NSF/GPAP SUMMER SCHOOL

on plasma physics for astrophysicists

Numerical Methods for MHD

Jim Stone

Institute for Advanced Study

Start with the equations of compressible viscous and resistive MHD:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot [\rho \mathbf{v}] = 0$$

$$\frac{\partial (\rho \mathbf{v})}{\partial t} + \nabla \cdot [\rho \mathbf{v}\mathbf{v} - \mathbf{B}\mathbf{B} + \mathsf{P}^* + \Pi] = 0$$

$$\frac{\partial E}{\partial t} + \nabla \cdot [(E + P^*)\mathbf{v} - \mathbf{B}\,(\mathbf{B} \cdot \mathbf{v}) + \Pi \cdot \mathbf{v} + \eta \mathbf{J} \times \mathbf{B}] = 0$$

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

$$P^* = P + \frac{\mathbf{B} \cdot \mathbf{B}}{2}$$

$$E = P/(\gamma - 1) + \frac{\rho(\mathbf{v} \cdot \mathbf{v})}{2} + \frac{\mathbf{B} \cdot \mathbf{B}}{2}$$

$$\Pi_{ij} = \rho\nu \left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_j} - \frac{2}{3}\delta_{ij} \nabla \cdot \mathbf{v} \right)$$

Also need an equation of state P = P(ρ,T). Usually adopt the ideal gas law $P=nkT$ or $P = (\gamma\text{-}1)e$ where for monoatomic gas (H), γ=5/3

Numerical methods for MHD equations can be classified by discretization strategy

Methods based on a spatial mesh

- Finite difference
- Finite volume
- Central schemes

Methods based on a spectral decomposition

- Global basis (Fourier, Chebyshev)
- Local polynomials (discontinuous Galerkin)

Methods based on Lagrangian (particle) mesh

- Smooth Particle Hydrodynamics (SPH)

Each method has various strengths/weaknesses.
In this lecture, focus on finite volume (FV) methods.

But before discussing details, a quick review of basic numerical numerical analysis…

Round-off error

Not all numbers on real axis can be represented.

If floating point operations result in a number that cannot be represented, some sort of rounding must be used.

Rounding is *correct* if no machine number lies between x and its rounded value x'. Difference between x and x' is the *round-off error*.

Can be rigorously proved that the relative error of a rounded value is bounded by a small, machine dependent number (the *machine precision*), that is

$$\frac{|x - x'|}{|x|} < \epsilon$$

Basis for all rigorous error analyses of numerical methods

Truncation error

Numerical algorithms approximate the true (analytic) solution using algebraic operations.

Difference between true and approximate (numerical) solution is ***truncation error***.

TE is not related to the finite precision of numbers on a computer (round-off error). Would exist even on a perfect machine with no round-off error.

TE is under programmer's control. Much of numerical analysis is trying to reduce it.

Convergence, consistency, and stability

Convergence: truncation error should decrease as grid spacing Δx decreases (numerical resolution increases).

Higher order schemes converge faster, so in general are better.

But, (1) cost and code complexity put a practical limit on the usefulness of high-order schemes, (2) all methods are first-order for discontinuities, so good shock capturing is essential.

In practice, "flops are free" on modern processors, which makes high-order schemes very attractive in the future.

Consistency: Solution of the discrete equations should be consistent with solutions of the underlying PDEs. Solution should converge to the analytic solution.

Stability: discrete equations must not admit any exponentially growing solutions, as they will be seeded by round-off error and eventually dominate.

Finite differencing

Simplest discretization of the simplest hyperbolic PDE (scalar linear advection equation) is forward-time centered-space (FTCS):

$$\frac{\partial u}{\partial t} + a\frac{\partial u}{\partial x} = 0 \quad \text{becomes} \quad \frac{u_j^{n+1} - u_j^n}{\Delta t} + a\left(\frac{u_{j+1}^n - u_{j-1}^n}{2\Delta x}\right) = 0$$

Perform von Neumann stability analysis.
For constant coefficients, the *analytic* solution to the difference equation is of the form: $u_j^n = \xi^n \exp ikj\Delta x$

Substitute this form into FDE, find $\xi(k) = 1 - i\left(\frac{a\Delta t}{\Delta x}\right)\sin k\Delta x$

Note that $|\xi(k)| > 1$ for ***all*** $\Delta t > 0$

Method is *unconditionally unstable*.

Lax-Friedrichs

Change time derivative in FTCS to use average of u at t^n, get LF:

$$\frac{u_j^{n+1} - (u_{j+1}^n + u_{j-1}^n)/2}{\Delta t} + a\left(\frac{u_{j+1}^n - u_{j-1}^n}{2\Delta x}\right) = 0$$

von Neumann stability analysis now gives: $\xi(k) = \cos k\Delta x - i\frac{a\Delta t}{\Delta x}\sin k\Delta x$

This has $|\xi(k)| < 1$ iff $\boxed{\frac{a\Delta t}{\Delta x} \leq 1}$

This is the *Courant-Levy-Friedrichs (CFL) stability criterion*.

Why does LF work and FTCS does not? Rewrite difference equation:

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

This is just FD form for the mixed PDE: $\frac{\partial u}{\partial t} + a\left(\frac{\partial u}{\partial x}\right) = \kappa\frac{\partial^2 u}{\partial x^2}$ $\kappa = \frac{\Delta x^2}{2\Delta t}$

LF adds explicit viscosity which makes algorithm stable.

Upwind methods

Write spatial derivative using one-sided differences that depend on sign of velocity

$$\frac{u_j^{n+1}-u_j^n}{\Delta t} = -a\left(\frac{u_j^n - u_{j-1}^n}{\Delta x}\right) \qquad \text{if a} > 0$$

$$-a\left(\frac{u_{j+1}^n - u_j^n}{\Delta x}\right) \qquad \text{if a} < 0$$

Remarkably, can rewrite upwind FDE in same form as LF:

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

So upwind method also adds explicit diffusion, with $\kappa = \frac{a\Delta x}{2}$

Define the "CFL number" $C = \frac{\Delta t}{\Delta x/a}$

which is the ratio of timestep to maximum stable timestep.

For C < 1, upwind methods add *less* diffusion than LF.

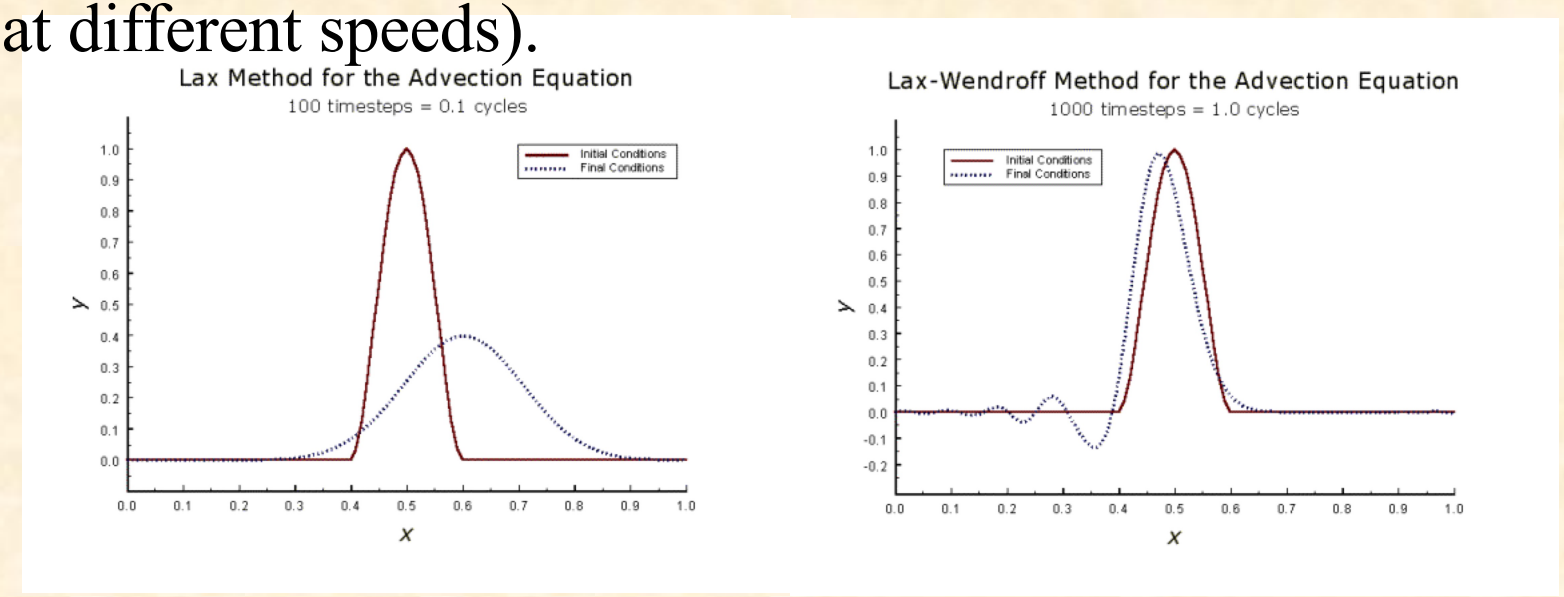
Diffusion versus dispersion error

Analytic solution to FDE may approximate a PDE (the *modified equation*) that is different from the one we are trying to solve.

e.g. we saw LF FDE actually solves: $\frac{\partial u}{\partial t} + a\left(\frac{\partial u}{\partial x}\right) = \kappa \frac{\partial^2 u}{\partial x^2}$

Clearly, LF (and first-order upwind) FDE add *diffusion error*.

Can show some methods (e.g. LW) add *dispersion error* (different k propagate at different speeds).



Dispersion error can be a serious problem for discontinuous solutions.

Finite Volume Methods

FV methods are the extension of upwind schemes to the full system of MHD equations.

Can show truncation error in FV methods is strictly diffusive. *Very desirable property for numerical methods.*

FV methods are implemented in many public codes.

Athena	VAC
AstroBEAR	BATS-R-US
RAMSES	FLASH
PLUTO	HARM
Enzo	Cosmos++
Einstein Toolkit	

This lecture will focus on Athena

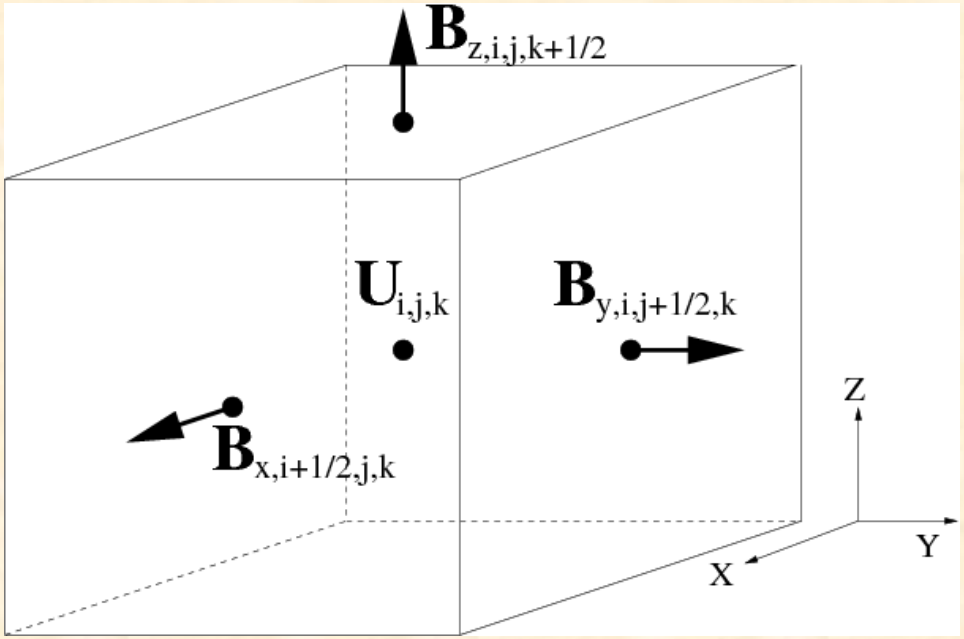
Discretization with FV Methods.

