An introduction to Adaptive Mesh Refinement (AMR) - Part 1: Numerical Methods and Tools

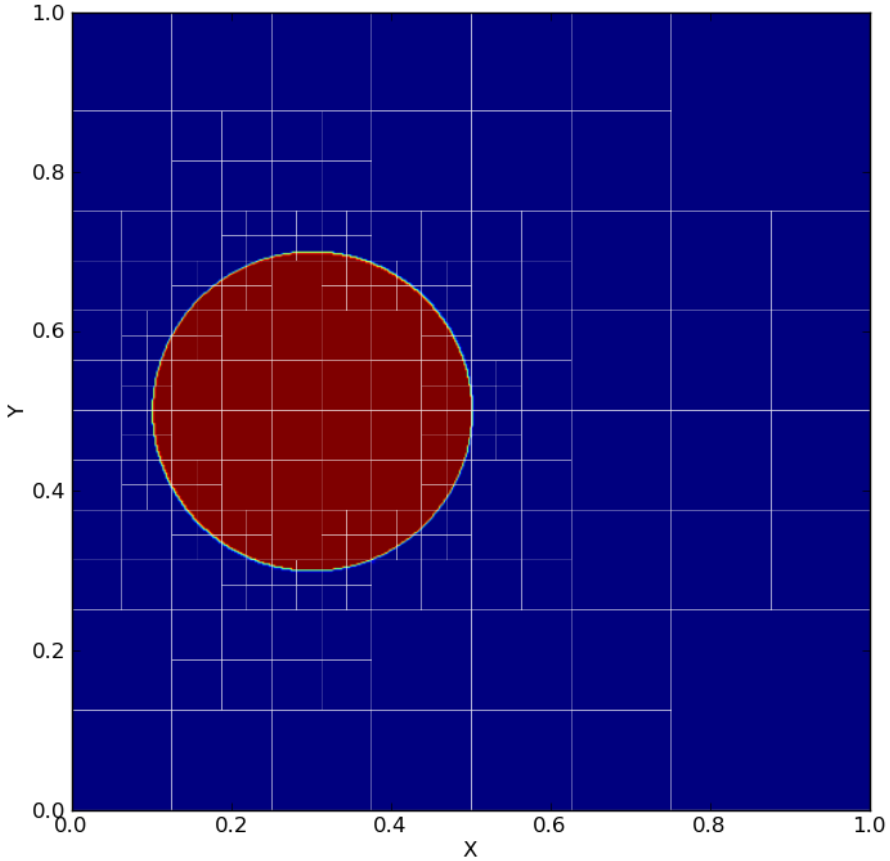
HPC Numerical Libraries
26-28 April 2016
CINECA – Casalecchio di Reno (BO)

Massimiliano Guarrasi- m.guarrasi@cineca.it
Super Computing Applications and Innovation Department

AMR - Introduction

- Solving Partial Differential Equations (PDEs)
  - PDEs solved using discrete domain
  - Algebraic equations estimate values of unknowns at the mesh points
  - Resolution/Spacing of mesh points determines error
  - Initial Solution and Boundary condition are needed
- Goal of grid adaptivity:
  - tracking features much smaller than overall scale of the problem providing adequate higher spatial and temporal resolution where needed.

