

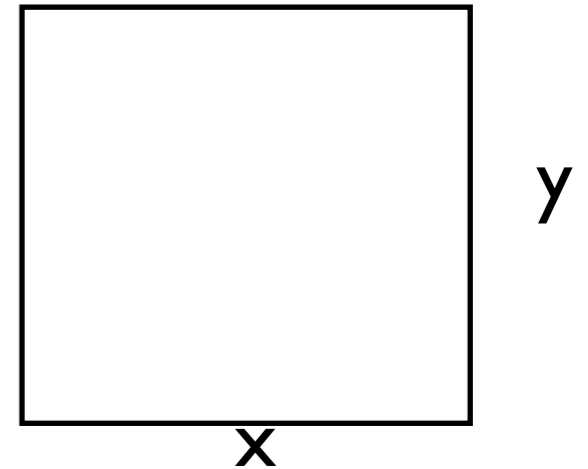
Partial Differential Equations, Part II

Scientific Computing for Astro Grad Students

- PDE basics & Definitions
- Convection-Diffusion Eqn
- Finite Difference approximations
- Finite Difference Method
- Stability; Implicit timestepping
- Accuracy
- Iterative Methods; Multigrid
- Finite Volume Method
- Eulerian vs Lagrangian
- SPH
- Spectral Methods

Elliptic Eqns

$$\frac{\partial u}{\partial t} + \vec{U} \cdot \nabla u = \kappa \nabla^2 u + f$$



- Smooth solutions
- Completely determined by boundary conditions, source function
- Domain of dependence of $u(x,y)$; $u(\text{all } x, \text{all } y)$

Elliptic Eqns

- Can use the same differential operator for Parabolic
- But now solving for **u**
- **D u = f**

$$\frac{D}{\Delta x^2} \begin{pmatrix} \dots & \dots & & & \\ 1 & -2 & 1 & & \\ & 1 & -2 & 1 & \\ & & 1 & -2 & 1 \\ & & & \dots & \dots \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \end{pmatrix}$$

Elliptic Eqns

- Exact
- This may be impractical!
- Entire matrix in memory at once
- 128^3 problem:
 - Matrix size $\sim 32 \times 10^9$
 - Cost of direct solve $\sim 512 \times 10^{12}$ ops.
- Other approaches?

$$\frac{D}{\Delta x^2} \begin{pmatrix} \dots & \dots & & & \\ 1 & -2 & 1 & & \\ & 1 & -2 & 1 & \\ & & 1 & -2 & 1 \\ & & & \dots & \dots \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \end{pmatrix}$$

Iterative Methods

- Elliptic like parabolic
but $du/dt = 0$
- Evolve in 'time'
towards steady state,
reducing error each
step

$$\frac{\partial u}{\partial t} = \kappa \nabla^2 u + f$$

Jacobi Iteration

- March forward some number of timesteps
- Choose dt to be as large as possible but still stable
- Fixed number of iterations, or until corrections become small

$$u_i^{N+1} = \frac{D\Delta t}{\Delta x^2} \left(u_{i+1}^N - \left(2 - \frac{\Delta x^2}{D\Delta t} \right) u_i^N + u_{i-1}^N \right)$$

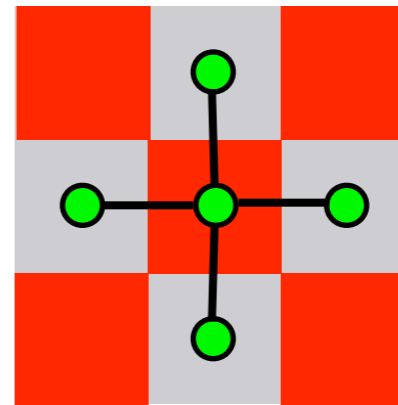
Jacobi Iteration

- **Downside: Memory**
(need space to store the whole next timestep)
- Seems unnecessarily slow -- why not use the new values somehow?

$$u_i^{N+1} = \frac{D\Delta t}{\Delta x^2} \left(u_{i+1}^N - \left(2 - \frac{\Delta x^2}{D\Delta t} \right) u_i^N + u_{i-1}^N \right)$$

Gauss-Seidel 'Red/Black'

- Imagine points (in 2d) on a checkerboard
- Updates to red squares only depend on themselves, black squares
- vice versa
- Do red sweep, then black sweep.
- About twice as fast, 1/2 the memory.

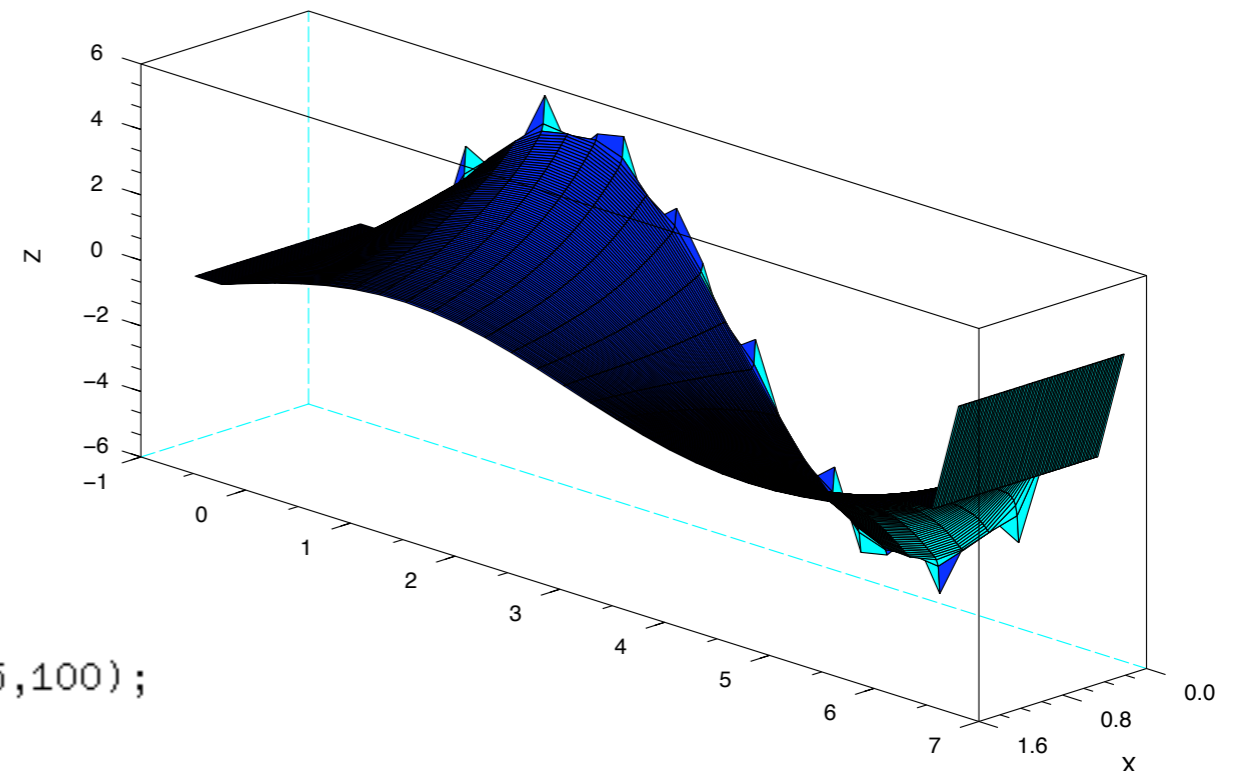


$$u_{i,j}^{N+1} = D\Delta t \frac{u_{i,j+1} + u_{i+1,j} - 4u_{i,j} + u_{i,j-1} + u_{i-1,j}}{\Delta x \Delta y}$$

Works, but slow

- Basically evolving the diffusion equation
- Small-scale noise gets wiped out quickly, but it takes a long time to diffuse large-scale modes away.

```
File Edit Search Execute Debug Scheme Options Windows
1 function [newu] = redblack(u, nx, h, dt, nguard)
2
3   odd = nguard+1:2:nx-2*nguard
4   even = nguard+2:2:nx-2*nguard
5
6   newu = u;
7
8   // red pass
9   du = dt/(h*h) * (u(even+1) -2.*u(even) + u(even-1));
10  newu(even) = u(even) + du;
11  // black pass
12  du = dt/(h*h) * (u(odd+1) -2.*u(odd) + u(odd-1));
13  newu(odd) = u(odd) + du;
14
15 endfunction
16
17 function [x,uo] = initialconditions(nx, h, nguard)
18
19   n = nx + 2*nguard
20   x = (-nguard:1:nx+nguard)*h;
21
22   uo = 5*sin(x) + sin(16*x);
23
24 endfunction
25
```



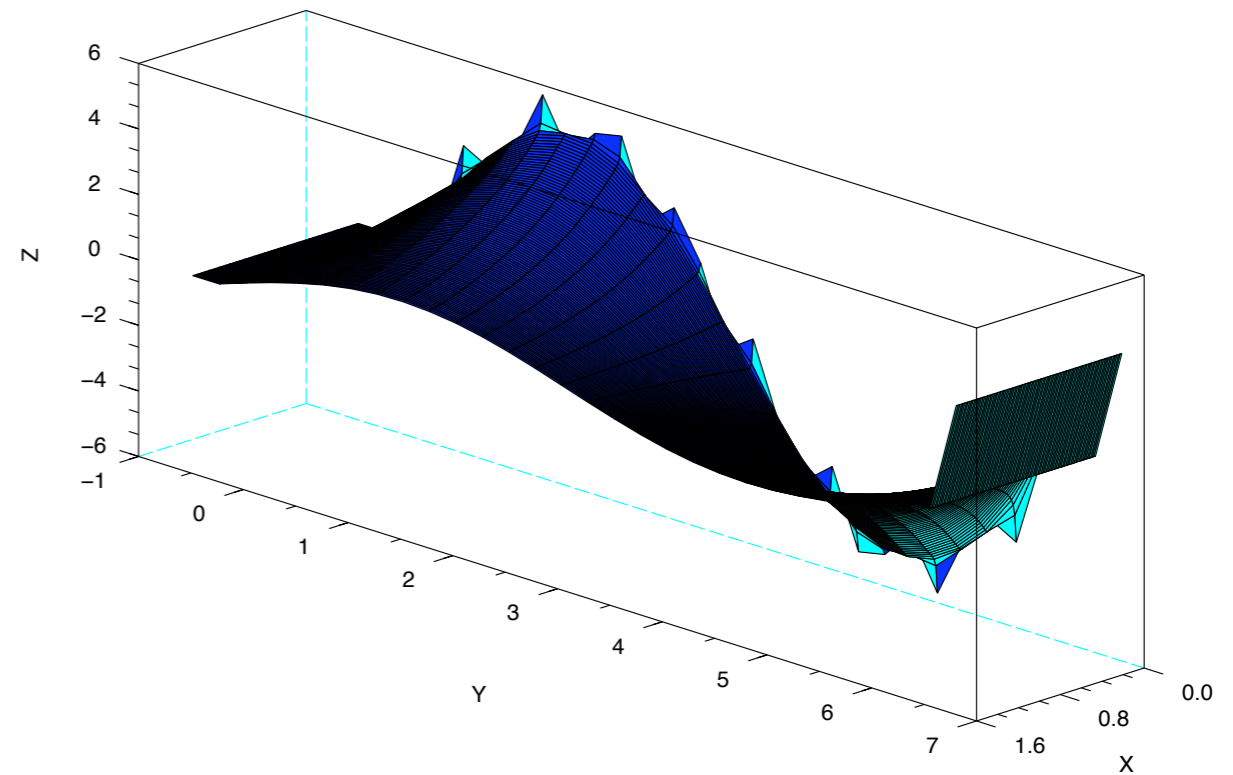
```
-->[allt,x,allu] = runredblack(25,100);
-->clf; plot3d(allt,x,allu)
-->■
```

Multigrid

- Newish (25 yrs) method
- Very well developed for linear elliptic equations, other areas still research topics
- Parallelizes fairly well, works with adaptive meshes nicely.

