Implementation of Adaptive Mesh Refinement in CFD-MHD code Antares : Hyperbolic Solvers

Somdeb Bandopadhyay

ASIAA/NTU-IAM, Taipei, ROC

December, 2017

Outline

Outline for Section 1

Preface

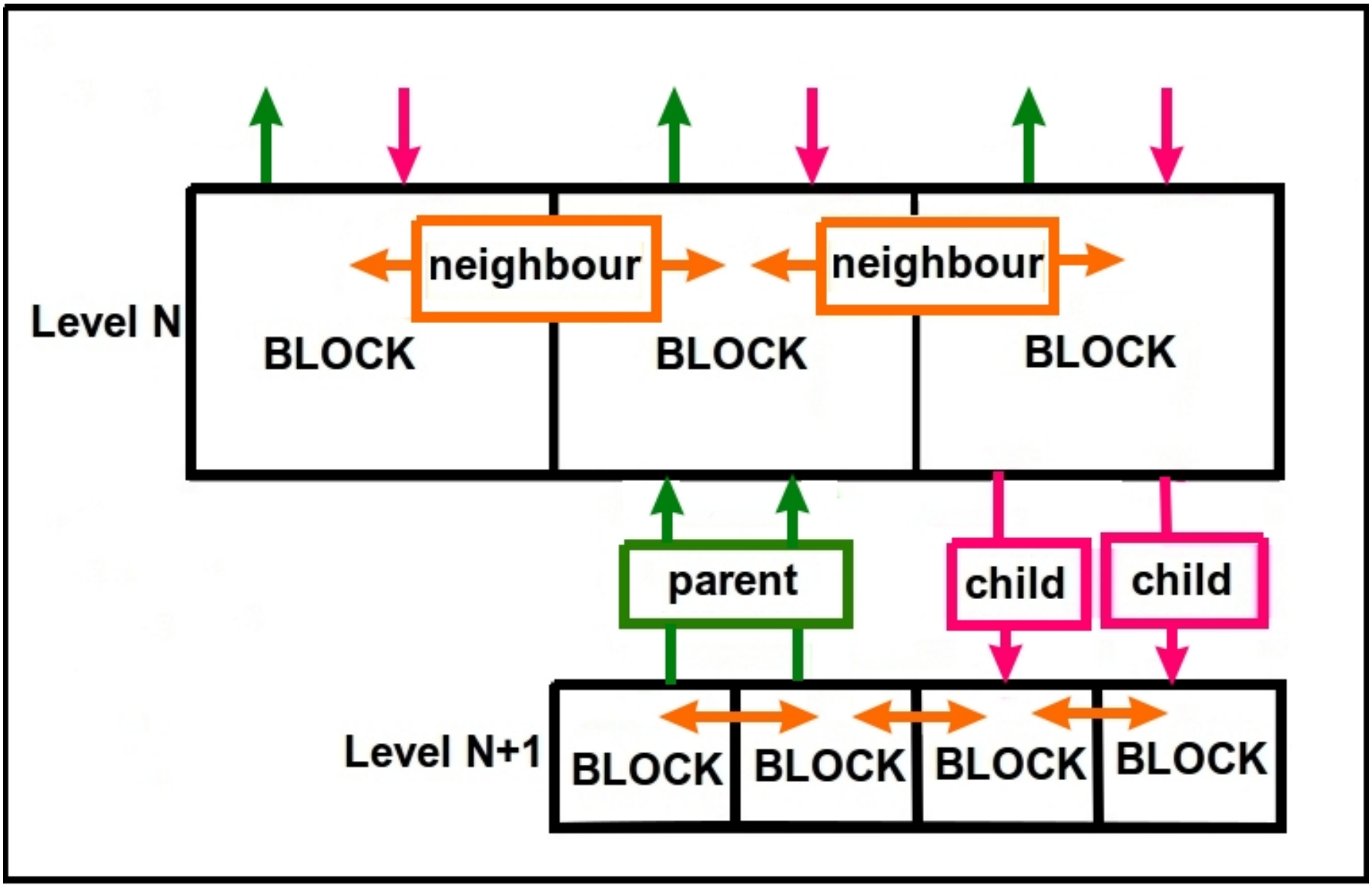
- This presentation is aimed to provide the key concepts and brief overview of the code and it's possible future development.
- This is not a thorough discussion about AMR method in general
- We shall go through the in depth analysis of different numerical schemes in another presentation

Current work

- Block Structured Adaptive Mesh Refinement has been implemented for hydrodynamic solvers in Antares
- Logical extension of the concept behind invoking nonuniform grids to efficiently utilize computer resources.
- Use finer grids where more accurate solution is desired.
- First introduced by Berger and Oliger (JCP,1984)
- Large variety of AMR approaches : different types of mesh topology, refinement criteria and data structure management. We shall only focus on block structured adaptive mesh refinement (BSAMR) procedures.