AMR - Introduction



Uniform meshes

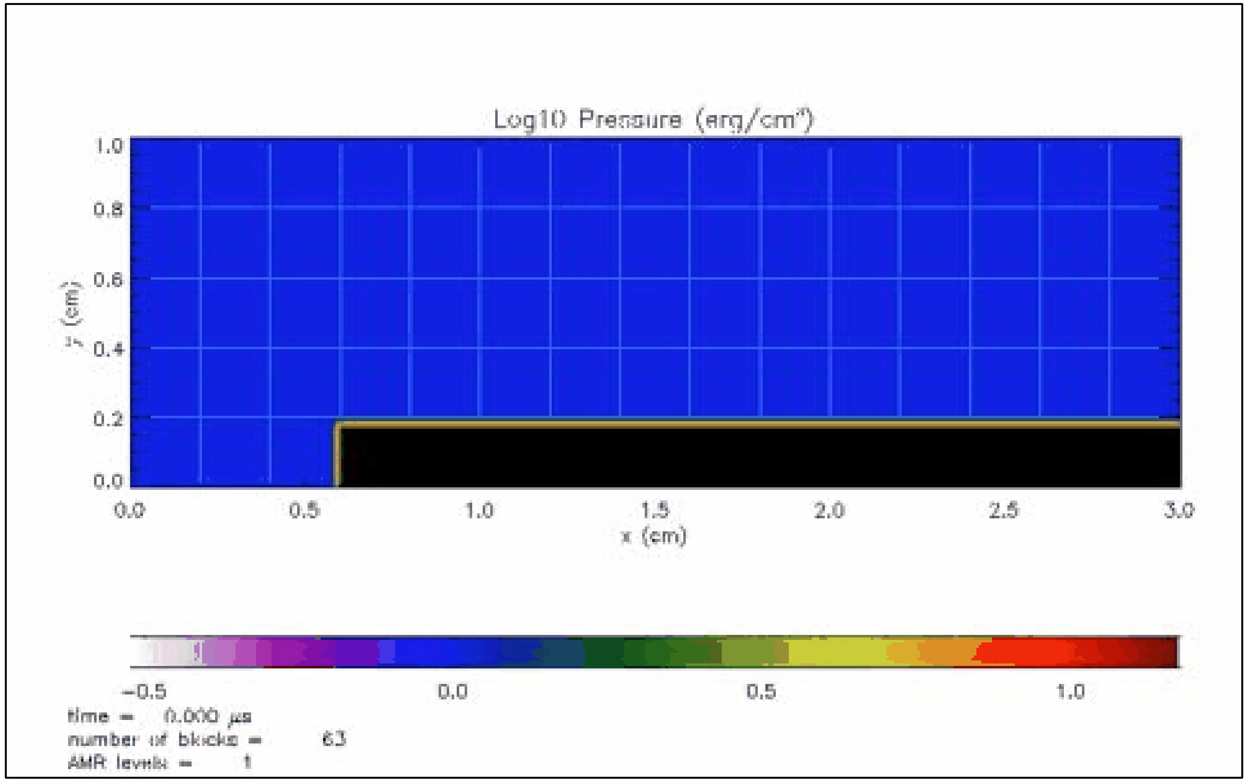
- High resolution required for handling difficult regions (discontinuities, steep gradients, shocks, etc.)
- Computationally extremely costly

Adaptive Mesh Refinement

- Start with a coarse grid
- Identify regions that need finer resolution
- Superimpose finer sub-grids only on those regions
- Increased computational savings over a static grid approach.
- Increased storage savings over a static grid approach.
- Complete control of grid resolution, compared to the fixed resolution of a static grid approach.

AMR makes it feasible to solve problems that are intractable on uniform grid

AMR - Applications



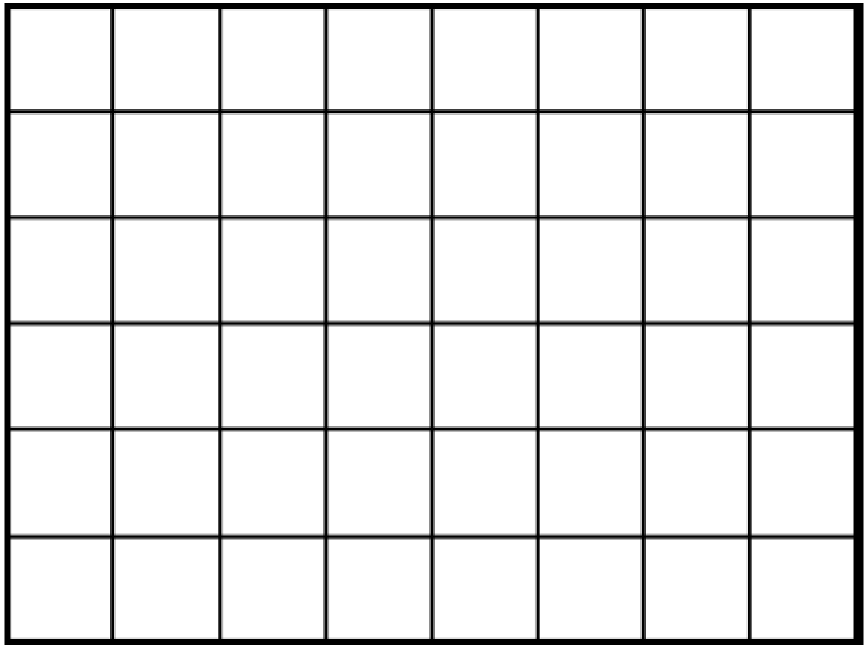
- CFD
- Astrophysics
- Climate Modeling
- Turbulence
- Mantle Convection Modeling
- Combustion
- Biophysics
- and many more

