Adding/Modifying FLASH capabilities

(… you **can** teach an old dog new tricks!)

FLASH center for Computational Science, University of Chicago

Outline

- General tips
- Implement a new geometry: Cylindrical MHD
- New boundary conditions for our new geometry
- Test and validate with an appropriate problem!

General tips

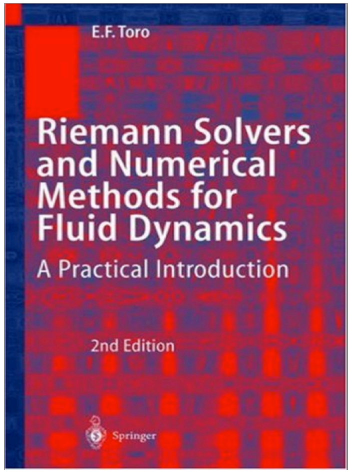
- ✓ Read the manual! (PDF, html, robodocs) explore flash.uchicago.edu
- ✓ Get to know the code's structure before you start implementing.
- ✓ Follow the general guidelines of existing implementations
- ✓ Get in touch with us! Mailing lists. Direct contacts are welcome!

FLASH User's Guide

Version 4.0-beta
February 2012 (last updated February 1, 2012)

FLASH Center for Computational Science
University of Chicago

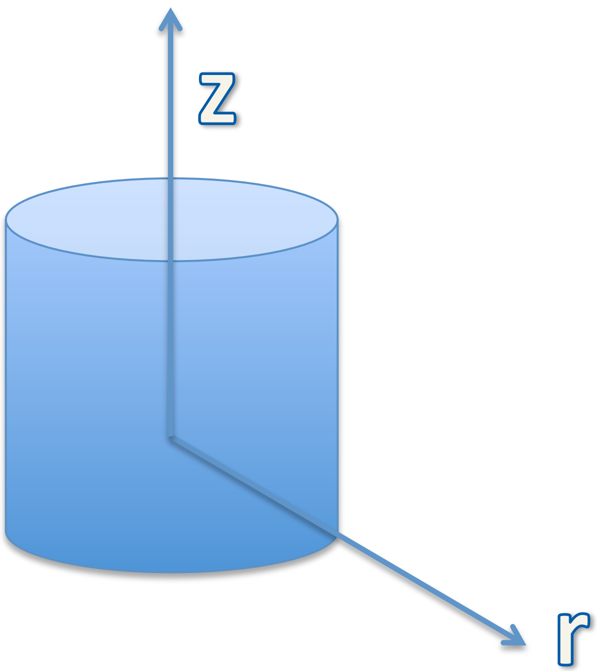
Know what you want to do



<u>General tips</u>

- ✓ A background in numerical algorithms is desirable but not strictly necessary (depends…).
- ✓ Search the literature for existing implementations, JCP, ApJs, J. Comput. Phys Com. etc
- ✓ See if what you need is already there in some from!

Example: RZ in unsplit MHD



Add a new geometry!

Literature examples

MHD equations

What needs to change&
how to go about and do it

!

Example: RZ in unsplit MHD

Guidelines: unsplit MHD in FLASH,
a modified CTU scheme

FLASH User's Guide

Version 4.0-beta
February 2012 (last updated February 1, 2012)

...ter for Computational Science
...niversity of Chicago

A Solution Accurate, Efficient and Stable Unsplit Staggered Mesh Scheme for Three Dimensional Magnetohydrodynamics

Dongwook Lee
The Flash Center for Computational Science, University of Chicago, 5747 S. Ellis, Chicago, IL 60637

Abstract

In this paper, we extend the unsplit staggered mesh scheme (USM) for 2D magnetohydrodynamics (MHD) [D. Lee, A. Deane, An Unsplit Staggered Mesh Scheme for Multidimensional Magnetohydrodynamics, J. Comput. Phys. 228 (2009) 952–975] to a full 3D MHD scheme. The 3D scheme uses the same set of fundamental algorithmic ideas that have been developed in the 2D USM scheme. The scheme is a finite-volume Godunov method consisting of (1) a constrained transport (CT) method for preserving the solenoidal magnetic field evolution on a staggered grid, and (2) an efficient and accurate single-step, directionally unsplit multidimensional data reconstruction-evolution algorithm,

Example: RZ in unsplit MHD

Guidelines: Mignone et al. 2007 and Skinner & Ostriker 2010, along with the implementation found in the PLUTO code.

PLUTO v. 3.1.1 (April 2011)

User's Guide

(http://plutocode.ph.unito.it)

Developer: A. Mignone[1,2] (mignone@ph.unito.it, mignone@oato.inaf.it)

Contributors: P. Tzeferacos[1] (Viscosity, MHD, STS, Finite-Difference

(petros.tzeferacos@ph.unito.it)

G. Bodo[2] (Parallelization)
T. Matsakos[3] (Resistivity, Thermal Con
O. Tesileanu[4] (Cooling)
C. Zanni[2] (AMR)

The Astrophysical Journal Supplement Series, 170:228–242, 2007 May

PLUTO: A NUMERICAL CODE FOR COMPUTATIONAL ASTROPHYSICS

A. Mignone,[1,2] G. Bodo,[2] S. Massaglia,[1] T. Matsakos,[1] O. Tesileanu,[1] C. Zanni,[3] and A. Ferrari[1]
Received 2006 November 5; accepted 2007 January 28

ABSTRACT

We present a new numerical code, PLUTO, for the solution of hypersonic flows in 1, 2, and 3 spatial dimensions and different systems of coordinates. The code provides a multiphysics, multialgorithm modular environment particularly oriented toward the treatment of astrophysical flows in presence of discontinuities. Different hydrodynamic modules and algorithms may be independently selected to properly describe Newtonian, relativistic, MHD, or relativistic MHD

The Astrophysical Journal Supplement Series, 188:290–311, 2010 May
doi:10.1088/0067-0049/188/1/290

THE *ATHENA* ASTROPHYSICAL MAGNETOHYDRODYNAMICS CODE IN CYLINDRICAL GEOMETRY

M. Aaron Skinner and Eve C. Ostriker
Astronomy Department, University of Maryland, College Park, MD 20742, USA; askinner@astro.umd.edu, ostriker@astro.umd.edu
Received 2009 December 22; accepted 2010 April 7; published 2010 May 4

ABSTRACT

A method for implementing cylindrical coordinates in the *Athena* magnetohydrodynamics (MHD) code is described.

Example: RZ in unsplit MHD

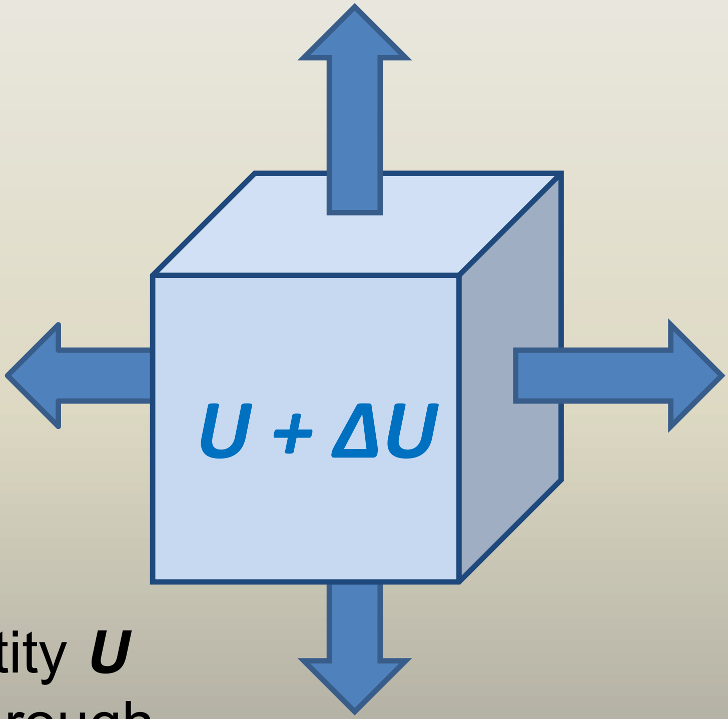
The ideal MHD system of equations can be written in compact form

$$\frac{\partial \boldsymbol{U}}{\partial t} = -\nabla \cdot \mathbf{F}\,(\boldsymbol{U}) + \boldsymbol{S}(\boldsymbol{U})$$

$$\boldsymbol{U} = \begin{pmatrix} \rho \\ \boldsymbol{m} \\ \boldsymbol{B} \\ E \end{pmatrix}, \quad \mathbf{F}\,(\boldsymbol{U}) = \begin{bmatrix} \rho\boldsymbol{v} \\ \boldsymbol{m}\boldsymbol{v} - \boldsymbol{B}\boldsymbol{B} + p_t\mathbf{I} \\ \boldsymbol{v}\boldsymbol{B} - \boldsymbol{B}\boldsymbol{v} \\ (E + p_t)\boldsymbol{v} - (\boldsymbol{v}\cdot\boldsymbol{B})\boldsymbol{B} \end{bmatrix}^T$$

where the source terms **S**(**U**) can account for geometrical corrections.

$$\frac{\partial \vec{U}}{\partial t} + \nabla \cdot \vec{F}(\vec{U}) = 0$$



It's a balance equation:

relates the *rate of change* of a quantity ***U*** with the *flux* of that quantity, ***F*(*U*)**, through the region boundary.