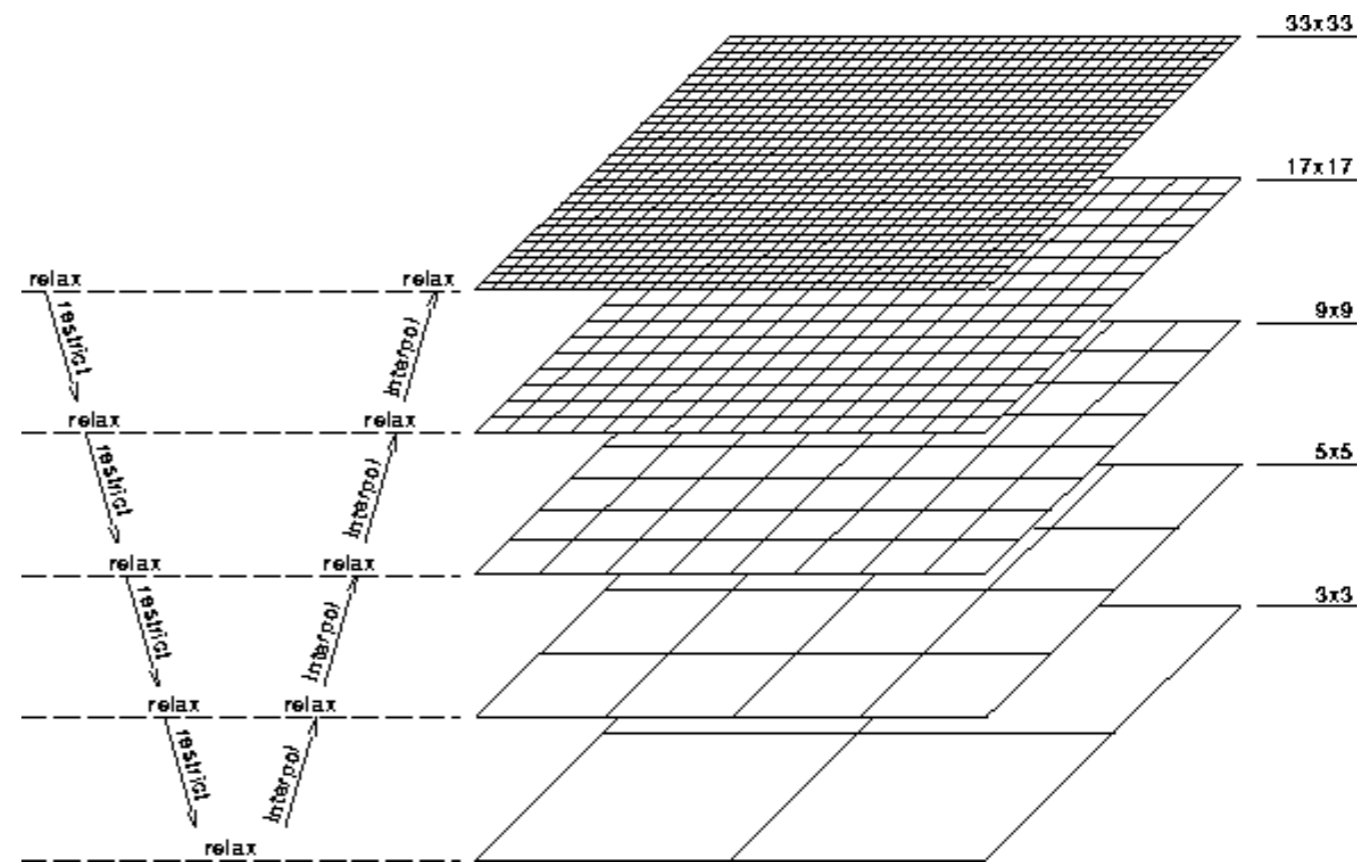
Multigrid

- **Key idea:** all modes are high frequency (and so diffuse quickly) if viewed with coarse enough resolution



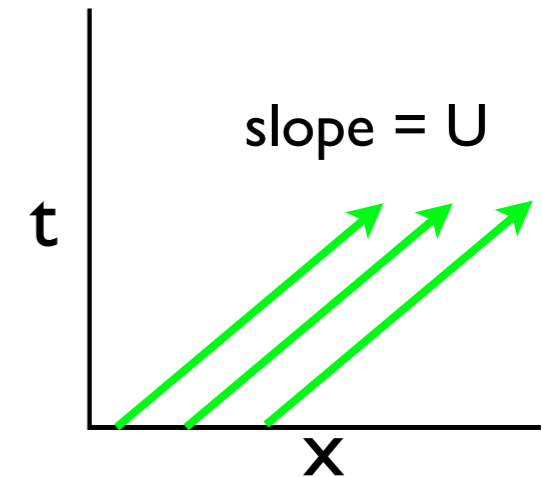
Multigrid

- Do a few iterations of simple technique at finest grid
- Coarsen
- iterate on this grid...



Hyperbolic Eqns

$$\frac{\partial u}{\partial t} + \vec{U} \cdot \nabla u = \kappa \nabla^2 u + f$$



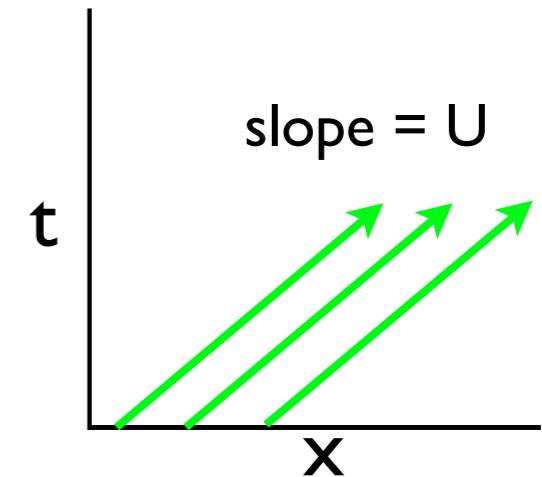
- Wave equation
- Non-smooth solutions
- Characteristics
- Domain of dependance: 'backwards light cone' of waves

Hyperbolic Eqns

$$\frac{\partial u}{\partial t} + \vec{U} \cdot \nabla u = f$$

$$\frac{\partial u}{\partial t} + \nabla \cdot (\vec{U} u) = f$$

$$\frac{\partial u}{\partial t} + \nabla \cdot F(U) = f$$



- Can typically be written in 'conservation law' form.
- Represents an important property of the equations

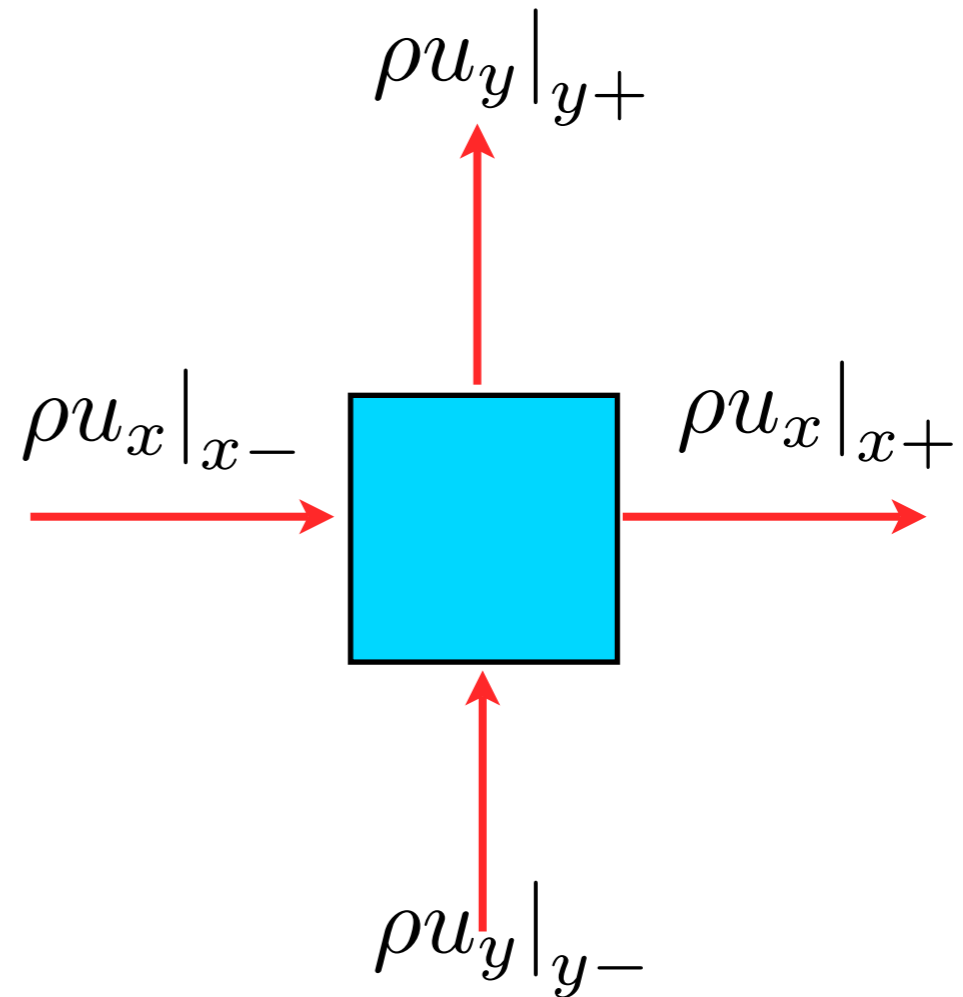
Hyperbolic Eqns

- Suggests that some quantities are conserved, with possible sources/sinks

$$\frac{\Delta M}{\Delta t} = \oint \rho \vec{u} \cdot \hat{n} dS$$

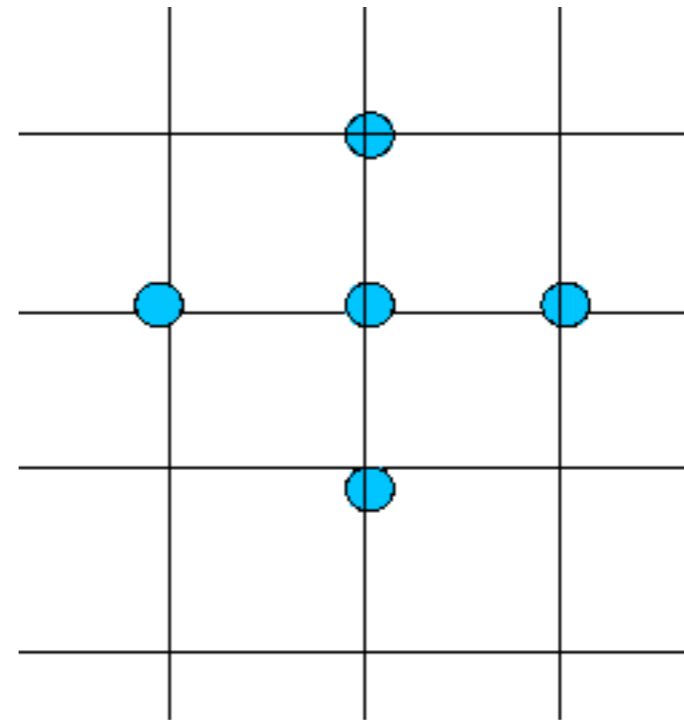
$$= \int \nabla \cdot \rho \vec{u}$$

$$\frac{\partial \rho}{\partial t} = \nabla \cdot \rho \vec{u}$$



Finite Difference Approaches

- Solution discretized as point values
- Build approximation to PDEs based on divided difference approximation to differentials
- Can be made very high-order
- Higher accuracy
- Faster convergence to PDEs as go to lower resolution



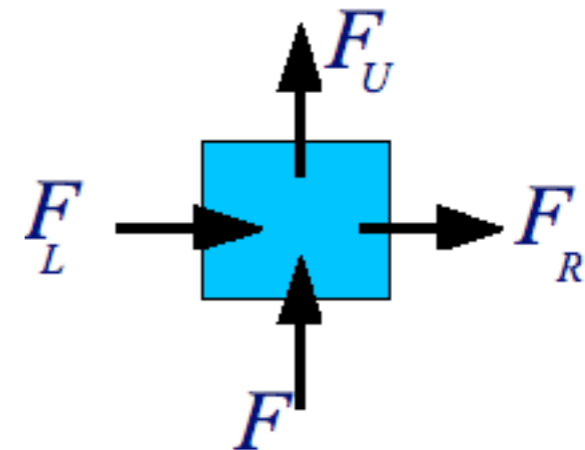
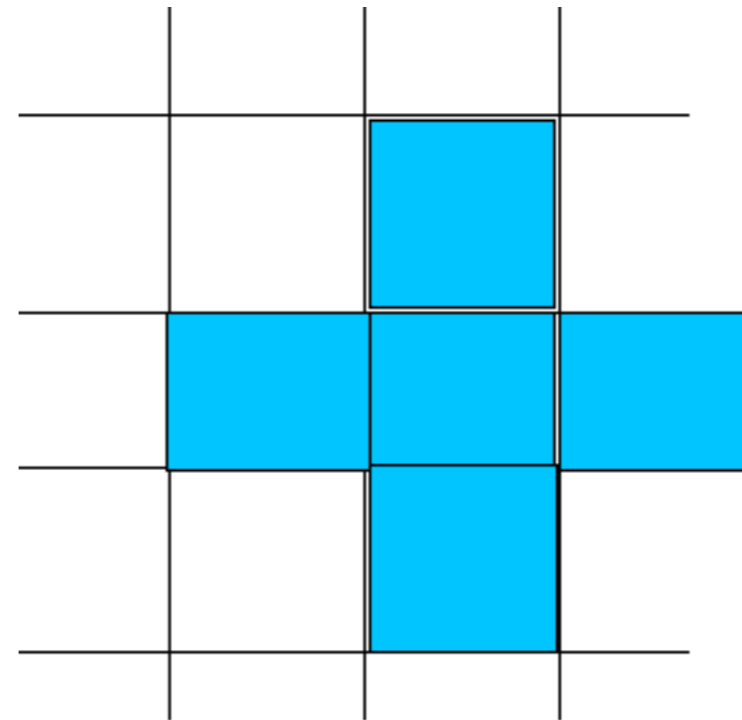
Finite Volume