Demo of a Shock wave passing over a step function (wind tunnel with a step), rendered using the FLASH code.

Courtesy of the Univ. of Chicago, Flash Code group

AMR Techniques

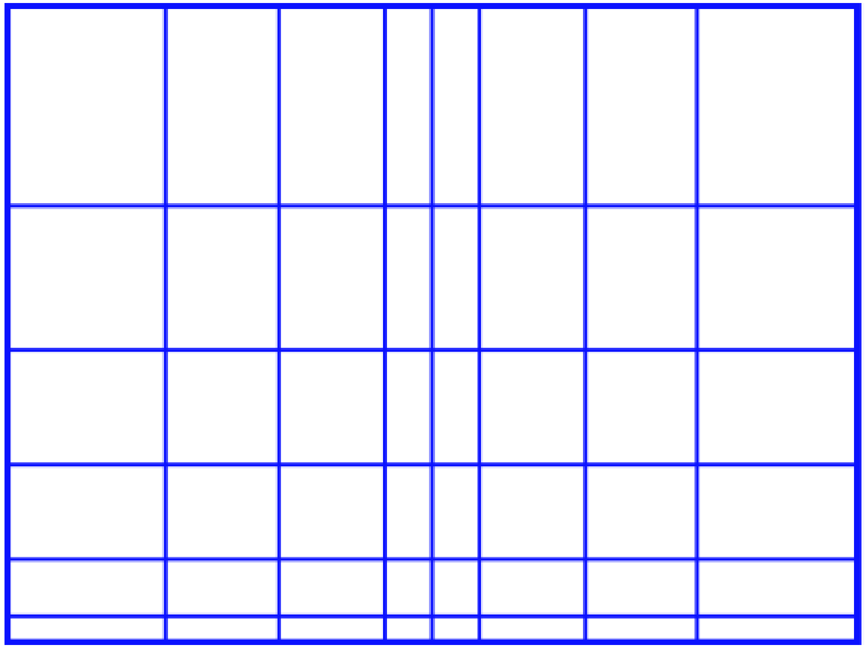
- mesh distortion



Courtesy of Dr. Andrea Mignone, University of Turin

AMR Techniques

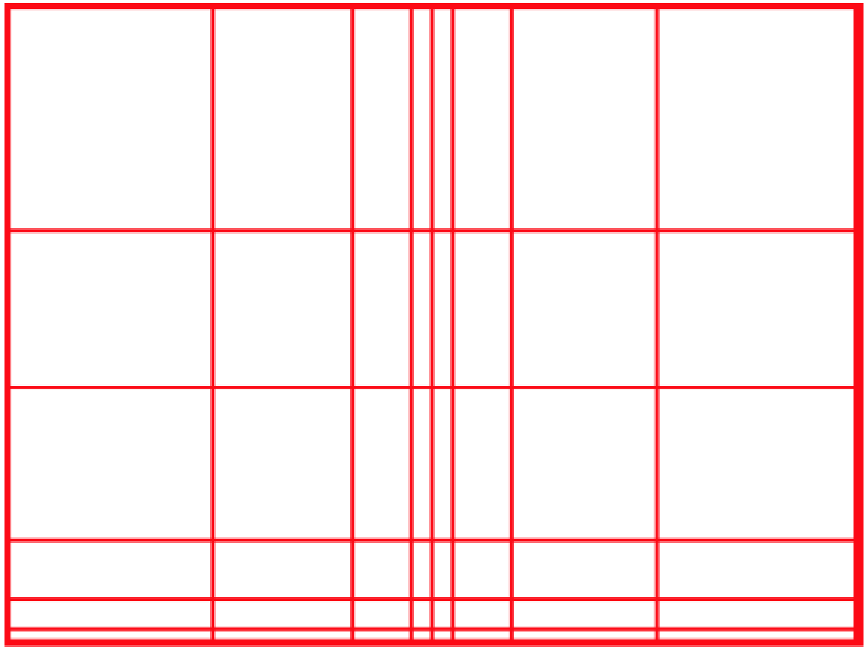