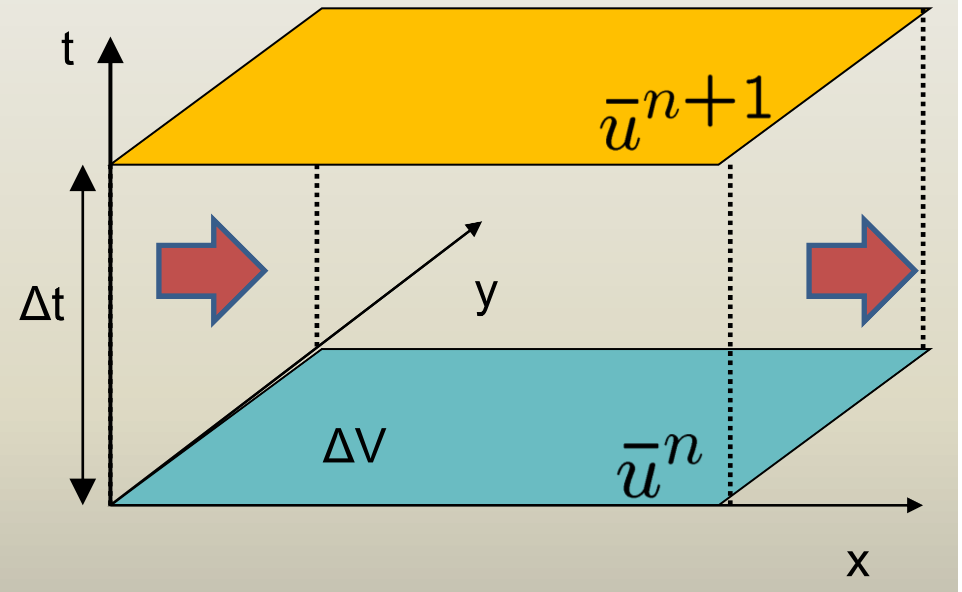
$$\frac{\partial \vec{U}}{\partial t} + \nabla \cdot \vec{F}(\vec{U}) = 0$$

Integral form

$$\int dt \int dV \left\{ \frac{\partial \boldsymbol{U}}{\partial t} + \nabla \cdot \boldsymbol{F}(\boldsymbol{U}) \right\} = 0$$

Finite volume formulation

$$\frac{\bar{u}^{n+1} - \bar{u}^{n}}{\Delta t} + \frac{1}{\Delta V} \oint \tilde{\boldsymbol{F}} \cdot d\boldsymbol{S} = 0$$



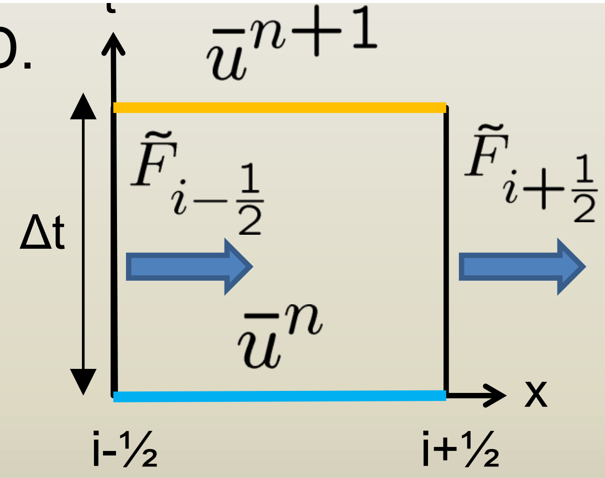
evolve volume averages

$$\bar{u}(t) = \frac{1}{\Delta V} \int u(x,t) dV$$

$$\tilde{F}(x) = \frac{1}{\Delta t} \int F(u(x,t)) dt$$

For the sake of exposition let's **discretize** in 1D.

$$\bar{u}_i^{n+1} = \bar{u}_i^n - \frac{\Delta t}{\Delta x}\left(\tilde{F}_{i+\frac{1}{2}} - \tilde{F}_{i-\frac{1}{2}}\right)$$



An iterative method!
The average quantities will now be

$$\bar{u}_i(t) = \frac{1}{\Delta x_i}\int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}} u(x,t)dx$$

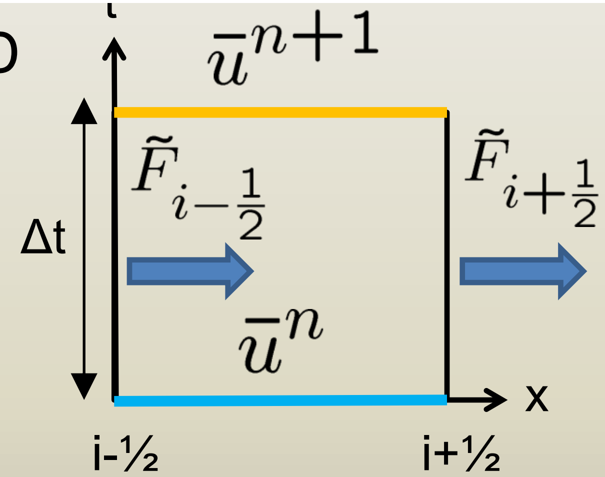
$$\tilde{F}_{i+\frac{1}{2}} = \frac{1}{\Delta t}\int_{t^n}^{t^{n+1}} F(u(x_{i+\frac{1}{2}},t))dt$$

All of this is exact!
No approximation yet.

Brief interlude

For the sake of exposition let's **discretize** in 1D

$$\bar{u}_i^{n+1} = \bar{u}_i^n - \frac{\Delta t}{\Delta x}\left(\tilde{F}_{i+\frac{1}{2}} - \tilde{F}_{i-\frac{1}{2}}\right)$$



An iterative method!
The average quantities will now be

$$\bar{u}_i(t) = \frac{1}{\Delta x_i}\int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}} u(x,t)\,dx$$

$$\tilde{F}_{i+\frac{1}{2}} = \frac{1}{\Delta t}\int_{t^n}^{t^{n+1}} F(u(x_{i+\frac{1}{2}},t))\,dt$$

All of this is exact!
No approximation yet.

Numerics in here
$F(u(x_{i\pm 1/2}, t^{n+1}))$!!!

Example: RZ in unsplit MHD

The ideal MHD system of equations can be written in compact form

$$\frac{\partial \boldsymbol{U}}{\partial t} = -\nabla \cdot \mathbf{F}\,(\boldsymbol{U}) + \boldsymbol{S}(\boldsymbol{U})$$

$$\boldsymbol{U} = \begin{pmatrix} \rho \\ \boldsymbol{m} \\ \boldsymbol{B} \\ E \end{pmatrix}, \quad \mathbf{F}\,(\boldsymbol{U}) = \begin{bmatrix} \rho \boldsymbol{v} \\ \boldsymbol{m}\boldsymbol{v} - \boldsymbol{B}\boldsymbol{B} + p_t \mathbf{I} \\ \boldsymbol{v}\boldsymbol{B} - \boldsymbol{B}\boldsymbol{v} \\ (E + p_t)\boldsymbol{v} - (\boldsymbol{v} \cdot \boldsymbol{B})\boldsymbol{B} \end{bmatrix}^T$$

where the source terms **S**(**U**) can account for geometrical corrections.

Example: RZ in unsplit MHD

Simple scalars such as density and energy will give

$$\frac{\partial q}{\partial t} + \nabla \cdot \boldsymbol{F} = 0 \Longrightarrow \frac{\partial q}{\partial t} + \frac{1}{r}\frac{\partial (rF_r)}{\partial r} + \frac{1}{r}\frac{\partial F_\phi}{\partial \phi} + \frac{\partial F_z}{\partial z} = 0$$

whereas momentum will write

$$\frac{\partial \boldsymbol{m}}{\partial t} + \nabla \cdot \boldsymbol{M} = 0 \Longrightarrow$$

$$\left\{ \begin{array}{l} \dfrac{\partial m_r}{\partial t} + \dfrac{1}{r}\dfrac{\partial (rM_{rr})}{\partial r} + \dfrac{1}{r}\dfrac{\partial M_{\phi r}}{\partial \phi} + \dfrac{\partial M_{zr}}{\partial z} = \dfrac{M_{\phi\phi}}{r} \\ \dfrac{\partial m_\phi}{\partial t} + \dfrac{1}{r}\dfrac{\partial (rM_{r\phi})}{\partial r} + \dfrac{1}{r}\dfrac{\partial M_{\phi\phi}}{\partial \phi} + \dfrac{\partial M_{z\phi}}{\partial z} = -\dfrac{M_{\phi r}}{r} \\ \dfrac{\partial m_z}{\partial t} + \dfrac{1}{r}\dfrac{\partial (rM_{rz})}{\partial r} + \dfrac{1}{r}\dfrac{\partial M_{\phi z}}{\partial \phi} + \dfrac{\partial M_{zz}}{\partial z} = 0 \end{array} \right.$$

Example: RZ in unsplit MHD

Similarly the induction equation will be

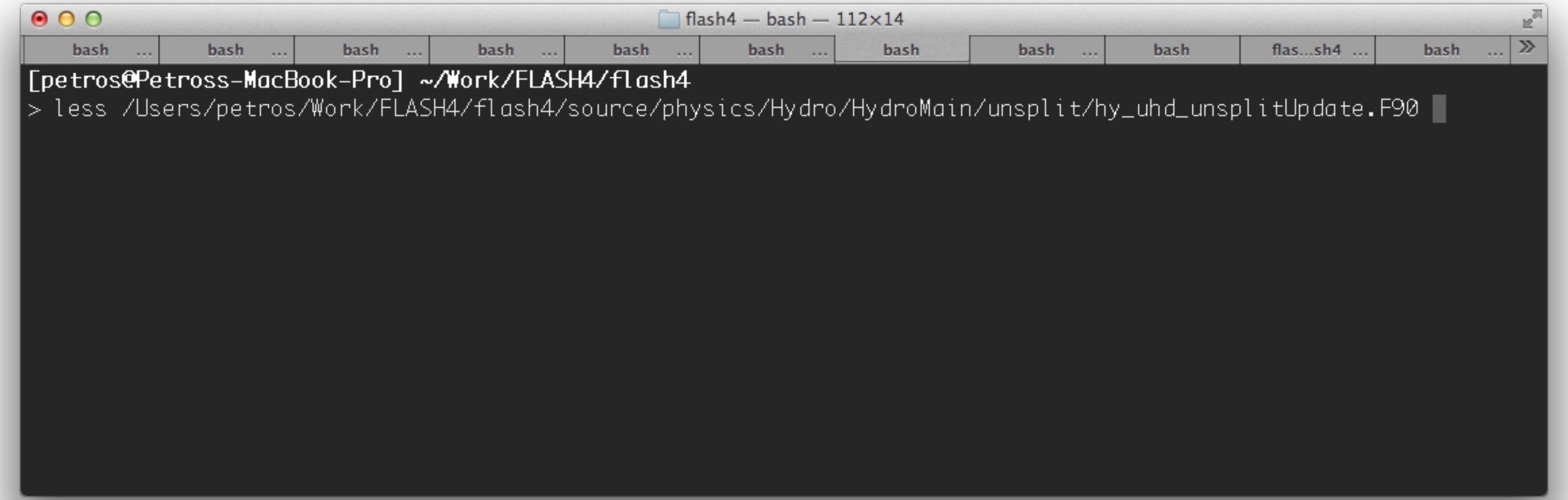
$$\frac{\partial \boldsymbol{B}}{\partial t} + \nabla \cdot \boldsymbol{\Omega} = 0 \implies \begin{cases} \dfrac{\partial B_r}{\partial t} + \dfrac{1}{r}\dfrac{\partial \Omega_{\phi r}}{\partial \phi} + \dfrac{\partial \Omega_{zr}}{\partial z} = 0 \\ \dfrac{\partial B_\phi}{\partial t} + \dfrac{1}{r}\dfrac{\partial (r\Omega_{r\phi})}{\partial r} + \dfrac{\partial \Omega_{z\phi}}{\partial z} = -\dfrac{\Omega_{\phi r}}{r} \\ \dfrac{\partial B_z}{\partial t} + \dfrac{1}{r}\dfrac{\partial (r\Omega_{rz})}{\partial r} + \dfrac{1}{r}\dfrac{\partial \Omega_{\phi z}}{\partial \phi} = 0 \end{cases}$$