- Solution discretized on control volumes
- Value is cell average
- For conservation eqn,

$$u_t + \nabla \cdot F(u) = 0$$

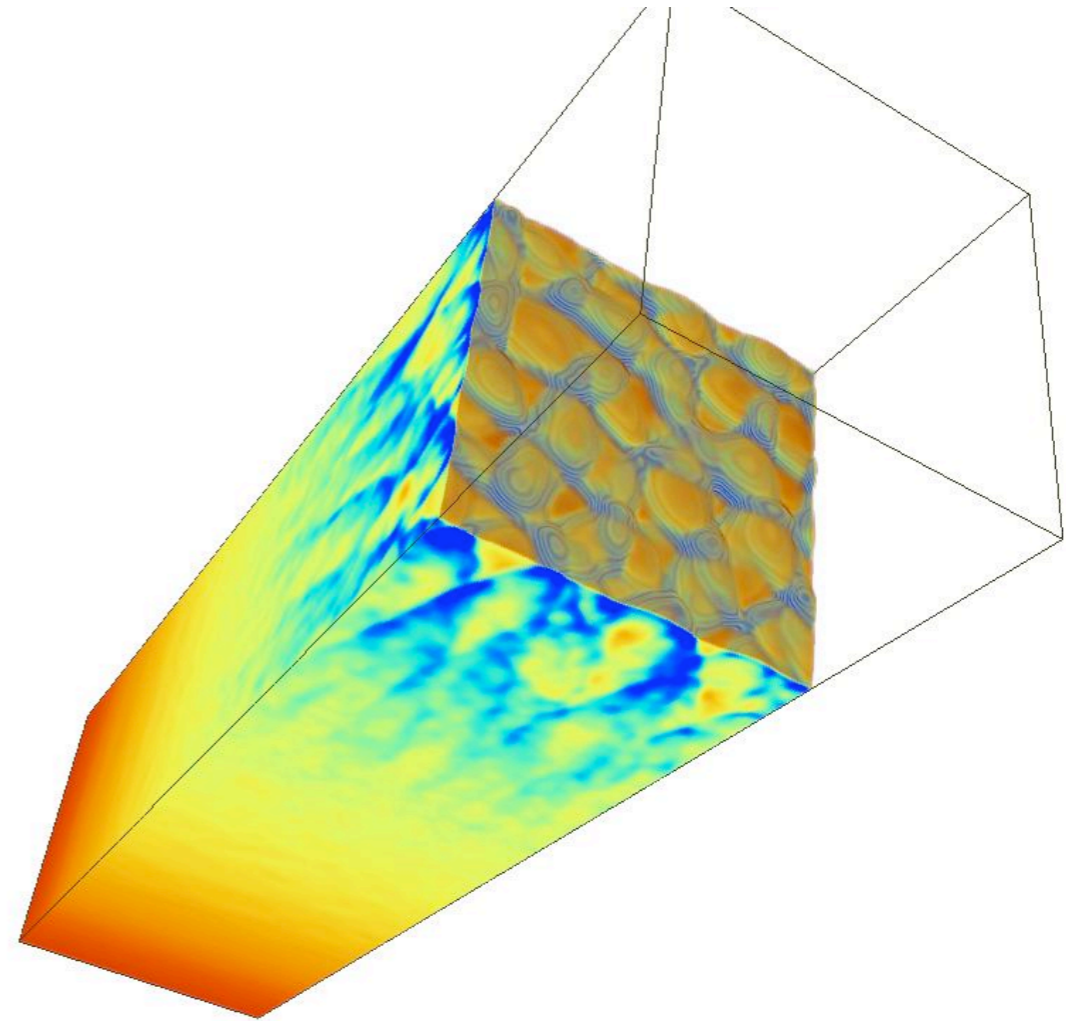
equation is updated as fluxes through interfaces.

- **Guaranteed** conservation



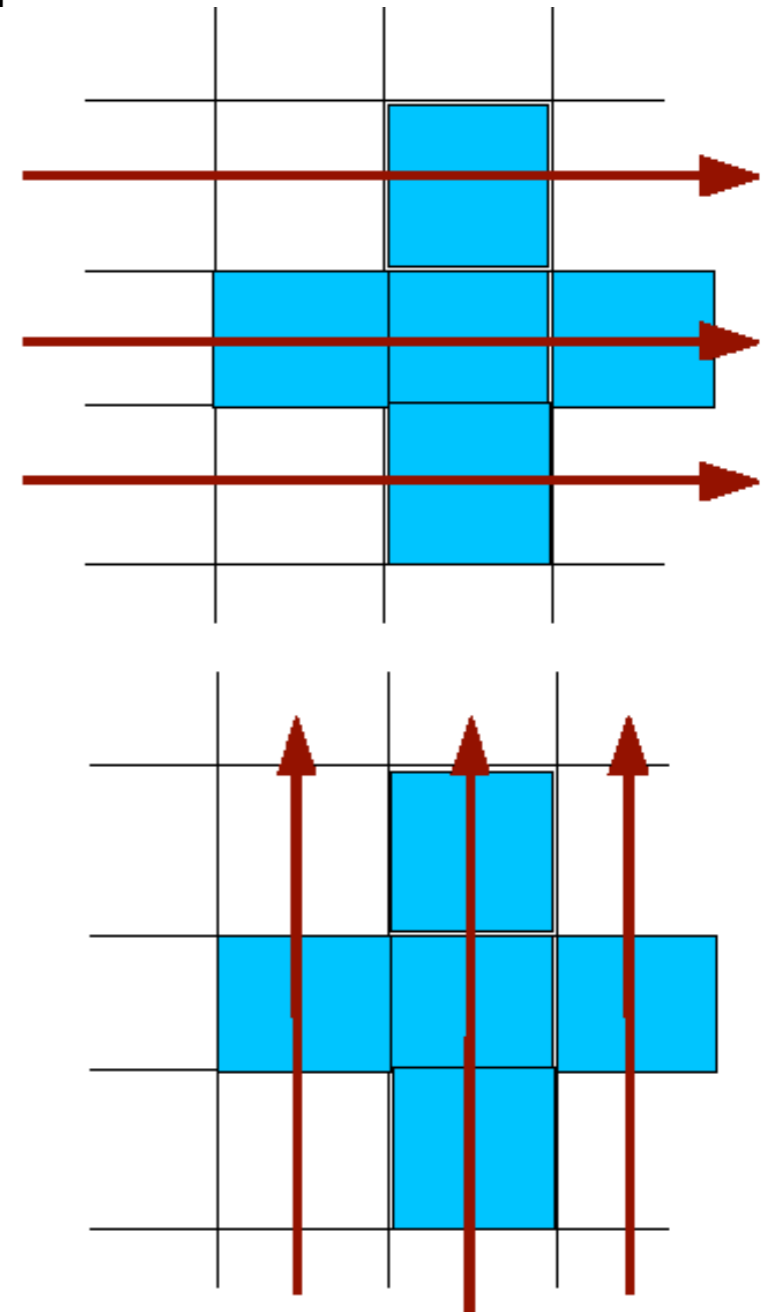
Finite Volume

- Slightly harder to make very high order
- Conservation may mean more than naïve order of accuracy
- Shock speed wrong with non-conservative discretization; can alter simulation results
- That one converges faster with resolution not a good indicator of accuracy with a given resolution



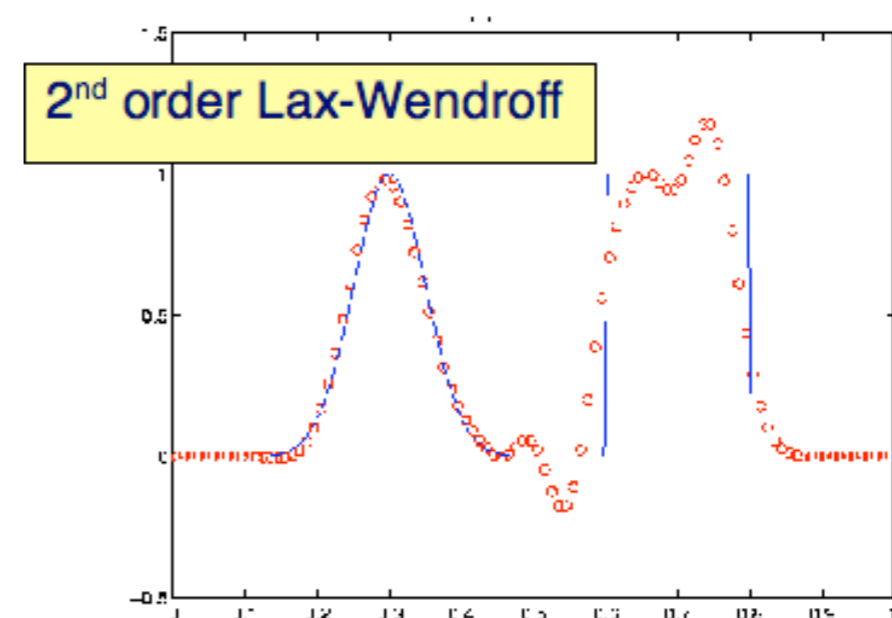
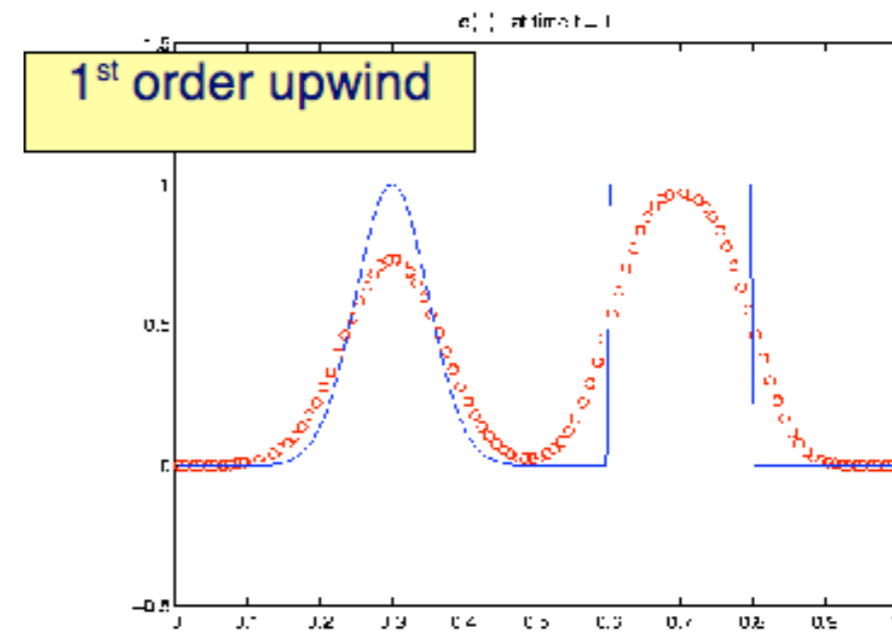
Multidimensional Finite Volume

- More difficult to make fully multidimensional FV scheme
- One approach which works well for many problems: 'dimensional splitting'
- Xsweep, Ysweep, Ysweep, Xsweep
- 2nd order accurate in time if sweeps are.