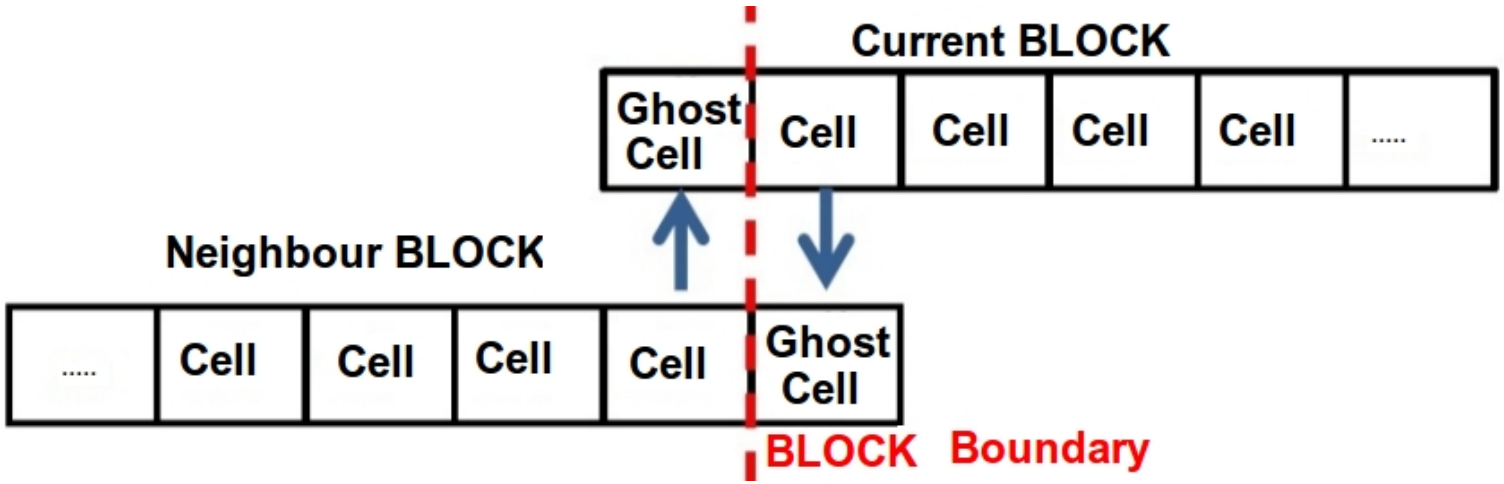
Outline for Section 2

The Parent-Child(ren) Arrangement



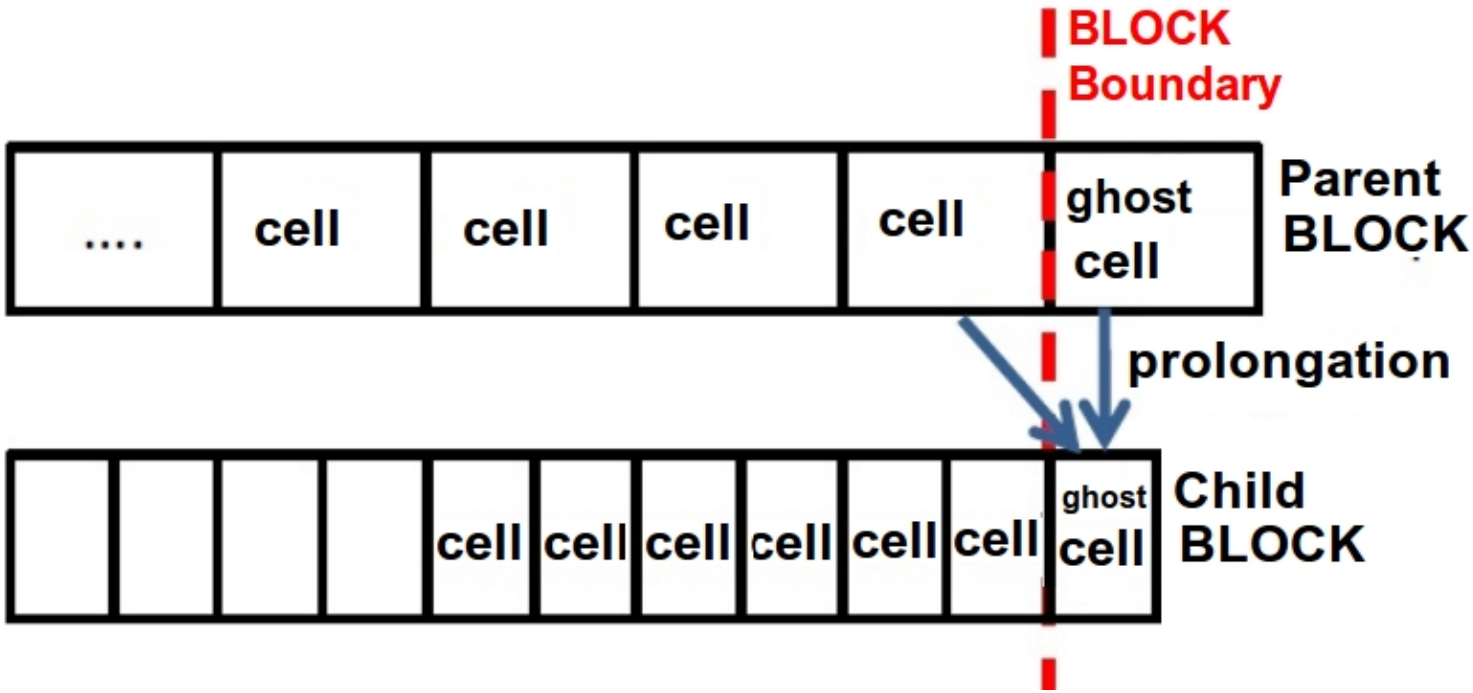
AMR Operations : Exchange/Copy

Transfer solution from inner cell of current block to the ghost cell(s) of neighbor block and receive data from neighbor's cell to the ghost cell of current block



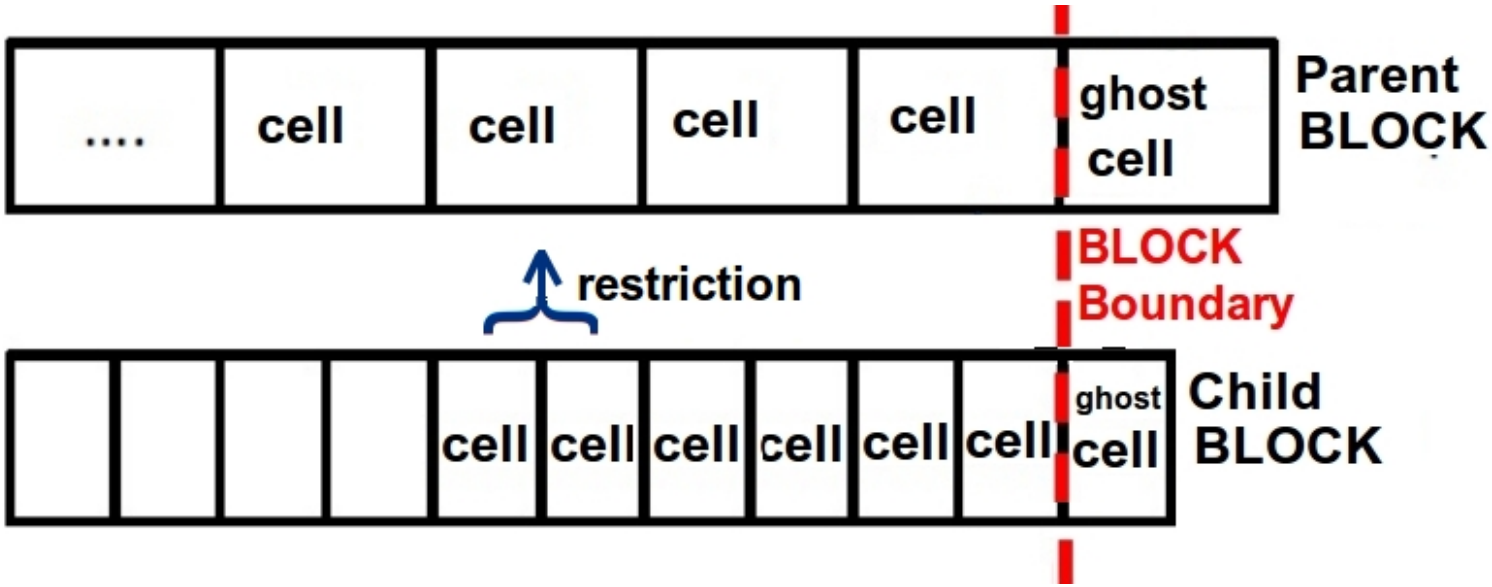
AMR Operations : Prolongation

Interpolation of the solution on a coarse mesh to the finer mesh



AMR Operations : Restriction

Mapping from the fine mesh solution to the coarse mesh



Block Structures

- The module bsamr_blocks defines and handles the block structures.
- Each block of our mesh configuration has three derived datatypes associated with it :
  - meshblocks contain all mesh information, pointers to neighbors, leaf and refinement flags, etc.
  - fieldblocks contain the primitive and conservative field variables which are being updated
  - infoblocks contain the information needed for domain decomposition

Derive Datatype : block_mesh for 2D

```
type block_mesh
type(block_mesh)   , pointer :: prev, next
type(block_mesh)   , pointer :: parent
type(pointer_mesh)           :: child(nchildren)
type(pointer_mesh)           :: edges(nsides,nsides,ndims)
type(pointer_mesh)           :: corners(nsides,nsides)
type(block_field) , pointer :: field
integer(kind=4)              :: id
integer(kind=4)              :: pid
integer(kind=4)              :: level
integer(kind=4)              :: conf
integer(kind=4)              :: refine
integer(kind=4)              :: pos(ndims)
integer(kind=4)              :: coords(ndims)
logical                      :: leaf
logical                      :: update
real(kind=8)                 :: xmin, xmax, ymin, ymax
end type block_mesh
```