What we end up with is three new source terms

as well as the need of redefining volumes and areas.

x^1	x	r
x^2	y	ϕ
x^3	z	z
$\Delta\mathcal{V}^1$	Δx	$\Delta^2 r$
$\Delta\mathcal{V}^2$	Δy	$r\Delta\phi$
$\Delta\mathcal{V}^3$	Δz	Δz
A^1_+	1	r_+
A^2_+	1	1
A^3_+	1	1

Ok, let's plug this in the variable update…

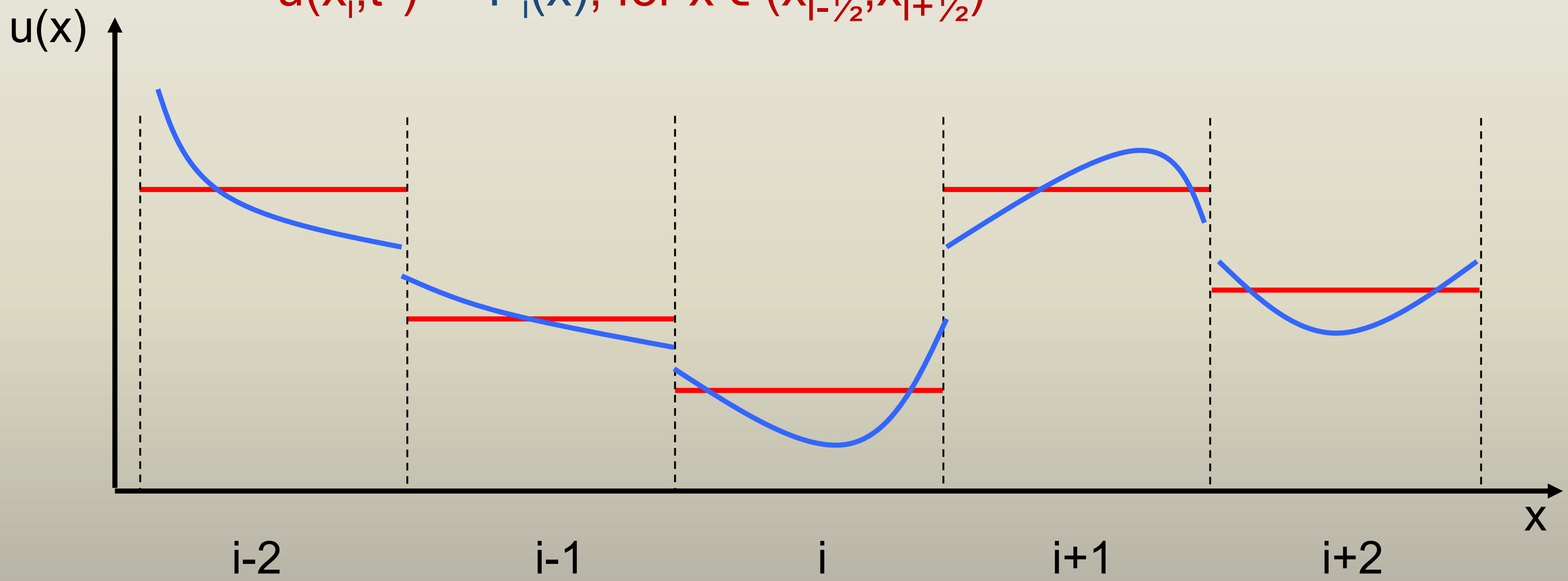


At any given time we do **not** have $u(x,t^n)$

We have discrete values at each cell, $u(x_i,t^n)$

piecewise polynomial reconstruction

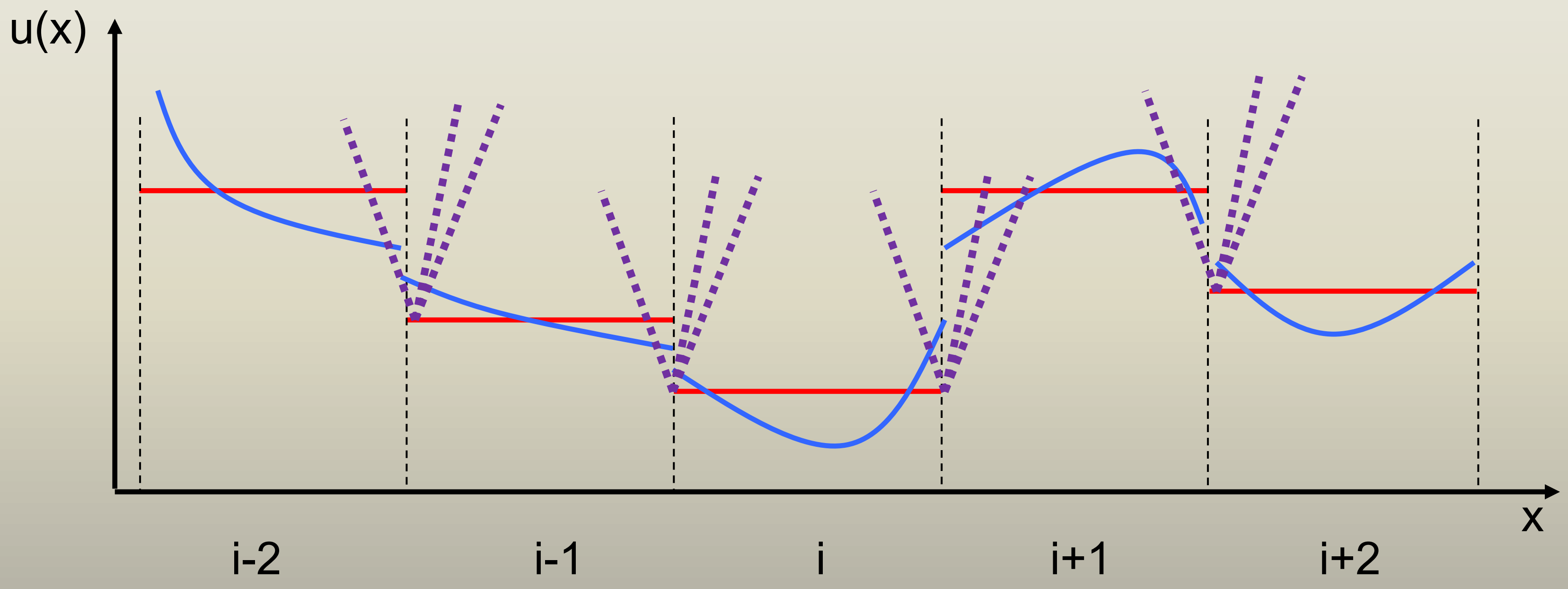
$u(x_i,t^n) = P_i(x)$, for $x \in (x_{i-½},x_{i+½})$



At each interface i+½ we have a $uL = P_i(x_{i+½})$ and a $uR = P_{i+1}(x_{i+½})$

