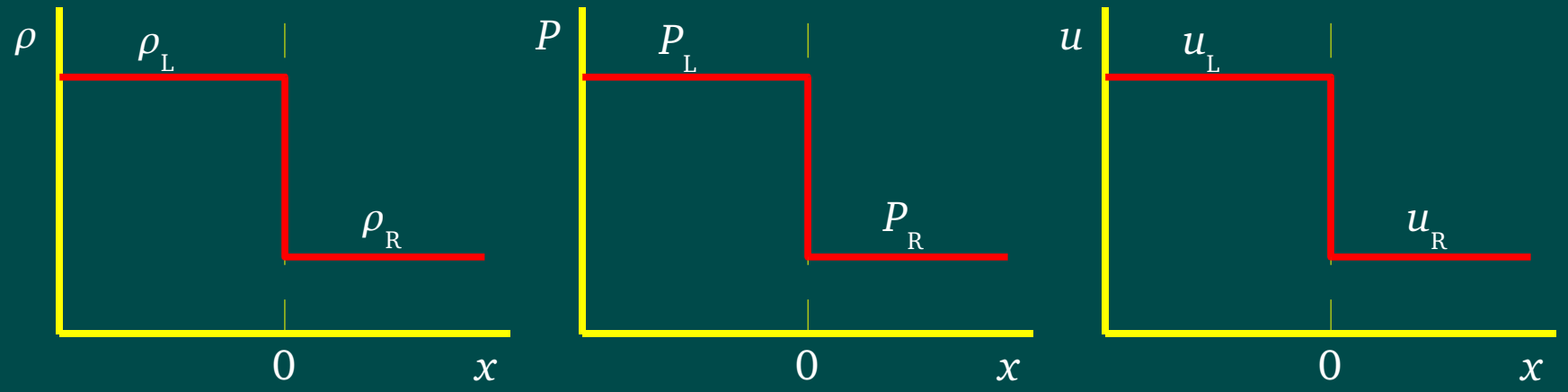
- Solve the time evolution over $[t_n, t_{n+1}]$ for this piecewise function *exactly*.
- The exact solution can then be used to produce fluxes.

Varieties:

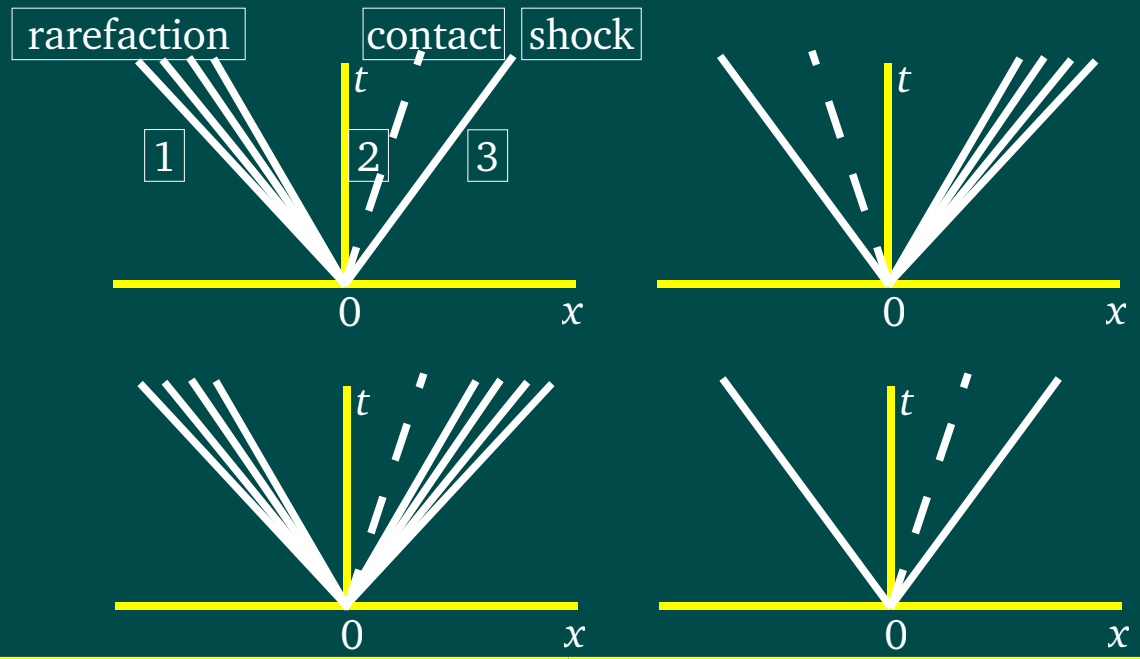
- Piecewise constant (Godunov's method, Godunov 1950)
- Piecewise linear (e.g., MUSCL, van Leer 1979)
- Piecewise parabolic (e.g., PPM, Colella & Woodward 1984)

The Riemann problem

Initial conditions: arbitrary discontinuity

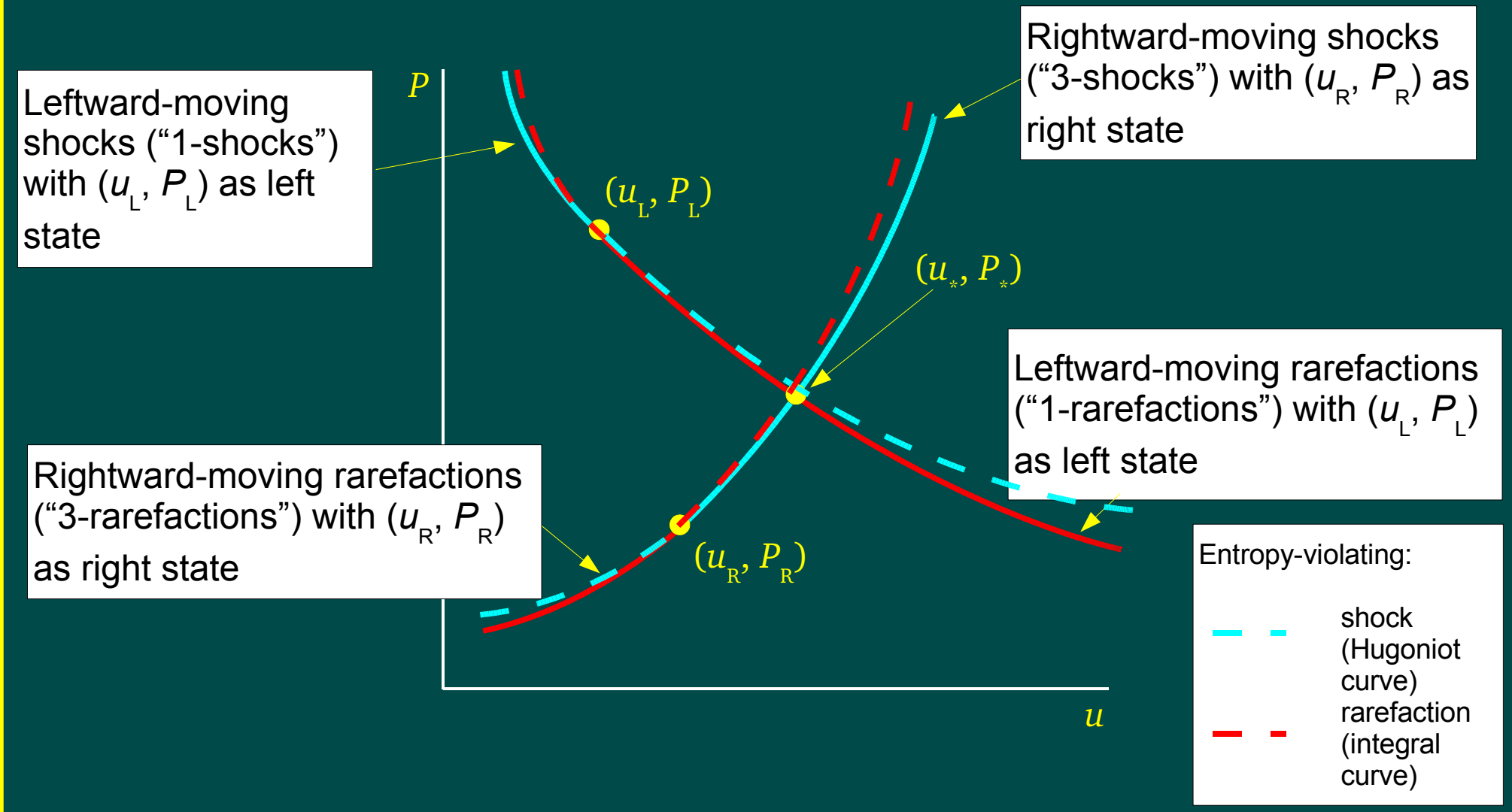


Admissible solutions: self-similar; three nonlinear waves



Solving the Riemann problem

Two states on either side of a nonlinear wave are connected by the *Hugoniot adiabat* (for shocks) or the *integral curve* of the appropriate characteristic (for rarefactions)