Finite Volume

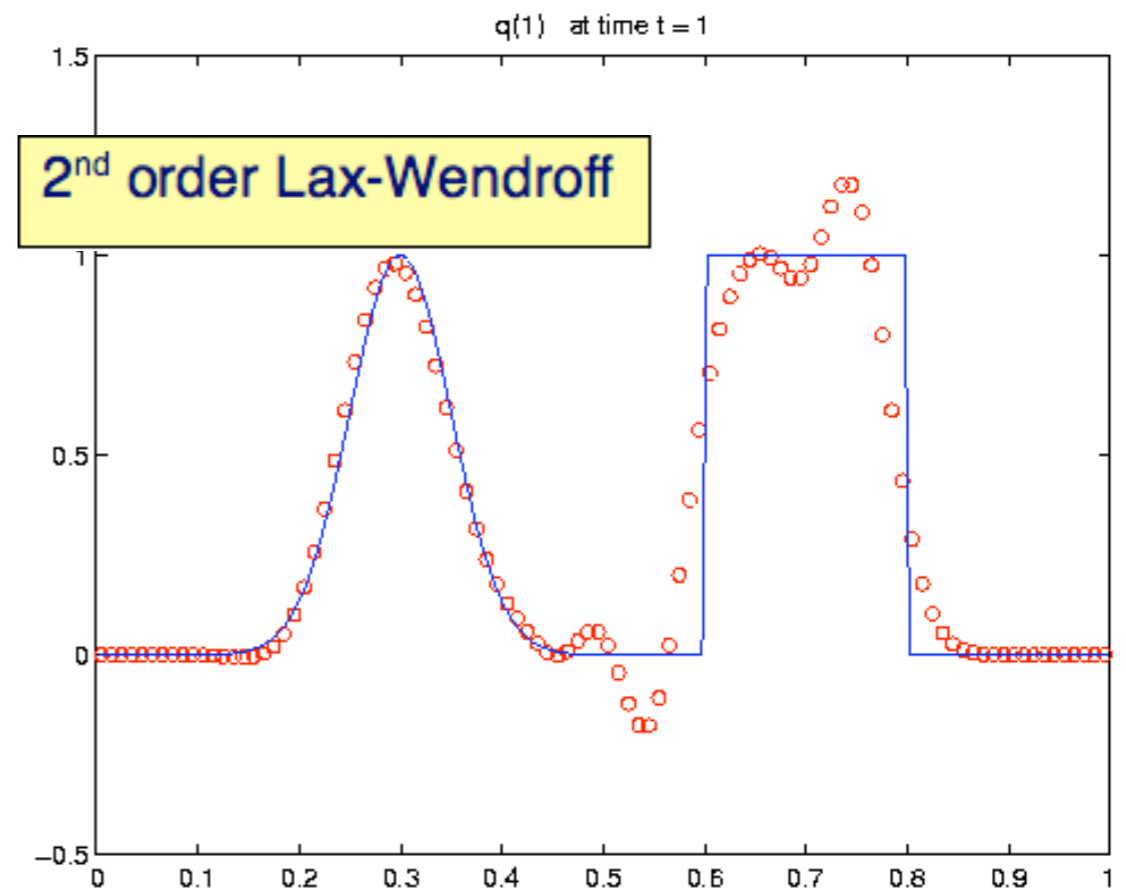
- Typical approach; interpolate data to cell faces, calculate flux function
- Low order interpolations typically add significant *diffusion* to flow features.
- High order can cause *dispersion* of sharp flow features -- but these can happen



LeVeque, <http://www.amath.washington.edu/~claw/>

Finite Volume

- Artificial 'viscosity' can stabilize (not eliminate) oscillations, but not general solution
- Great improvement can be made by limiting slopes implied by methods – disallow new min/max
- Best handling of shocks – numerical methods that 'know' detailed structure of solutions
- Nonlinear eqns: for 2nd order in time, need to know how flux will change in time.

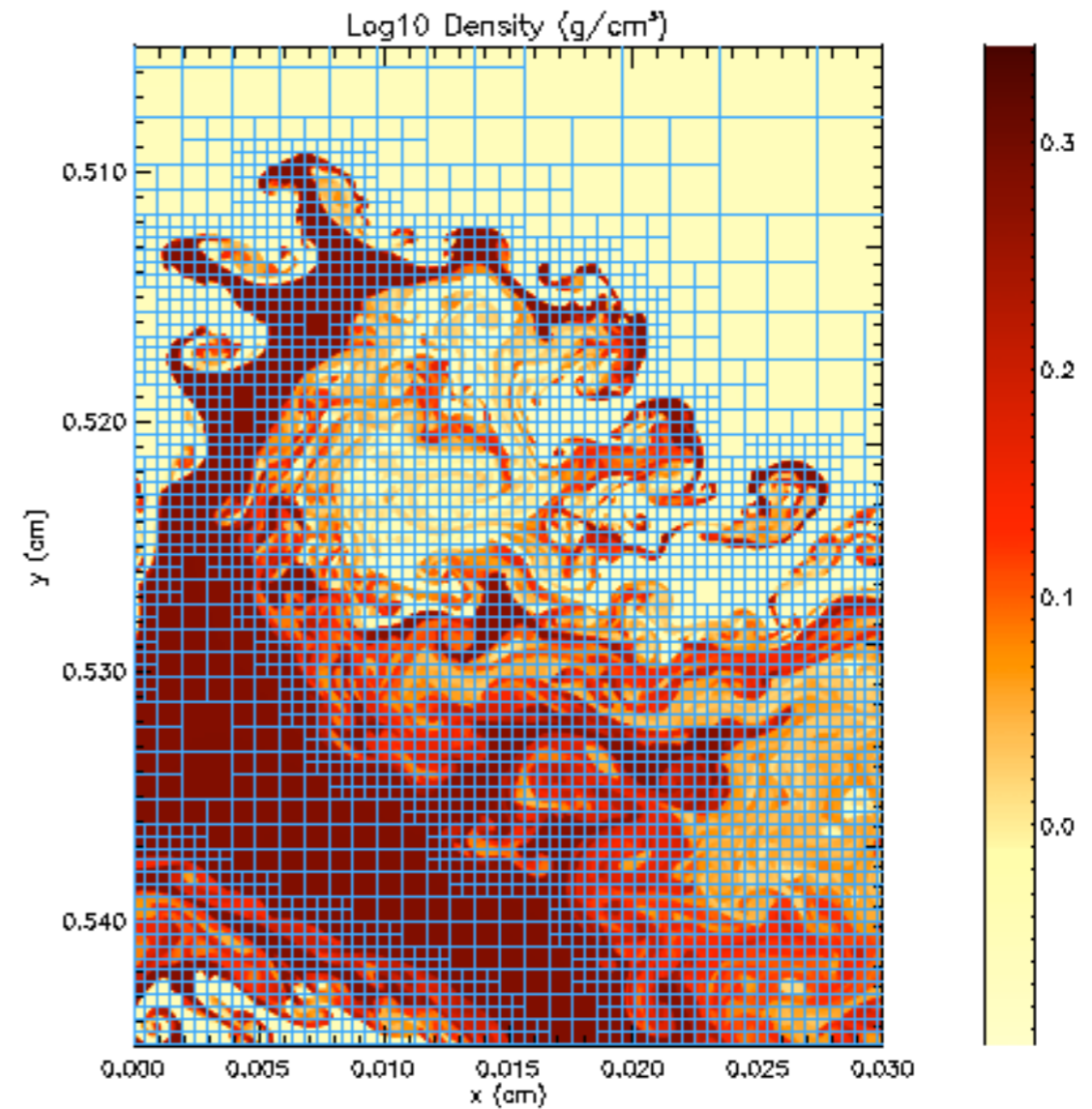


Lagrangian methods and Smoothed Particle Hydrodynamics (SPH)

Eulerian Grid Methods

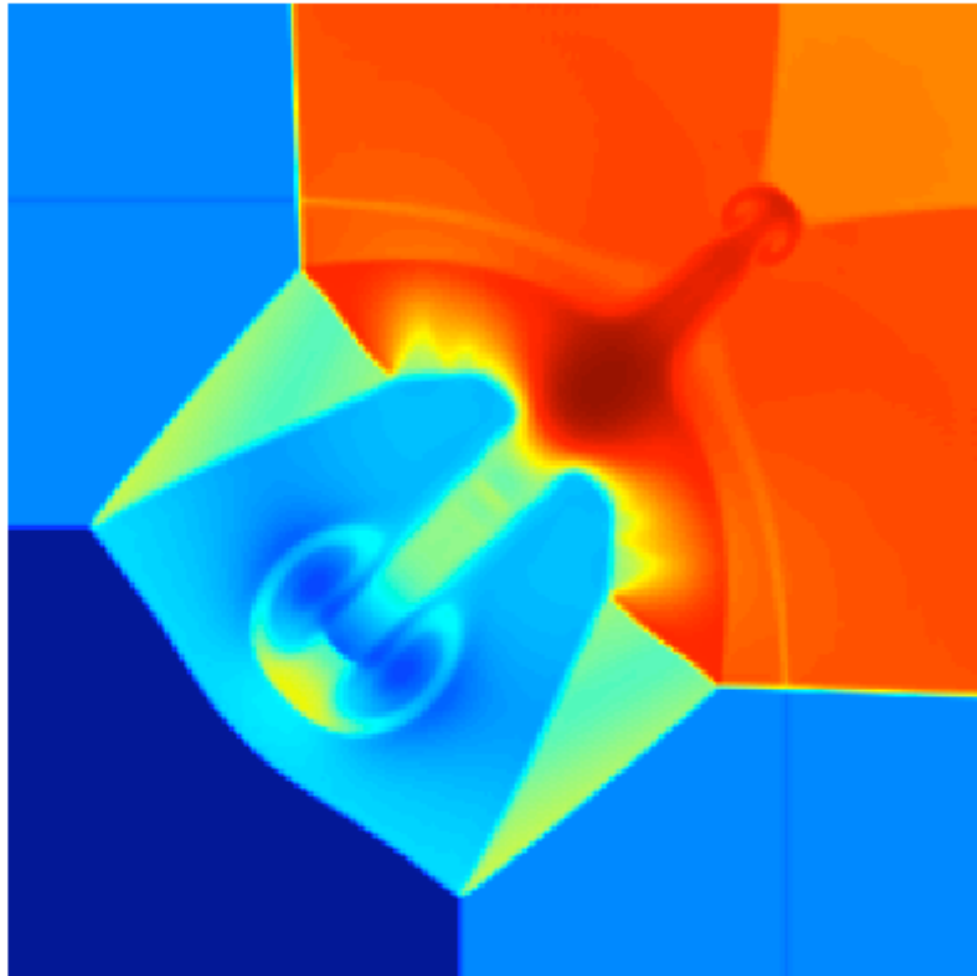
.The methods covered so far use an Eulerian grid:

- . Prescribed coordinates
- . In 'lab frame'
- . Fluid elements flow through grid zones