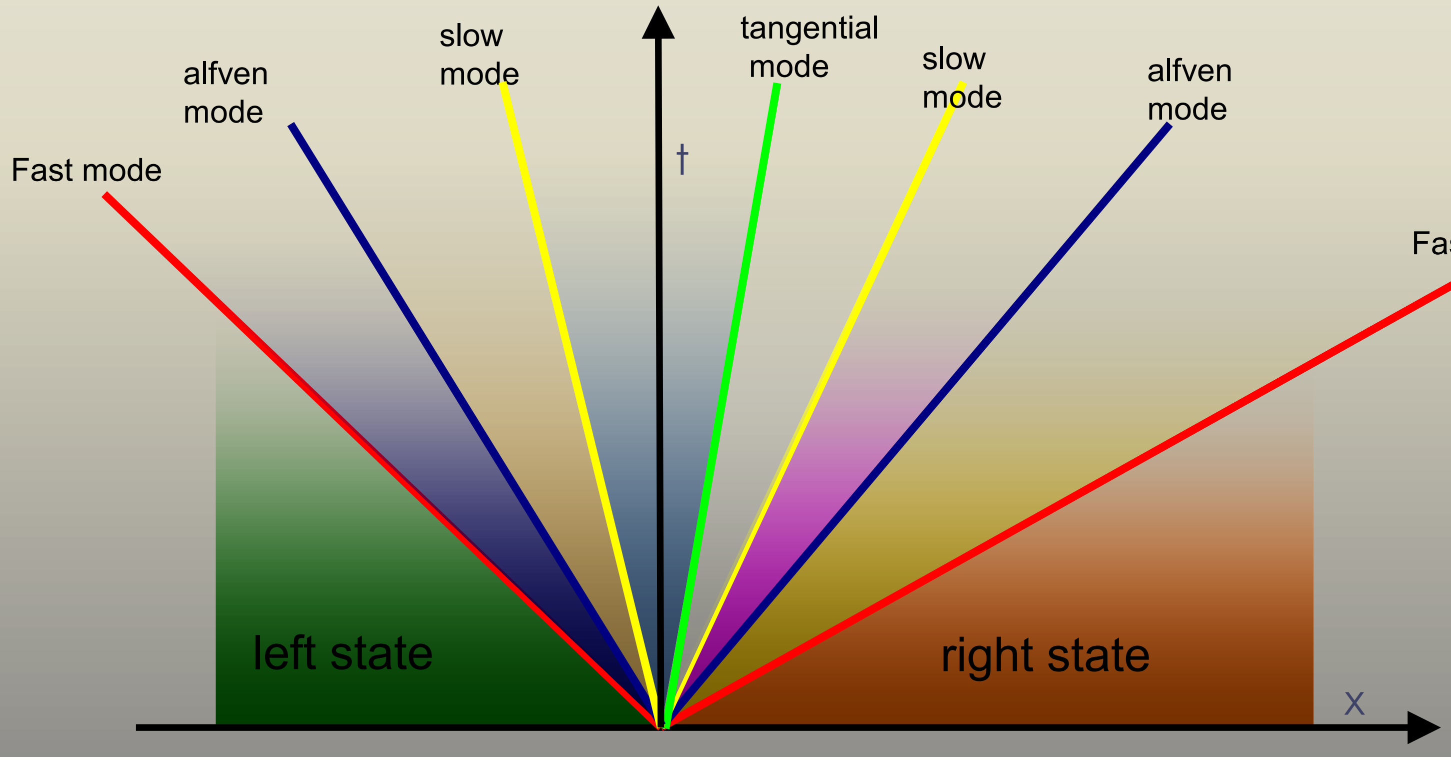
Brief interlude

At each interface i+½ we solve a Riemann problem and obtain $\tilde{F}_{i+\frac{1}{2}}$



Brief interlude

The flux computation $F_{i+\frac{1}{2}}$ requires the solution of a Riemann problem at the cell interface. It involves the temporal evolution of a discontinuity.



Having all the needed terms we can now advance our solution in time and form new averages.

$$\bar{u}_i^{n+1} = \bar{u}_i^n - \frac{\Delta t}{\Delta x}\left(\tilde{F}_{i+\frac{1}{2}} - \tilde{F}_{i-\frac{1}{2}}\right)$$

This technique is called **R-S-A**

- ✓ **R**econstruct (obtain uL, uR)
- ✓ **S**olve (the Riemann problem)
- ✓ **A**verage (evolve in time and form the new averages)

Example: RZ in unsplit MHD

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

Compact hyperbolic form, CTU formulation

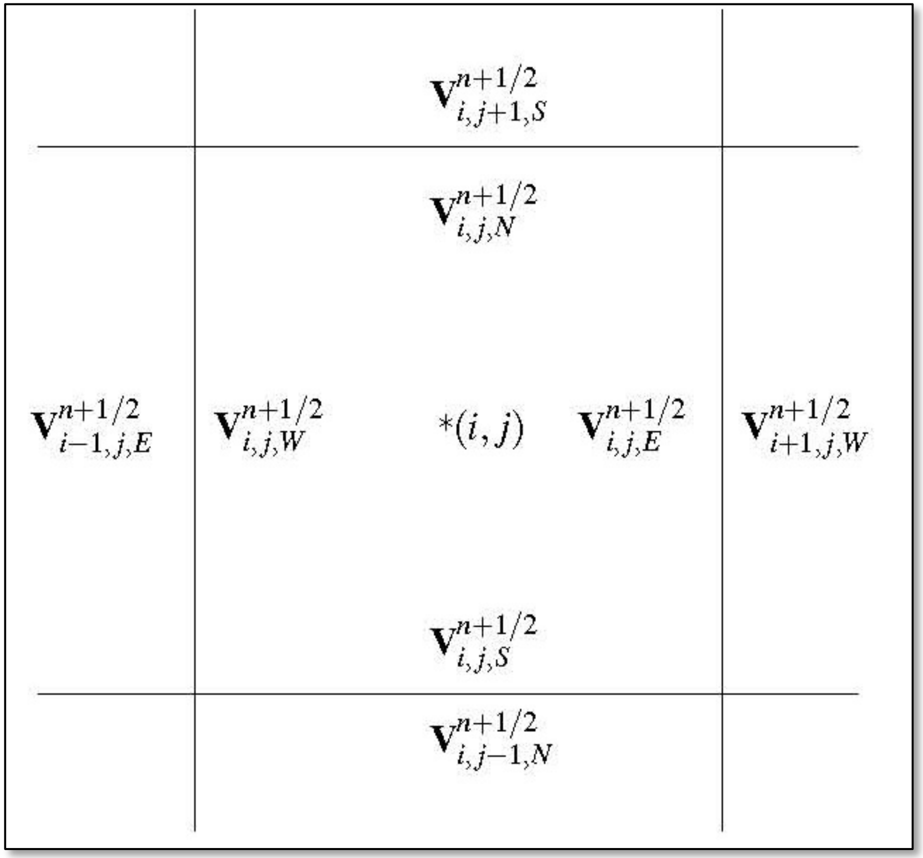
$$\tilde{\mathbf{F}}^{*,n+1/2}_{i-1/2,j,k} = \mathrm{RP}\left(\mathbf{V}^{n+1/2}_{i-1,j,k,E}, \mathbf{V}^{n+1/2}_{i,j,k,W}\right), \quad \tilde{\mathbf{F}}^{*,n+1/2}_{i+1/2,j,k} = \mathrm{RP}\left(\mathbf{V}^{n+1/2}_{i,j,k,E}, \mathbf{V}^{n+1/2}_{i+1,j,k,W}\right),$$

$$\tilde{\mathbf{G}}^{*,n+1/2}_{i,j-1/2,k} = \mathrm{RP}\left(\mathbf{V}^{n+1/2}_{i,j-1,k,N}, \mathbf{V}^{n+1/2}_{i,j,k,S}\right), \quad \tilde{\mathbf{G}}^{*,n+1/2}_{i,j+1/2,k} = \mathrm{RP}\left(\mathbf{V}^{n+1/2}_{i,j,k,N}, \mathbf{V}^{n+1/2}_{i,j+1,k,S}\right),$$

$$\tilde{\mathbf{H}}^{*,n+1/2}_{i,j,k-1/2} = \mathrm{RP}\left(\mathbf{V}^{n+1/2}_{i,j,k-1,T}, \mathbf{V}^{n+1/2}_{i,j,k,B}\right), \quad \tilde{\mathbf{H}}^{*,n+1/2}_{i,j,k+1/2} = \mathrm{RP}\left(\mathbf{V}^{n+1/2}_{i,j,k,T}, \mathbf{V}^{n+1/2}_{i,j,k+1,B}\right).$$

Example: RZ in unsplit MHD

Where the notation for the discretization is given by



Example: RZ in unsplit MHD

Compact, linearized primitive form:

$$\boxed{\mathbf{V} = (\rho, u, v, w, B_x, B_y, B_z, p)^T}$$

$$\frac{\partial \mathbf{V}}{\partial t} + \mathbf{A}_x \frac{\partial \mathbf{V}}{\partial x} + \mathbf{A}_y \frac{\partial \mathbf{V}}{\partial y} + \mathbf{A}_z \frac{\partial \mathbf{V}}{\partial z} = 0.$$

$$\mathbf{A}_y = \begin{pmatrix} v & 0 & \rho & 0 & 0 & 0 & 0 & 0 \\ 0 & v & 0 & 0 & -\frac{B_y}{\rho} & -\frac{B_x}{\rho} & 0 & 0 \\ 0 & 0 & v & 0 & \frac{B_x}{\rho} & -\frac{B_y}{\rho} & \frac{B_z}{\rho} & \frac{1}{\rho} \\ 0 & 0 & 0 & v & 0 & -\frac{B_z}{\rho} & -\frac{B_y}{\rho} & 0 \\ 0 & -B_y & B_x & 0 & v & -u & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & B_z & -B_y & 0 & -w & v & 0 \\ 0 & 0 & \gamma p & 0 & 0 & -k\mathbf{u}\cdot\mathbf{B} & 0 & v \end{pmatrix}$$

$$\mathbf{A}_x = \begin{pmatrix} u & \rho & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & u & 0 & 0 & -\frac{B_x}{\rho} & \frac{B_y}{\rho} & \frac{B_z}{\rho} & \frac{1}{\rho} \\ 0 & 0 & u & 0 & -\frac{B_y}{\rho} & -\frac{B_x}{\rho} & 0 & 0 \\ 0 & 0 & 0 & u & -\frac{B_z}{\rho} & 0 & -\frac{B_x}{\rho} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & B_y & -B_x & 0 & -v & u & 0 & 0 \\ 0 & B_z & 0 & -B_x & -w & 0 & u & 0 \\ 0 & \gamma p & 0 & 0 & -k\mathbf{u}\cdot\mathbf{B} & 0 & 0 & u \end{pmatrix}, \quad \mathbf{A}_z = \begin{pmatrix} w & 0 & 0 & \rho & 0 & 0 & 0 & 0 \\ 0 & w & 0 & 0 & -\frac{B_z}{\rho} & 0 & -\frac{B_x}{\rho} & 0 \\ 0 & 0 & w & 0 & 0 & -\frac{B_z}{\rho} & -\frac{B_y}{\rho} & 0 \\ 0 & 0 & 0 & w & \frac{B_x}{\rho} & \frac{B_y}{\rho} & -\frac{B_z}{\rho} & \frac{1}{\rho} \\ 0 & -B_z & 0 & B_x & w & 0 & -u & 0 \\ 0 & 0 & -B_z & B_y & 0 & w & -v & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \gamma p & 0 & 0 & -k\mathbf{u}\cdot\mathbf{B} & w \end{pmatrix}$$