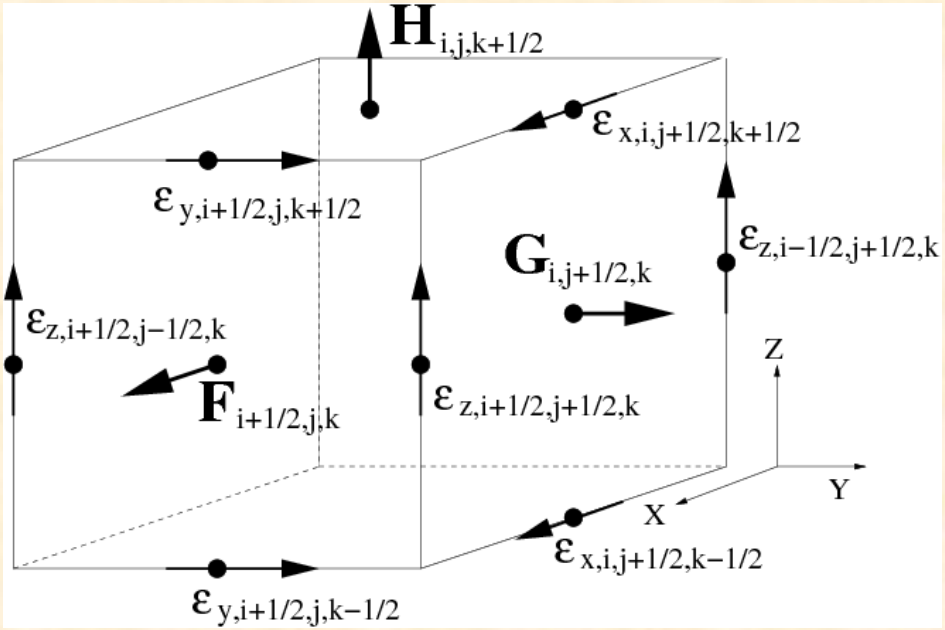
Uses cell-centered mass, momentum, energy and face-centered fields where cell-centered values are averaged over the volume:

$$\mathbf{U}^n_{i,j,k} = \frac{1}{\delta x \delta y \delta z} \int_{z_{k-1/2}}^{z_{k+1/2}} \int_{y_{j-1/2}}^{y_{j+1/2}} \int_{x_{i-1/2}}^{x_{i+1/2}} \mathbf{U}(x, y, z, t^n)\, \mathrm{d}x\, \mathrm{d}y\, \mathrm{d}z$$

and face centered values are averaged over areas, e.g.;

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





Also requires face-centered fluxes, and edge-centered EMFs.

Finite Volume Discretization

Conservations laws for mass, momentum and energy can all be written as

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

Integrate over the volume of a grid cell, and over a timestep dt, apply the divergence theorem to give

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

(This equation is exact -- no approximations have been made!)

Where, in the previous equations:

$$\mathbf{U}^{n}_{i,j,k} = \frac{1}{\delta x \delta y \delta z} \int_{z_{k-1/2}}^{z_{k+1/2}} \int_{y_{j-1/2}}^{y_{j+1/2}} \int_{x_{i-1/2}}^{x_{i+1/2}} \mathbf{U}(x, y, z, t^n)\, \mathrm{d}x\, \mathrm{d}y\, \mathrm{d}z$$

are *volume-averaged* values, while

$$\mathbf{F}^{n+1/2}_{i-1/2,j,k} = \frac{1}{\delta y \delta z \delta t} \int_{t^n}^{t^{n+1}} \int_{z_{k-1/2}}^{z_{k+1/2}} \int_{y_{j-1/2}}^{y_{j+1/2}} \mathbf{F}(x_{i-1/2}, y, z, t)\, \mathrm{d}y\, \mathrm{d}z\, \mathrm{d}t$$

$$\mathbf{G}^{n+1/2}_{i,j-1/2,k} = \frac{1}{\delta x \delta z \delta t} \int_{t^n}^{t^{n+1}} \int_{z_{k-1/2}}^{z_{k+1/2}} \int_{x_{i-1/2}}^{x_{i+1/2}} \mathbf{G}(x, y_{j-1/2}, z, t)\, \mathrm{d}x\, \mathrm{d}z\, \mathrm{d}t$$

$$\mathbf{H}^{n+1/2}_{i,j,k-1/2} = \frac{1}{\delta x \delta y \delta t} \int_{t^n}^{t^{n+1}} \int_{y_{j-1/2}}^{y_{j+1/2}} \int_{x_{i-1/2}}^{x_{i+1/2}} \mathbf{H}(x, y, z_{k-1/2}, t)\, \mathrm{d}x\, \mathrm{d}y\, \mathrm{d}t$$

are *area- and time-averaged* fluxes.

Importantly, FV discretization preserves volume-integrated values of **U** to machine precision.

Finite-Area discretization of the induction equation.

Integrate the induction equation over each cell face, apply Stokes Law

$$
\begin{aligned}
B^{n+1}_{x,i-1/2,j,k} = B^{n}_{x,i-1/2,j,k} \quad &- \quad \frac{\delta t}{\delta y}(\mathcal{E}^{n+1/2}_{z,i-1/2,j+1/2,k} - \mathcal{E}^{n+1/2}_{z,i-1/2,j-1/2,k}) \\
&+ \quad \frac{\delta t}{\delta z}(\mathcal{E}^{n+1/2}_{y,i-1/2,j,k+1/2} - \mathcal{E}^{n+1/2}_{y,i-1/2,j,k-1/2})
\end{aligned}
$$

$$
\begin{aligned}
B^{n+1}_{y,i,j-1/2,k} = B^{n}_{y,i,j-1/2,k} \quad &+ \quad \frac{\delta t}{\delta x}(\mathcal{E}^{n+1/2}_{z,i+1/2,j-1/2,k} - \mathcal{E}^{n+1/2}_{z,i-1/2,j-1/2,k}) \\
&- \quad \frac{\delta t}{\delta z}(\mathcal{E}^{n+1/2}_{x,i,j-1/2,k+1/2} - \mathcal{E}^{n+1/2}_{x,i,j-1/2,k-1/2})
\end{aligned}
$$

$$
\begin{aligned}
B^{n+1}_{z,i,j,k-1/2} = B^{n}_{z,i,j,k-1/2} \quad &- \quad \frac{\delta t}{\delta x}(\mathcal{E}^{n+1/2}_{y,i+1/2,j,k-1/2} - \mathcal{E}^{n+1/2}_{y,i-1/2,j,k-1/2}) \\
&+ \quad \frac{\delta t}{\delta y}(\mathcal{E}^{n+1/2}_{x,i,j+1/2,k-1/2} - \mathcal{E}^{n+1/2}_{x,i,j-1/2,k-1/2})
\end{aligned}
$$

Again, these equations are exact -- no approximation has been made.

Where, in the induction equation,

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

$$B^n_{y,i,j-1/2,k} = \frac{1}{\delta x \delta z} \int_{z_{k-1/2}}^{z_{k+1/2}} \int_{x_{i-1/2}}^{x_{i+1/2}} B_y(x, y_{j-1/2}, z, t^n)\ \mathrm{d}x\ \mathrm{d}z$$

$$B^n_{z,i,j,k-1/2} = \frac{1}{\delta x \delta y} \int_{y_{j-1/2}}^{y_{j+1/2}} \int_{x_{i-1/2}}^{x_{i+1/2}} B_z(x, y, z_{k-1/2}, t^n)\ \mathrm{d}x\ \mathrm{d}y$$

are *area-averaged* components of the magnetic field, and

$$\mathcal{E}^{n+1/2}_{x,i,j-1/2,k-1/2} = \frac{1}{\delta x \delta t} \int_{t^n}^{t^{n+1}} \int_{x_{i-1/2}}^{x_{i+1/2}} \mathcal{E}_x(x, y_{j-1/2}, z_{k-1/2}, t)\ \mathrm{d}x\ \mathrm{d}t$$

$$\mathcal{E}^{n+1/2}_{y,i-1/2,j,k-1/2} = \frac{1}{\delta y \delta t} \int_{t^n}^{t^{n+1}} \int_{y_{j-1/2}}^{y_{j+1/2}} \mathcal{E}_y(x_{i-1/2}, y, z_{k-1/2}, t)\ \mathrm{d}y\ \mathrm{d}t$$

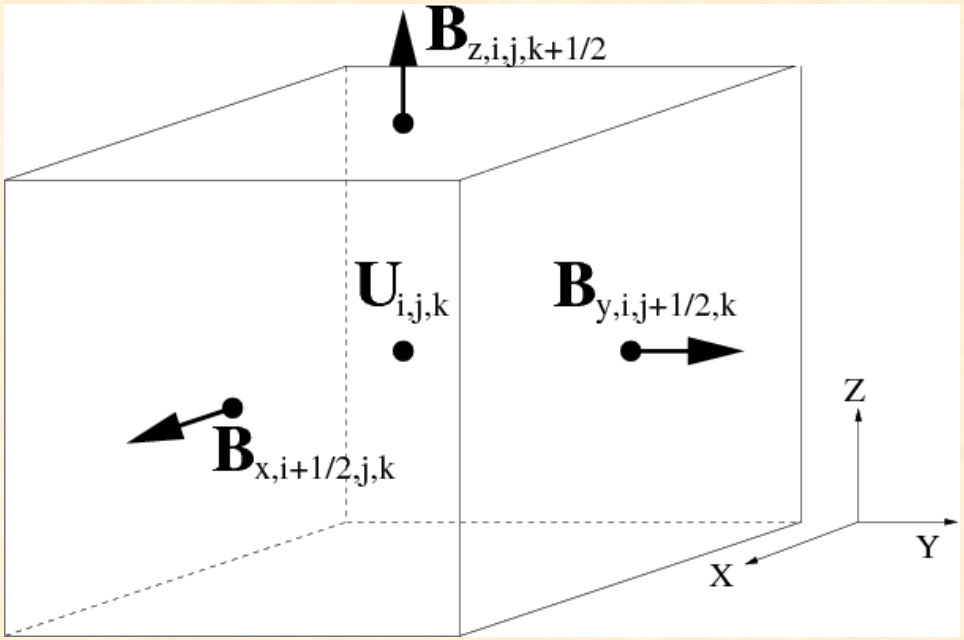
$$\mathcal{E}^{n+1/2}_{z,i-1/2,j-1/2,k} = \frac{1}{\delta z \delta t} \int_{t^n}^{t^{n+1}} \int_{z_{k-1/2}}^{z_{k+1/2}} \mathcal{E}_z(x_{i-1/2}, y_{j-1/2}, z, t)\ \mathrm{d}z\ \mathrm{d}t$$

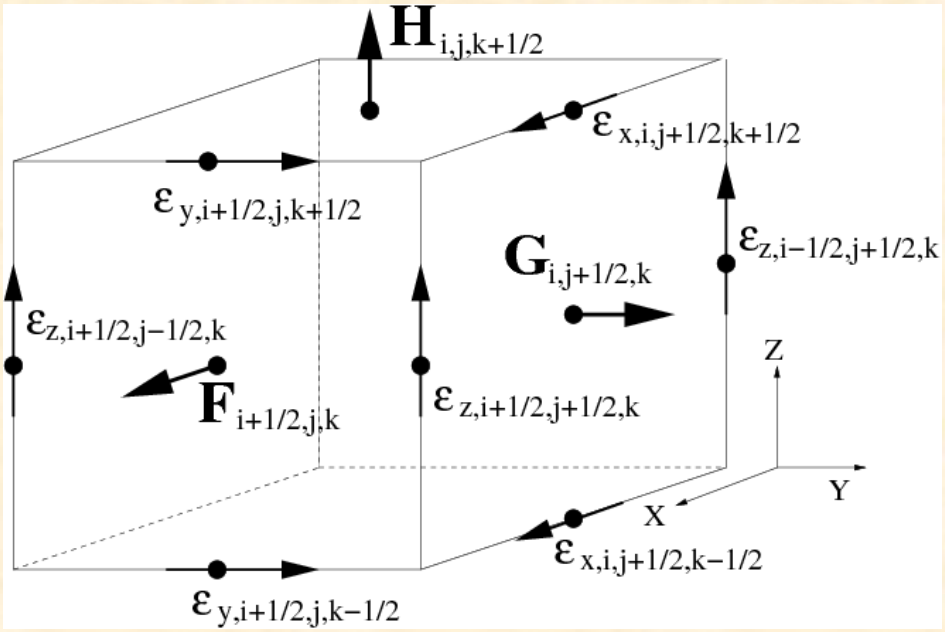
are *line- and time-averaged* electro-motive forces (v x B).