- mesh distortion



Courtesy of Dr. Andrea Mignone, University of Turin

AMR Techniques

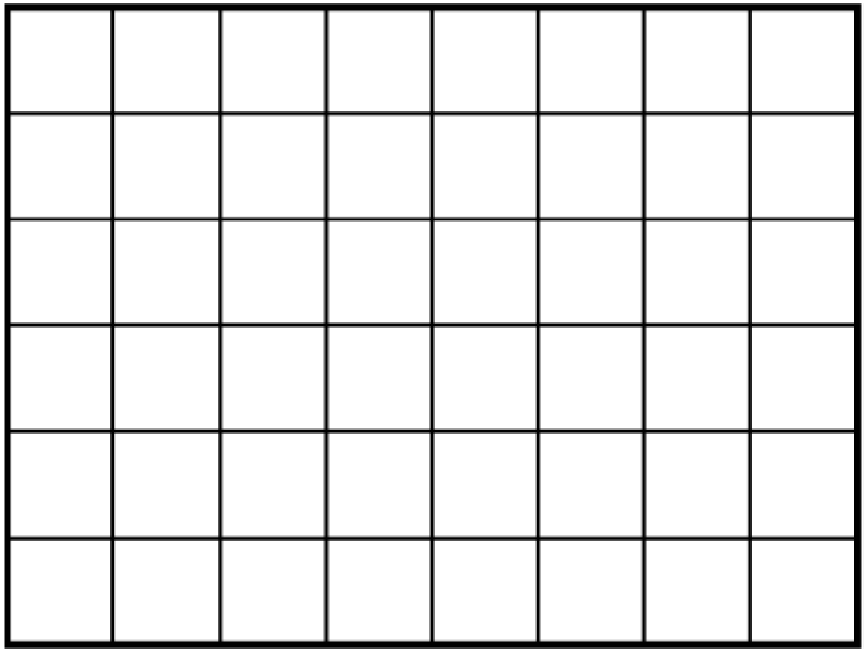
- mesh distortion



Courtesy of Dr. Andrea Mignone, University of Turin

AMR Techniques

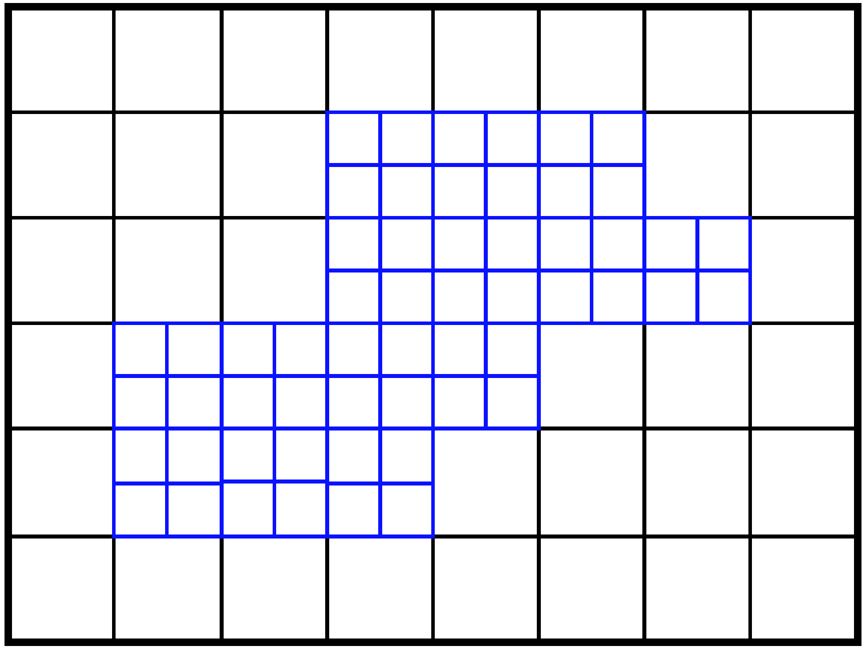
- mesh distortion
- point-wise structured refinement (tree-based)