Example: RZ in unsplit MHD

Characteristic tracing on both **normal predictor** and **transverse corrector**

$$\mathbf{V}^{n+1/2}_{i,j,k,E,W} = \mathbf{V}^{n}_{i,j,k} + \frac{1}{2}[\pm\mathbf{I} - \frac{\Delta t}{\Delta x}\mathbf{A}_x(\mathbf{V}^{n}_{i,j,k})]\Delta^{n}_{x,i,j,k} - \frac{\Delta t}{2\Delta y}\mathbf{A}_y(\mathbf{V}^{n}_{i,j,k})\Delta^{n}_{y,i,j,k} - \frac{\Delta t}{2\Delta z}\mathbf{A}_z(\mathbf{V}^{n}_{i,j,k})\Delta^{n}_{z,i,j,k},$$

$$\mathbf{V}^{n+1/2}_{i,j,k,N,S} = \mathbf{V}^{n}_{i,j,k} - \frac{\Delta t}{2\Delta x}\mathbf{A}_x(\mathbf{V}^{n}_{i,j,k})\Delta^{n}_{x,i,j,k} + \frac{1}{2}[\pm\mathbf{I} - \frac{\Delta t}{\Delta y}\mathbf{A}_y(\mathbf{V}^{n}_{i,j,k})]\Delta^{n}_{y,i,j,k} - \frac{\Delta t}{2\Delta z}\mathbf{A}_z(\mathbf{V}^{n}_{i,j,k})\Delta^{n}_{z,i,j,k},$$

$$\mathbf{V}^{n+1/2}_{i,j,k,T,B} = \mathbf{V}^{n}_{i,j,k} - \frac{\Delta t}{2\Delta x}\mathbf{A}_x(\mathbf{V}^{n}_{i,j,k})\Delta^{n}_{x,i,j,k} - \frac{\Delta t}{2\Delta y}\mathbf{A}_y(\mathbf{V}^{n}_{i,j,k})\Delta^{n}_{y,i,j,k} + \frac{1}{2}[\pm\mathbf{I} - \frac{\Delta t}{\Delta z}\mathbf{A}_z(\mathbf{V}^{n}_{i,j,k})]\Delta^{n}_{z,i,j,k},$$

Example: RZ in unsplit MHD

In cylindrical coordinates the primitive formulation does not come without source terms if we keep the same formalism by substituting x with R… For example, the continuity equation can be written as

$$\partial_t \rho + \rho \partial_R v_R + v_R \partial_R \rho = -\frac{1}{R} \rho v_R$$

whereas momenta will be

$$\partial_t v_R + v_R \partial_R v_R + \frac{1}{\rho} \partial_R P + \frac{1}{\rho} B_\phi \partial_R B_\phi + \frac{1}{\rho} B_z \partial_R B_z = \frac{1}{R} \left(v_\phi^2 - \frac{1}{\rho} B_\phi^2 \right)$$

$$\partial_t v_\phi + v_R \partial_R v_\phi - \frac{1}{\rho} B_R \partial_R B_\phi = -\frac{1}{R} (v_\phi v_R - \frac{1}{\rho} B_\phi B_R),$$

$$\partial_t v_z + v_R \partial_R v_z - \frac{1}{\rho} B_R \partial_R B_z = 0.$$

Example: RZ in unsplit MHD

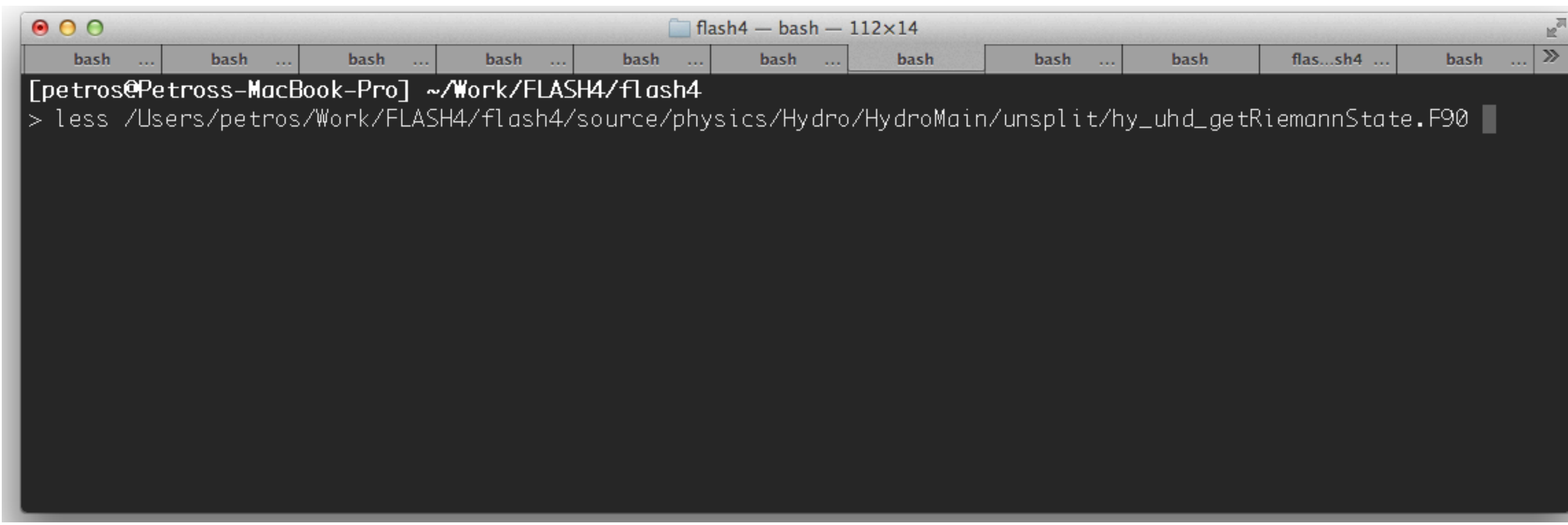
The source terms needed for the state calculation will be

$$\boldsymbol{s}_{\mathrm{geom}} \equiv \begin{bmatrix} -\dfrac{1}{R}\rho v_R \\ \dfrac{1}{R}\left(v_\phi^2 - \dfrac{1}{\rho}B_\phi^2\right) \\ -\dfrac{1}{R}\left(v_\phi v_R - \dfrac{1}{\rho}B_\phi B_R\right) \\ 0 \\ -\dfrac{1}{R}\gamma P v_R \\ -\dfrac{1}{R}v_\phi B_R \\ -\dfrac{1}{R}v_R B_z \end{bmatrix}$$

for the primitive variable vector:

$$\begin{bmatrix} \rho \\ v_R \\ v_\phi \\ v_z \\ P \\ B_\phi \\ B_z \end{bmatrix}$$

Ok, let's plug this in the state calculation…



Example: RZ in unsplit MHD

- There are more considerations and details to take care of but what we saw was mainly the core

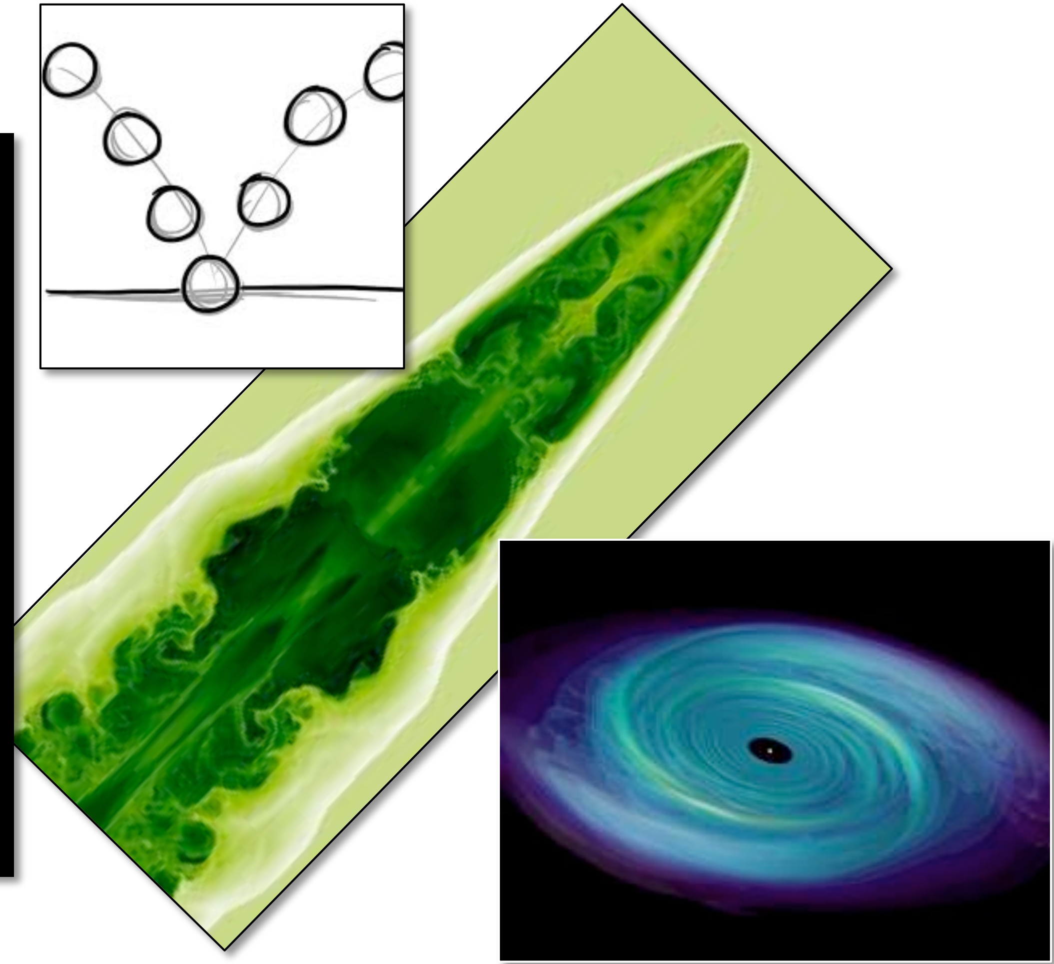
- Grep is your friend! Use it to find variables and functions you may need from existing implementations.

- OK, our new geometry requires new boundary conditions!

Example: Axisym., Eqtsym. BC

The new coordinate system benefits from symmetries that can be exploited if proper boundary conditions are present.