Derive Datatype : block_field

```
type block_field

type(block_field), pointer                              :: prev
type(block_field), pointer                              :: next
type(block_mesh), pointer                               :: mesh
real(kind=8), dimension(:,:,:,:)   , pointer            :: u
real(kind=8), dimension(:,:,:,:)   , allocatable        :: u0, u1
real(kind=8), dimension(:,:,:,:)   , allocatable        :: q
real(kind=8), dimension(:,:,:,:,:) , allocatable        :: f

end type block_field
```

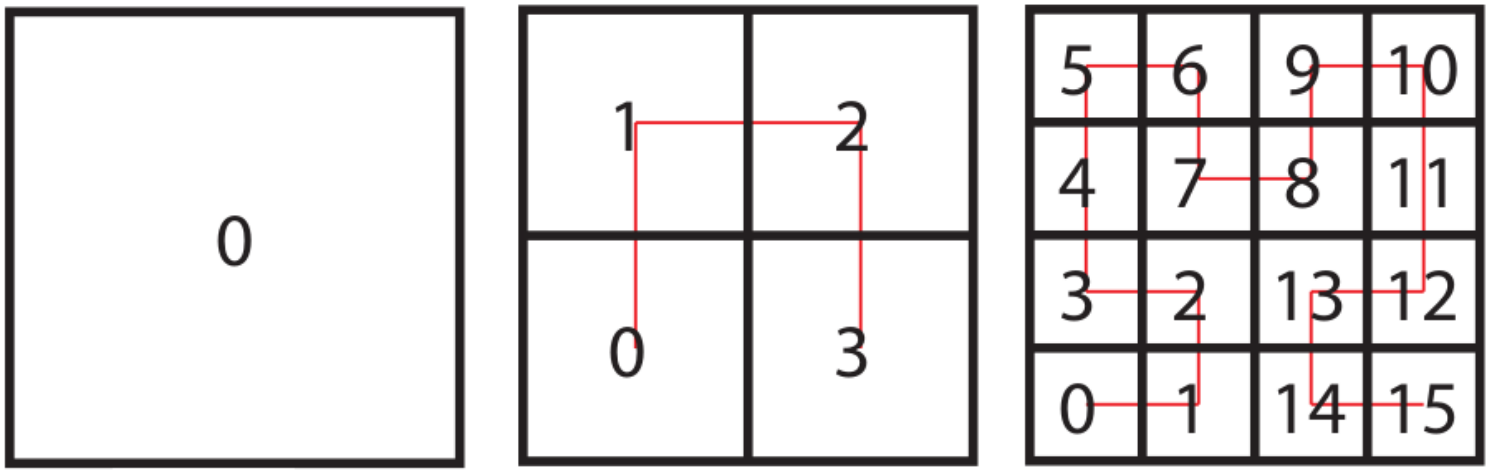
Derive Datatype : block_info

```
type block_info
type(block_info)   , pointer :: prev, next
type(block_mesh)   , pointer :: block
type(block_mesh)   , pointer :: neigh
integer(kind=4)              :: direction
integer(kind=4)              :: corner(NDIMS)
integer(kind=4)              :: level_difference
end type block_info
```

Control of AMR

- We calculate the local second derivative error of a variable, and find its maximum value within each block
- This maximum values is then compared to the refinement and derefinement criterions namely crefmin (default value 0.2) and crefmax (default value 0.8)
- If the maximum error is larger than parameter crefmax, the block is selected for refinement.All blocks which are marked for refinement are refined.
- If maximum error is smaller than crefmin, the block is selected for derefinement. Whenever the block is marked for derefinement, it is actually derefined only if all its siblings are marked for derefinement and all neighbors are on the same or lower level.
- By default we use density and pressure for calculation of secod derivative error, but any primitive variables can be used by settingvalue of refinement_variables in the input file