time = 1.950 s
number of blocks = 224756
AMR levels = 10

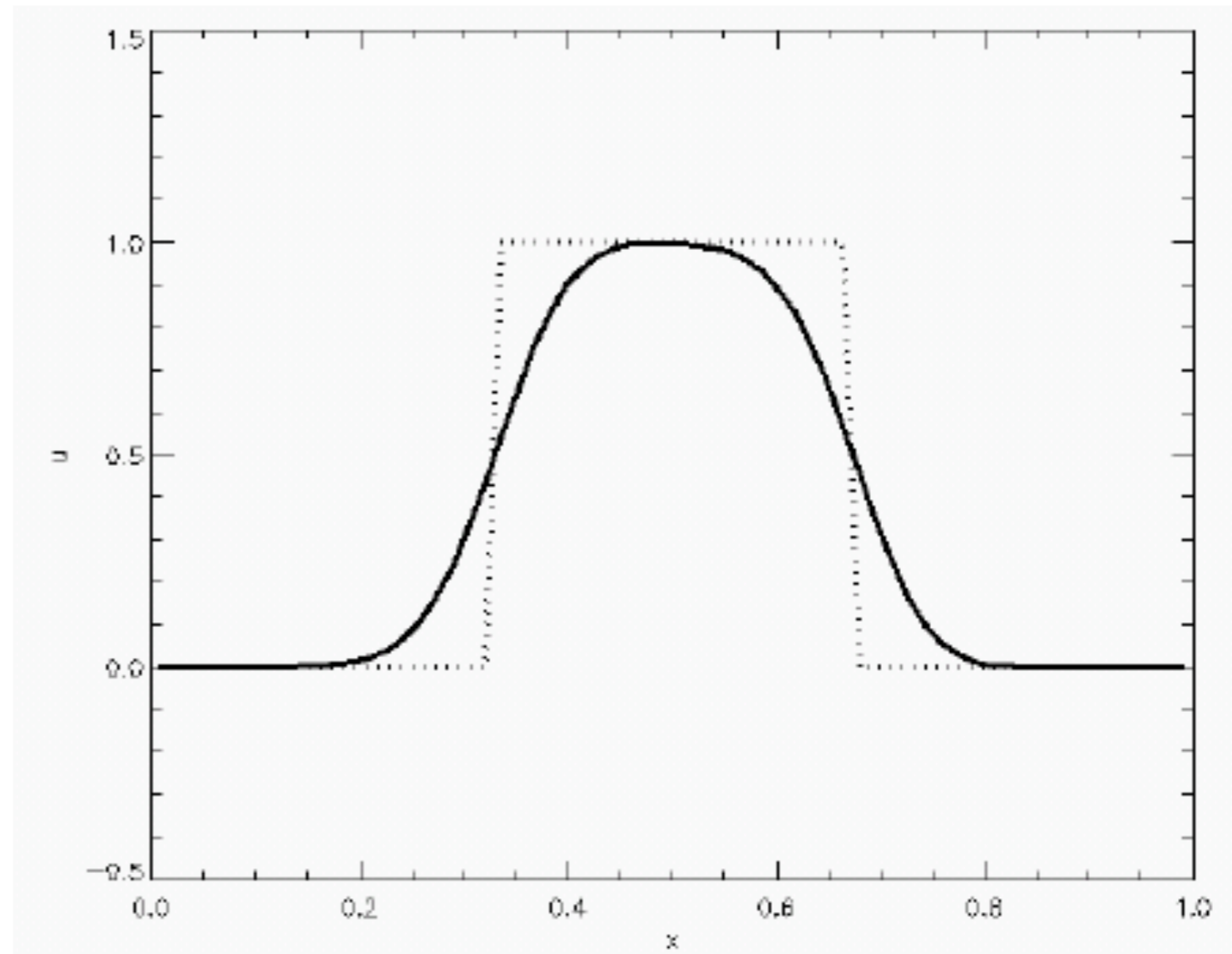
Eulerian Grid Methods



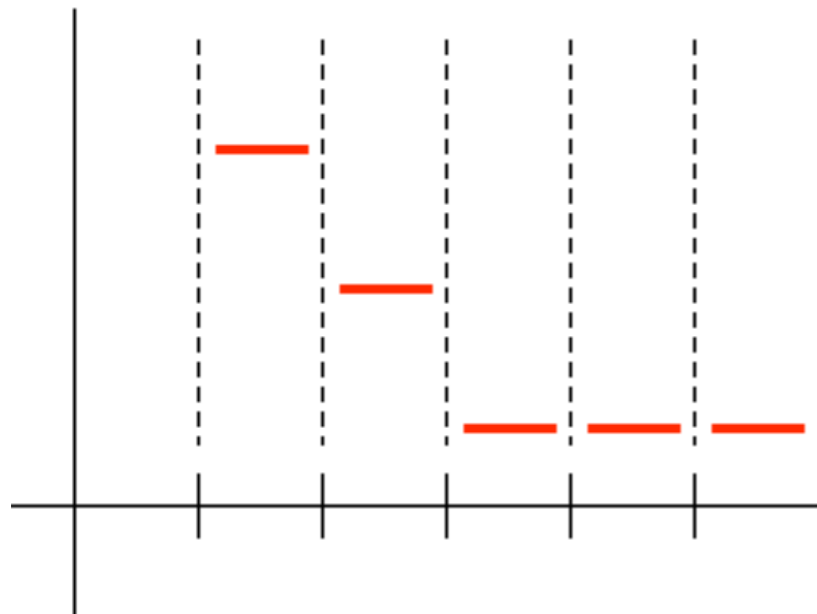
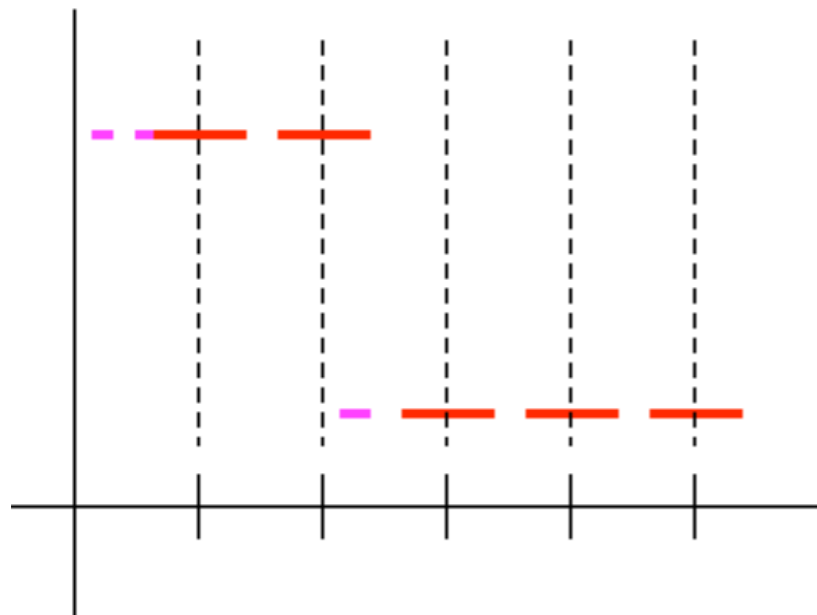
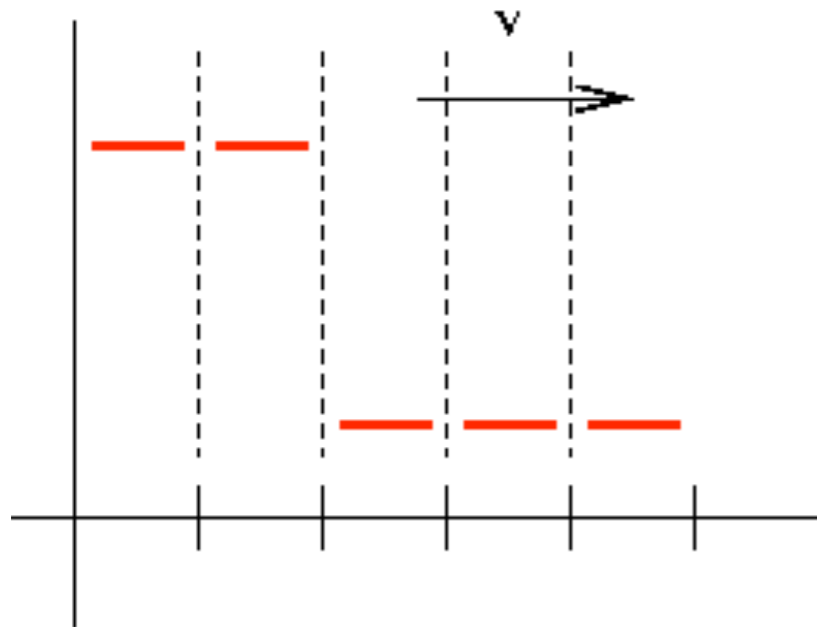
- This is probably the standard approach to solving the equations of fluid motions in most disciplines.
- Many decades of research into solution techniques
- Extremely sophisticated, high-order accuracy methods
- Can accurately describe very complex phenomena.

Eulerian Grid Methods

- However, simplest possible dynamics fares surprisingly poorly
- Even high-order methods are quite diffusive when advection over a large number of grid cells is necessary



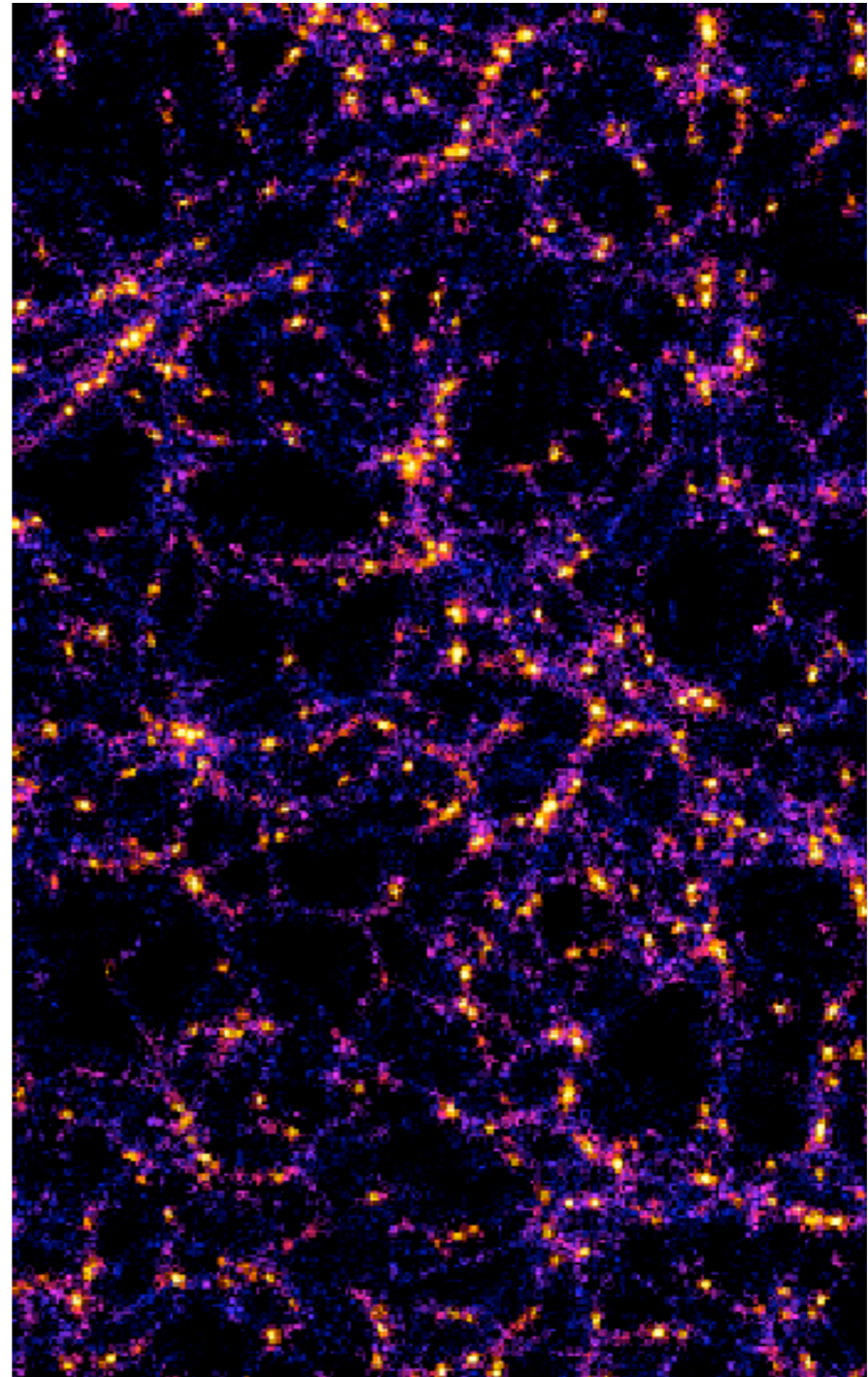
Eulerian Grid Methods



- This is fundamental to how Eulerian grid codes work.
- Can be ameliorated but not fixed.
- Once some of a quantity enters a grid cell its contribution is spread throughout domain through some averaging procedure.
- Higher order methods do this to a lesser degree than lower-order methods, but the effect remains.
- Occurs for any evolved quantity.
- “Numerical diffusion”

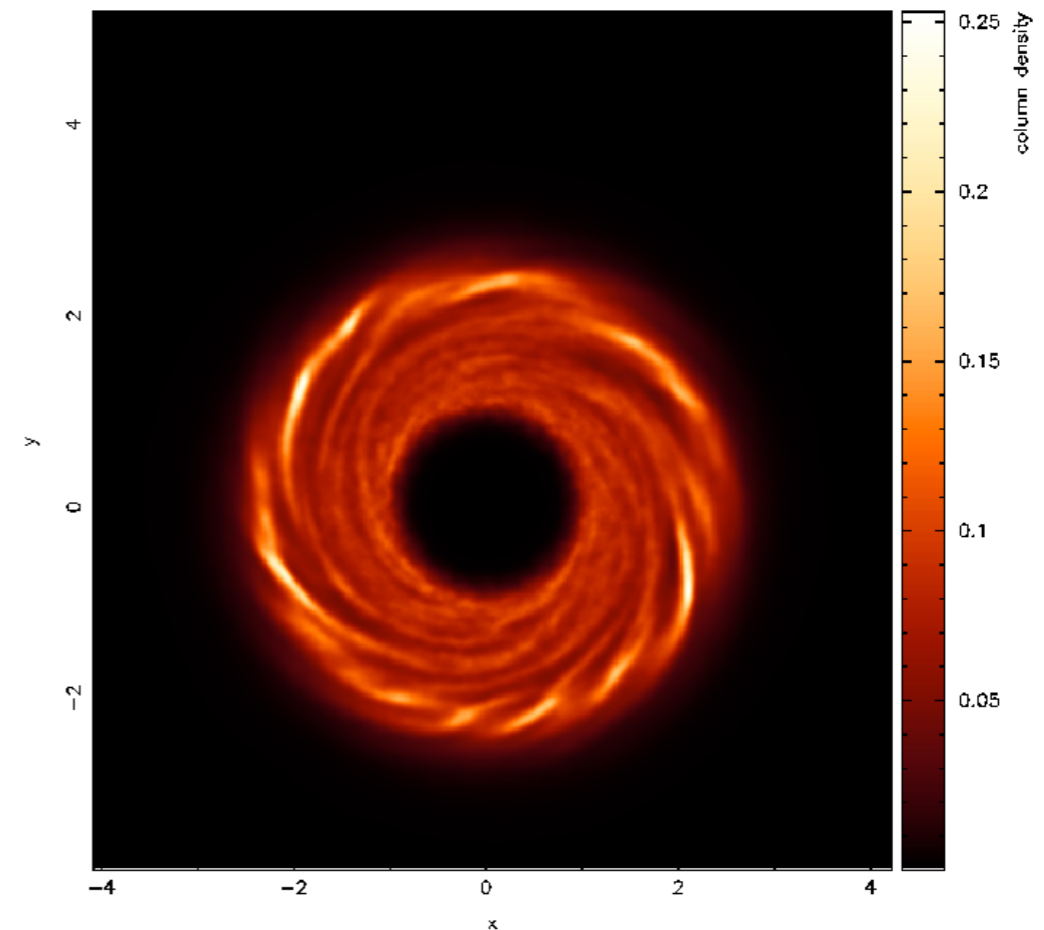
Advection-Dominated Flows

- There are many systems in astrophysics which are dominated by large-scale advection of fluid
- Eulerian grid is not necessarily the most natural approach in these systems
- Cosmology: evolution is dominated by large scale falling of material onto local density enhancements



Advection-Dominated Flows

- There are many systems in astrophysics which are dominated by large-scale advection of fluid
- Eulerian grid is not necessarily the most natural approach in these systems
- Accretion disks: Flow is dominated by differential Keplerian rotation around central object