Finite-area discretization of induction equation preserves magnetic flux (and therefore div(B) constraint) to machine precision.

Discretization with FV Methods.

Uses cell-centered mass, momentum, energy; face-centered field:



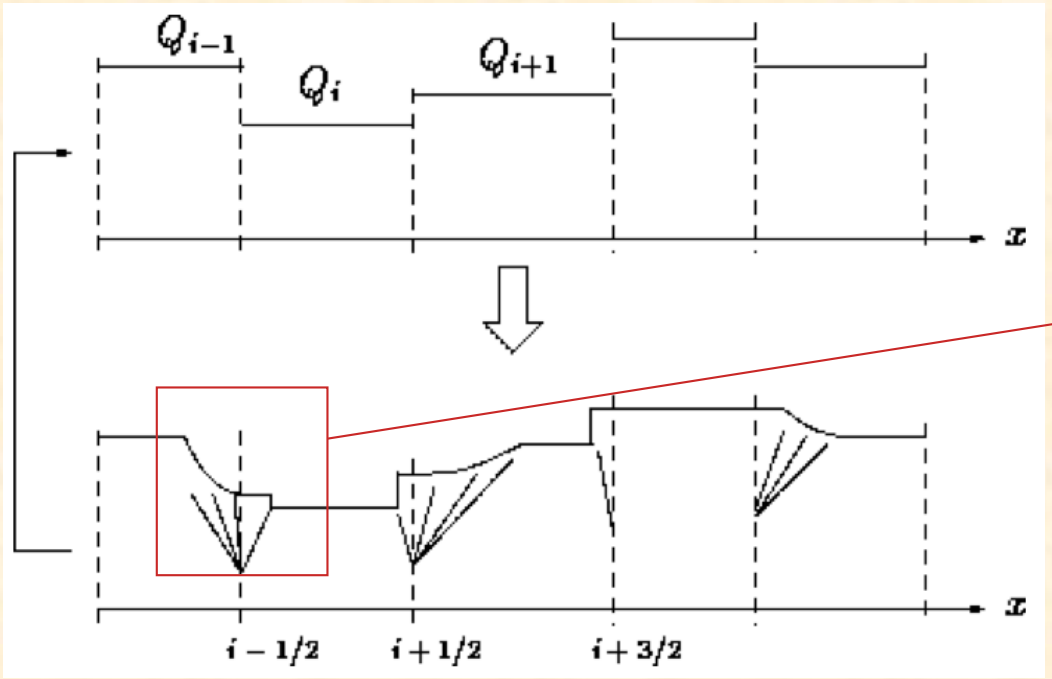


Uses face-centered fluxes, and edge-centered EMFs.

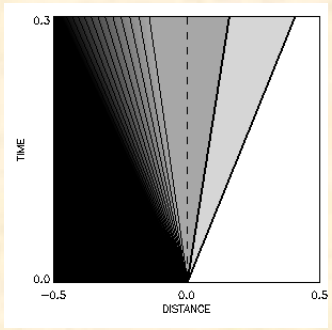
The key is how to compute these fluxes and EMFs all at once!

Godunov's orginal (first-order) method

- Difference in cell-averaged values at each grid interface define set of Riemann problems (evolution of initially discontinuous states).
- Solution of Riemann problems averaged over cell give time-evolution of cell-averaged values, until waves from one interface crosses the grid and interacts with the other, that is for $\Delta t \leq \Delta x/(v+C)$
- Due to conservation, don't actually need to solve Riemann problem exactly. Just need to compute state *at location of interface* to compute fluxes.



Flux given by solution along x=0



Then, solution evolved according to

$$\frac{\mathbf{U}_j^{n+1} - \mathbf{U}_j^n}{\Delta t} = -\frac{\mathbf{F}_{j+1/2}^{n+1/2} - \mathbf{F}_{j-1/2}^{n+1/2}}{\Delta x}$$

Riemann solvers

For pure hydrodynamics of ideal gases, exact/efficient nonlinear Riemann solvers are possible.

In MHD, nonlinear Riemann solvers are complex because:

1. There are 3 wave families in MHD – 7 characteristics
2. In some circumstances, 2 of the 3 waves can be degenerate (e.g. $V_{Alfven} = V_{slow}$)

Equations of MHD are not *strictly hyperbolic*
(Brio & Wu, Zachary & Colella)

Thus, in practice, MHD Godunov schemes use approximate and/or linearized Riemann solvers.

Many different approximations are possible:

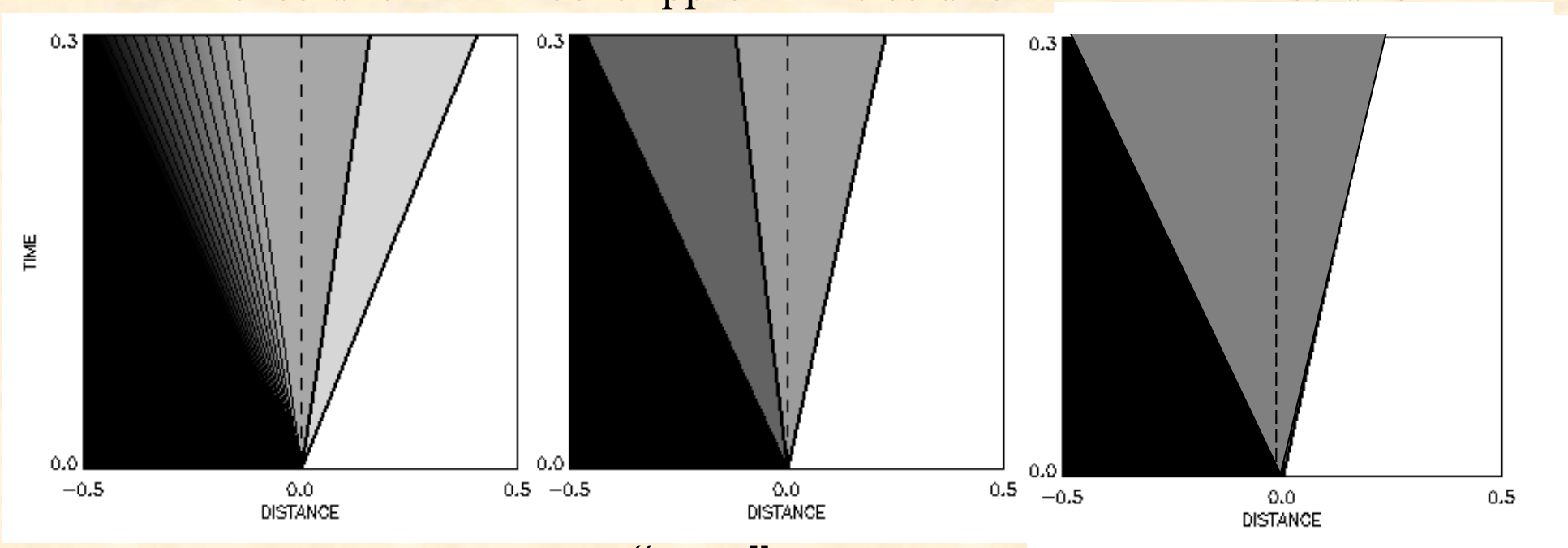
1. Roe's method – keeps all 7 characteristics, but treats each as a simple wave.
   - Good resolution of all waves
   - Requires characteristic decomposition in conserved variables
   - Expensive and difficult to add new physics
   - Fails for strong rarefactions

2. Harten-Lax-van Leer-Einfeldt (HLLE) method – keeps only largest and smallest characteristics, averages intermediate states in-between.
   - Very simple and efficient
   - Guarantees positivity in 1D
   - Very diffusive for contact discontinuities

3. HLLC(HLLD) methods – Adds entropy (and Alfven) wave back into HLLE method, giving two (four) intermediate states.
   - Reasonably simple and efficient
   - Guarantees positivity in 1D
   - Better resolution of contact discontinuities

Effect of various approximations on the solution to the Riemann problem in hydrodynamics

Exact solution Roe's approximate solution HLLE solution



So which approximation is "best"? Must explore the use of each.

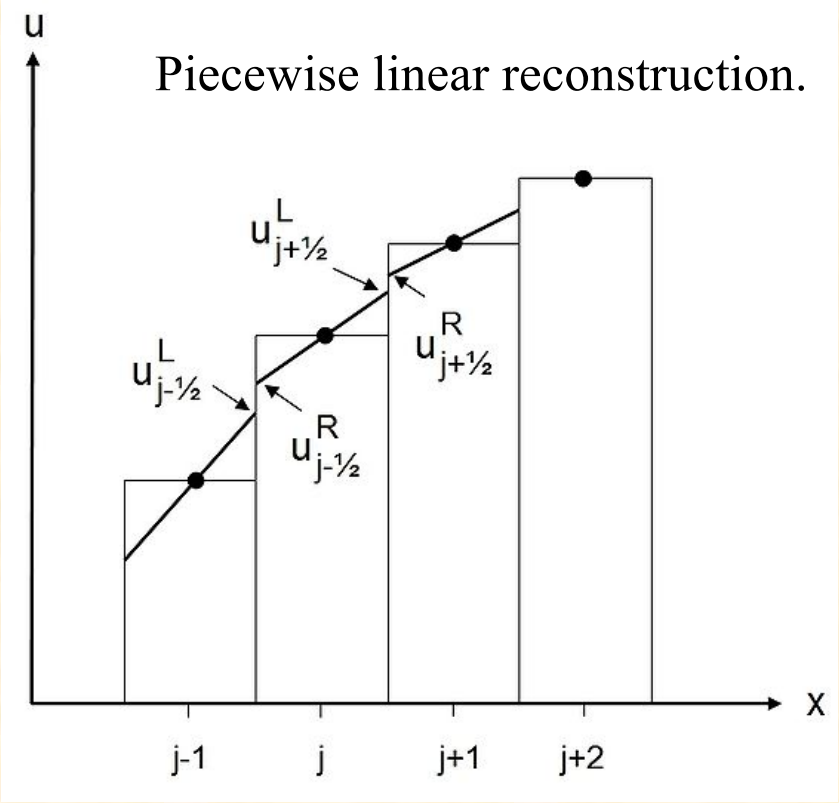
Use of a Riemann solvers is a benefit, not a weakness, of a Godunov method: makes shock capturing more accurate.

Higher-order reconstruction

• Using cell-centered values for left- and right-states to define Riemann problems at cell interfaces is first-order and very diffusive.

• Higher-order methods use piecewise linear (MUSCL), piecewise-parabolic (PPM), or WENO reconstruction within cells.

• Difference between L/R states is small for smooth flow, large near shocks. Riemann solver automatically gives correct dissipation for shocks.



Piecewise linear reconstruction.