Domain decomposition : Hilbert Curve



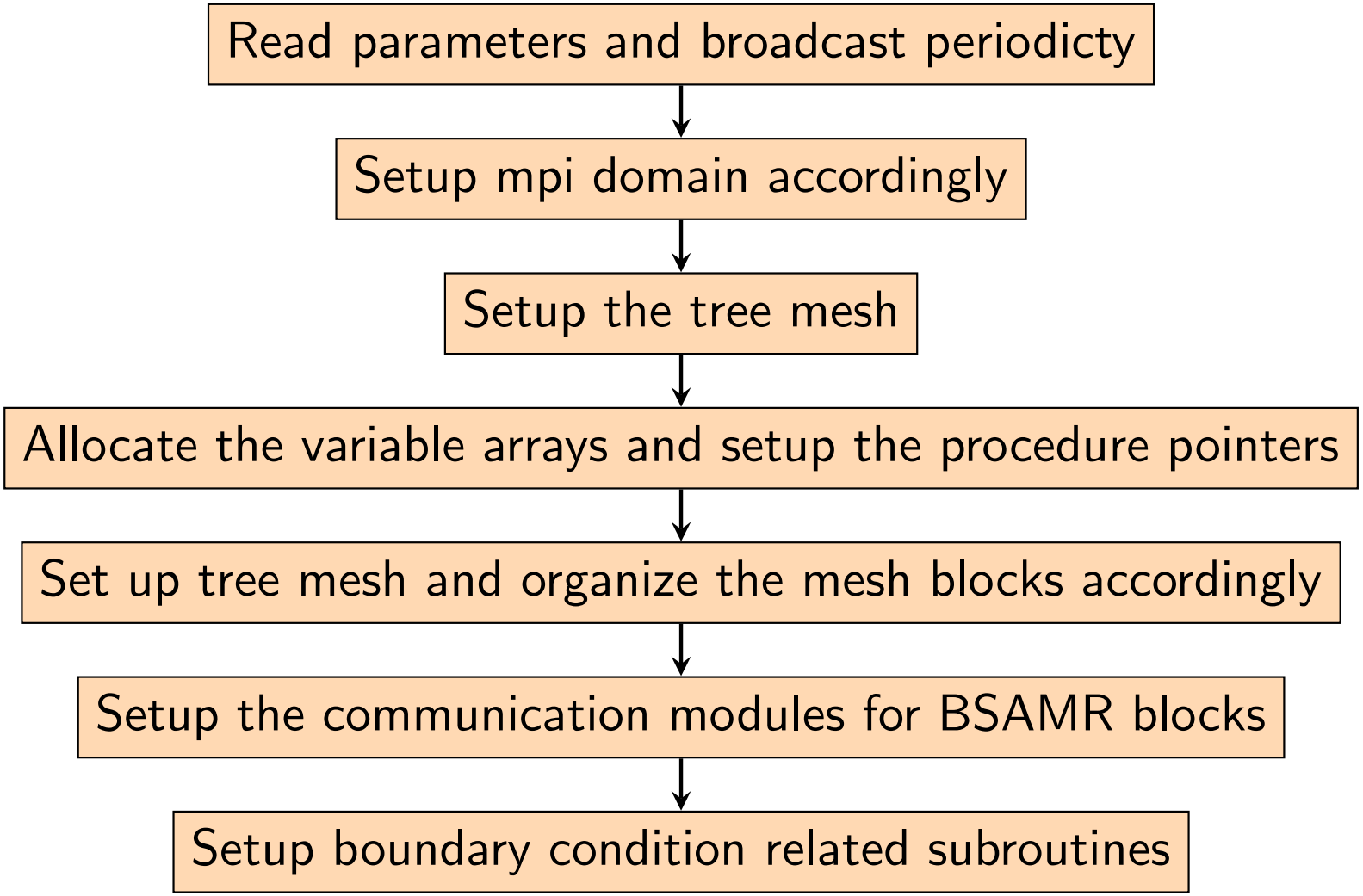
Level 0 **Level 1** **Level 2**

Outline for Section 3

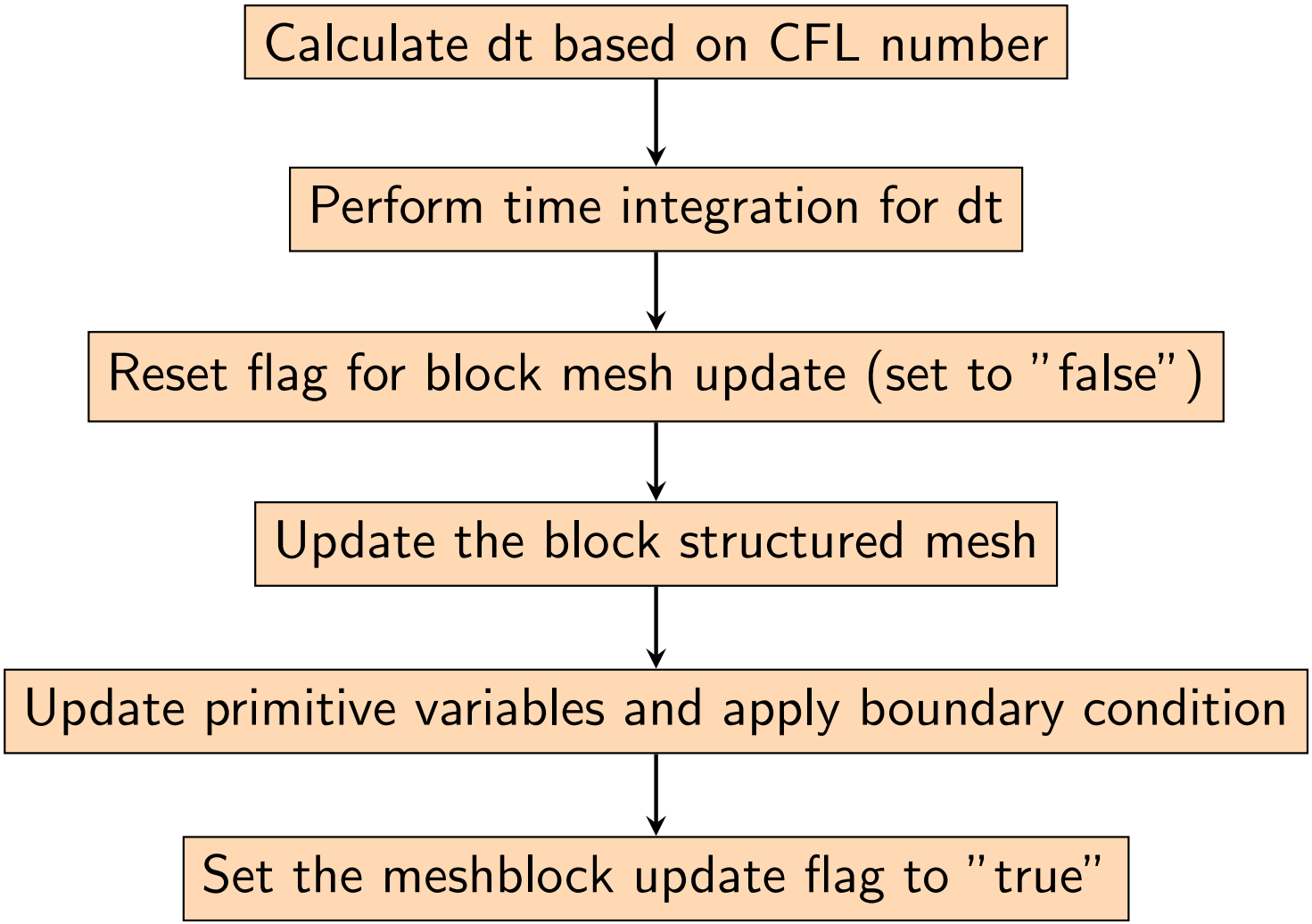
Input File

- Each line will contain the name and value of one parameter.
- Comments are added at each line by putting # in from of it.
- Each line has the form :
  $< \textit{parameter name} >=< \textit{parameter value} >$
- parameter value can be a string, interger of real type