Courtesy of Dr. Andrea Mignone, University of Turin

AMR Techniques

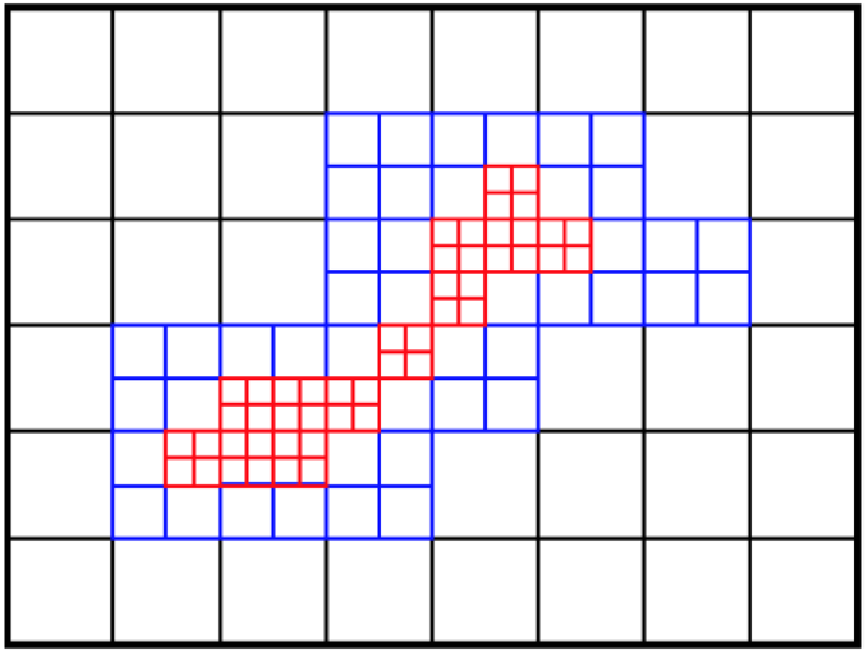
- mesh distortion
- point-wise structured refinement (tree-based)



Courtesy of Dr. Andrea Mignone, University of Turin

AMR Techniques

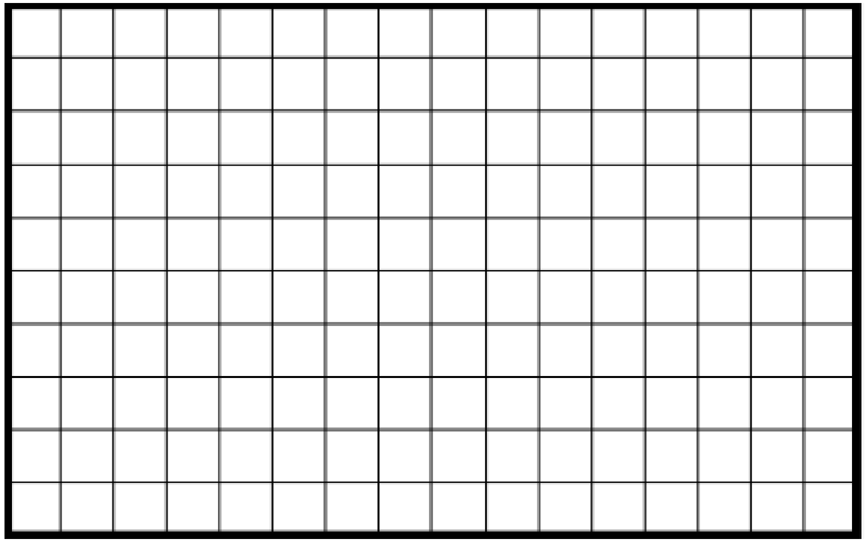
- mesh distortion
- point-wise structured (tree-based) refinement