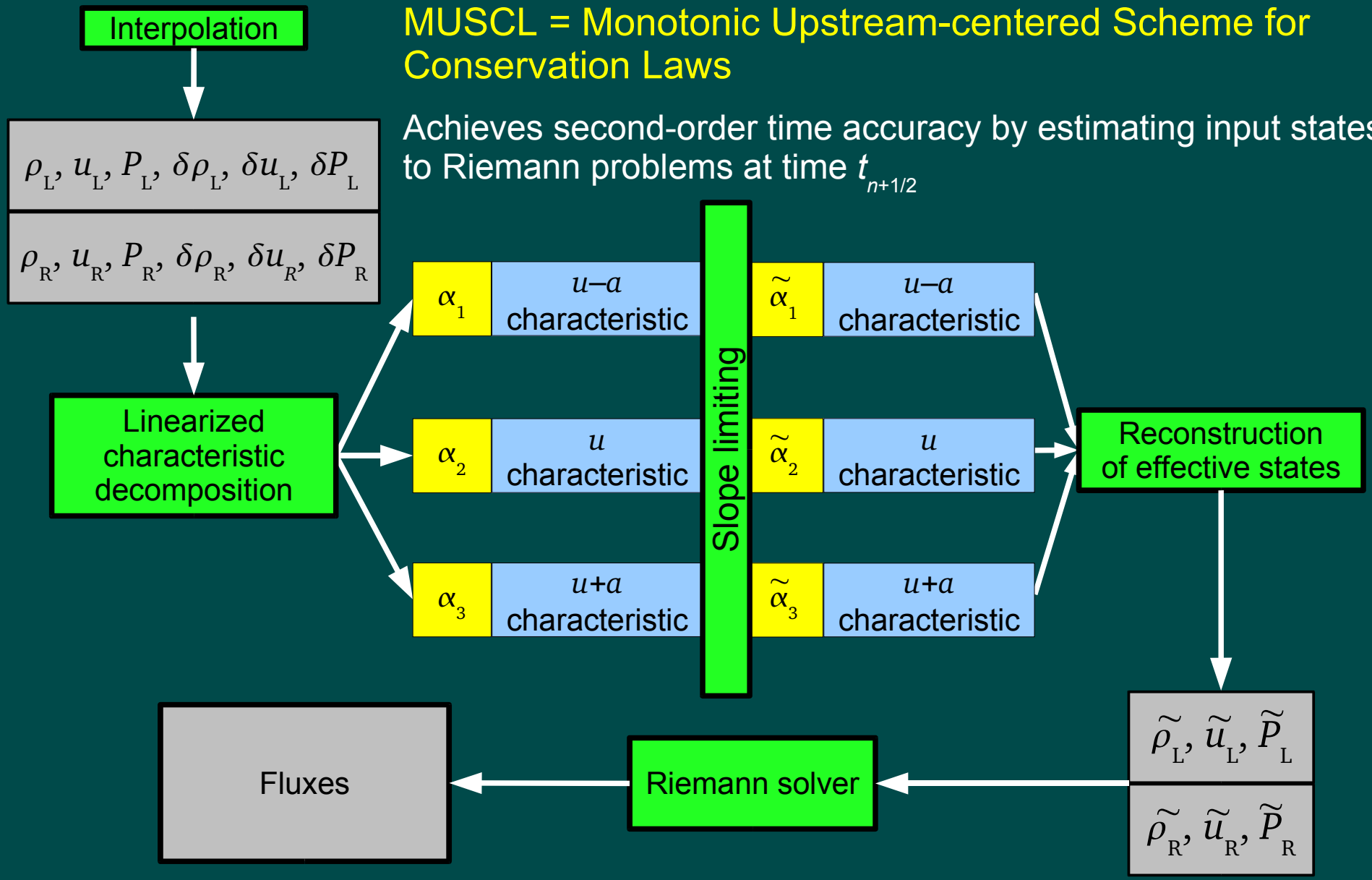


Generalization to higher order in *x* and *t*

MUSCL approach (van Leer 1980s)

MUSCL = Monotonic Upstream-centered Scheme for Conservation Laws

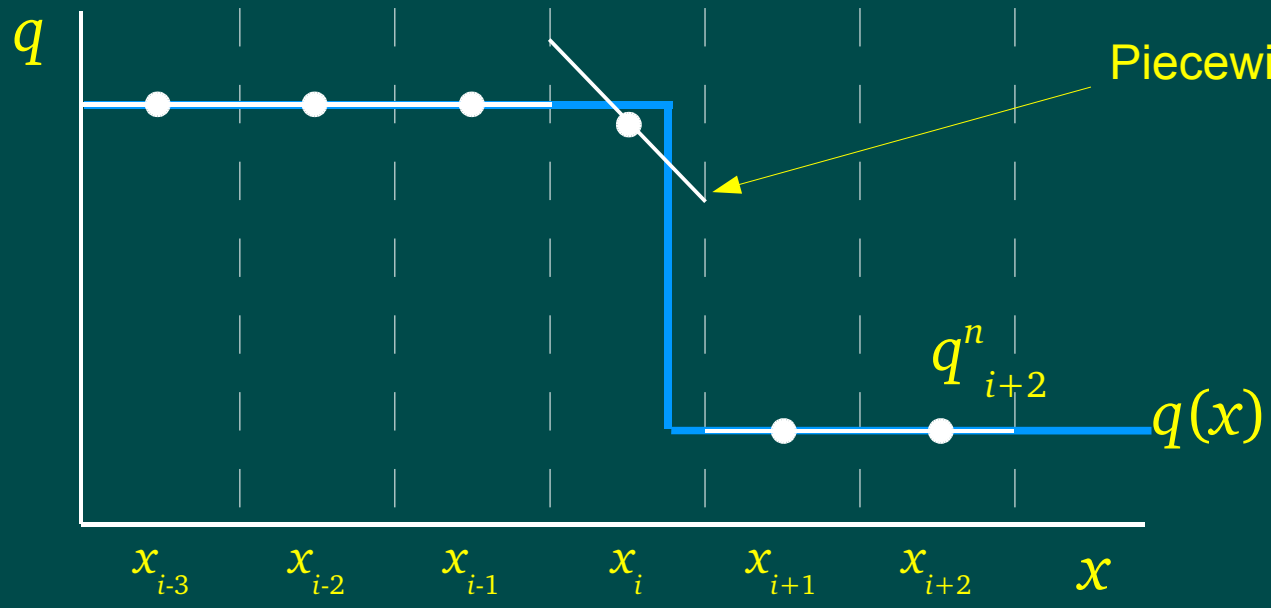
Achieves second-order time accuracy by estimating input states to Riemann problems at time $t_{n+1/2}$



The need to limit slopes

As long as $\frac{1}{\Delta x}\int_{x_{i-1/2}}^{x_{i+1/2}} q(x,t_n)\,dx = q_i^n$ for our polynomials, slopes can be whatever we need

Notice that at discontinuities, divided differences give meaningless slopes:



Unless we “flatten” the interpolating polynomial at discontinuities, we will introduce oscillations at these locations.