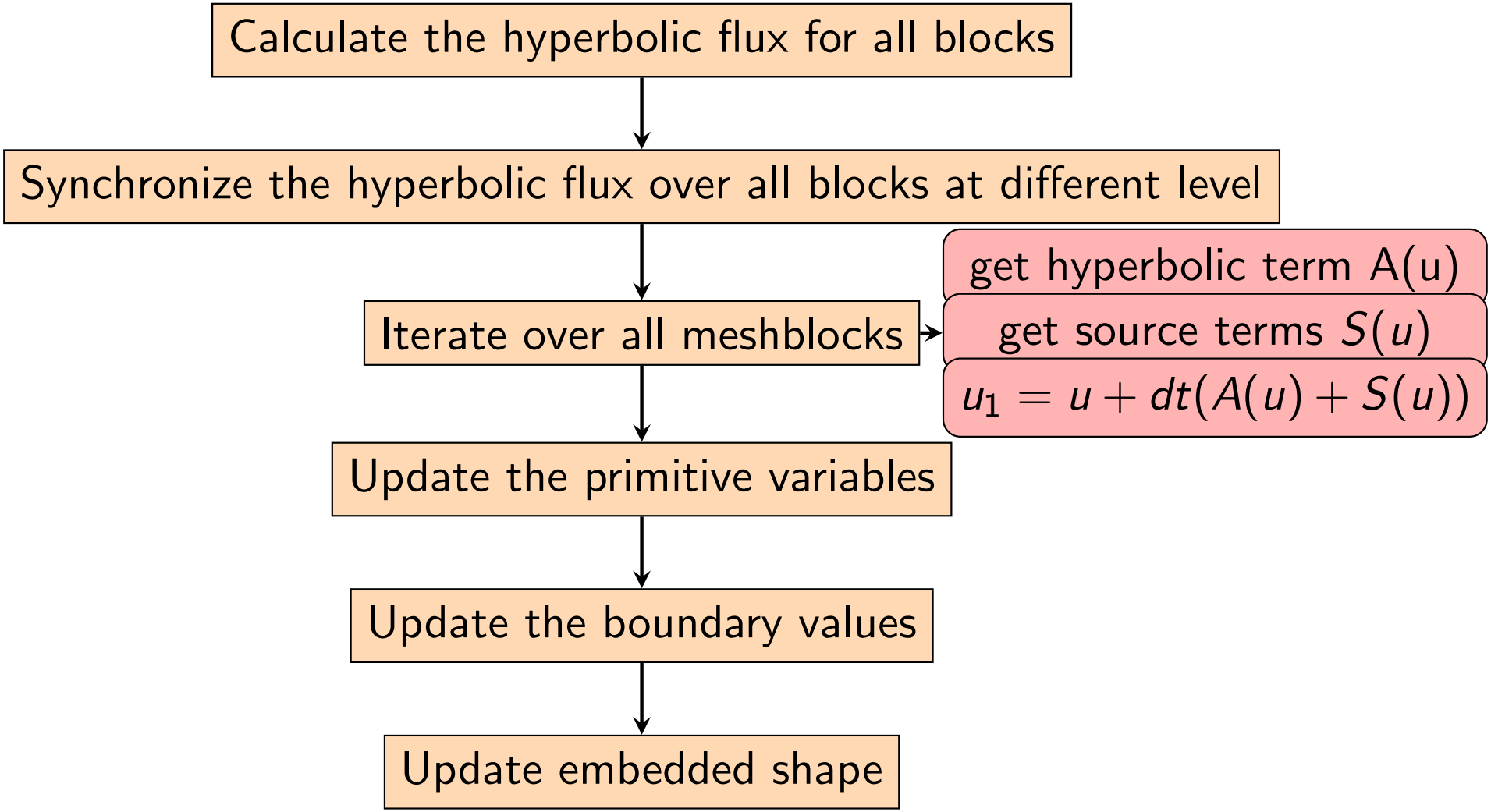
Pre-Time Loops Operations



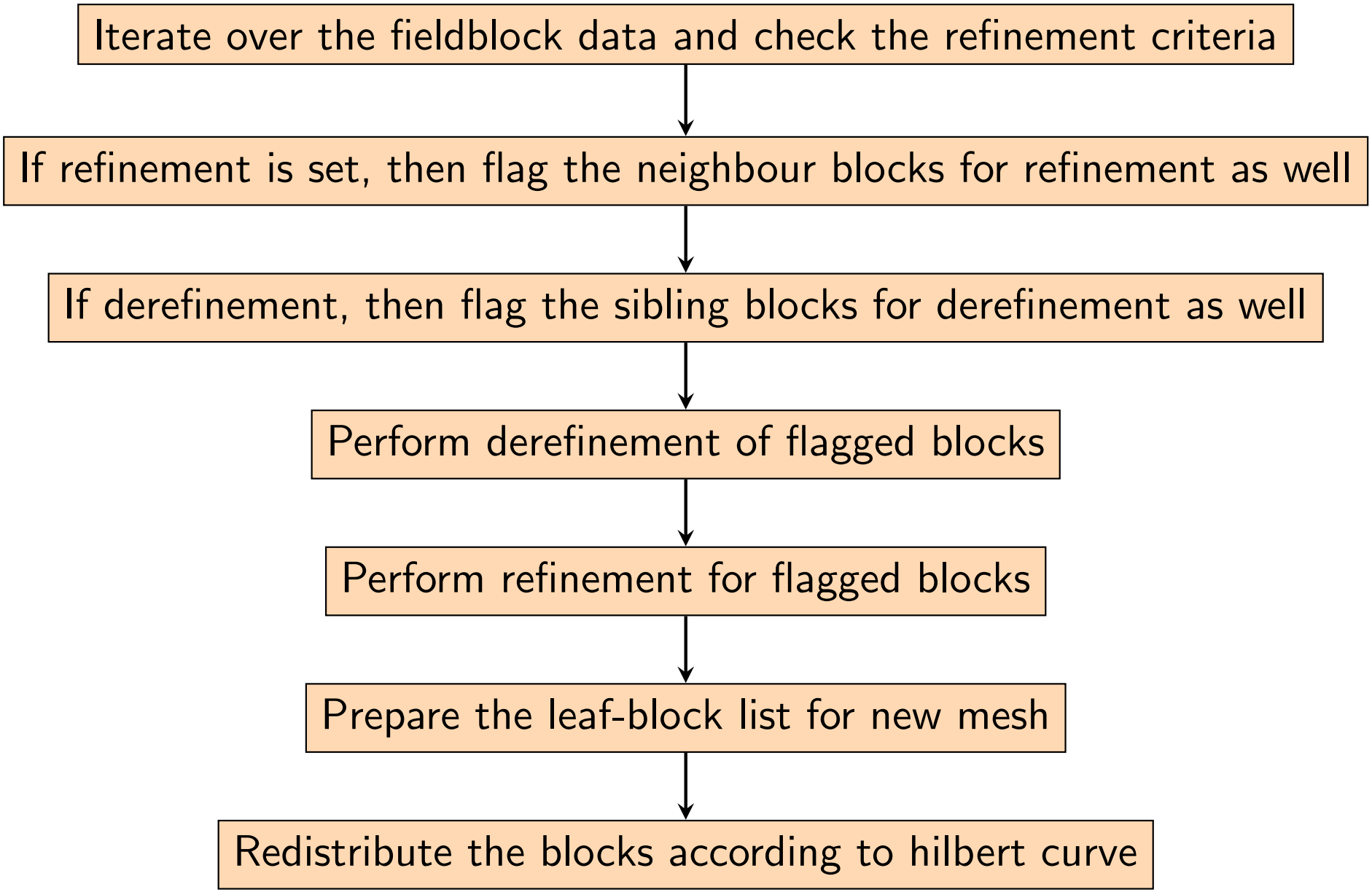
Main Time Loop



Time Integration (single step)



Update Procedure for Block Structured Mesh



Outline for Section 4

Why Should we care about Higher-order schemes?

Let's a space-time dependent function $f(x,t)$ in terms of its Laplace-Fourier transform by

$$f(x,t) = \int F(k,t)\, e^{ikx} dk \tag{4.1}$$

Analytically we get the spacial derivative of $f(x,t)$

$$\frac{\partial f}{\partial x} = \int ikF(t,k)\, e^{ikx} dk \tag{4.2}$$

When this is evaluated numerically, one writes an equivalent representation as

$$\frac{\partial f}{\partial x} = \int ik_{eq}F(t,k)\, e^{ikx} dk \tag{4.3}$$

with equivalent wavenumber k_{eq} different for different discretization schemes.

Evaluation of k_{eq}

If use 2nd Order Central differencing (CD2) for evaluation of first derivative in an uniform grid in the physical plane :

$$\left.\frac{\partial f}{\partial x}\right|_m = \frac{f_{m+1} - f_{m-1}}{2h} \tag{4.4}$$