Courtesy of Dr. Andrea Mignone, University of Turin

AMR Techniques

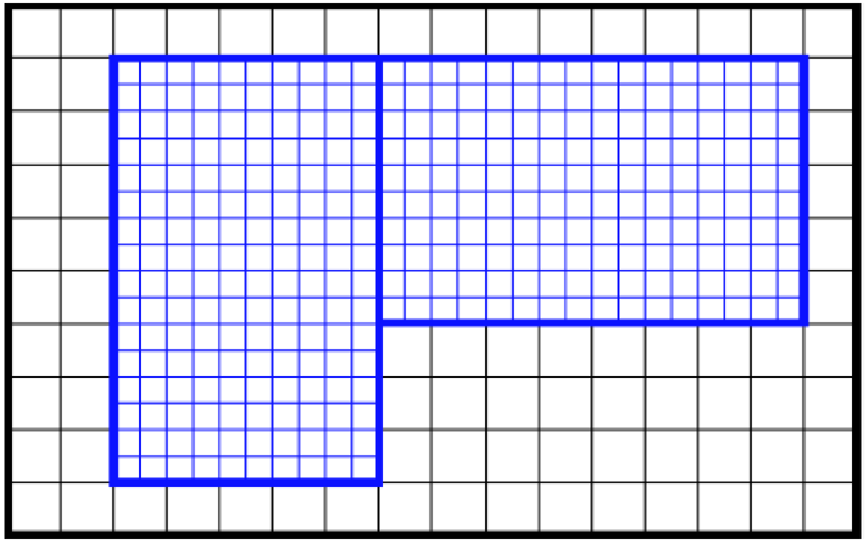
- mesh distortion
- point-wise structured (tree-based) refinement
- block structured



Courtesy of Dr. Andrea Mignone, University of Turin

AMR Techniques

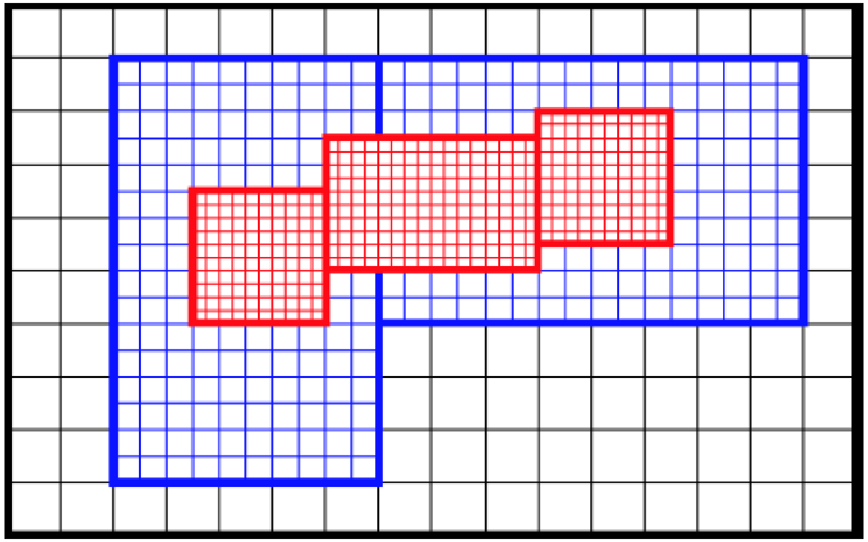
- mesh distortion
- point-wise structured (tree-based) refinement
- block structured



Courtesy of Dr. Andrea Mignone, University of Turin

AMR Techniques

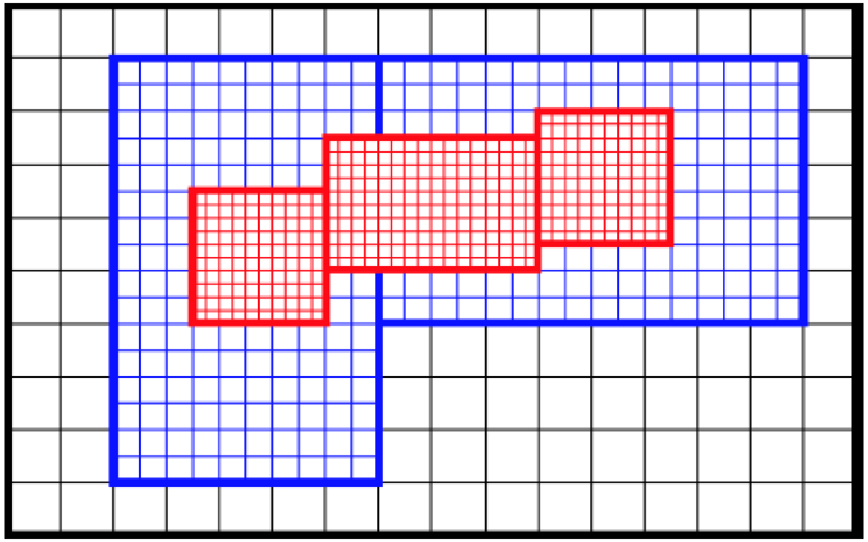
- mesh distortion
- point-wise structured (tree-based) refinement
- block structured:



Courtesy of Dr. Andrea Mignone, University of Turin

AMR Techniques

- mesh distortion
- point-wise structured (tree-based) refinement
- *block structured:*



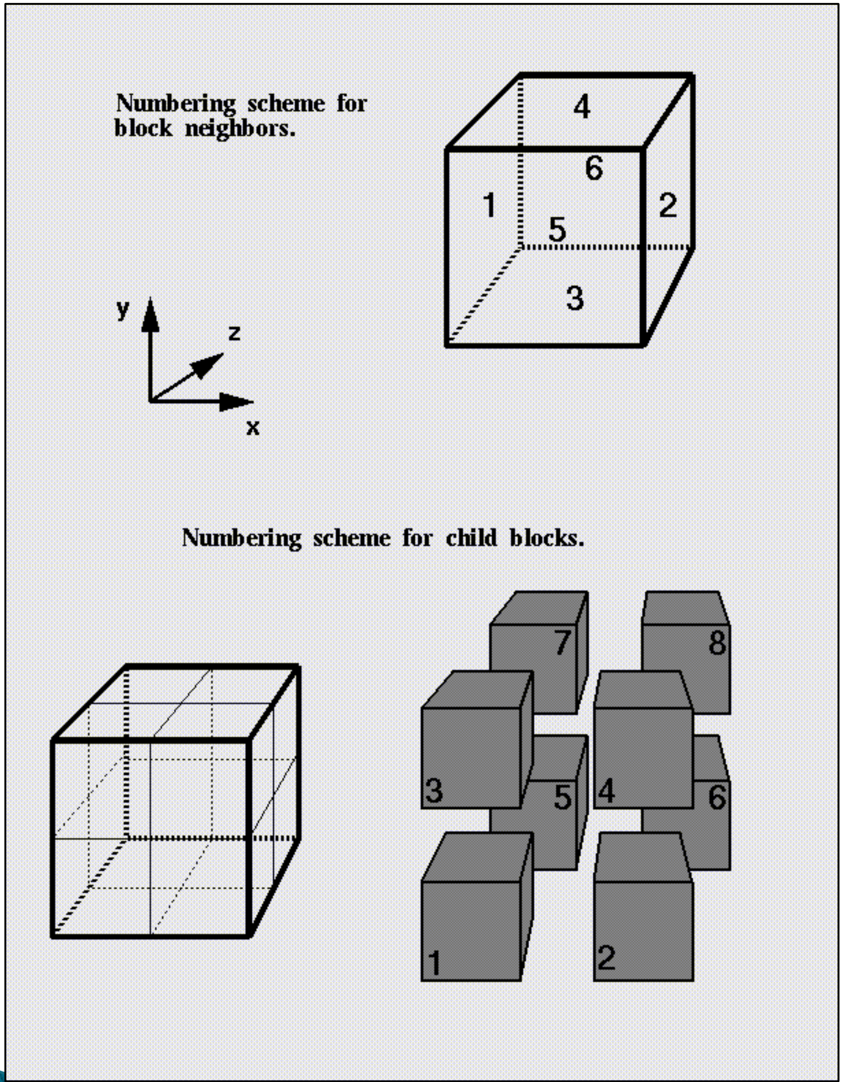
Courtesy of Dr. Andrea Mignone, University of Turin

- data blocks are created so that the same stencil can be used for all points and no special treatment is required.
- High level objects that encapsulate the functionality for AMR and its parallelization are independent of the details of the physics algorithms and the problem being solved.
- Simplifies the process of adding/replacing physics modules as long as they adhere to the interface requirements.