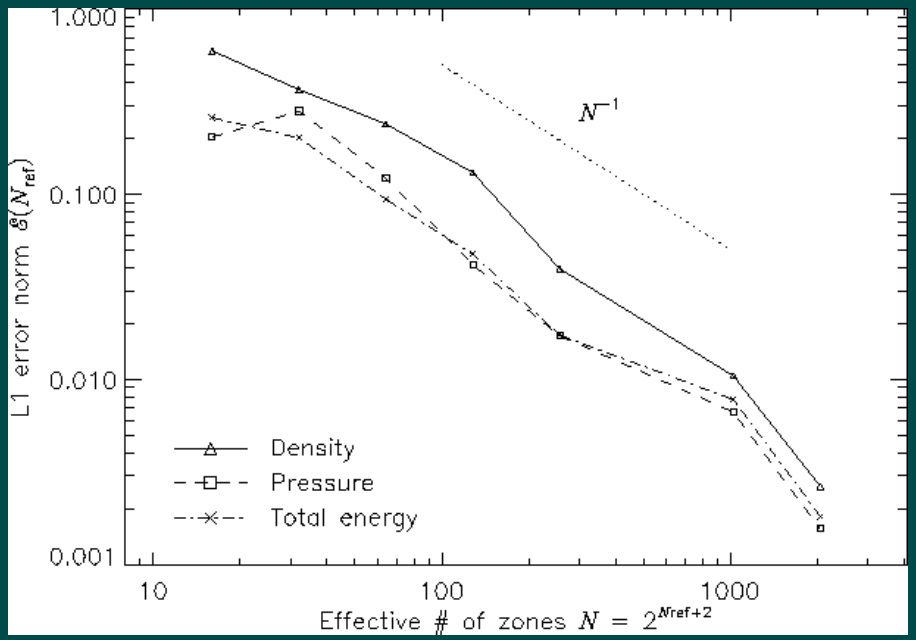
We only want to do this when we have to...

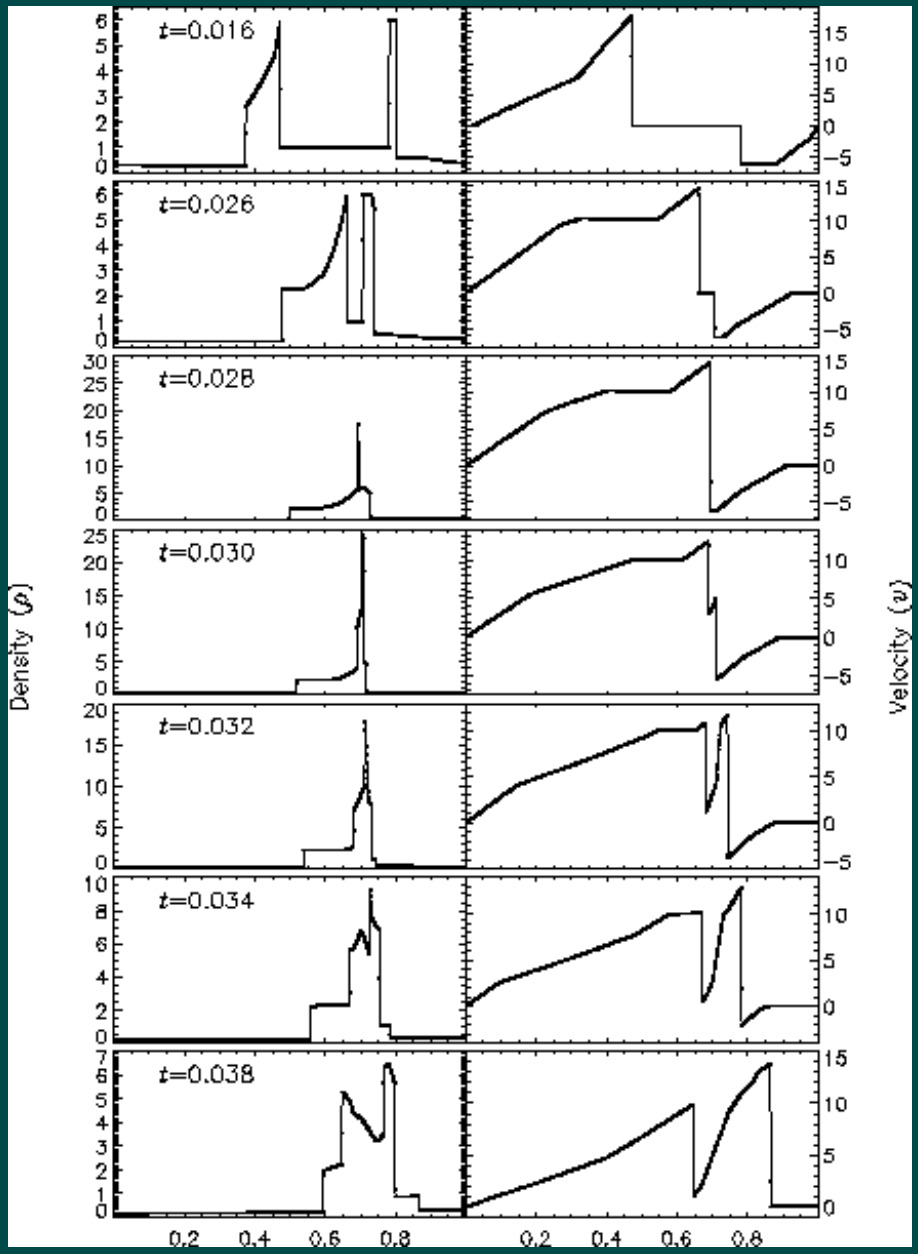
Piecewise parabolic method (PPM)

Colella & Woodward (1984)

- Piecewise parabolic interpolants
- MUSCL approach to time integration
- Slope limiting and (weak) artificial viscosity at strong shocks
- Contact steepening

Example: interaction of two strong blast waves (Woodward & Colella 1984) computed using FLASH (Fryxell et al. 2000)





Operator splitting (fractional-step method)

Suppose we have a difference operator D with truncation error $O(\Delta t)$ or better that can be written

$$D = D_1 + D_2 + D_3$$

Then

$$D[q] = D_1[D_2[D_3[q]]] + O(\Delta t)$$

If D is $O(\Delta t^2)$, we can do better by symmetrizing (Strang splitting):

$$D[q] = D_1^{1/2}[D_2^{1/2}[D_3^{1/2}[D_3^{1/2}[D_2^{1/2}[D_1^{1/2}[q]]]]]] + O(\Delta t^2)$$

Apply each operator for ½ Δt. Operators can be

- Different directions: $D_1 = \frac{\partial \rho u_x}{\partial x}, \; D_2 = \frac{\partial \rho u_x}{\partial y}, \; D_3 = \frac{\partial \rho u_x}{\partial z}$
- Different physics: $D_1 = \frac{\partial \rho u}{\partial x}, \; D_2 = \nu \frac{\partial^2 \rho u}{\partial x^2}, \; D_3 = \Lambda(\rho)$

Boundary conditions for CFD – practicalities

Typically we establish *ghost cells* around the boundary of our domain and set their values at the beginning of each timestep, independent of the integration scheme. Common types of boundary condition include:

Periodic boundaries

Here we require that all components of the solution (ρ, P, $\mathbf{u}$, etc.) be periodic functions of the computational box. So we set

$$\rho_0 = \rho_N \qquad P_0 = P_N \qquad u_{x0} = u_{xN} \qquad u_{y0} = u_{yN} \qquad u_{z0} = u_{zN}$$
$$\rho_{-1} = \rho_{N-1} \qquad P_{-1} = P_{N-1} \qquad u_{x,-1} = u_{x,N-1} \qquad u_{y,-1} = u_{y,N-1} \qquad u_{z,-1} = u_{z,N-1}$$
$$\rho_1 = \rho_{N+1} \qquad P_1 = P_{N+1} \qquad u_{x,1} = u_{x,N+1} \qquad u_{y,1} = u_{y,N+1} \qquad u_{z,1} = u_{z,N+1}$$
$$\ldots$$

We include one ghost cell on each side for each point in our stencil on that side. For example: upwind and Lax-Wendroff require one ghost cell on each side (three-point stencil).