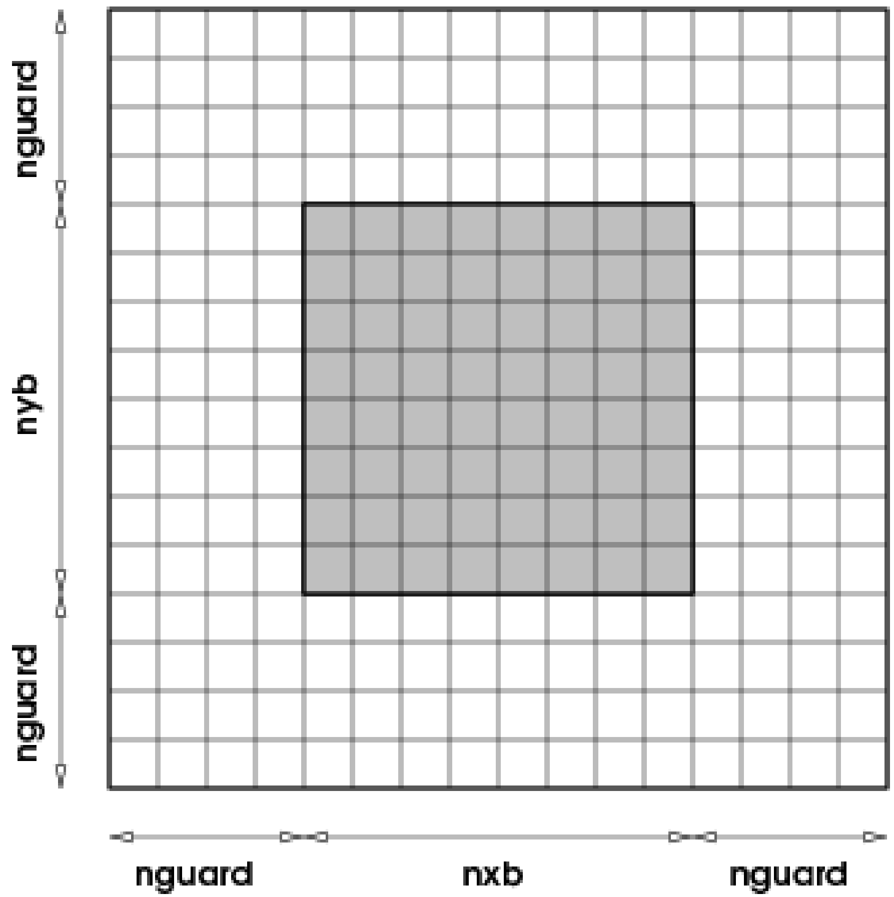
```
!!      Reflective
!!  Vn -> -Vn,  Bn -> -Bn
!!  Vp ->  Vp,  Bp ->  Bp
!!  Vt ->  Vt,  Bt ->  Bt
!!
!!      Axisymmetric
!!  Vn -> -Vn,  Bn -> -Bn
!!  Vp ->  Vp,  Bp ->  Bp
!!  Vt -> -Vt,  Bt -> -Bt
!!
!!      Eqtsymmetric
!!  Vn -> -Vn,  Bn ->  Bn
!!  Vp ->  Vp,  Bp -> -Bp
!!  Vt ->  Vt,  Bt -> -Bt
```



Example: Axisym., Eqtsym. BC

The new coordinate system benefits from symmetries that can be exploited if proper boundary conditions are present.

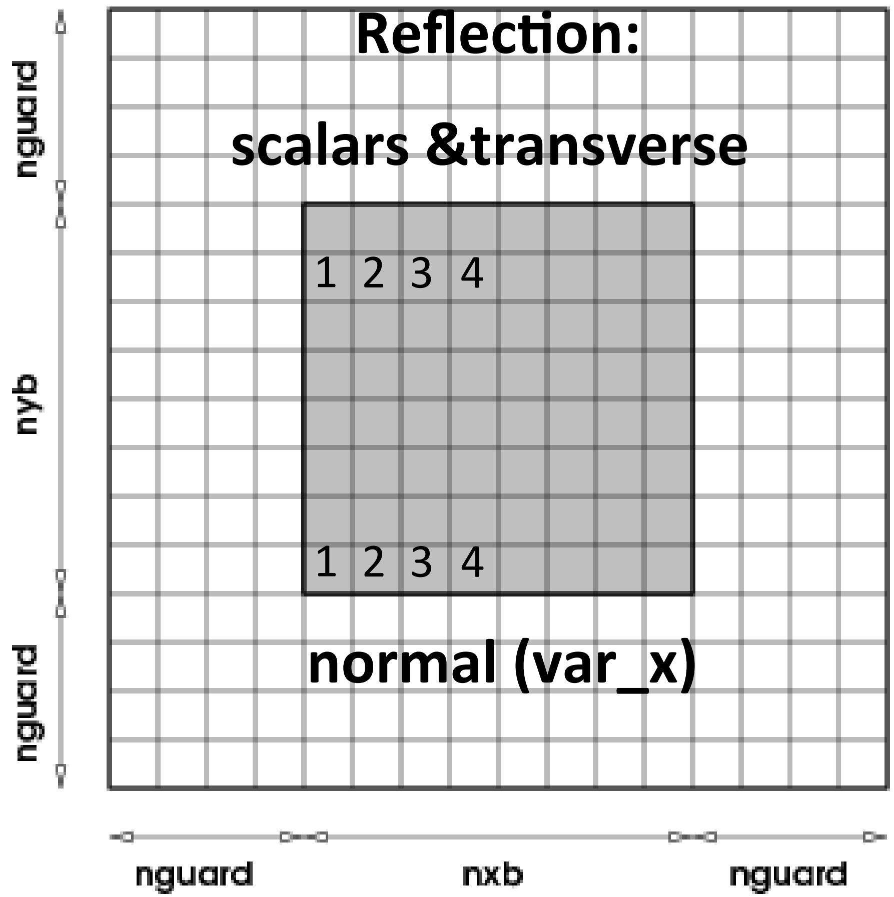
```
!!        Reflective
!!   Vn -> -Vn,   Bn -> -Bn
!!   Vp ->  Vp,   Bp ->  Bp
!!   Vt ->  Vt,   Bt ->  Bt
!!
!!        Axisymmetric
!!   Vn -> -Vn,   Bn -> -Bn
!!   Vp ->  Vp,   Bp ->  Bp
!!   Vt -> -Vt,   Bt -> -Bt
!!
!!       Eqtsymmetric
!!   Vn -> -Vn,   Bn ->  Bn
!!   Vp ->  Vp,   Bp -> -Bp
!!   Vt ->  Vt,   Bt -> -Bt
```



Example: Axisym., Eqtsym. BC

The new coordinate system benefits from symmetries that can be exploited if proper boundary conditions are present.

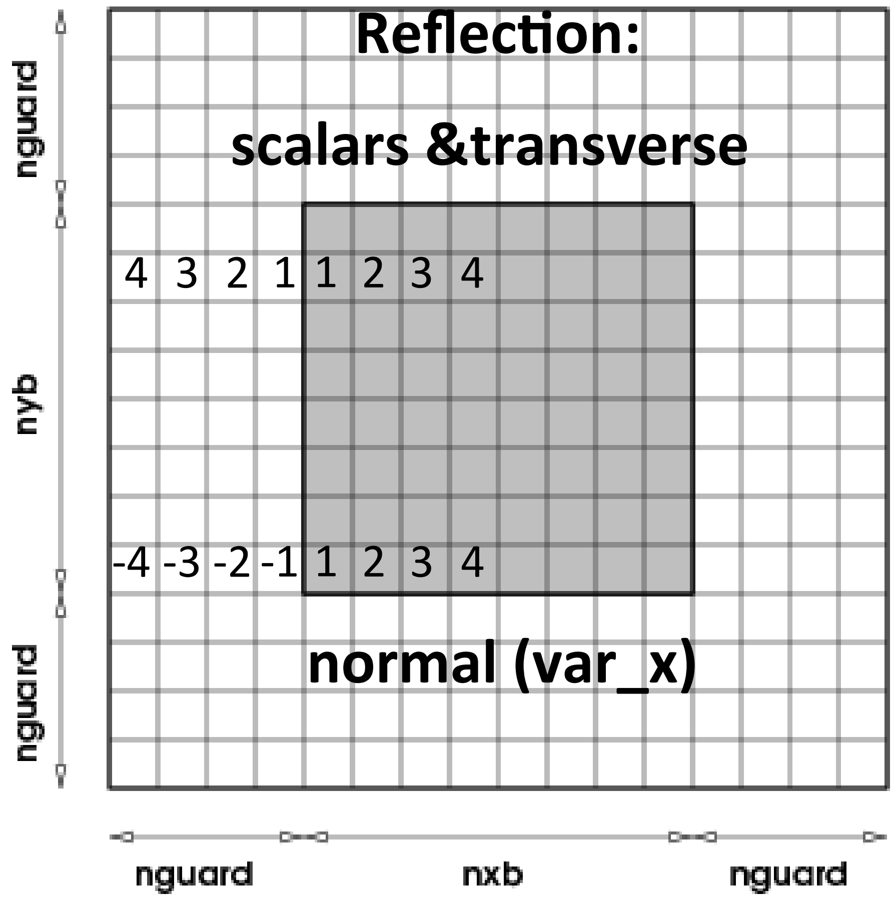
```
!!      Reflective
!!  Vn -> -Vn,  Bn -> -Bn
!!  Vp ->  Vp,  Bp ->  Bp
!!  Vt ->  Vt,  Bt ->  Bt
!!
!!      Axisymmetric
!!  Vn -> -Vn,  Bn -> -Bn
!!  Vp ->  Vp,  Bp ->  Bp
!!  Vt -> -Vt,  Bt -> -Bt
!!
!!     Eqtsymmetric
!!  Vn -> -Vn,  Bn ->  Bn
!!  Vp ->  Vp,  Bp -> -Bp
!!  Vt ->  Vt,  Bt -> -Bt
```