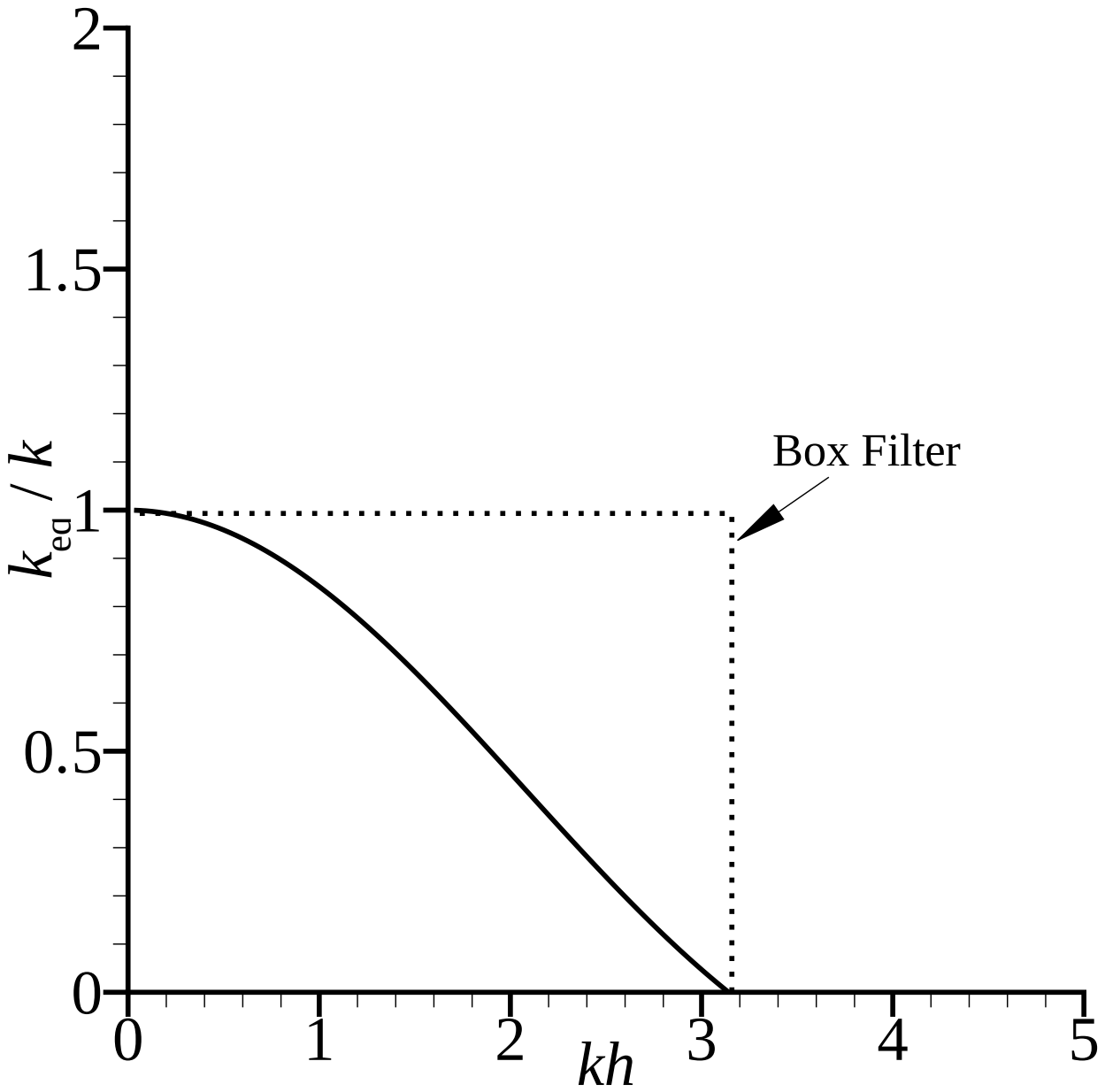
Since $x = mh$, one can use Eqn. (4.1) in Eqn. (4.4) to get the following :

$$\frac{\partial f}{\partial x} = \int F(t,k)\frac{e^{ik(m+1)h} - e^{ik(m-1)h}}{2h}\, dk = \int F\frac{e^{ikh} - e^{-ikh}}{2h}\, e^{ikx_m} dk \tag{4.5}$$

Comparing expressions in Eqns. (4.3) and (4.5) one gets

$$\left(\frac{k_{eq}}{k}\right)_{CD_2} = \frac{\sin\ kh}{kh} \tag{4.6}$$

Plot of k_{eq} for CD2 vs Fourier Spectral Case



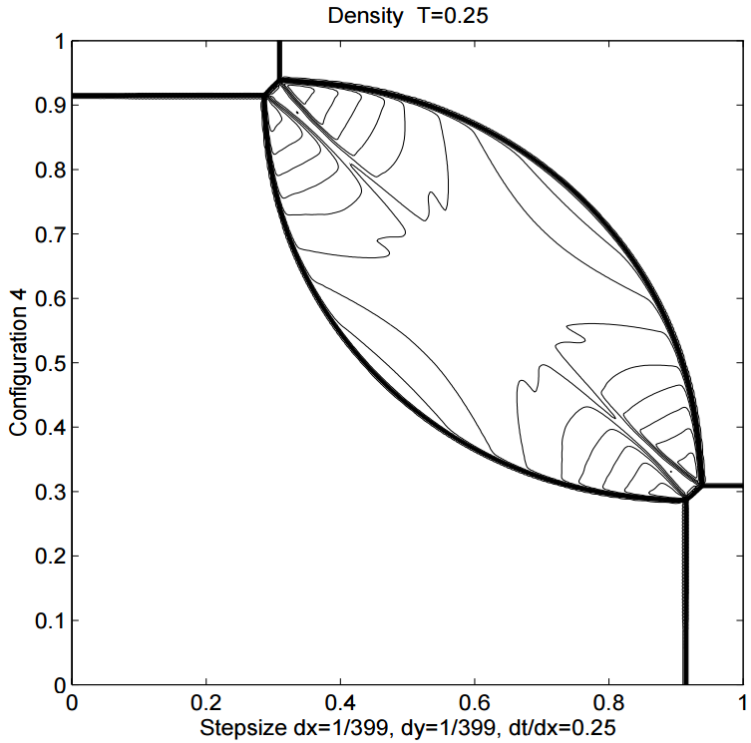
Discretization Schemes (Need separate presentation)

1. **Shock capturing schemes** => Riemann solvers currently implemented :
   - HLL
   - HLLC(adiabatic only) : *The tangential components of the velocity are discontinuous across the contact discontinuity[Toro et al,Shock Waves, 1994]*
   - Roe-solver : *[Roe, P. L. Journal of Computational Physics, 1981]*
2. **Interpolation schemes** => Available interpolation methods :
   - 2nd order TVD (default)
   - 3rd order WENO : Yamaleev & Carpenter,Journal of Computational Physics, 2009
   - 3rd order Monotonically Preserving scheme : He, Z.-W.et al, Science China: Physics, Mechanics & Astronomy, 2011
   - Compact Reconstruction based WENO and MP schemes (CRWENO, Ghosh and Baeder, Journal on Scientific Computing, 2012)

Outline for Section 5

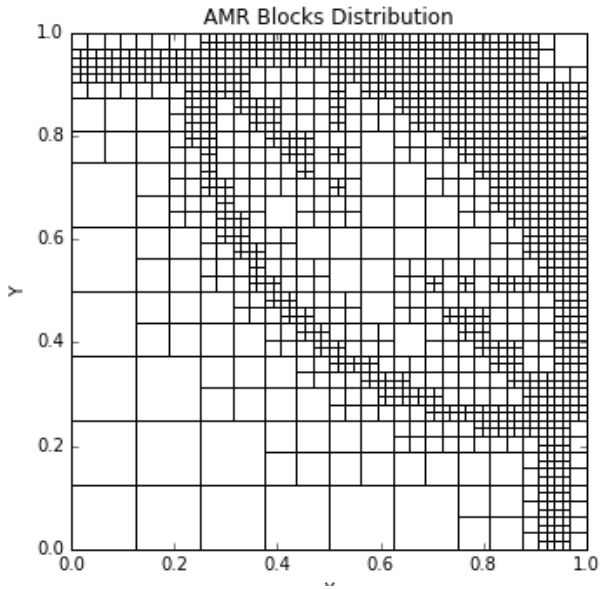
2D Riemann Problem

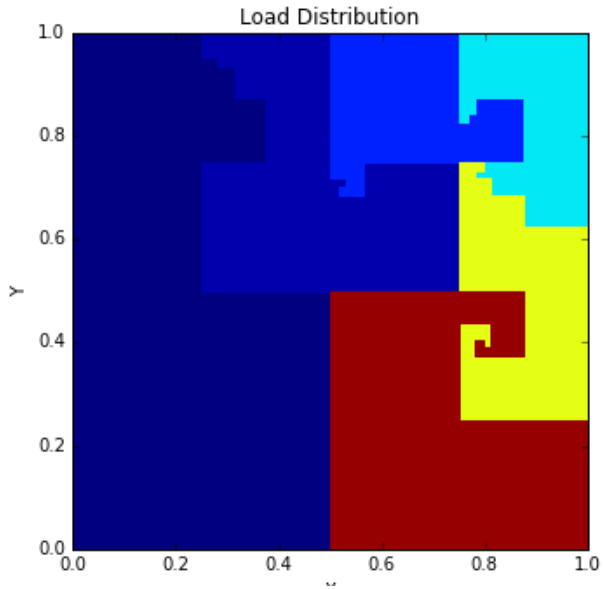
Case : P. Lax and X.-D. Liu, "Solution of two-dimensional Riemann problems of gas dynamics by positive schemes," SIAM J Sci Comp 19 (1998), case 4