Extensions to multidimensions

Traditionally, multidimensional methods are constructed using dimensional (directional) splitting:

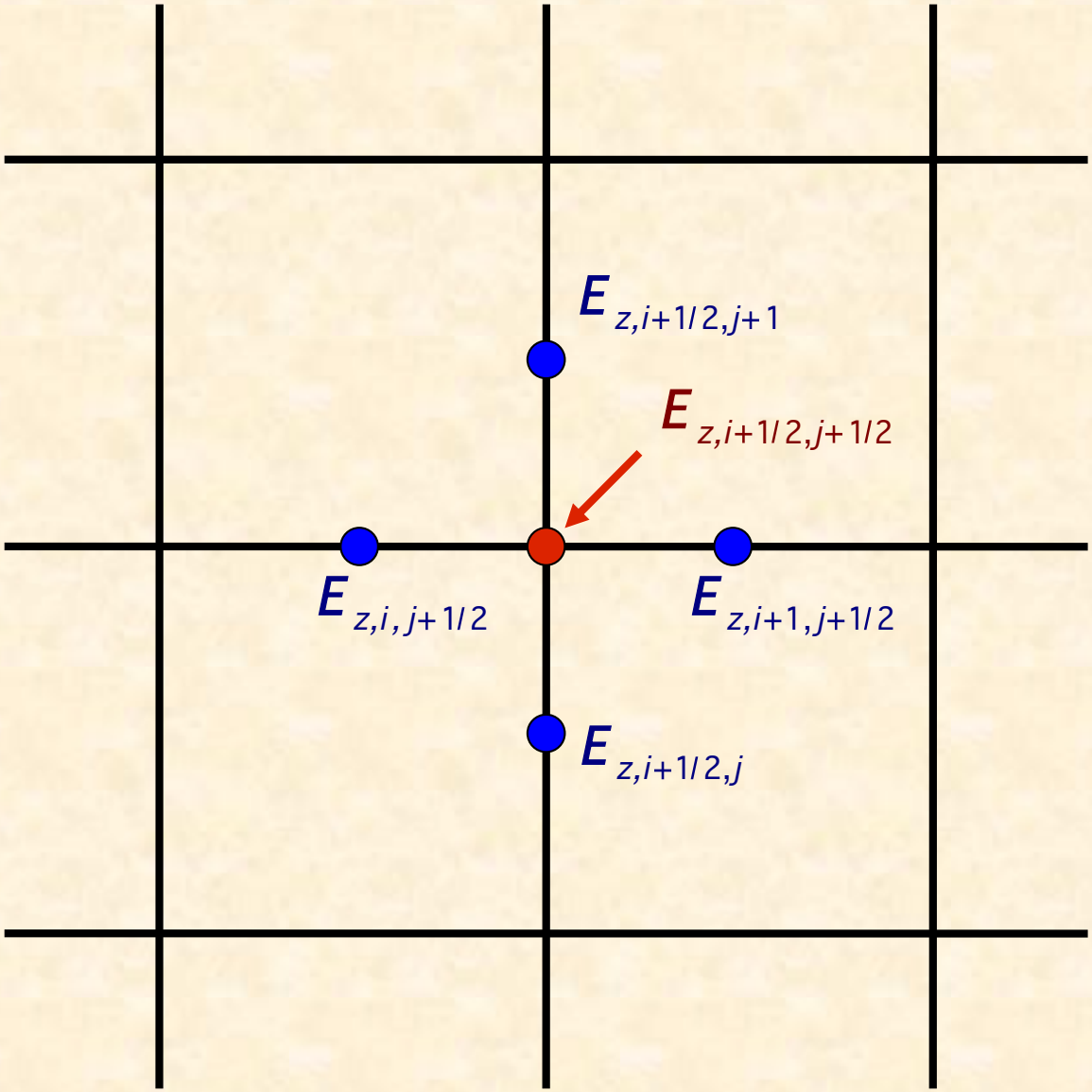
1. Solve $U_t = F_x$
2. Solve $U_t = G_y$, with G constructed from result of x-update
3. Solve $U_t = H_z$, with H constructed from result of y-update

Sometimes these sweeps are symmetrized to make splitting 2nd order in time (Strang splitting)

In MHD, must use *directionally unsplit* schemes to preserve div(B)=0, e.g. **constrained transport** (Evans & Hawley 1988).

In addition, higher than 2nd-order FV schemes require special treatment of spatial reconstruction, and conversion between conserved and primitive variables. Most codes are only 2nd-order, but 4th-order is becoming popular.

Constrained Transport in multidimensions



- Finite Volume / Godunov algorithm gives E-field at face centers.
- "CT Algorithm" needs E-field at grid cell corners.
- Arithmetic averaging: 2D plane-parallel flow does not reduce to equivalent 1D problem
- Algorithms which reconstruct E-field at corner are superior Gardiner & Stone 2005

Time integration via method of lines.

To achieve high-order accuracy in time, multistep strong-stability-preserving (SSP) Runge-Kutta integrators can be used, e.g. RK2, RK3, RK4 (Suresh & Huynh).

Let $\mathcal{L}$ represent RHS operator of MHD equations. Write Forward Euler as:

$$\mathcal{F}[f,t] = f + \Delta t \mathcal{L}(f,t)$$

Then, second order RK2 algorithm is:

$$f^{(1)} = \mathcal{F}[f^n, t^n]$$

$$f^{n+1} = \frac{1}{2} f^n + \frac{1}{2} \mathcal{F}[f^{(1)}, t^n + \Delta t]$$

Third-order RK3 algorithm is:

$$f^{(1)} = \mathcal{F}[f^n, t^n]$$

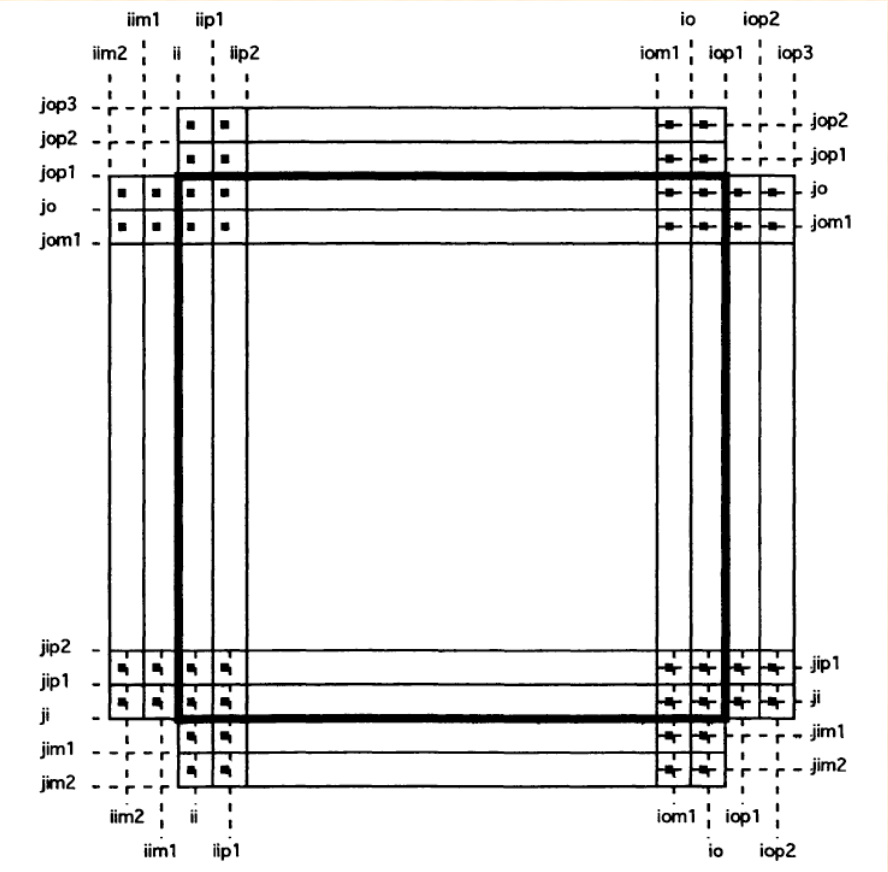
$$f^{(2)} = \frac{3}{4} f^n + \frac{1}{4} \mathcal{F}[f^{(1)}, t^n + \Delta t]$$

$$f^{n+1} = \frac{1}{3} f^n + \frac{2}{3} \mathcal{F}[f^{(2)}, t^n + \Delta t/2]$$

Note algorithms are dimensionally unsplit. All spatial operators added at once.

Boundary conditions

Most grid codes apply boundary conditions by specifying solution in extra rows of cells ("ghost" or "guard" zones) at boundary of grid. This algorithm requires 2 or 3 rows (for 2nd or 4th order upwind reconstruction respectively)



There are different ways of specifying solution in ghost zones for different BCs, e.g.

1. Reflecting
2. Inflow
3. Outflow
4. Periodic

BCs must be applied after every stage in multi-stage integrators.

Practical time step control

For stability, the integration time step is limited by the Courant-Freidrichs-Lewy (CFL) condition:

$$\Delta t \leq \Delta x/(v + C)$$

Must take minimum timestep over whole grid.

Can be thought of as limiting distance fastest wave travels in one time step is less than cell size everywhere on mesh.

Also can be derived more rigorously from formal von Neumann stability analysis.

Define Courant number: $C = \dfrac{\Delta t}{\Delta x/(v + C)}$

For stability C < 1

In multi-dimensions, $C < 1/N_{dim}$ for most RK integrators.

Some Tests

Five test problems we have found very useful (all drawn from basic physics of fluids):

1. Linear wave convergence
2. Nonlinear circularly polarized Alfven waves
3. Brio & Wu, and Ryu-Jones shocktubes
4. Field loop advection
5. MHD instabilities (KH, RT, MRI, etc.)

For MHD, ***must*** focus on multidimensional tests.

Convergence rate and ability to capture shocks are equally important.

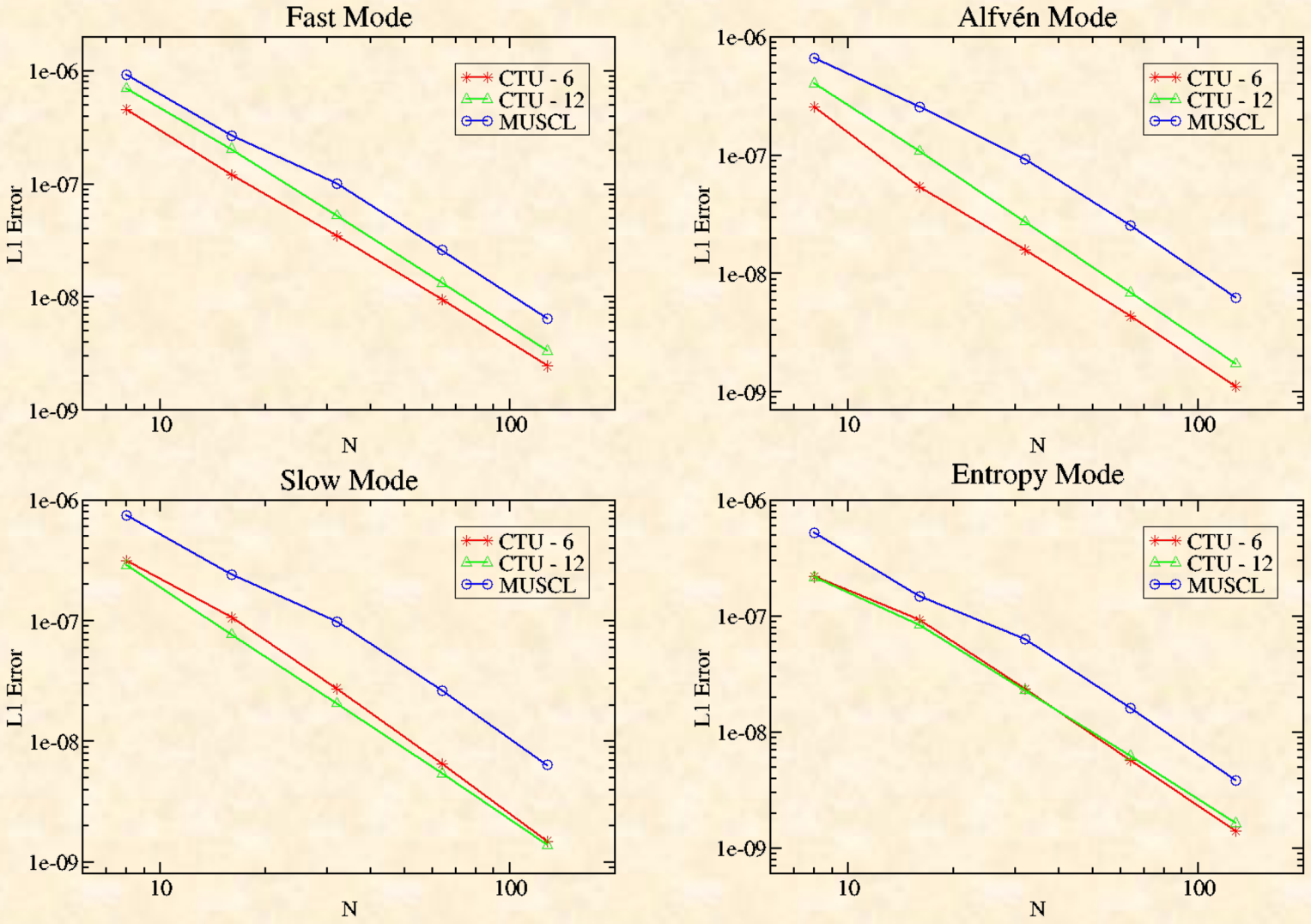
See http://www.astro.princeton.edu/~jstone/athena.html