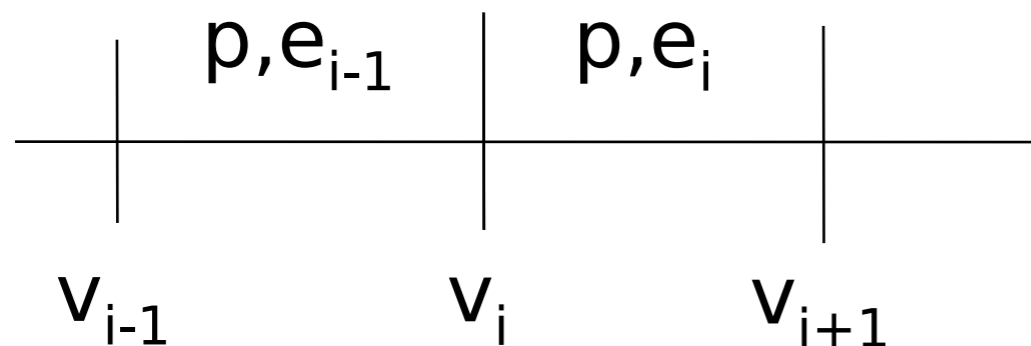


1D Lagrangian Formulation

Lagrangian formulation

No fluxes 'through' fluid element interfaces, as no transport through interfaces*

Typically implemented on a staggered grid:



$$\frac{D\rho}{Dt} = -\rho \frac{\partial u}{\partial x}$$

$$\rho \frac{Du}{Dt} = -\frac{\partial p}{\partial x}$$

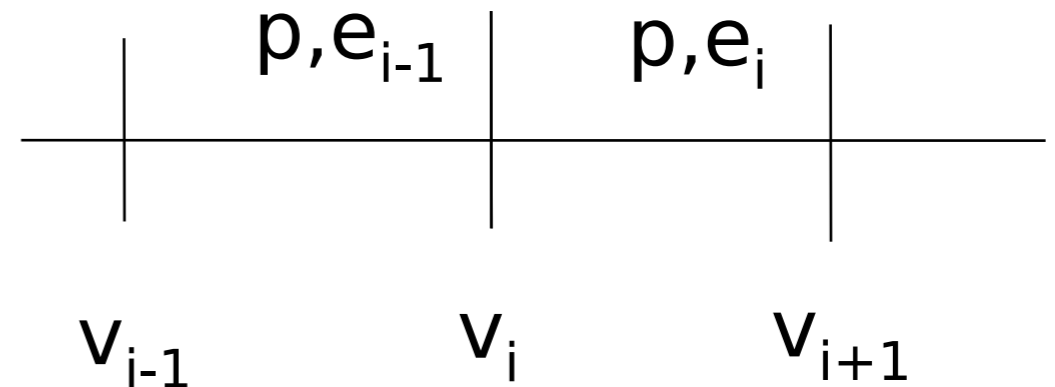
$$\rho \frac{De}{Dt} = -p \frac{\partial u}{\partial x}$$

No purely advective fluxes!

*absent other physics like dissipative transport

1D Lagrangian Formulation

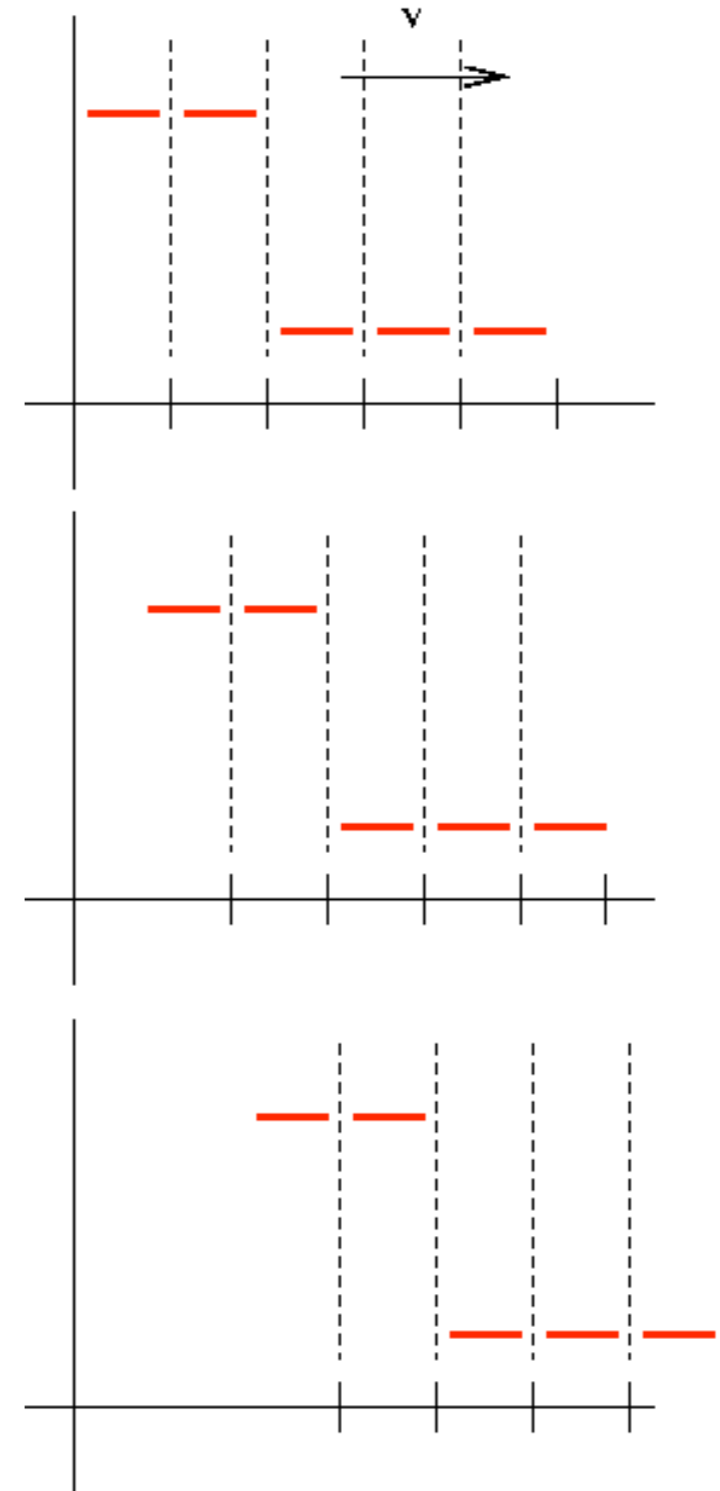
- Each lagrangian cell has a fixed mass; density is just mass over (evolved) volume element
- Guaranteed conservation of mass; shared cell edges guarantee conservation of volume, energy



1D Lagrangian Formulation

Huge benefit: open boundaries – mesh can expand as necessary

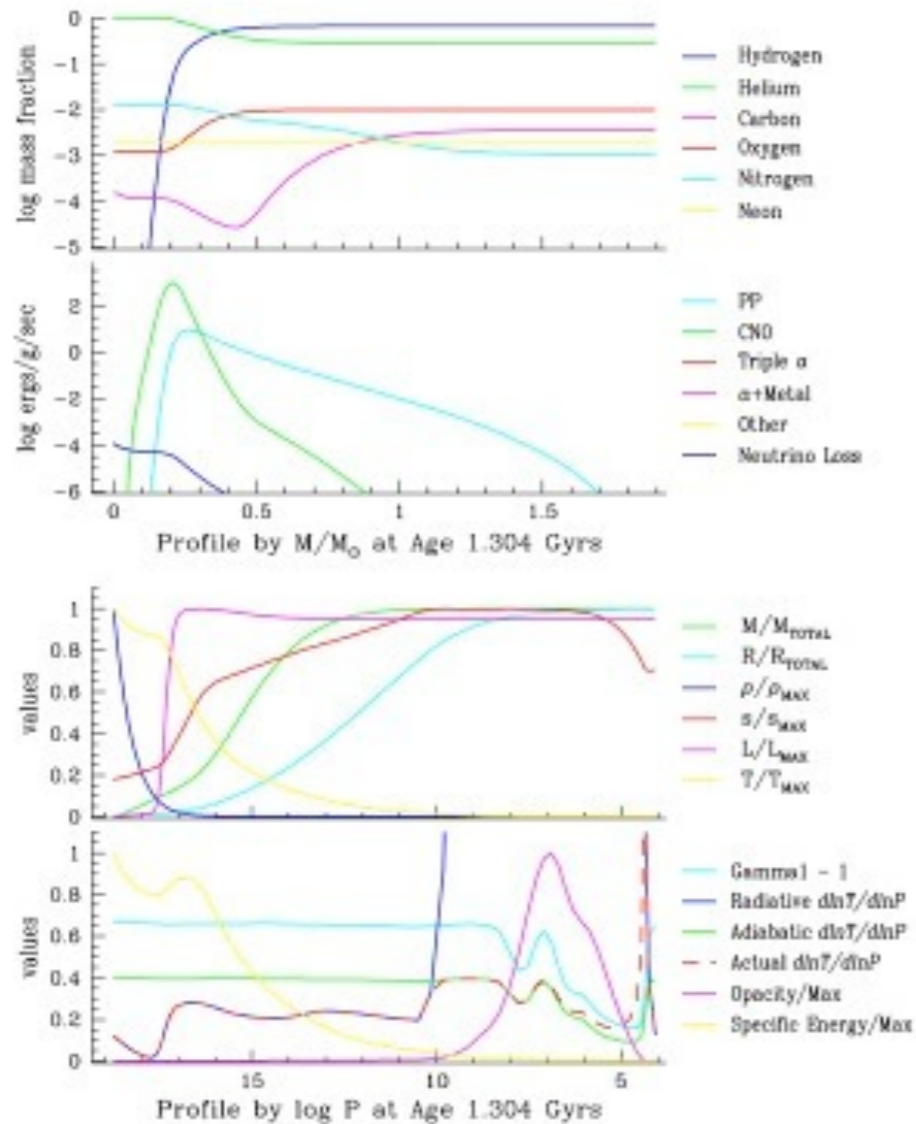
And, of course, no numerical diffusion from purely moving fluid around



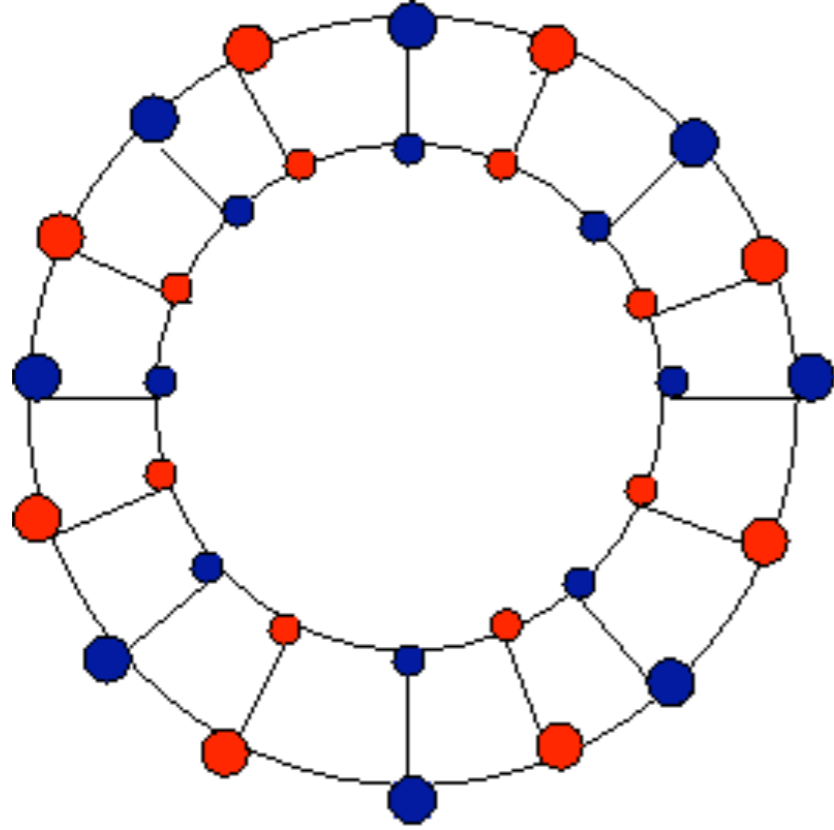
1D Lagrangian Formulation

These advantages make 1d lagrangian grid methods a natural choice for applications such as stellar evolution

Typically use 'mass coordinates' even outside of numerical context



Multi-D Lagrangian Gridding

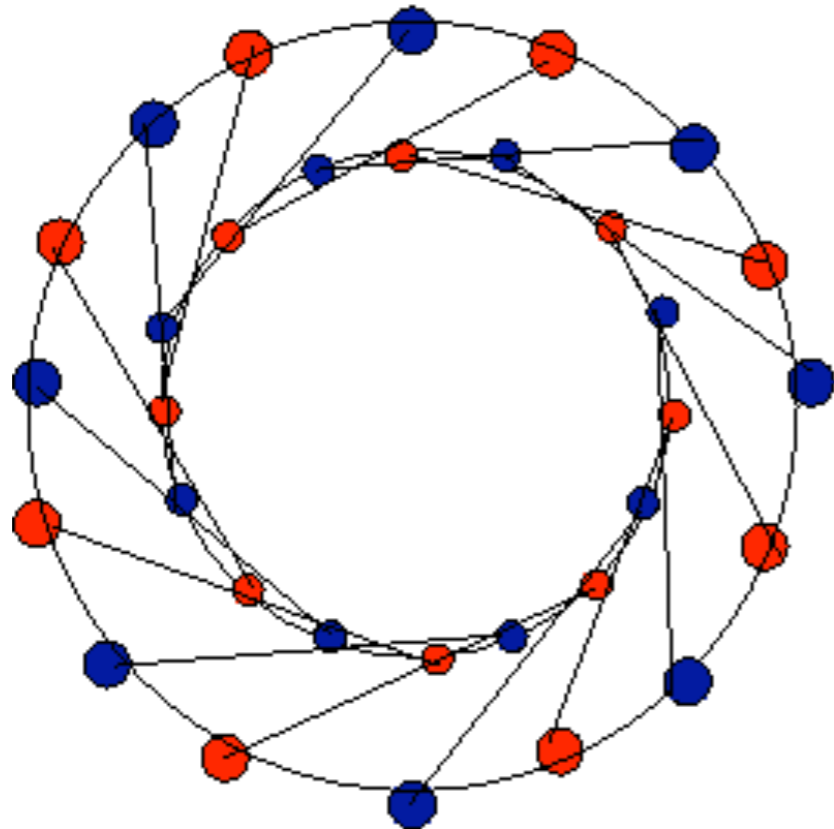


Works extremely well in 1d.