Boundary conditions – 2

Outflow boundaries

Here we wish to allow material to flow out of the grid, but not to flow onto the grid. For supersonic flows we can set all gradients to zero on the boundary:

$$\rho_{-1} = \rho_0 \quad P_{-1} = P_0 \quad u_{x,-1} = u_{x,0} \quad u_{y,-1} = u_{y,0} \quad u_{z,-1} = u_{z,0}$$
$$\rho_{N+1} = \rho_N \quad P_{N+1} = P_N \quad u_{x,N+1} = u_{x,N} \quad u_{y,N+1} = u_{y,N} \quad u_{z,N+1} = u_{z,N}$$
$$\ldots$$

For subsonic flows this is usually not sufficient: waves can be reflected from the boundary. Characteristic-tracing methods are sometimes used in this case.

Reflecting boundaries

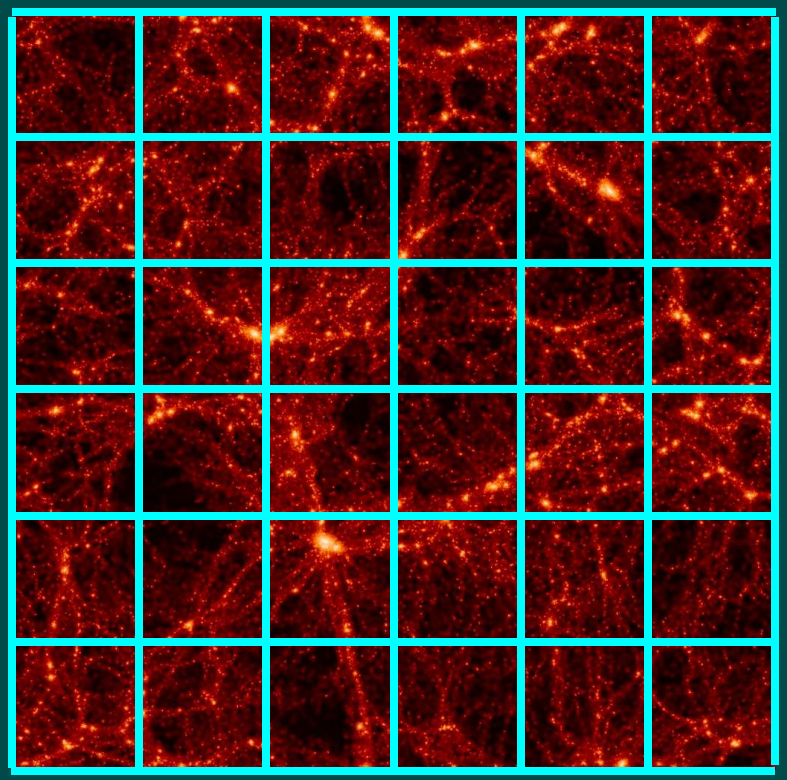
These are used for solid surfaces and symmetry axes. The Euler equations preserve reflection symmetry, so reflect density, pressure, and parallel velocity as even functions and velocity normal to the surfaces as an odd function.

$$u_{x0} = u_{xN} = 0$$
$$\rho_{-1} = \rho_1 \quad P_{-1} = P_1 \quad u_{x,-1} = -u_{x,1} \quad u_{y,-1} = u_{y,1} \quad u_{z,-1} = u_{z,1}$$
$$\rho_{N+1} = \rho_{N-1} \quad P_{N+1} = P_{N-1} \quad u_{x,N+1} = -u_{x,N-1} \quad u_{y,N+1} = u_{y,N-1} \quad u_{z,N+1} = u_{z,N-1}$$
$$\ldots$$

The need for mesh refinement

Ideally we would cover the Universe with a uniform mesh:

- Numerical diffusivity/resistivity constant
- Properties mathematically simple
- Easy to parallelize (domain decomposition)



But this would be a colossal waste of resources!

For explicit hydro schemes in 3D,

$$\text{Storage} \propto N^3$$
$$\text{Cost} \propto N^4$$

If we wanted to resolve interstellar distances in a cosmological volume, we would need

$$\left.\begin{aligned} L &\sim 1 \text{ Gpc} \\ \Delta &\sim 1 \text{ pc} \end{aligned}\right\} \Rightarrow N \sim 10^9$$

We would need 10^{28} bytes per variable... a computer requiring 1 ns to update each zone would take 3×10^{29} yr for one timestep! And most of it would be voids...

Adaptive mesh types

Types of mesh refinement

- r refinement – move or stretch the mesh points

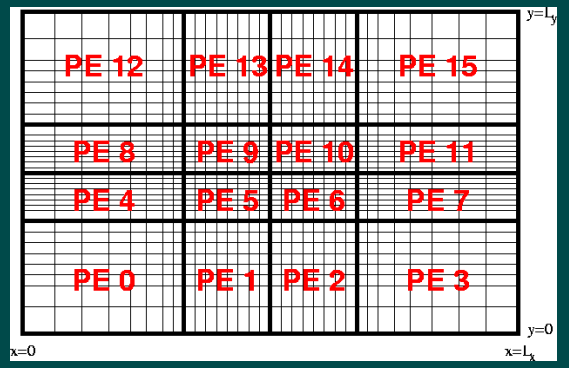
  Nonuniform meshes
  Lagrangian meshes
  Arbitrary Lagrangian-Eulerian (ALE)

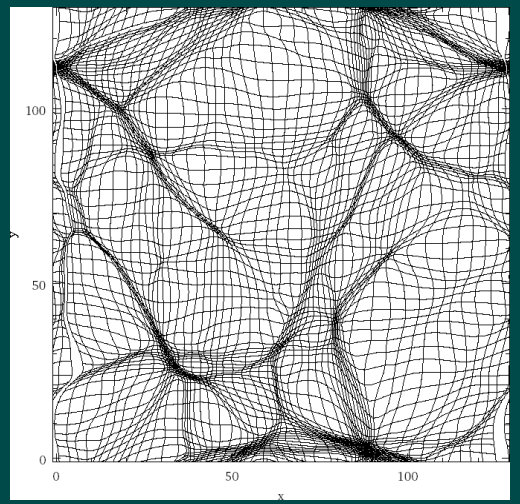
- p refinement – adjust the order of the method

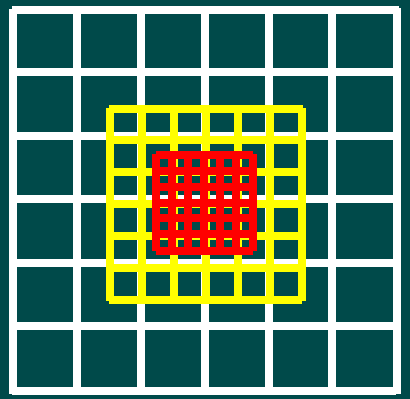
  Discontinuous Galerkin
  Spectral elements