Linear Wave Convergence: 3D (2N x N x N) grid

Initialize pure eigenmode for each wave family

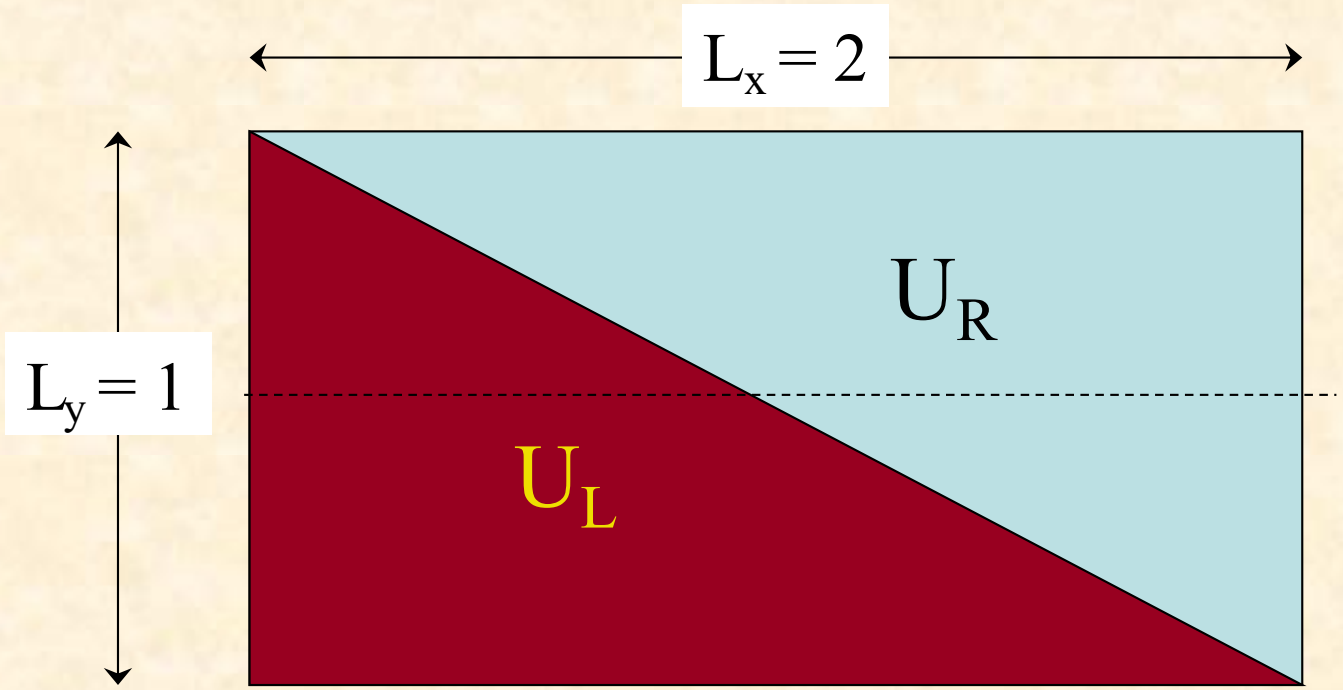
Measure RMS error in U after propagating one wavelength

quantitative test of accuracy of scheme



RJ2a Riemann problem rotated to grid

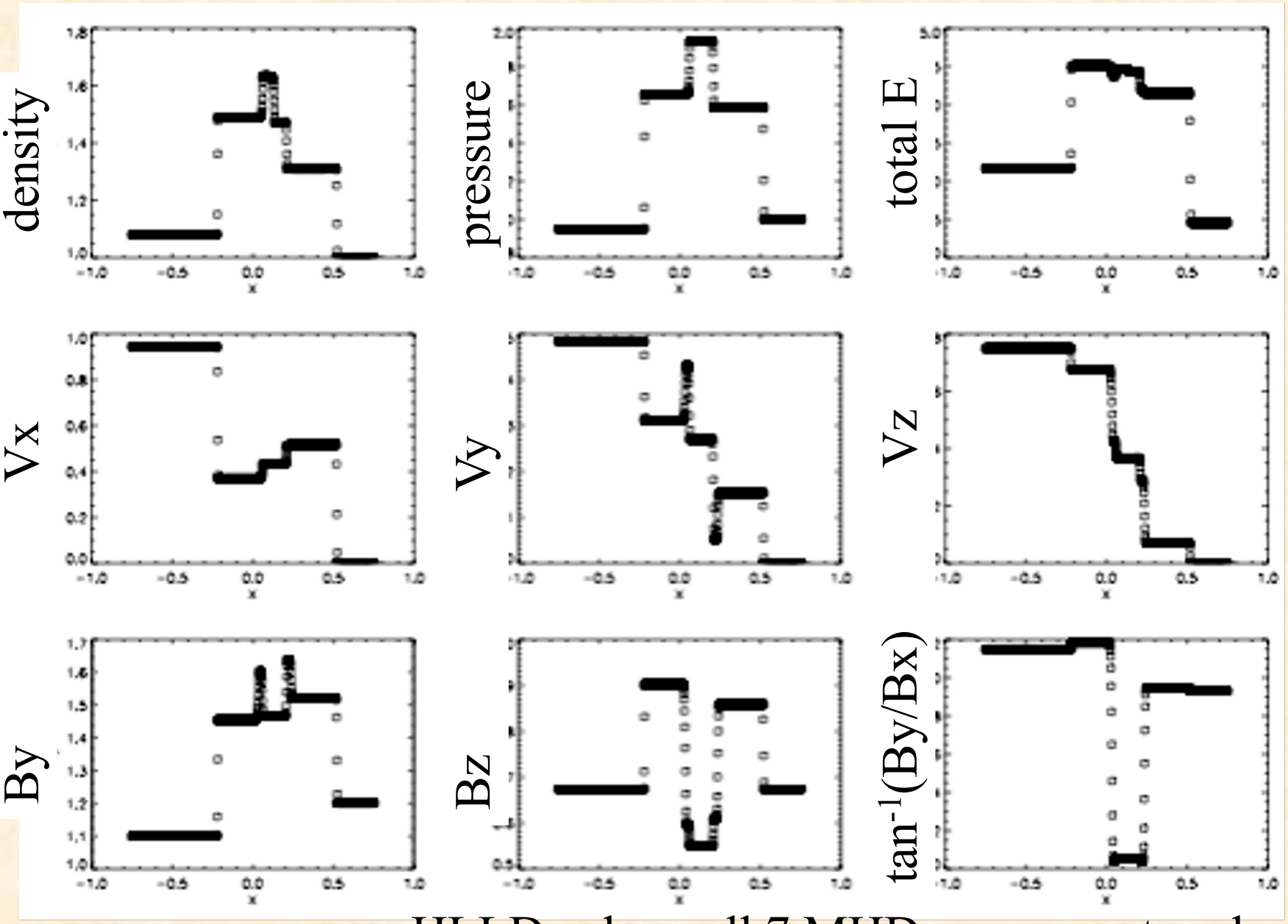
Initial discontinuity inclined to grid at $\tan^{-1}\theta = 1/2$
Magnetic field initialized from vector potential to ensure div(B)=0
$\Delta x = \Delta y$, 512 x 256 grid



Problem is Fig. 2a from Ryu & Jones 1995

Final result plotted along horizontal line at center of grid

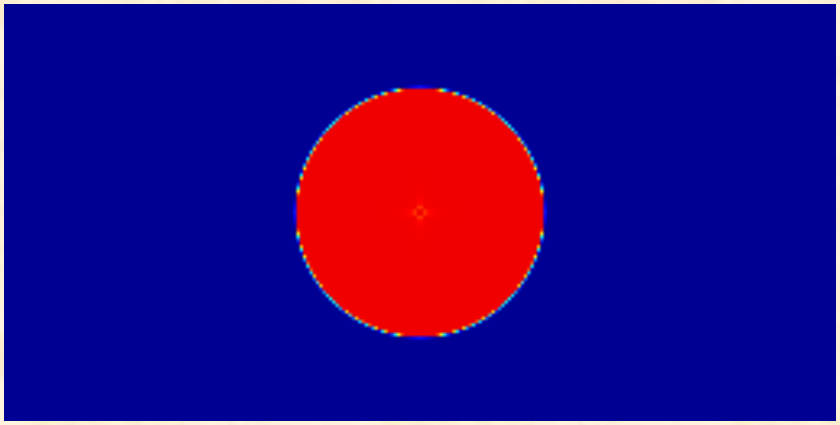
RJ2a shocktube in 3D (2N x N x N grid)



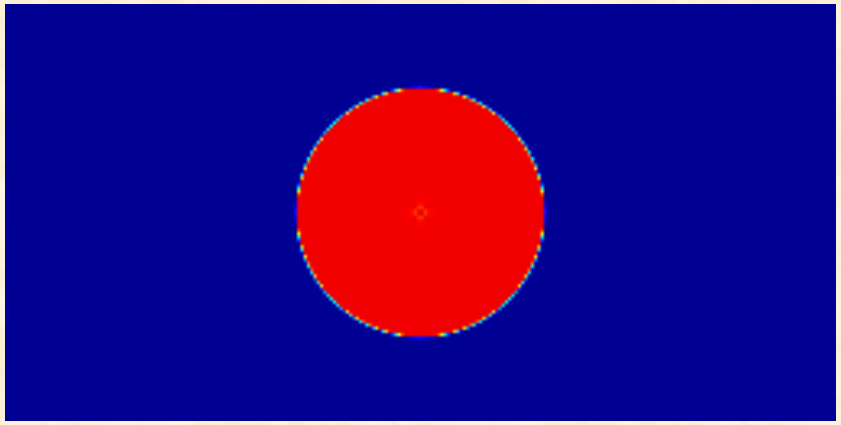
HLLD solver, all 7 MHD waves captured well.

Advection of a field loop (2N x N grid)

Field Loop Advection ($\beta = 10^6$): MUSCL - Hancock integrator
Movies of B^2



Arithmetic average
(Balsara & Spicer 1999)