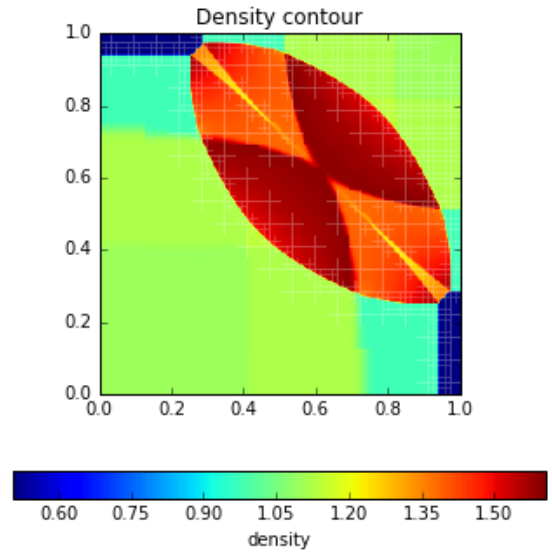


2D Riemann Problem

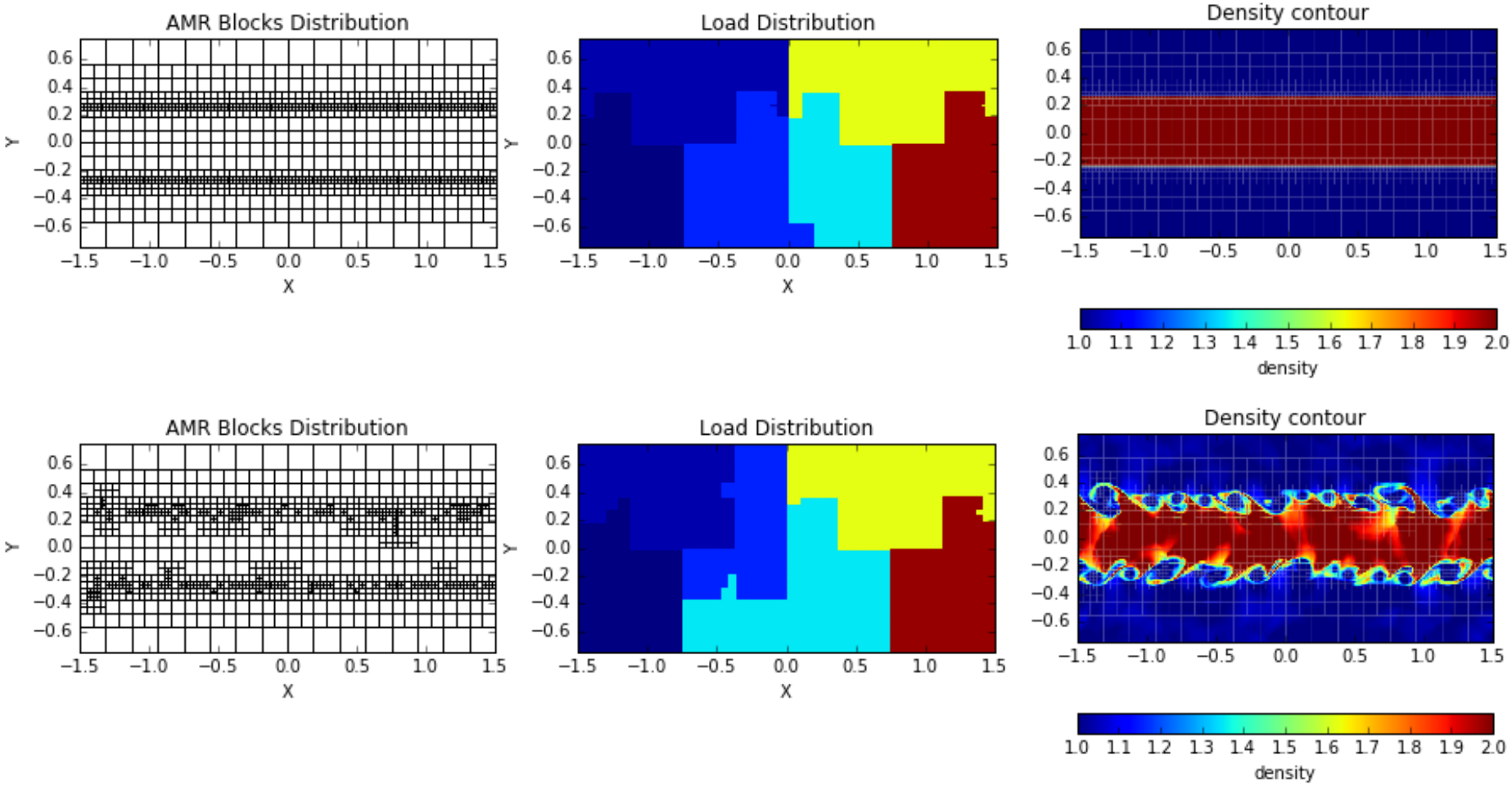
Results : HLLC, RK2,WENO3, cfl=0.4, maxlevel of refinement = 7