- h refinement – change the mesh spacing

  Nested grids
  Adaptive mesh refinement (AMR)
  Finite elements



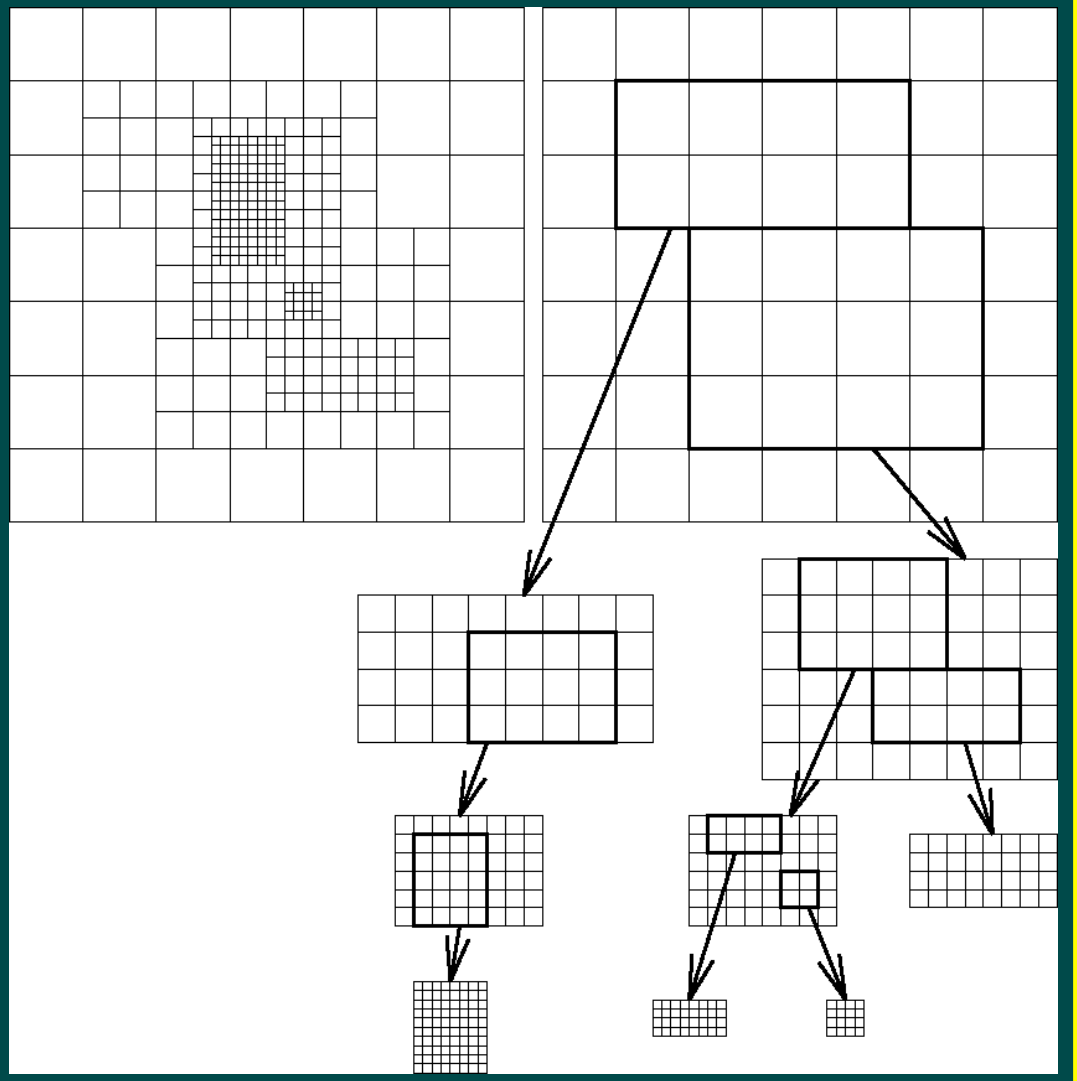




Block-structured adaptive mesh refinement

Berger & Oliger (1984)
Berger & Colella (1989)

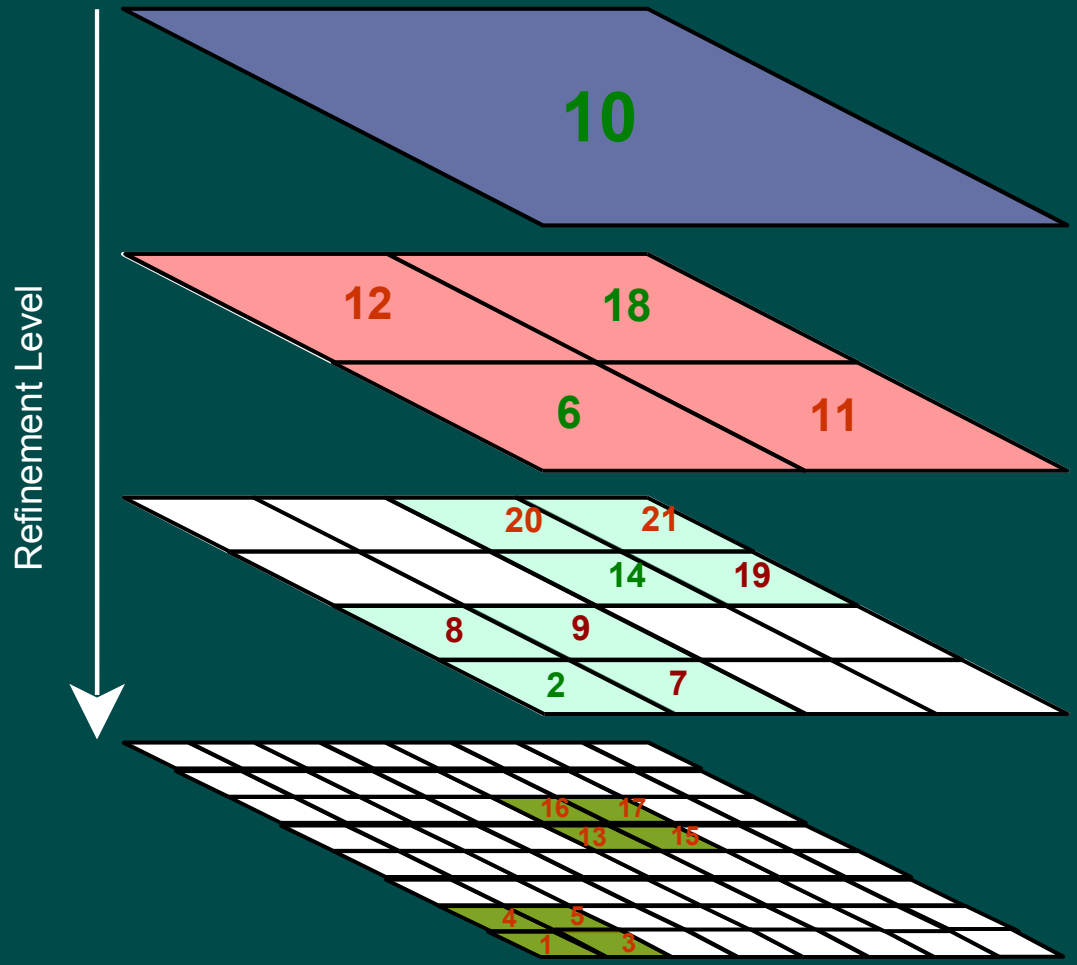
- Hierarchy of structured meshes (**patches**)
- Individual patches:
  - Different sizes
  - Different numbers of zones
  - Arbitrary integer resolution increment from one level to next
  - Patches can overlap