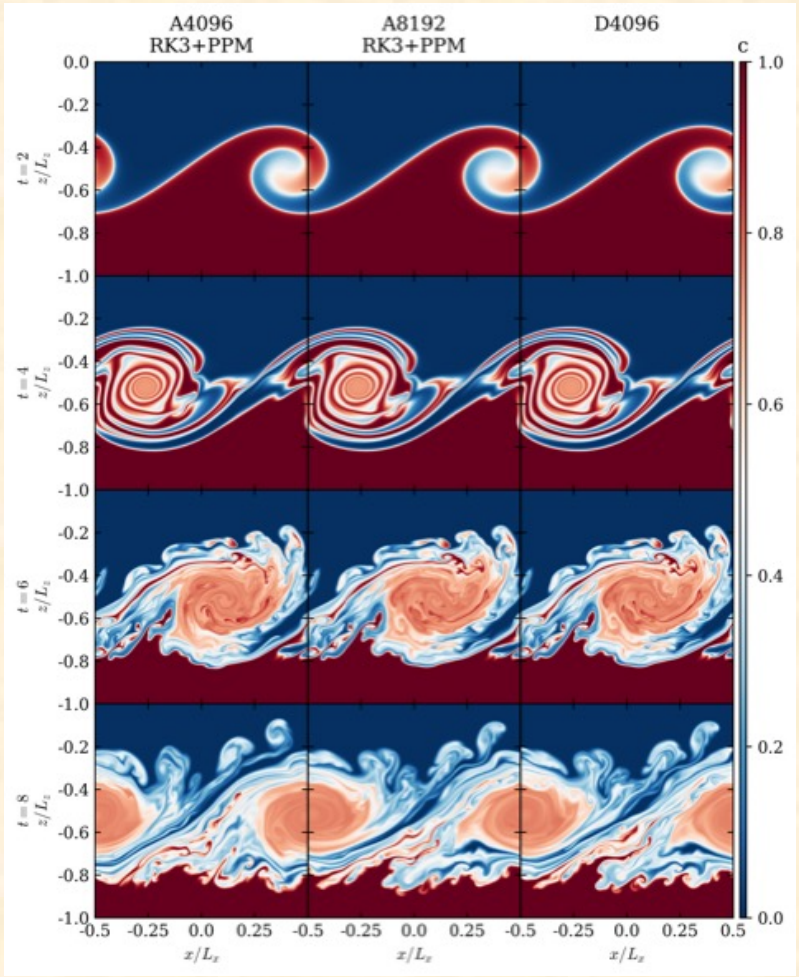
Gardiner & Stone 2005

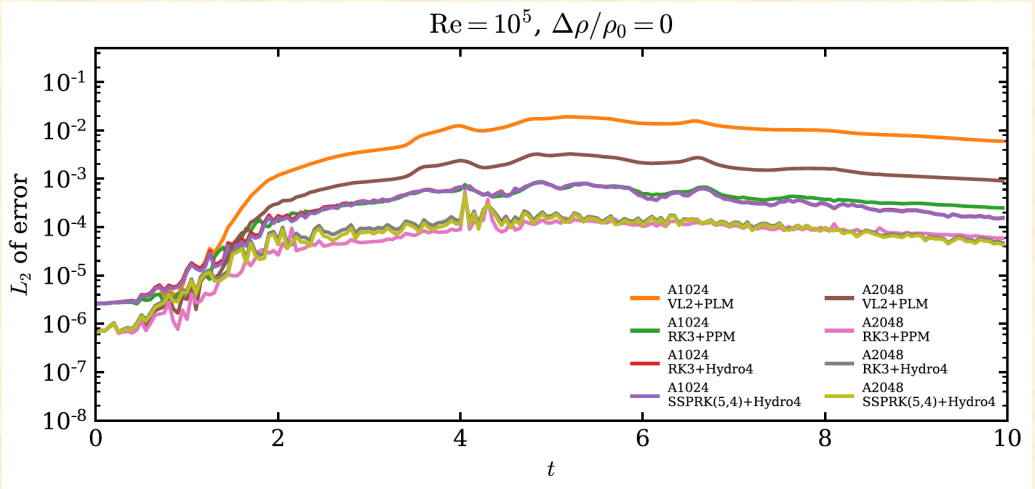
Good test of proper upwinding of electric fields in CT.

Good test of whether codes preserves div(B) on appropriate stencil: Run in 3D with non-zero V_z. Does method keep B_z zero?

Nonlinear regime of KH instability

- Resolved shear layer with tanh profile. Single more perturbation.
- Explicit viscosity and heat conduction: ***resolved reference solution***
- Good agreement between Athena and Dedalus (spectral code)

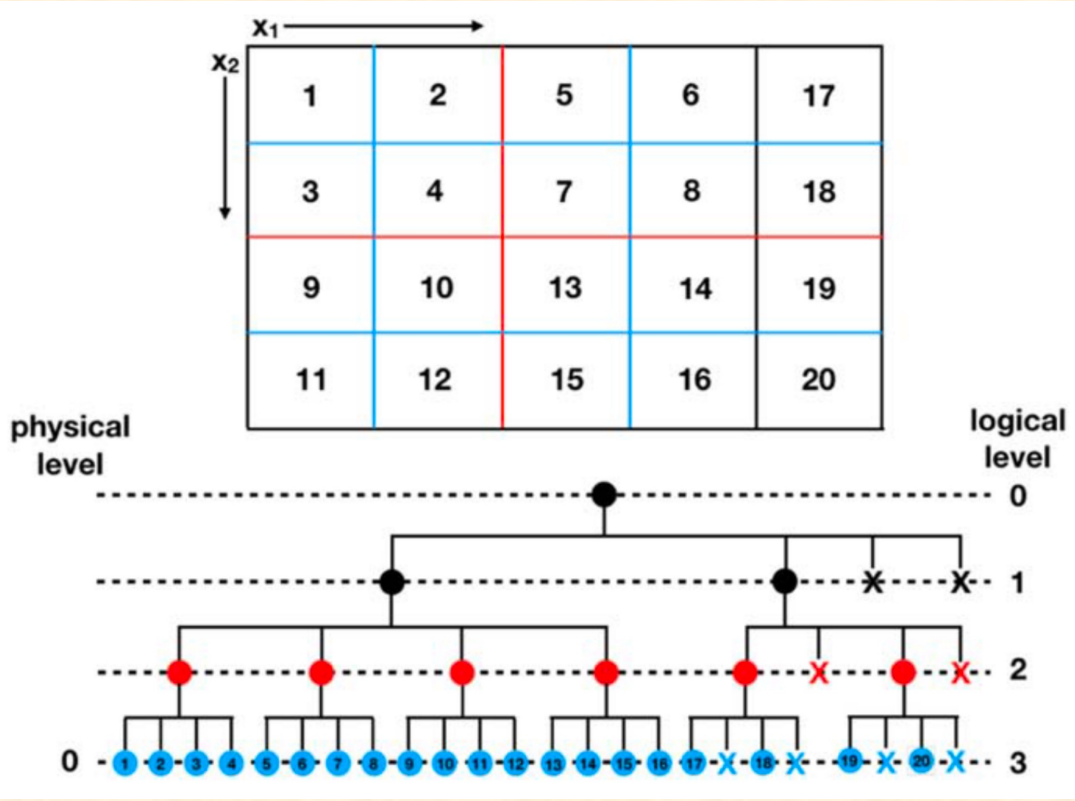




Lecoanet+ 2016

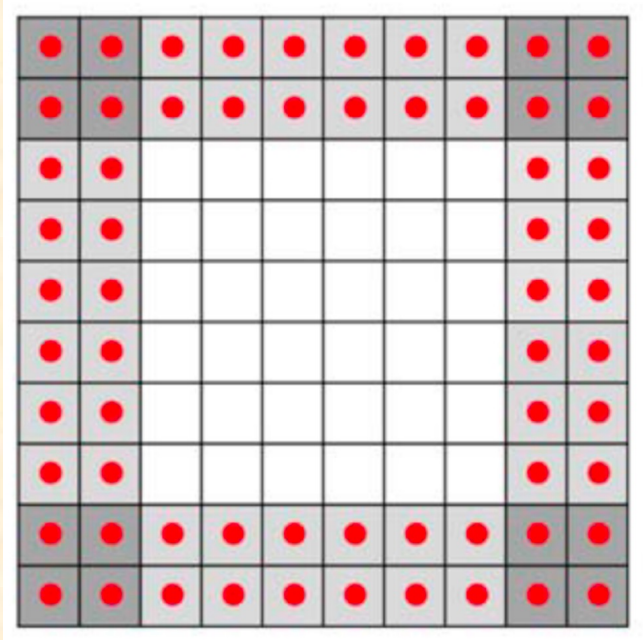
Parallelization

Distributed memory parallelization relies on domain decomposition using MPI. Grid is divideded into equally sized blocks, with each block assigned to different processors.



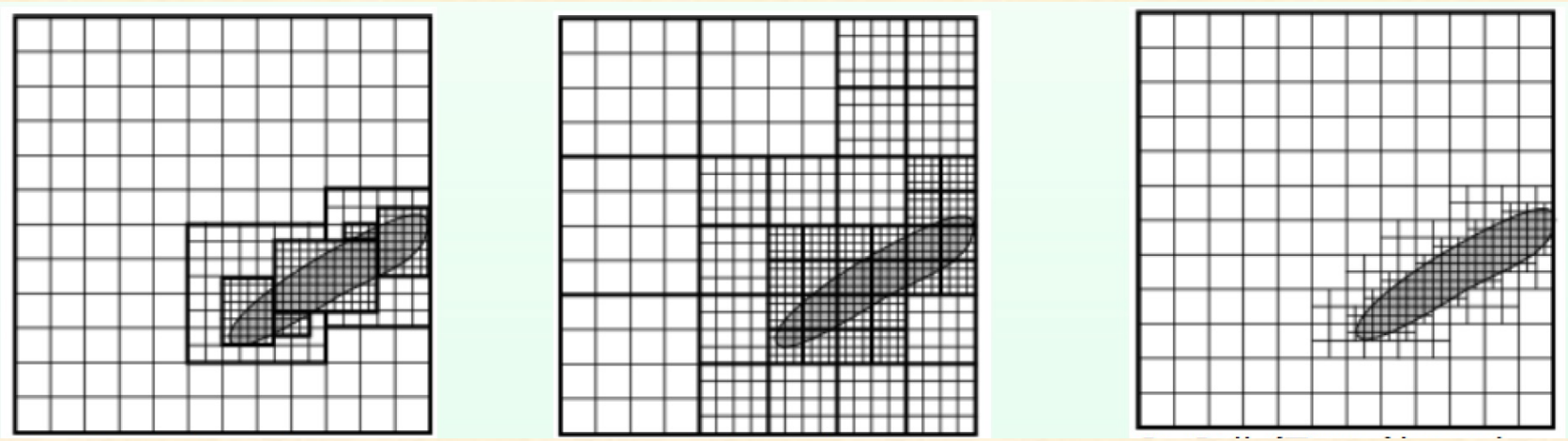
Blocks are organized into a quad(oct)-tree in 2D(3D) using Z-ordering to improve locality.

Ghost cells are communicated between blocks using MPI.



Adaptive Mesh Refinement

Blocks can be refined to improve resolution where it is needed.