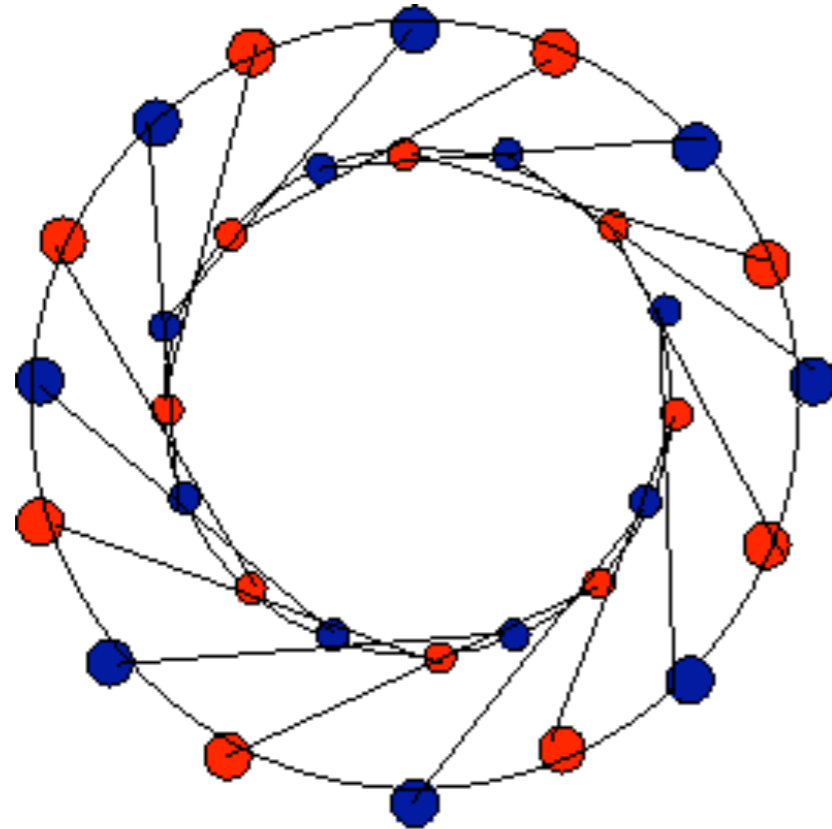
In multidimensions, more complexity possible in geometry

Even differential rotation / shear can lead to disastrously tangled meshes.

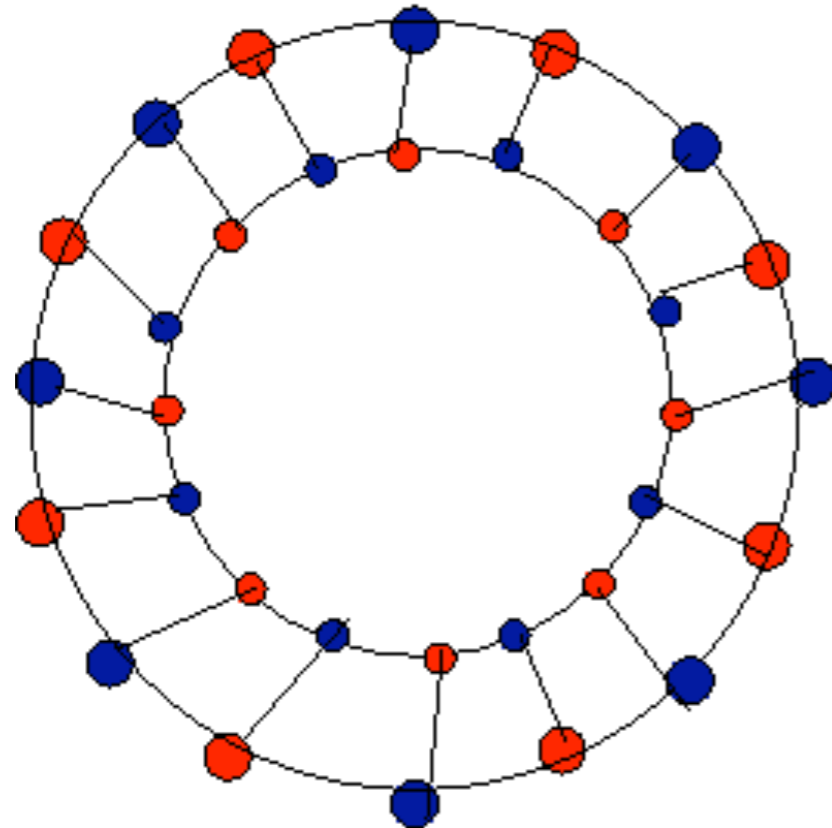
More complex motions almost hopeless



Multi-D Lagrangian Gridding



Remesh



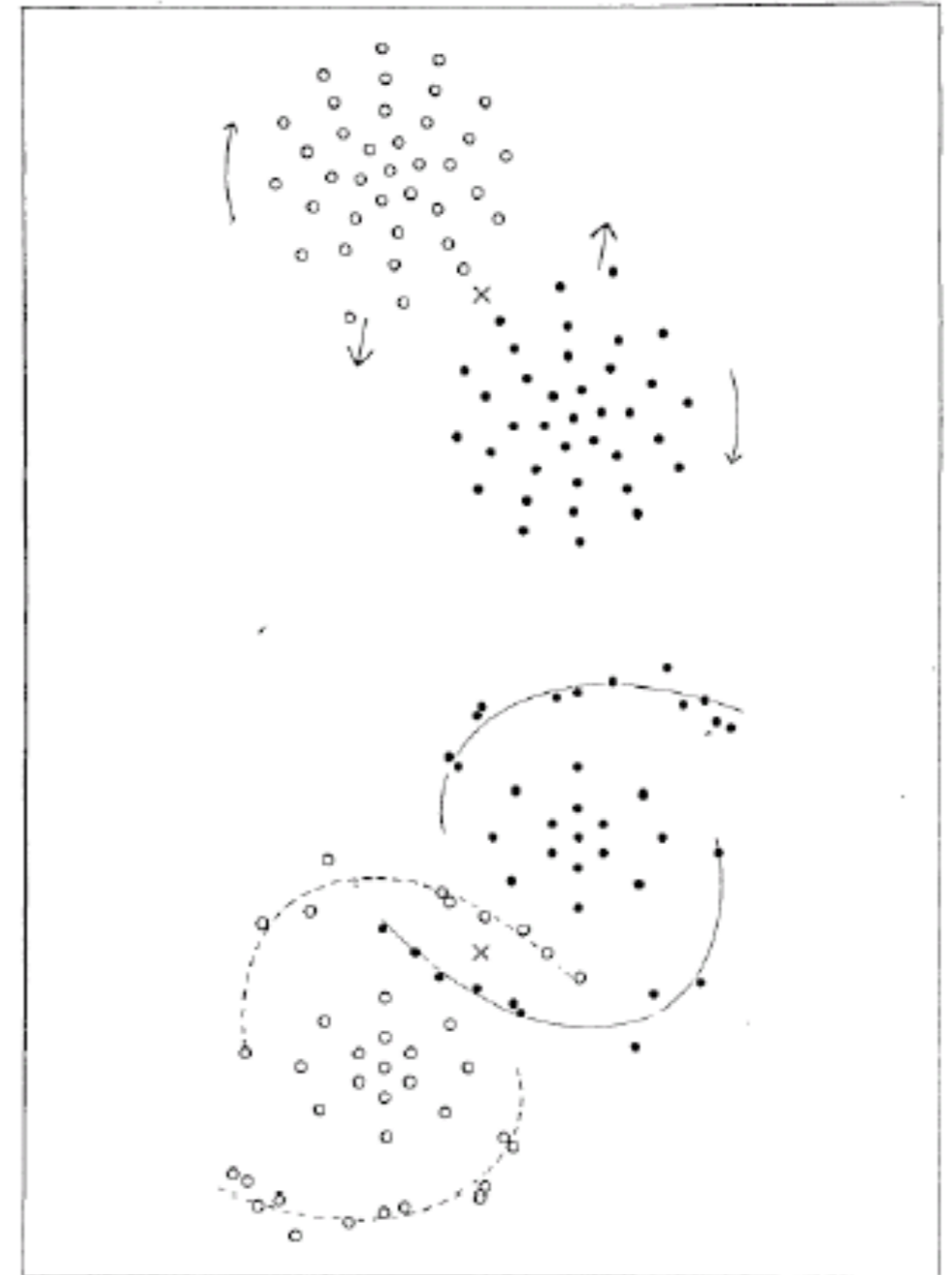
Can deal with this problem by remeshing every so often

Remeshing can be a very expensive step (choosing an optimal new mesh for a set of points is difficult)

Loss of main benefit of Lagrangian method – diffusive (as fluid ‘moves through’ remeshing)

Gridless methods

- Grid is a way of assigning neighbors to structure local interactions.
- If can determine local neighbors without discretizing on a grid can avoid the issues with tangling/remapping
- Astrophysics has a long (>60 yr) history with one gridless method – gravitational N-body calculations



Holmberg (1941)

FIG. 4a

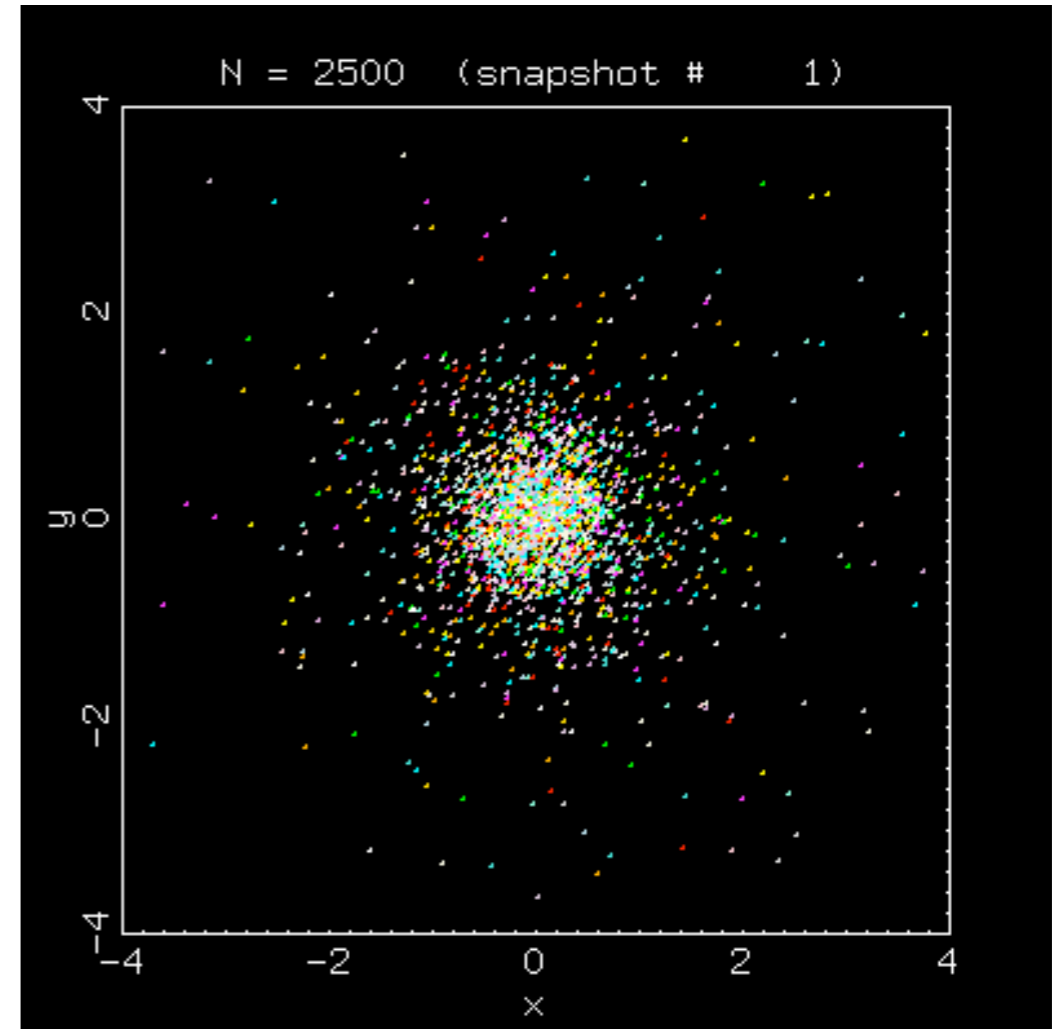
N-body calculation

It is clearly true that one can write the density field in a domain as integral over infinite number of point particles:

$$\rho(\vec{r}) = \frac{1}{V} \int_V \delta(\vec{r}' - \vec{r}) dm$$

N-body calculation approximates this by using a finite number of particles

$$\rho(\vec{r}) \approx \frac{1}{V} \sum_{i: \vec{r}_i \in V} m_i$$



<http://www.physics.drexel.edu/~steve/n-body.html>

Smoothed Particle Hydrodynamics

For hydrodynamics, interactions need to be local

Quantities stored at N 'free' particles

If infinite number of particles, any hydrodynamic quantity A could be defined as

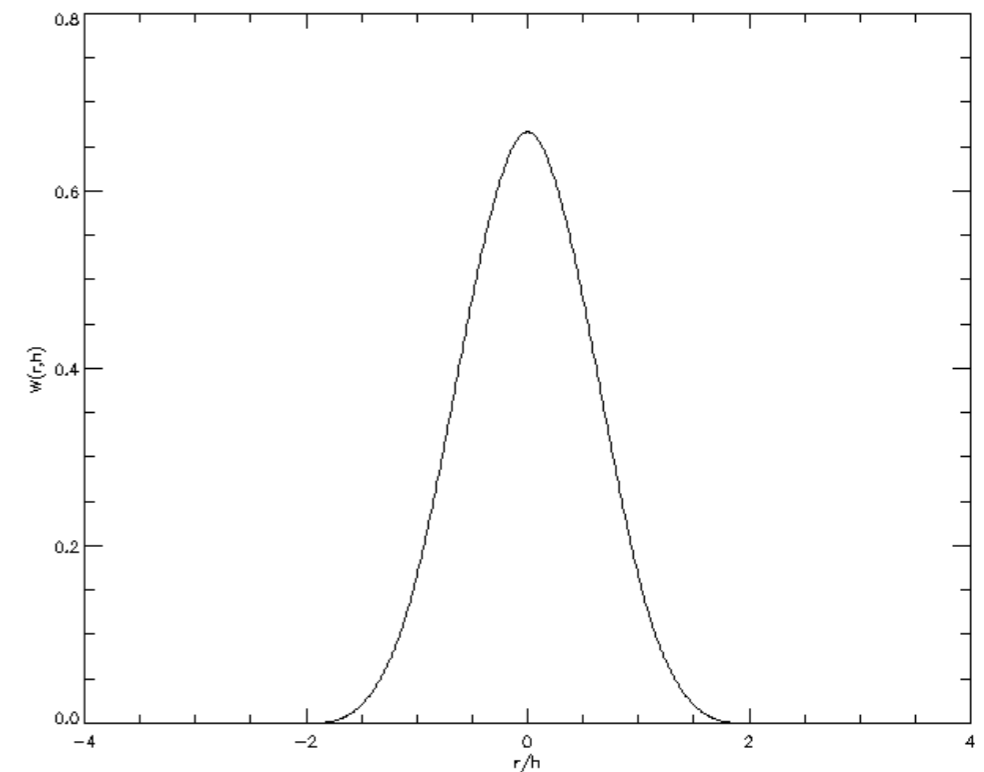
$$A(\vec{r}) = \int A(\vec{r}') \delta(\vec{r} - \vec{r}') dV$$

Finite number of particles – quantities must be smoothed over some finite smoothing length h

$$A(\vec{r}) = \int A(\vec{r}') W(|\vec{r} - \vec{r}'|, h) dV$$

Properties of smoothing function $W(r,h)$

- In small h limit, goes to delta function
- (usually) symmetric about $r=0$
- Compact support – W is exactly zero outside of some finite radius around the particle
- Cubic spline is typical choice



SPH 'Discretization' Error

$$A(\vec{r}) = \int A(\vec{r}') W(|\vec{r} - \vec{r}'|, h) dV$$

Even in this limit, smoothed quantity has error $O(h^2)$ [Why?]

Very difficult (likely impossible) to have robust, stable smoothing with higher order accuracy.

With finite number of particles,

$$A(\vec{r}) \approx \sum_b A_b \frac{m_b}{\rho_b} W(|\vec{r} - \vec{r}_b|, h)$$

SPH 'Discretization' Error

$$A(\vec{r}) \approx \sum_b A_b \frac{m_b}{\rho_b} W(|\vec{r} - \vec{r}_b|, h)$$

Taylor expand this around particle 'a's position,
and define $W_{ab} = W(r_a - r_b, h)$

$$A(r) = A_a \sum_b \frac{m_b}{\rho_b} W_{ab} + \nabla A_a \sum_b \frac{m_b}{\rho_b} \vec{r}_{ab} W_{ab} + \dots$$

Even for constant function (say, $A = 1$), not
guaranteed exact; must divide by first term, eg
do SPH interpolation of A / SPH interpolation
of 1.

SPH `Discretization' Error

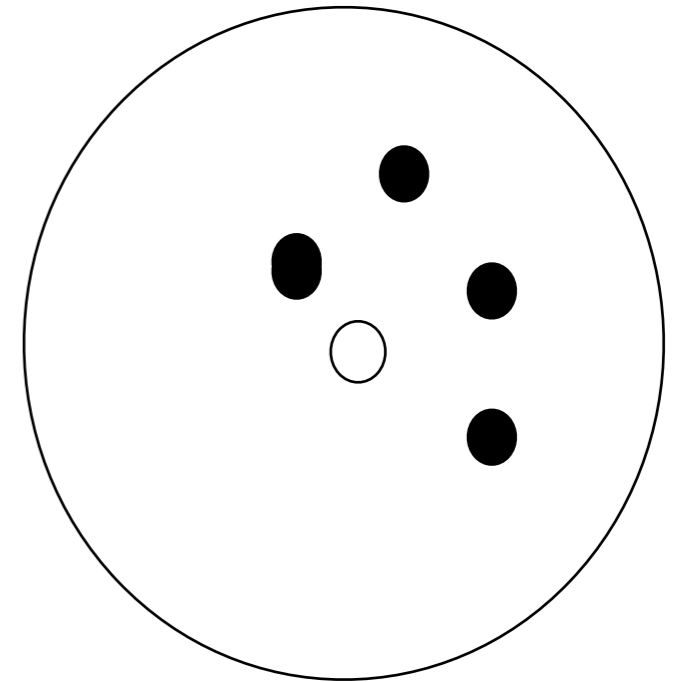
Error comes from discretization – finite number of particles

No guarantee that there will be enough particles well enough distributed so that

$$\sum_b \frac{m_b}{\rho_b} W_{ab} = 1$$

although corrections can be made

This problem is much worse for derivatives; we know that numerical derivatives of `noisy' data is hard.



Equation of Continuity

Mass conservation is already guaranteed;
each particle has its own mass, which
doesn't change

Density has particularly simple form in SPH
summation interpolant

$$\rho_a = \sum_b m_b W_{ab}$$

Taking lagrangian time derivative results in

$$\frac{d\rho_a}{dt} = \sum_b m_b (\vec{v}_a - \vec{v}_b) \cdot \nabla_a W_{ab}$$

Lagrangian Formulation

Can now express other equations in terms of
Lagrangian, Hamiltonian dynamics

Lagrangian for Hydrodynamics is

$$L = \int \left(\frac{1}{2} \rho v^2 - \rho u \right) dV$$
$$L = \sum_b m_b \left(\frac{1}{2} v_b^2 - u_b(\rho_b, s_b) \right) dV$$

from Euler-Lagrange equations, get eqn of motion

$$\frac{d\vec{v}_a}{dt} = - \sum_b m_b \left(\frac{P_a}{\rho_a^2} + \frac{P_b}{\rho_b^2} \right) \nabla_a W_{ab}$$

Momentum Equation

$$\frac{d\vec{v}_a}{dt} = - \sum_b m_b \left(\frac{P_a}{\rho_a^2} + \frac{P_b}{\rho_b^2} \right) \nabla_a W_{ab}$$

Note symmetry; contribution to momentum of particle a from b equal and opposite to b from a

Conserves momentum exactly

This form of gradient of pressure has some discreteness inaccuracies, but the symmetry is much more important

Energy Equation

Similarly, Hamiltonian can be written

$$H = \sum_b m_b \left(\frac{1}{2} v_a^2 + u_a \right)$$

implying a specific energy equation

$$\frac{de_a}{dt} = \sum_b m_b \left(\frac{P_a}{\rho_a^2} \vec{v}_b + \frac{P_b}{\rho_b^2} \vec{v}_a \right) \cdot \nabla_a W_{ab}$$

again, note symmetry.

Because of troubles with internal energy evolution in high-speed flows, some SPH practitioners evolve entropy rather than energy; applies to grid codes too.

Meshless Lagrangian vs Grid Codes

- Handles open, free boundaries much better
- Much less diffusion for bulk motion
- Automatically resolves high density region
- No need to waste computation on empty space
- Couples naturally to N-body gravity
- Very robust
- Poor at dealing with shocks
- Low-order spatial accuracy
- Derivatives harder, making some physics (MHD) harder
- More caution required with initial conditions
- Hard to follow interesting dynamics in low-density regions
- Too robust?

Spectral Methods

Spectral methods

- Different approach to finite difference/
volume
- Instead of local low-order
approximations, use global basis functions

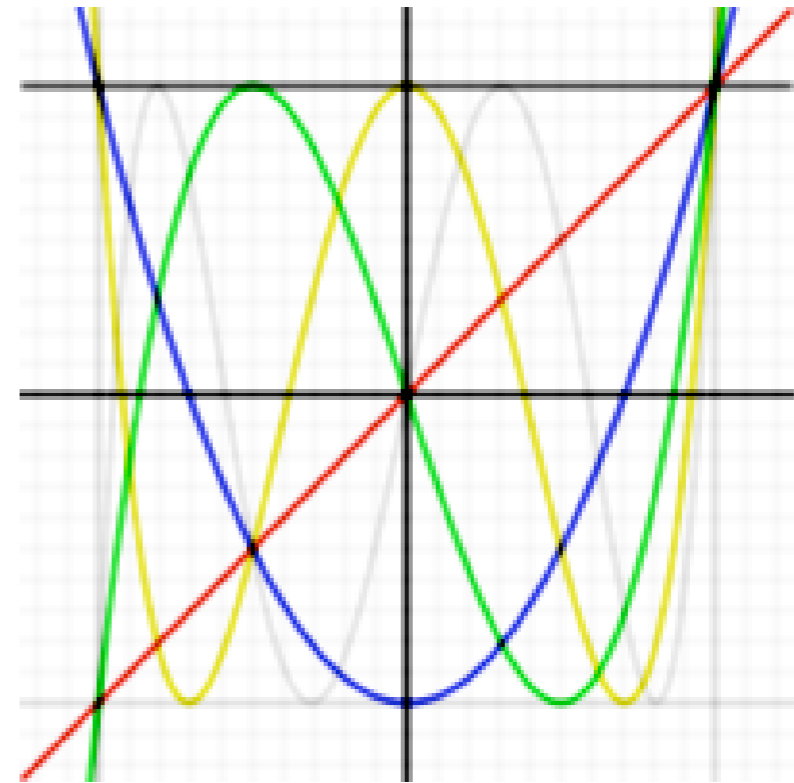
Spectral methods

- Works extremely well for linear PDEs -- differentials become multiplications!
- Poisson Equation

Spectral methods

- Typical Example:
Fourier decomposition
- FFT helps
- But works best with
periodic BCs
- Can use other bases
- Can constrain
coefficients to enforce
BCs

$$T_n(\cos \theta) = \cos(n\theta)$$



Wikipedia

<http://en.wikipedia.org/wiki/Image:Chebyshev.png>

Spectral Methods

- Advantages:
 - Highly accurate ('exponential convergence')
 - Not that much harder to deal with than FV, FD methods
- Disadvantages
 - Because global, hard to parallelize
 - Difficulty dealing with non-smooth quantities