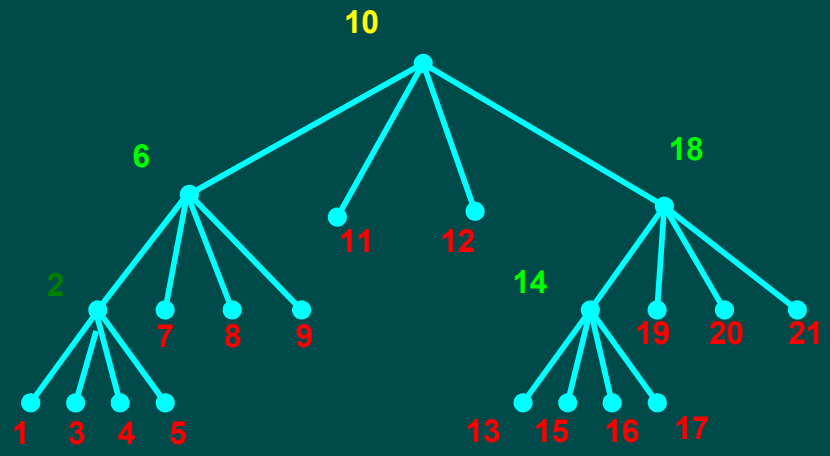
Norman & Bryan (1998)

Oct-tree AMR

PARAMESH (MacNeice et al. 2000)



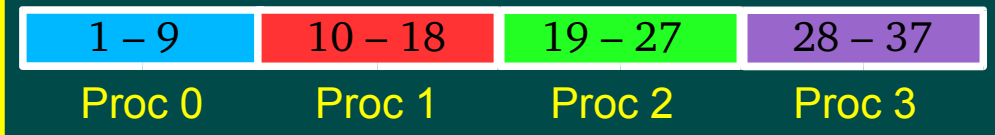
- Each block contains n^d zones in d dimensions
- Blocks stored in 2^d-tree data structure
- Factor of 2 refinement per level
- Blocks assigned indices via space-filling curve



Parallel decomposition

When implementing AMR on a parallel computer, commonly one uses a *space-filling curve* to assign a 1D ordering to the 2D/3D blocks on different levels of refinement

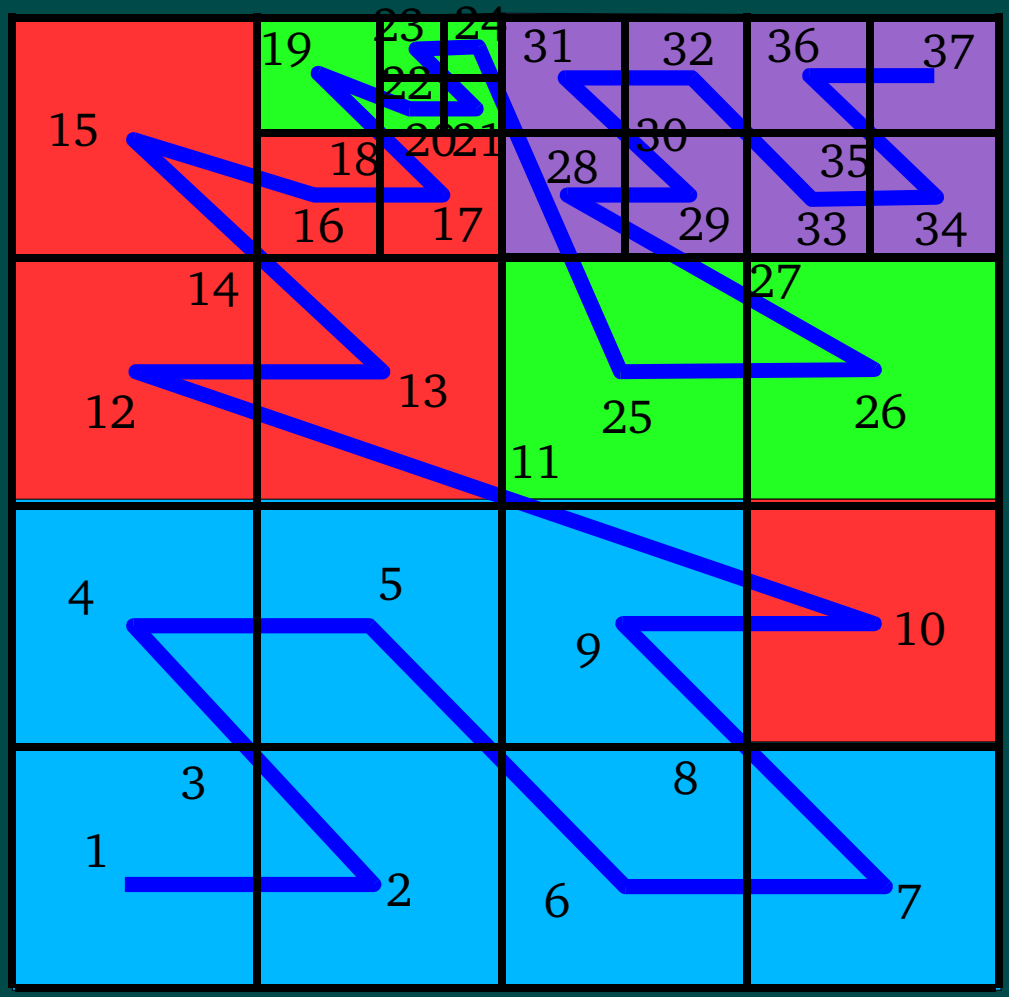
Once blocks are ordered, the list can be divided up evenly among processors:



Blocks can also be weighted by the work needed to update them

Space-filling curves tend to preserve *locality* – nearby blocks tend to have adjacent indices

Example: Morton curve used by FLASH



(Fryxell et al. 2000)

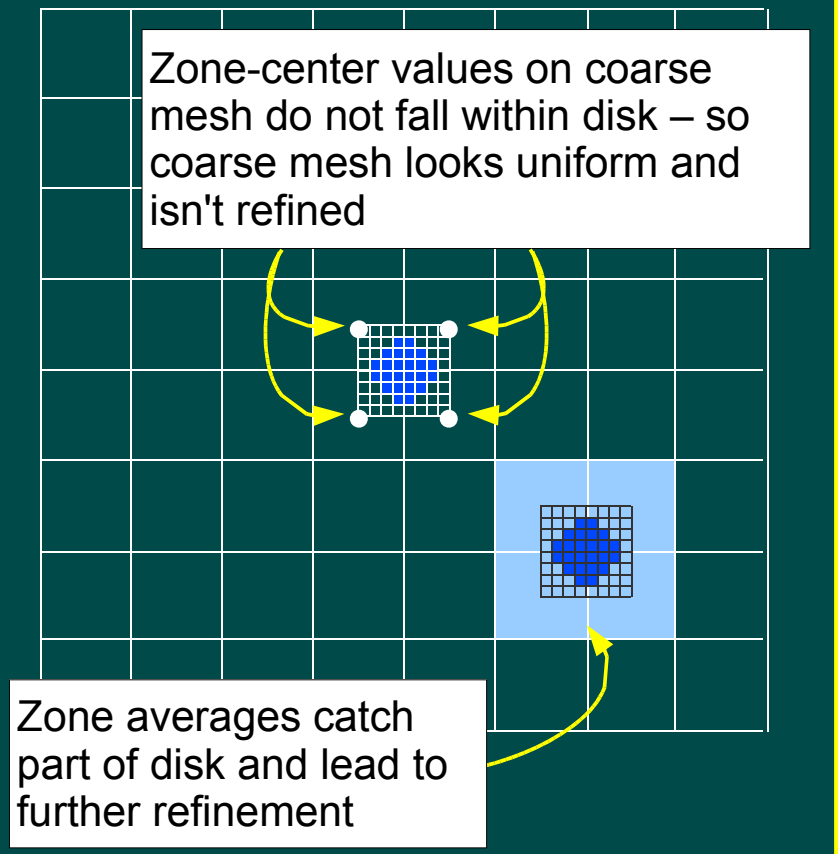
Initializing AMR calculations (non-cosmological)

Different from initializing a uniform mesh:

- Multiple blocks or patches of zones
- Don't know block structure in advance (depends on solution!)

Initialization is done one block/patch at a time.

1. We start by initializing the coarsest block.
2. Decide where to refine; create new blocks.
3. Redistribute blocks among processors if necessary.
4. Initialize solution on the new blocks (rather than interpolating from coarse blocks as during the run).
5. Go to 2; if no new blocks required, we are done.