Patch based

AMRex
Chombo
PLUTO

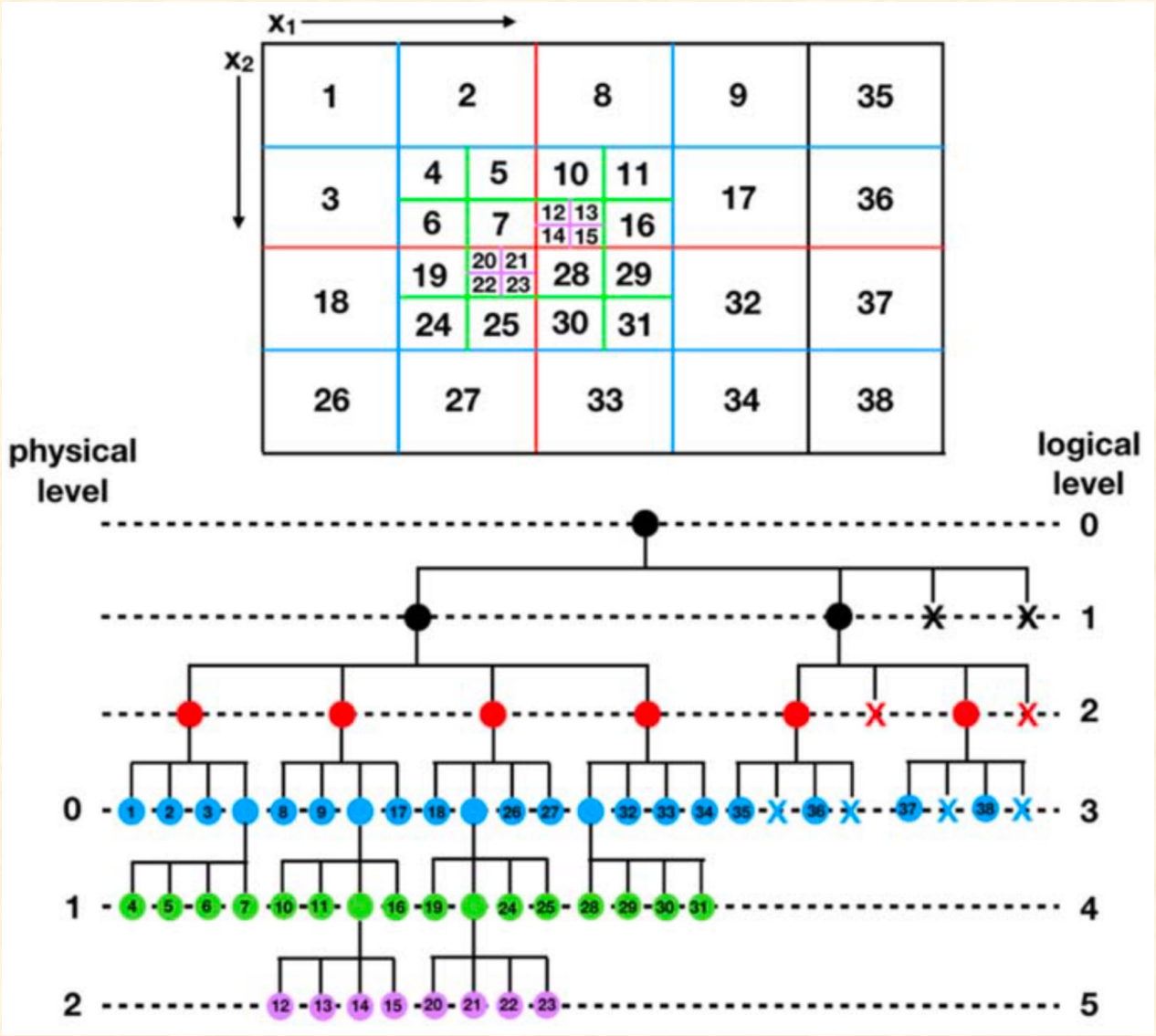
Block based

Athena++/AthenaK
FLASH
ENZO-E

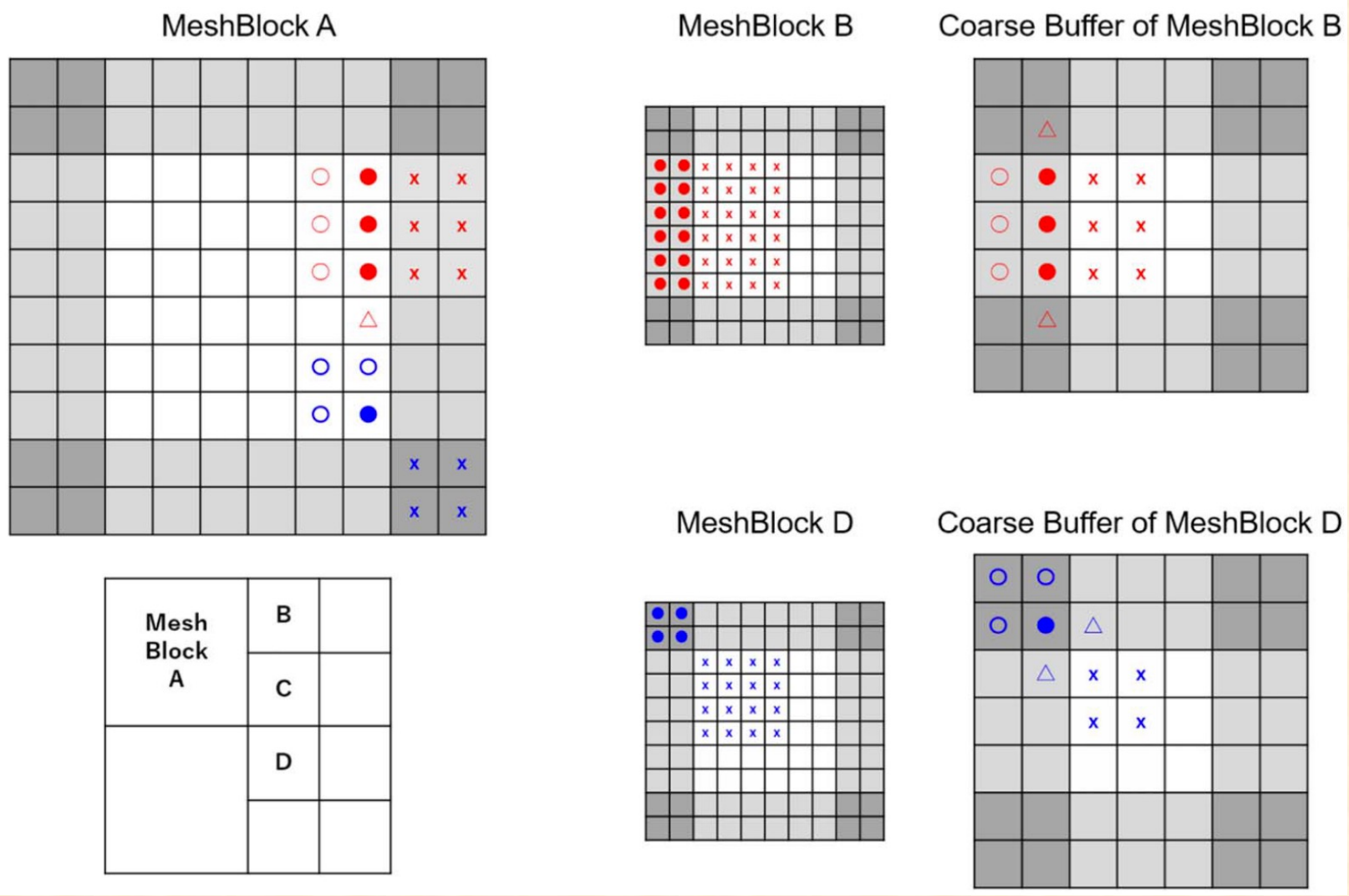
Cell based

RAMSES
ART

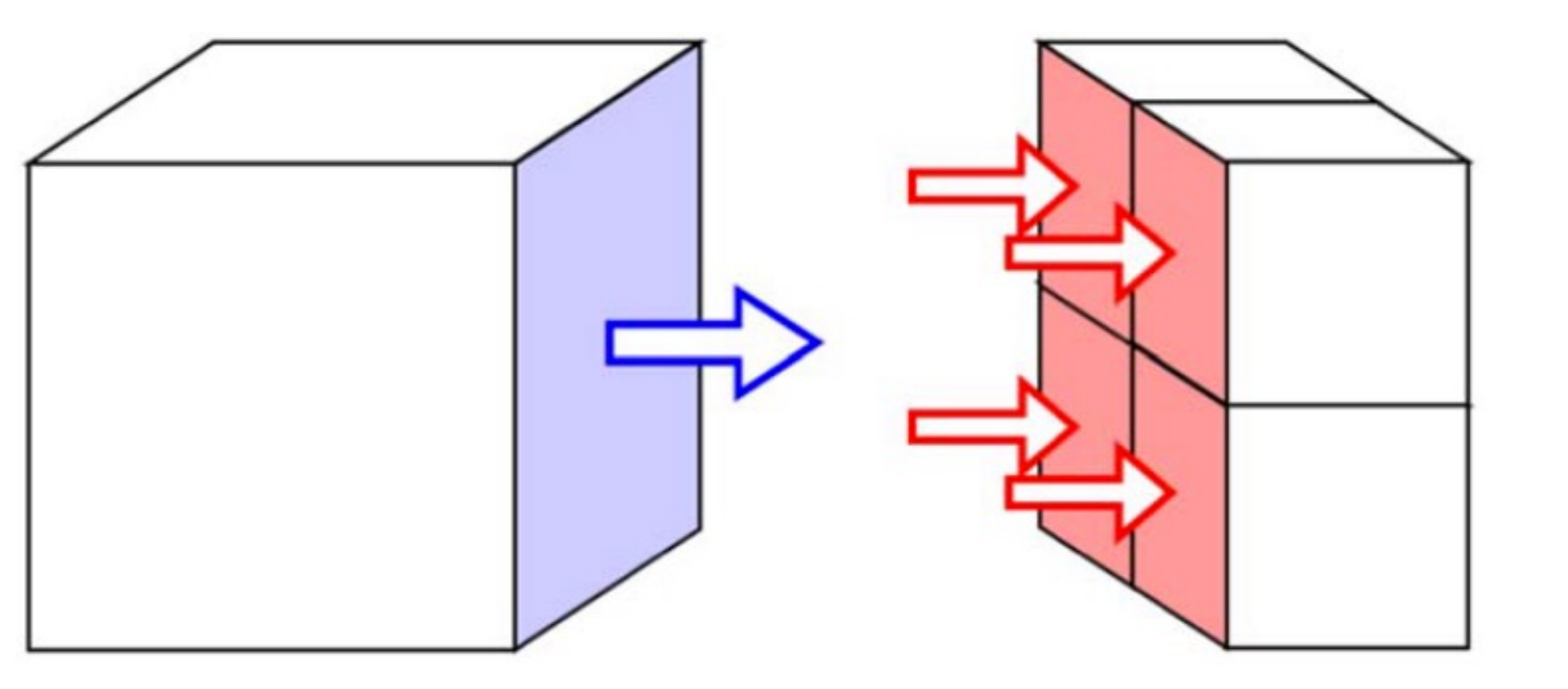
Quad(oct)-tree in 2D(3D) is especially useful for organizing blocks with AMR.



Communicating ghost cells for cell-centered variables with AMR becomes much more complicated.

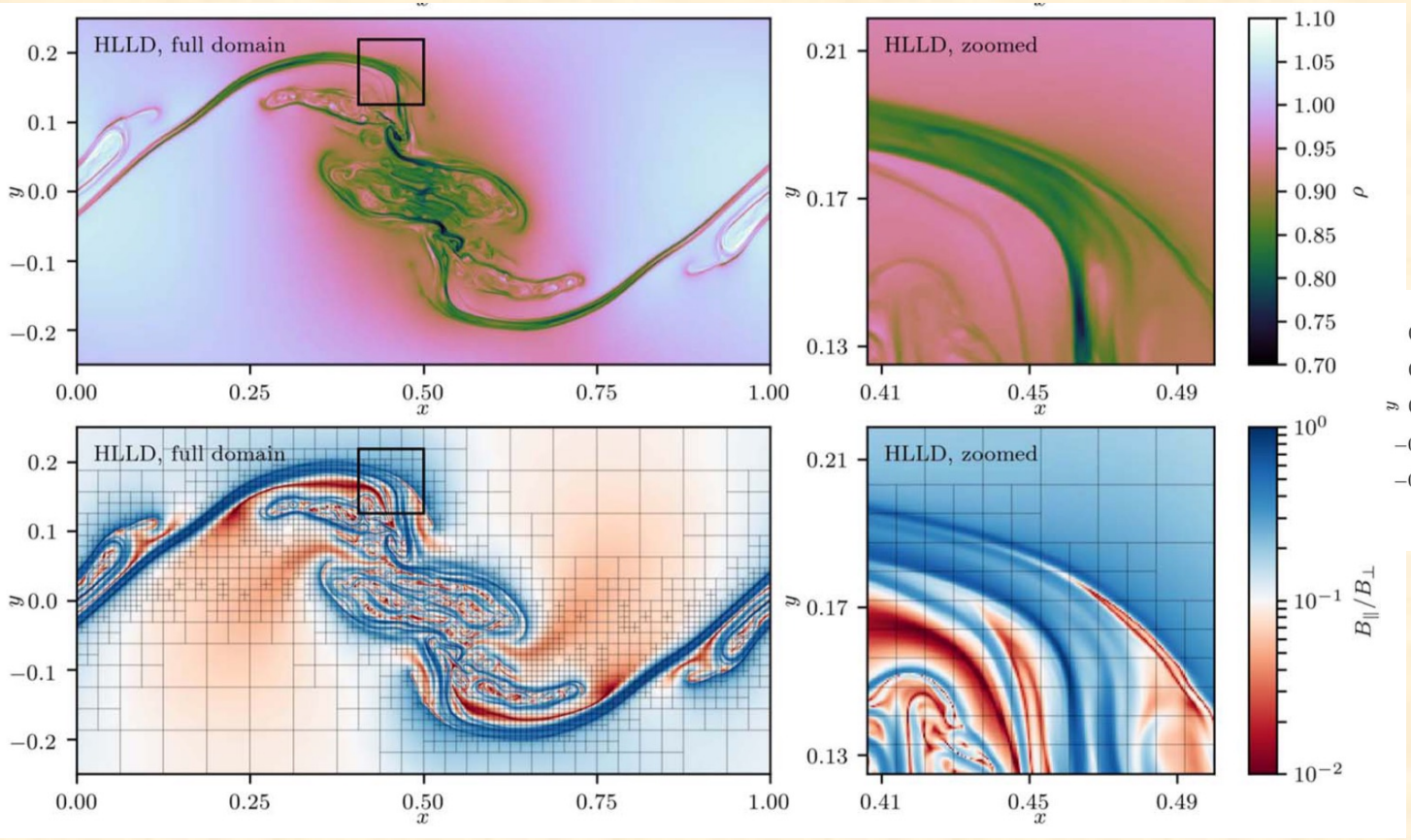


In addition, flux correction at fine/coarse boundaries is needed with AMR to conserve volume integrals of **U**. For example (cell-centered variables):

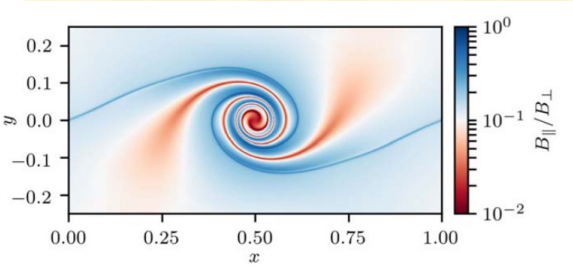


Additional corrections needed for electric fields (fluxes of face-centered variables).