Example: Axisym., Eqtsym. BC

The new coordinate system benefits from symmetries that can be exploited if proper boundary conditions are present.

```
!!      Reflective
!!  Vn -> -Vn,  Bn -> -Bn
!!  Vp ->  Vp,  Bp ->  Bp
!!  Vt ->  Vt,  Bt ->  Bt
!!
!!      Axisymmetric
!!  Vn -> -Vn,  Bn -> -Bn
!!  Vp ->  Vp,  Bp ->  Bp
!!  Vt -> -Vt,  Bt -> -Bt
!!
!!     Eqtsymmetric
!!  Vn -> -Vn,  Bn ->  Bn
!!  Vp ->  Vp,  Bp -> -Bp
!!  Vt ->  Vt,  Bt -> -Bt
```



Let's implement these ideas…

```
[petros@Petross-MacBook-Pro] ~/Work/FLASH4/flash4
> less /Users/petros/Work/FLASH4/flash4/source/Grid/GridBoundaryConditions/Grid_bcApplyToRegion.F90
```

Example: Axisym., Eqtsym. BC

In order to make them play well with others…

```
source — bash — 126×14
bash … | bash … | bash … | bash … | bash … | bash … | bash | bash … | bash | flas...ash4 … | bash … | bash
[petros@Petross-MacBook-Pro] ~/Work/FLASH4/flash4/source
> less /Users/petros/Work/FLASH4/flash4/source/RuntimeParameters/RuntimeParametersMain/RuntimeParameters_mapStrToInt.F90
[petros@Petross-MacBook-Pro] ~/Work/FLASH4/flash4/source
> less /Users/petros/Work/FLASH4/flash4/source/Simulation/constants.h
[petros@Petross-MacBook-Pro] ~/Work/FLASH4/flash4/source
>
```

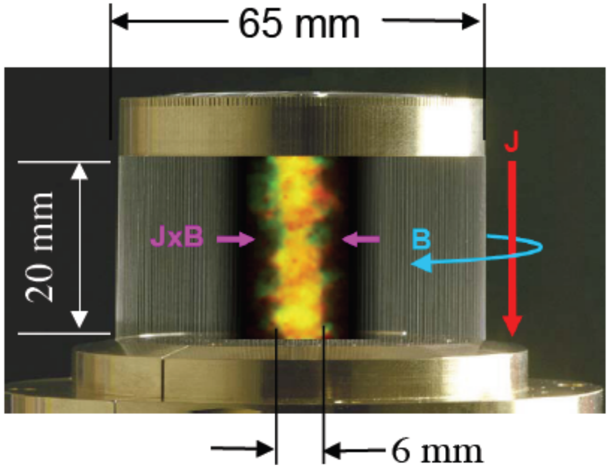
Example: Implementation verification

OK, now we have to test our implementation!

Magnetized Noh, take two: Cylindrical geometry!

OK, now we have to test our implementation and the new boundary conditions!

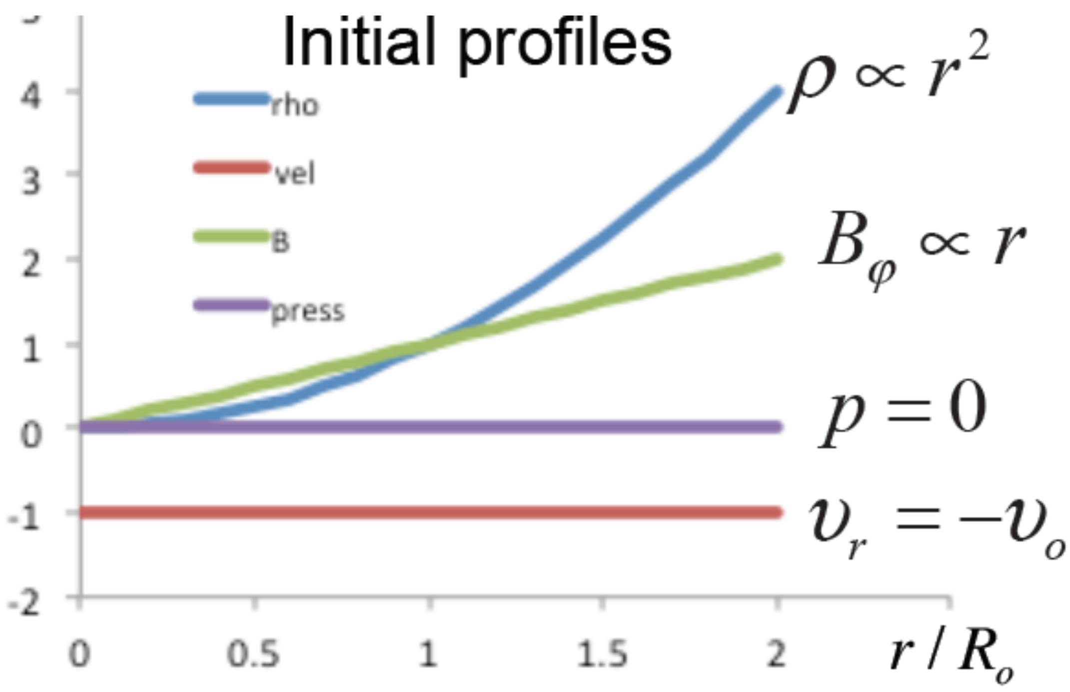
Magnetized Noh, take two: Cylindrical geometry!



>The magnetized Noh is an inherently cylindrical setup

>Even easier to initialize than the Cartesian setup we saw.

>Let’s see the initial conditions



Example: Implementation verification

We'll view the initialization

Modify the par file

Setup the test problem

Compile

Run

Visualize the data & compare with the analytical solution

HANDS ON!

Example: Implementation verification

```
object — bash — 126×12
[petros@Petross-MacBook-Pro] ~/Work/FLASH4/flash4/object
> less /Users/petros/Work/FLASH4/flash4/source/Simulation/SimulationMain/magnetoHD/NohCylindrical/Simulation_initBlock.F90
```

Let's take a look…

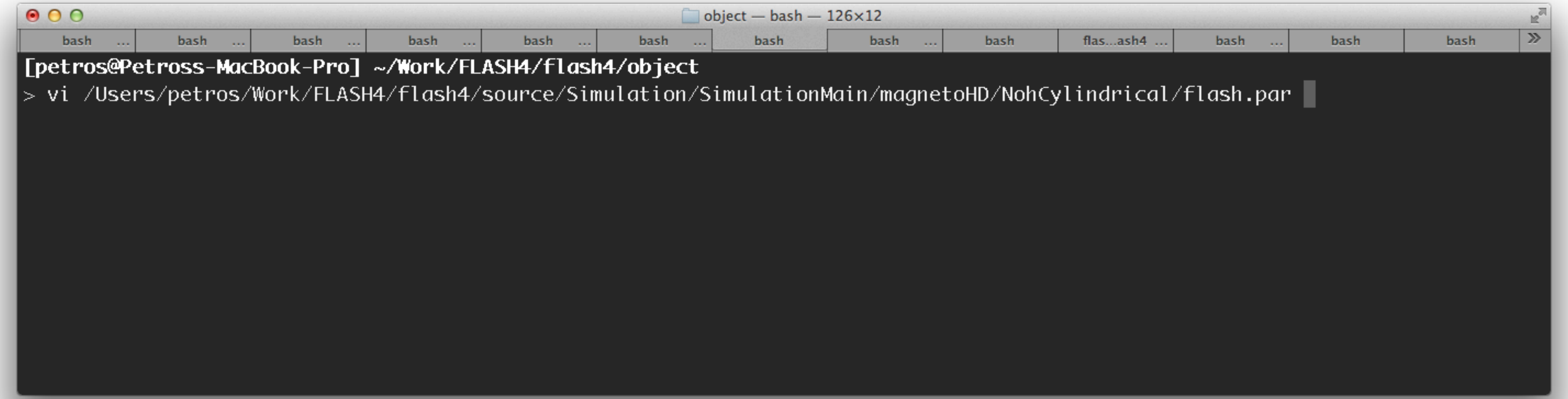
Example: Implementation verification

```
[petros@Petross-MacBook-Pro] ~/Work/FLASH4/flash4/object
> vi /Users/petros/Work/FLASH4/flash4/source/Simulation/SimulationMain/magnetoHD/NohCylindrical/flash.par
```

Modify the par file: use your preferred text editor, e.g. vim, emacs and so on.

ASK FOR HELP IF YOU SHOULD HAVE ANY TROUBLE

Example: Implementation verification



We will set the refinement conditions and comment some conflicting arguments.

Define: lrefine_min, lrefine_max, nrefs, refine_var_1

ASK FOR HELP IF YOU SHOULD HAVE ANY TROUBLE

Example: Implementation verification

```
[petros@Petross-MacBook-Pro] ~/Work/FLASH4/flash4
> ./setup -auto magnetoHD/NohCylindrical +usm -2d +pm4dev +cylindrical -site=brassica.asci.uchicago.edu
```