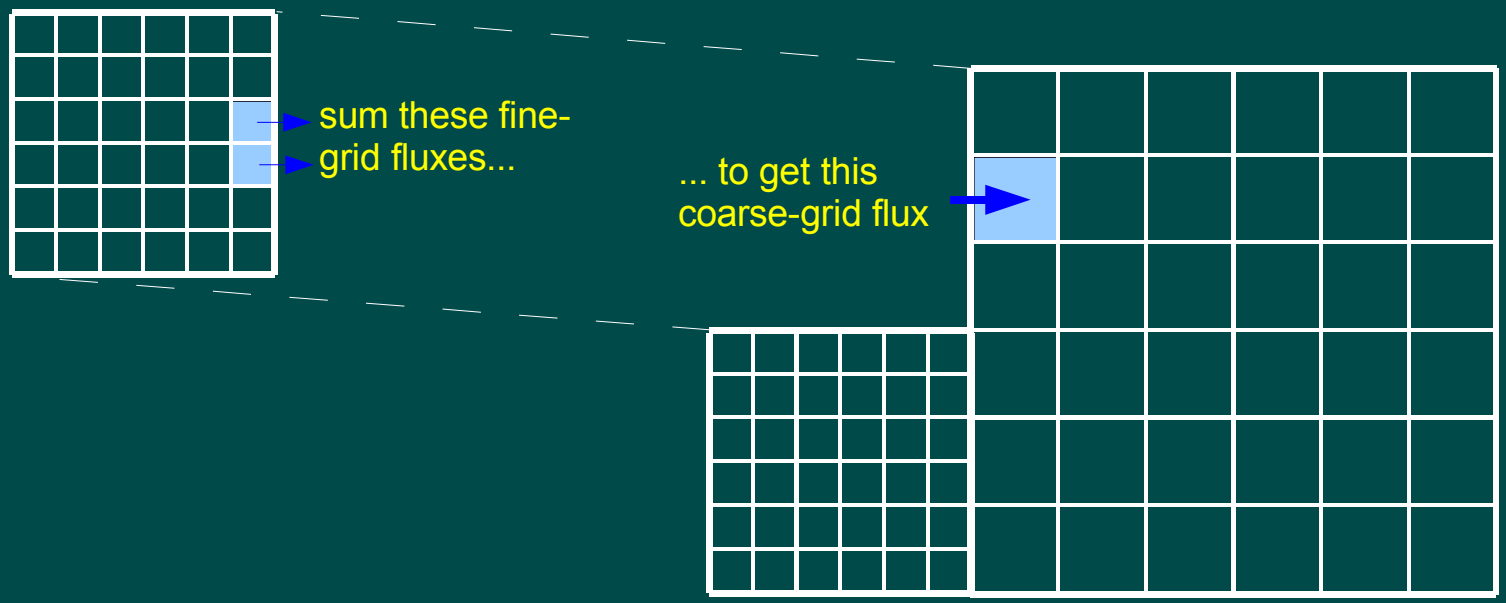


Be careful – desired features may not be resolvable on the coarsest mesh, so expected refinements might not occur.

- Use subzone averaging to make coarse zones reflect correct zone averages
- Force refinement in regions where initial conditions contain fine features

Basic AMR hydro algorithm – global timestep

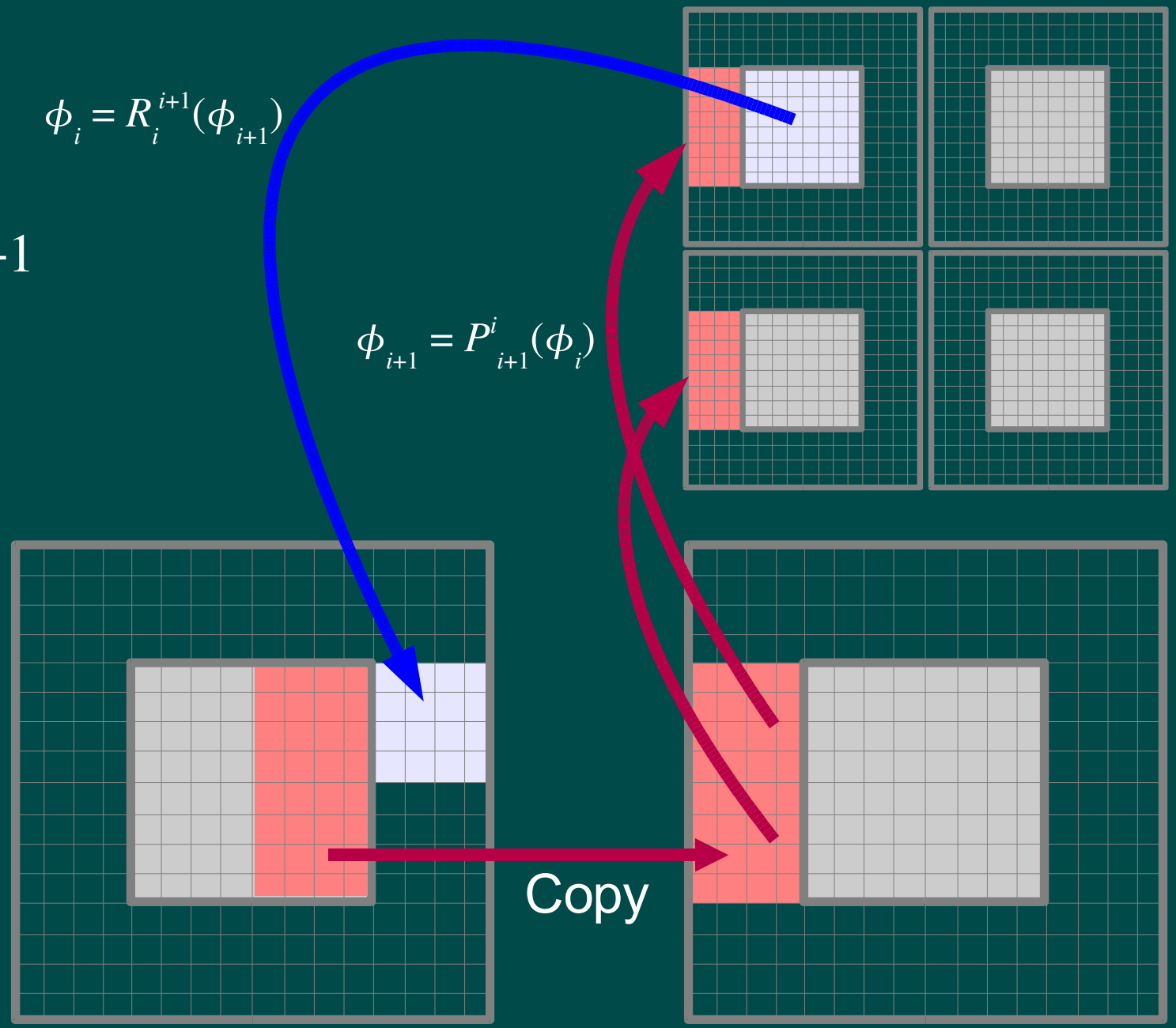
1. **Update boundary information.**
   a) Blocks restrict interior information to parent blocks
   b) Blocks trade information with neighbors on same refinement level
   c) Blocks with external boundaries set those values explicitly
   d) Fine blocks at fine-coarse boundaries set boundary values by interpolation from parents
2. **Apply hydrodynamics operator to each block, treating it as independent.**
   a) Compute flux for each cell interface on each block
   b) Override coarse boundary fluxes at coarse-fine interfaces with fine fluxes
   c) Difference fluxes to perform time update