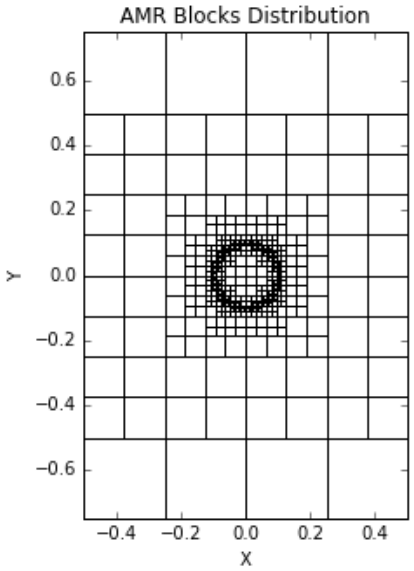


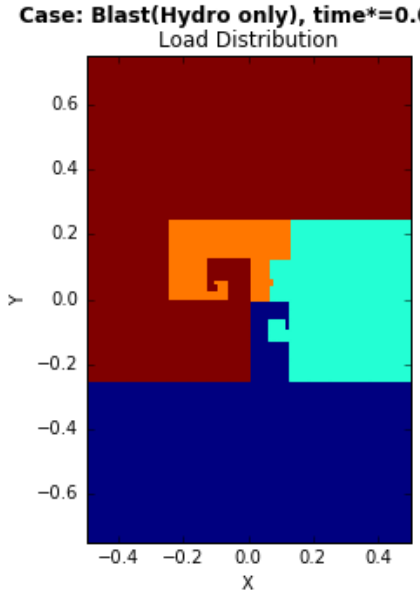


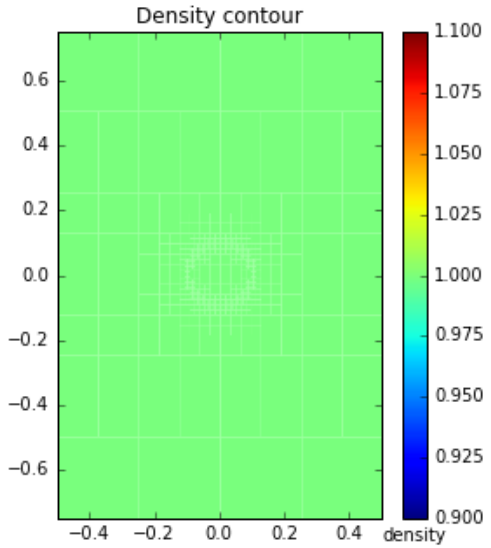
Test Case : KHI

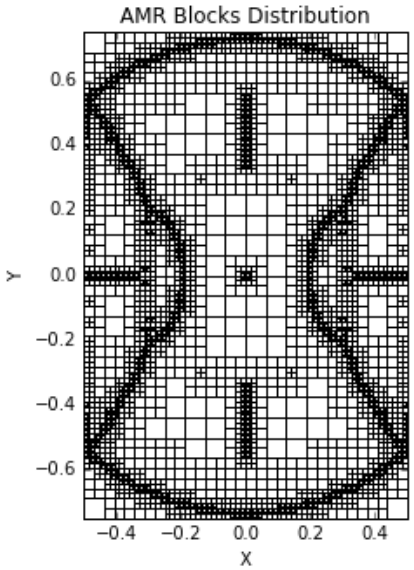


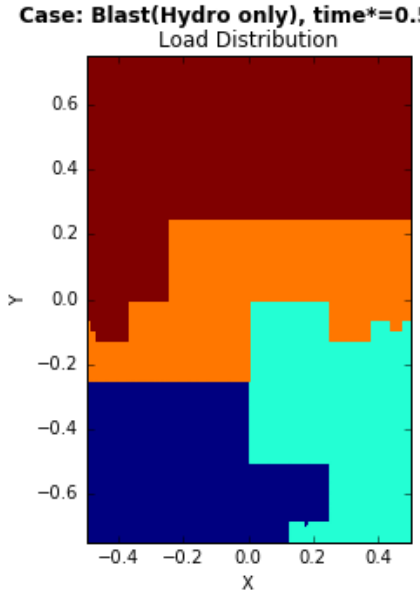
Test Case : Blastwave

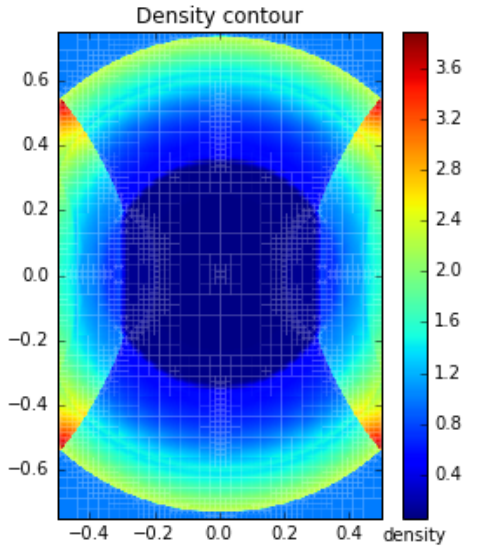




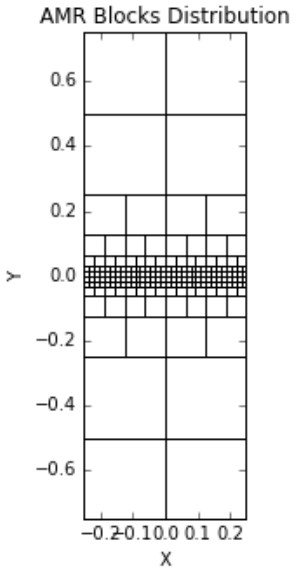


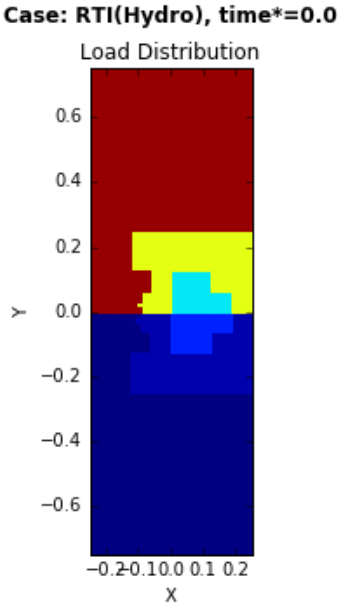


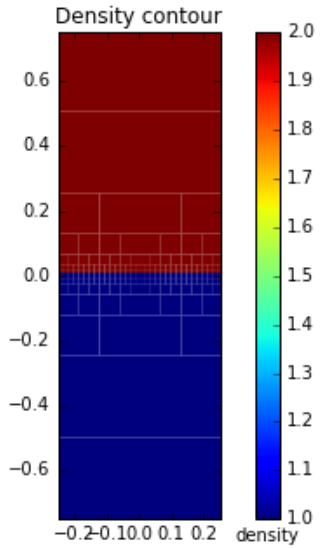


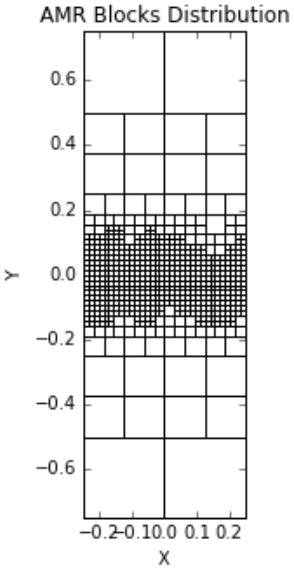


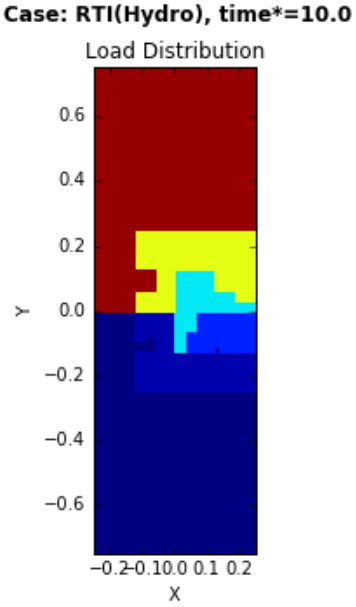
Test Case : RTI

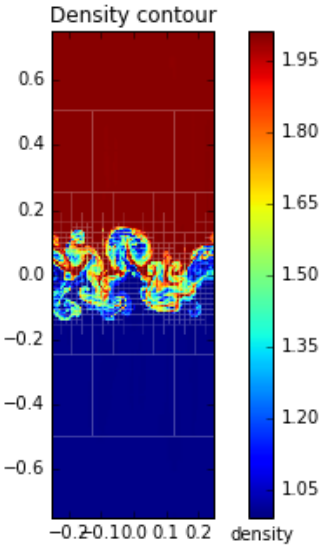












Current issues

- The python script for post-process is having some memory related issues.
- We still need to check the scaling for the load balancing
- As of now, all variables are stored in collocated arrangement, I am not sure if we need to modify some parts of the code for MHD case

Conclusion and Future Work

- We have successfully implemented adaptive mesh refinement based on nested grid topology, to solve hyperbolic problems
- Dynamic load balancing is done using hilbert space filling curve which is widely accepted in scientific-computation community
- Multiple options are available for spacial discretization including second order TVD to higher order WENO and compact schemes
- Flexibility of the domain is achieved by embedded boundary type approach as shown by the blast wave case
- Addition of active and passive particle transport can be considered
- Development of poisson solvers is ongoing
- Implementation of MHD algorithm will be done in future

Thanks!