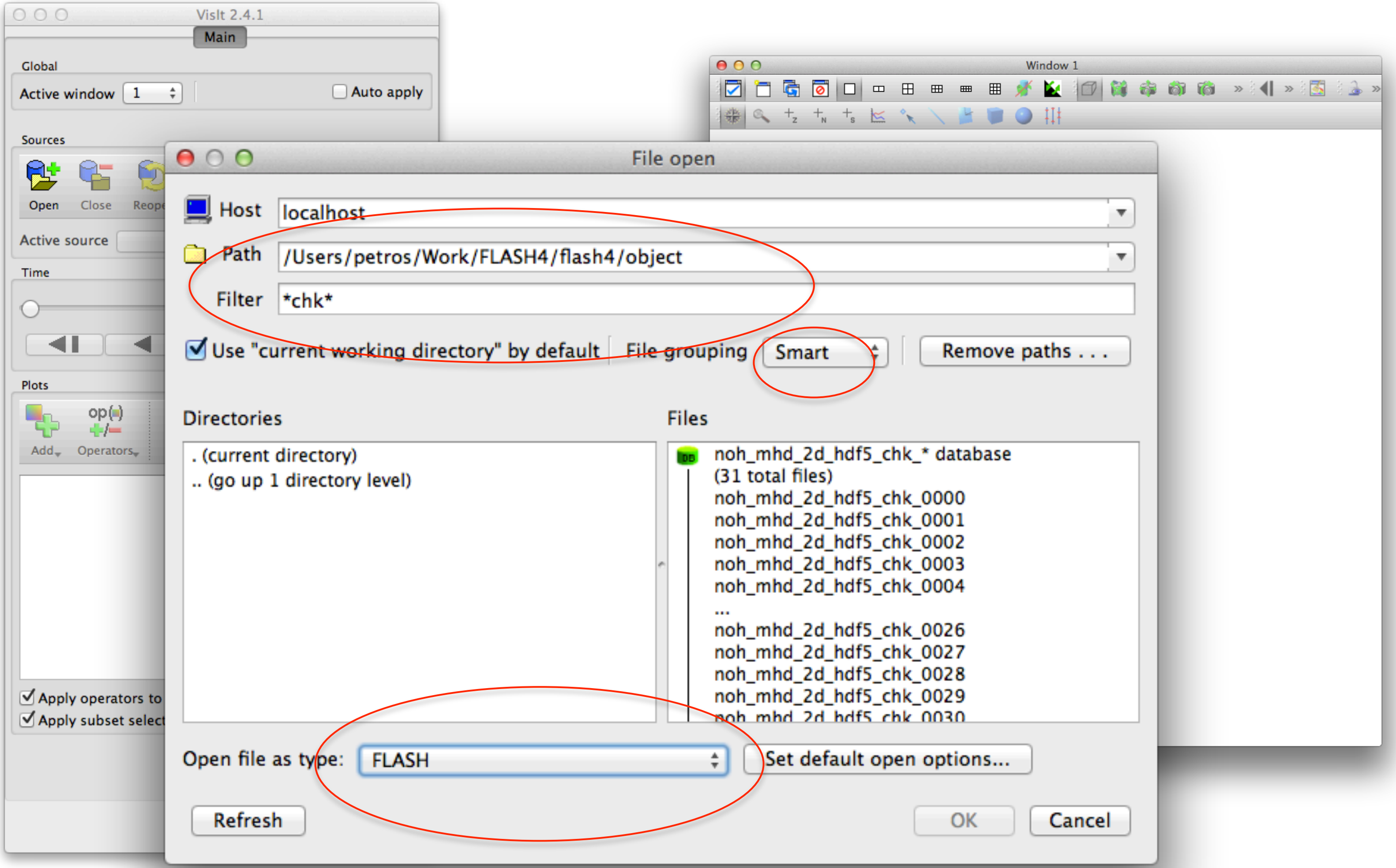
```
[petros@Petross-MacBook-Pro] ~/Work/FLASH4/flash4
> cd object
[petros@Petross-MacBook-Pro] ~/Work/FLASH4/flash4/object
> make -j4
```

```
[petros@Petross-MacBook-Pro] ~/Work/FLASH4/flash4/object
> mpirun -np 4 ./flash4
```

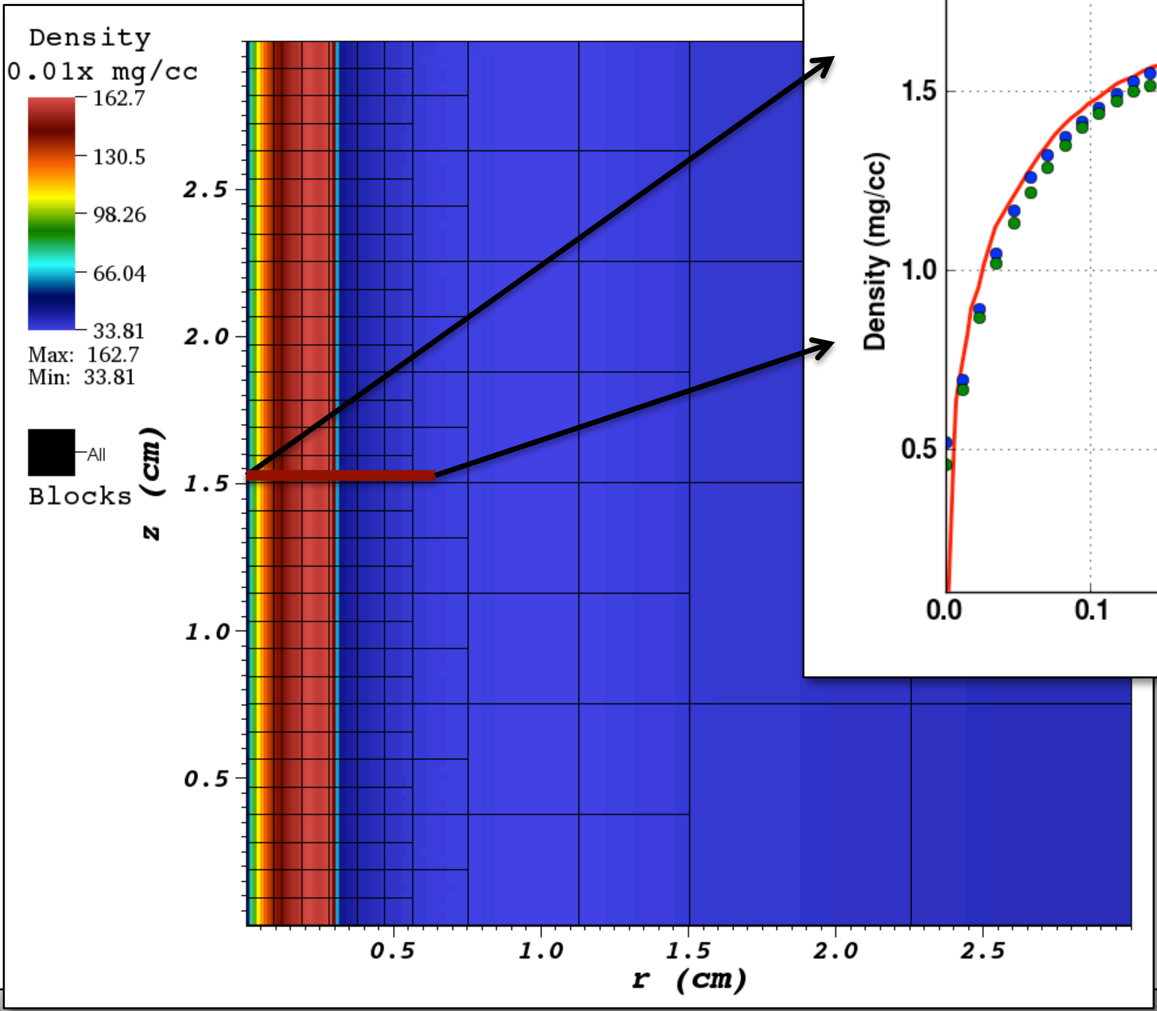
Setup, compile, and run the problem

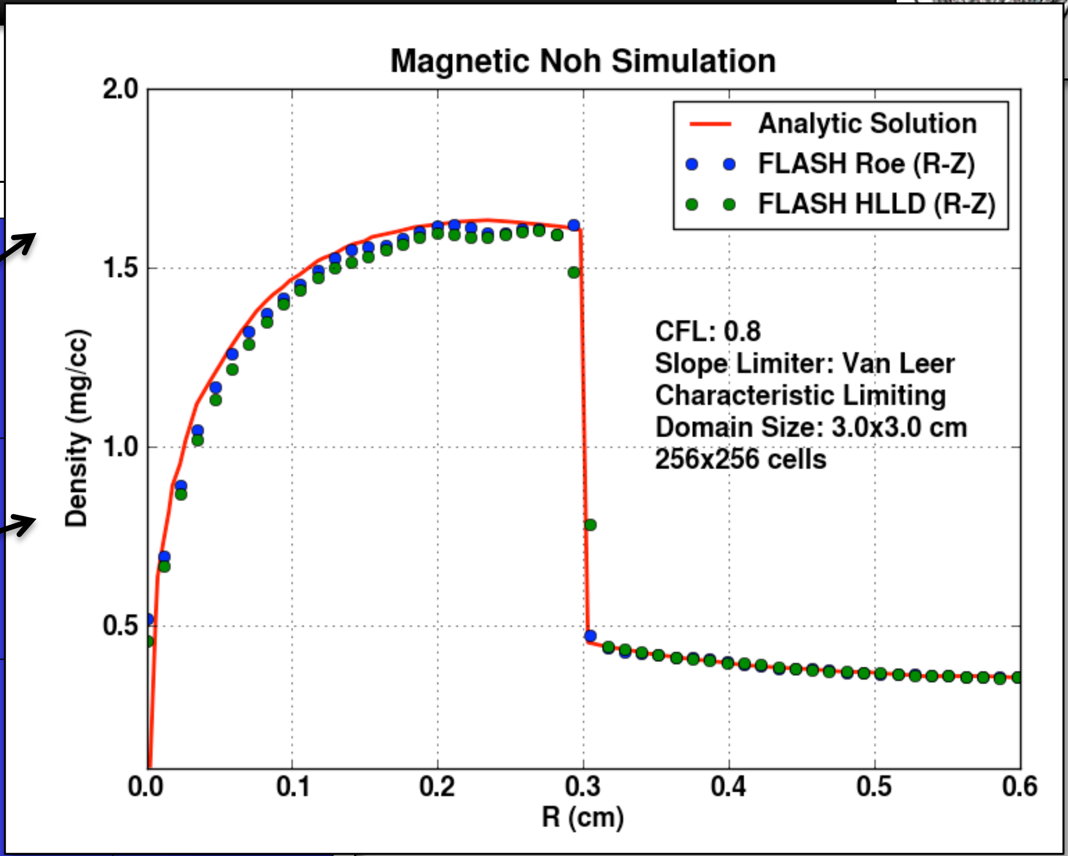
ASK FOR HELP IF YOU SHOULD HAVE ANY TROUBLE

That was it! Now let's take a look at the results.



Example: Implementation verification





Done!

- Follow the guidelines in existing implementations

- Make sure that your implementation does not conflict (#ifdefs are the salt and pepper of coding life!)

- Test and validation are important, have more than one problems to capture your bugs!

Thanks for listening! Happy coding!