Should we expect solutions on an AMR mesh to be identical to those on a uniform grid?



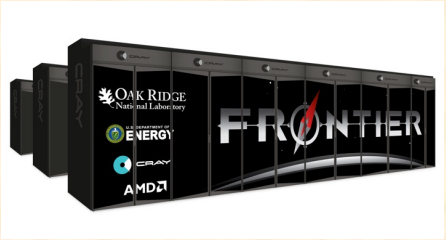
MHD KHI with AMR grid solution



Uniform grid solution

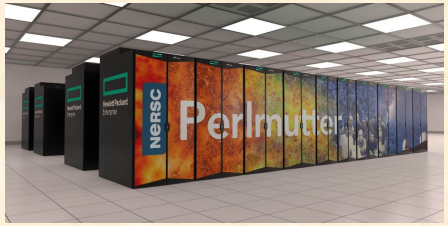
Programming for Modern HPC Systems

Largest open-science computers adopt a wide variety of designs



Frontier (USA)
600k AMD EPYC cores +
38k AMD Mi250X GPUs

Aurora (USA)
200k Intel Xeon cores +
55k Intel Ponte Vecchio GPUs



Perlmutter (USA)
200k AMD EPYC cores +
6k NVidia A100 GPUS

Each of these architectures requires a different programming model (C++, HIP, SYCL, CUDA).

Need a single programming model that runs on all architectures.

Solution: Kokkos (Trott et al. 2021)

- Open source code project at Sandia Nat. Lab. https://github.com/kokkos/
- Set of C++ templates and abstractions for data containers and parallel execution strategies.
- Depending on build options, code compiles into C++, CUDA, OpenMP, OneAPI, etc.
- This results in ***performance portability.*** In principle, a code built on top of Kokkos can run efficiently on any architecture.

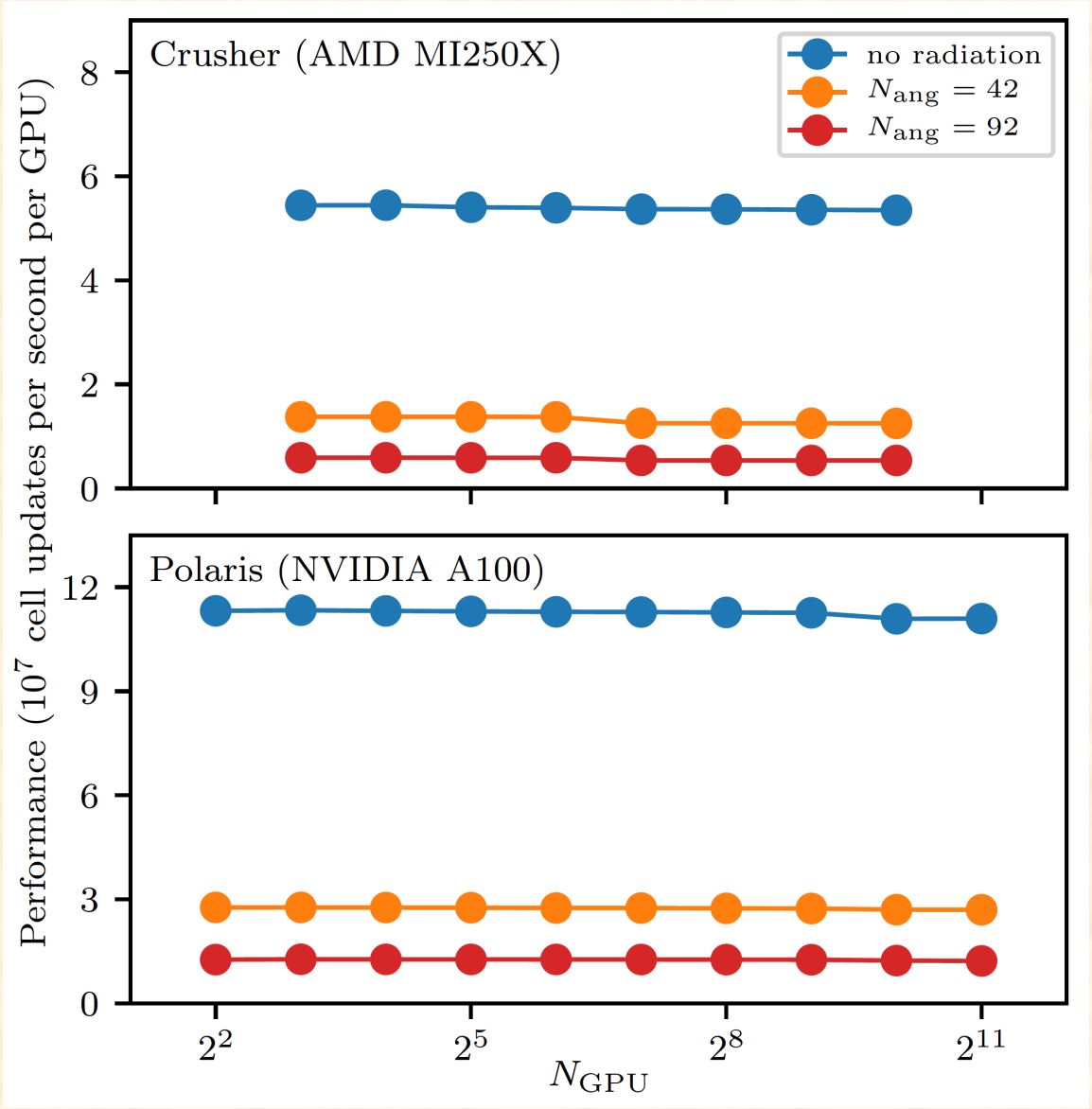
Performance portability

PLM, RK2, HLLE solver

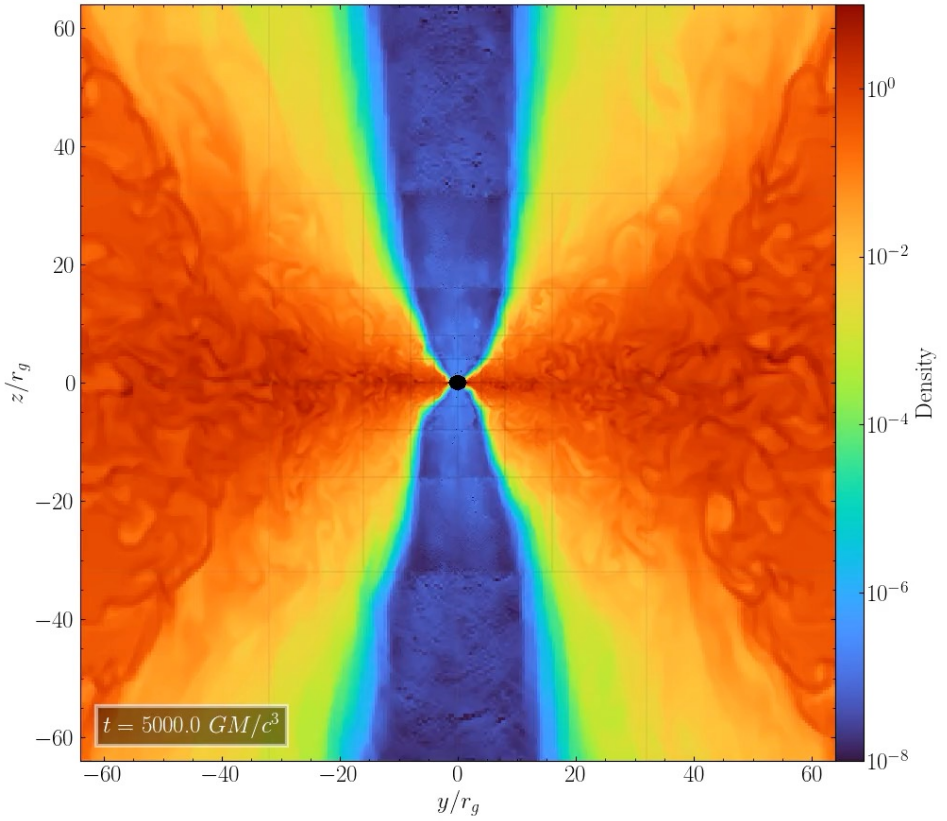
zone-cycles/sec ($\times 10^6$)	AthenaK			Athena++	
	CPU		GPU	CPU	
	single core※	one node(96 cores)※	A100	single core	one node(96 cores)
Hydro	4.89	184	555	4.94	186
MHD	2.00	80.8	260	2.24	90.5
SR Hydro	1.51	88.4	263	2.30	136
SR MHD	0.883	55.4	142	1.02	65.5
GR Hydro	0.728	44.1	220	1.57	95.3
GR MHD	0.350	22.3	98.7	0.550	35.0

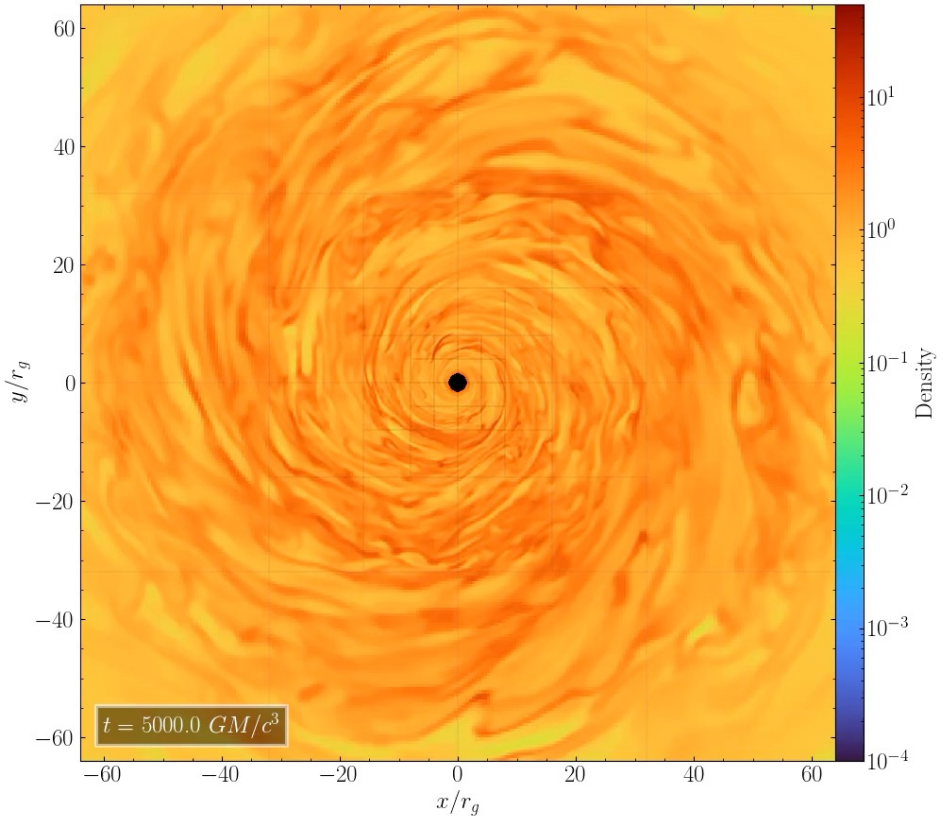
*Must guarantee inner loops are vectorized.

Excellent weak scaling on GPUs:



Example application: Radiation dominated BH accretion flow





Spin = 0.9, SANE initial conditions, 10 M_{Sun} black hole,
initial $P_{rad}/P_{gas} = 10^3$, scattering and absorption opacity
Accretion rate approximately Eddington rate.
Runs cost ~4K GPU hours. Much bigger/longer runs possible.

Summary

- There are many numerical algorithms for solving the compressible MHD equations based on different discretizations.
- Finite volume methods based on uniform meshes are popular since they are excellent for shock capturing.
- Modern codes must be able to run on today's heterogeneous computing systems, such as GPU accelerators.
- Numerical methods are a powerful tool for exploring nonlinear and time dependent MHD.