AMR boundary conditions

$$\phi_i = R_i^{i+1}(\phi_{i+1})$$

$$\phi_{i+1} = P_{i+1}^{i}(\phi_i)$$



Basic AMR hydro algorithm – global timestep

3. Update mesh refinement.

a) For each block, compute a figure of merit that will determine whether the block should be refined or derefined ("refinement marking")

b) For each block:

i) If it is to be refined, create child blocks and set their solution information by interpolating from the parent

ii) If it is to be derefined, restrict solution information to its parent block, then free up the memory allocated to the block to be derefined

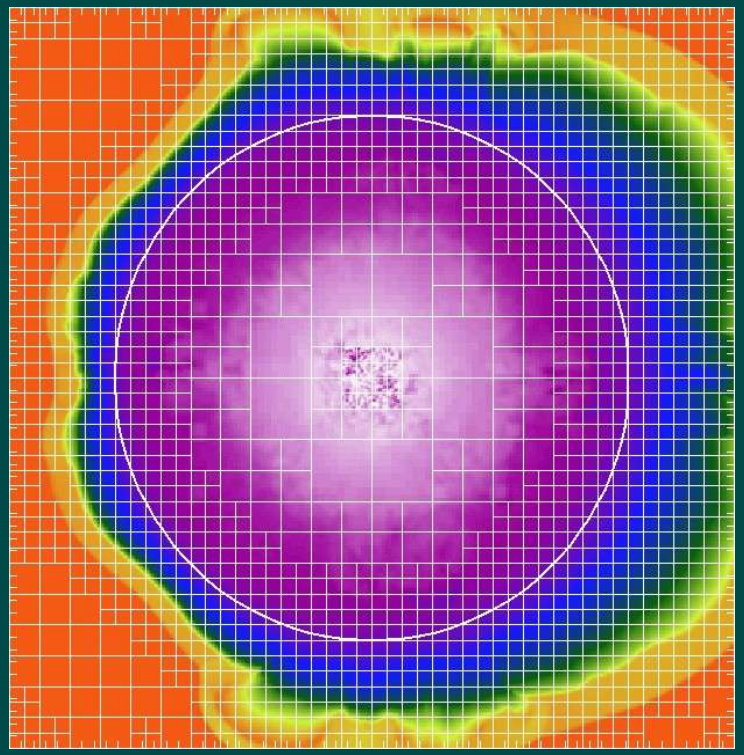
c) Compute block indices using space-filling curve

d) Redistribute blocks by communication among processors so that all processors have about the same amount of work to do during the next step

Refinement criteria

Geometrical

Refine blocks containing certain points – useful for doing spherically symmetric problems on Cartesian grids (refine center and edge of object)



Second derivative of density or temperature

Used to capture shocks. Compute scaled second derivative for each zone after Löhner (1987):

$$E_i \equiv \frac{|u_{i+1} - 2u_i + u_{i-1}|}{|u_{i+1} - u_i| + |u_i - u_{i-1}| + \epsilon\left(|u_{i+1}| + 2|u_i| + |u_{i-1}|\right)}$$

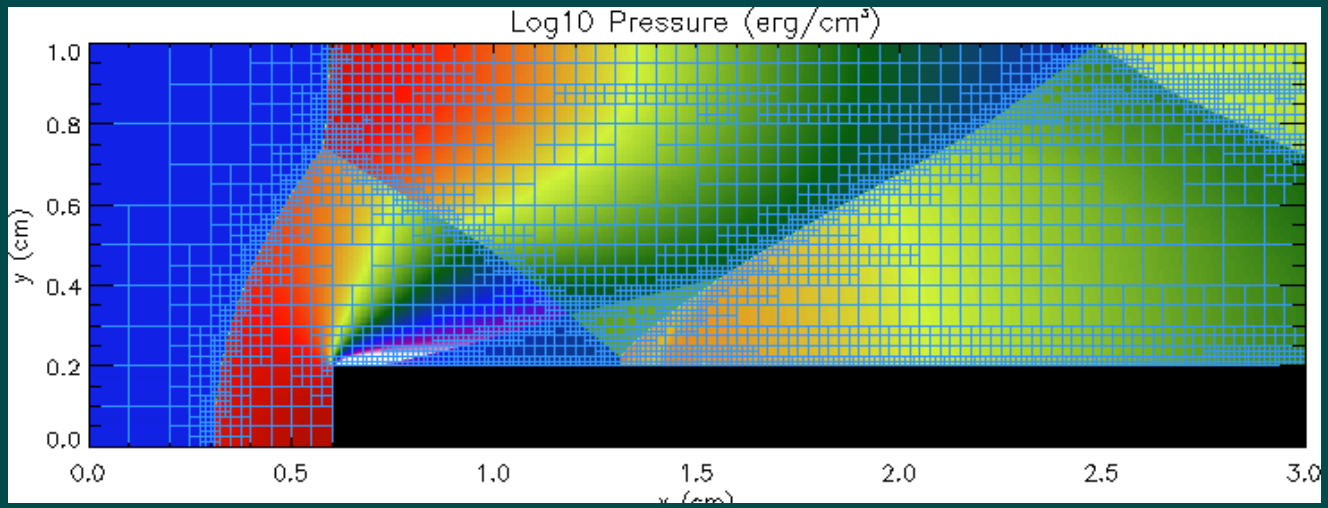
Refine block if

$$\max_{\text{block}} E_i > C_1$$

Derefine block if

$$\max_{\text{block}} E_i < C_2$$

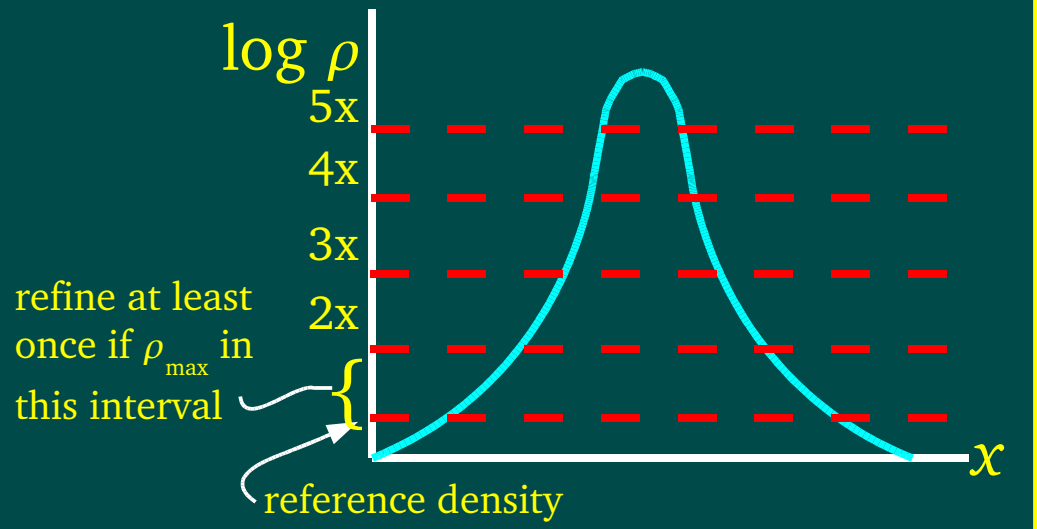
Typical values $C_1 = 0.8$, $C_2 = 0.2$, $\epsilon = 0.01$



Refinement criteria

Magnitude of density

Refine blocks to different levels depending on the ratio of their maximum densities to some reference density.



Truncation error estimate

Use Richardson extrapolation of solution on coarser grids to estimate mesh spacing required for a given level of error.

Physical criteria

Keep instabilities resolved by computing local critical wavelengths $\lambda_{\rm crit}$ and asking whether $\lambda_{\rm crit} < \Delta x$. *Example*: Truelove et al. (1997) criterion for Jeans-unstable fluid:

$$\Delta x < 0.25\,\lambda_J, \quad \lambda_J \equiv c_s \left(\frac{\pi}{G\rho}\right)^{1/2}$$