Existing Frameworks

- PARAMESH – http://www.physics.drexel.edu/~olson/paramesh
- SAMRAI – https://computation.llnl.gov/casc/SAMRAI/
- p4est – http://www.p4est.org/
- Chombo – https://commons.lbl.gov/display/chombo/Chombo
- and many more

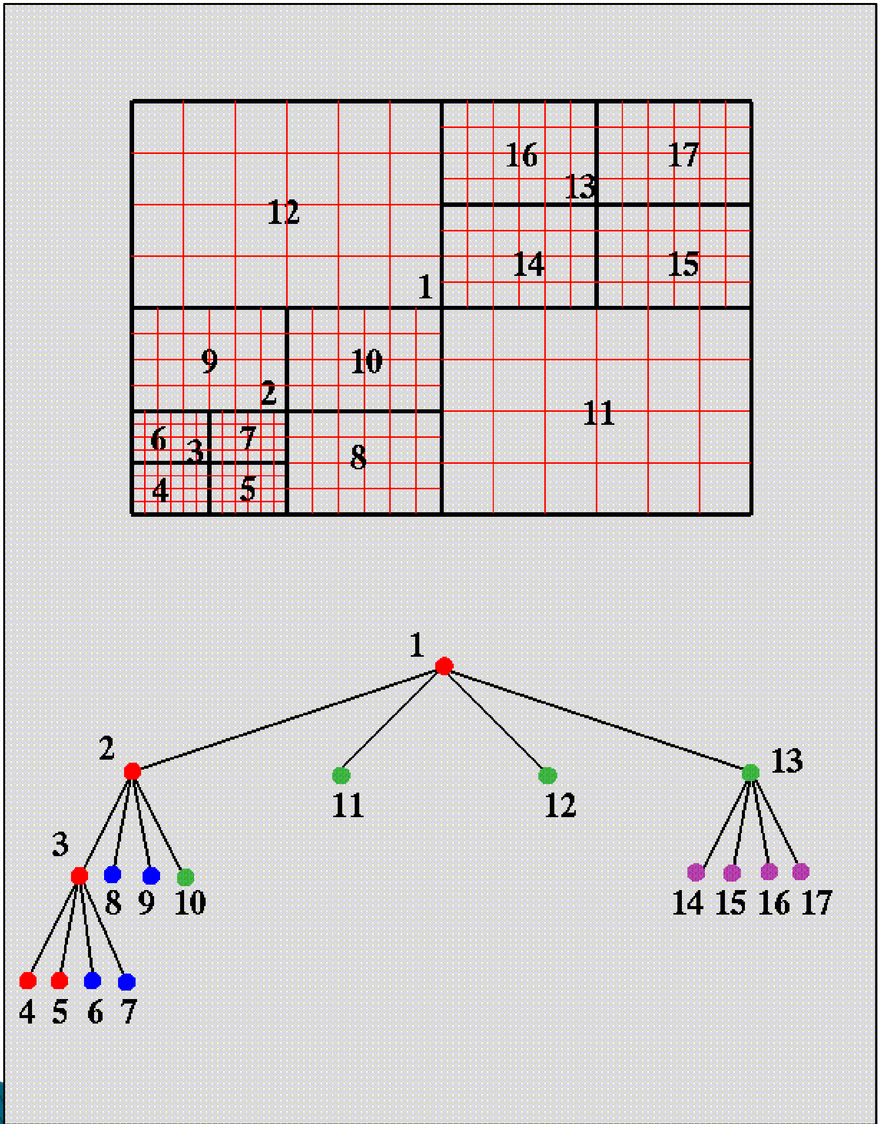
Block Numbering



From Paramesh User Guide

- All the grid blocks are related to one another as the nodes of a tree.
- The starting block is called root block, and the blocks with an higher resolution are called leaf blocks.
- When a leaf block is designated for refinement, it spawns 2 child blocks in 1D, 4 child blocks in 2D or 8 child blocks in 3D, and the original block is called mother (or parent) block.
- These child blocks cover the same physical line, area or volume as their parent but with twice the spatial resolution.
- Usually it is helpful to use a particular numbering algorithm (see next slides).

Typical grid hierarchy

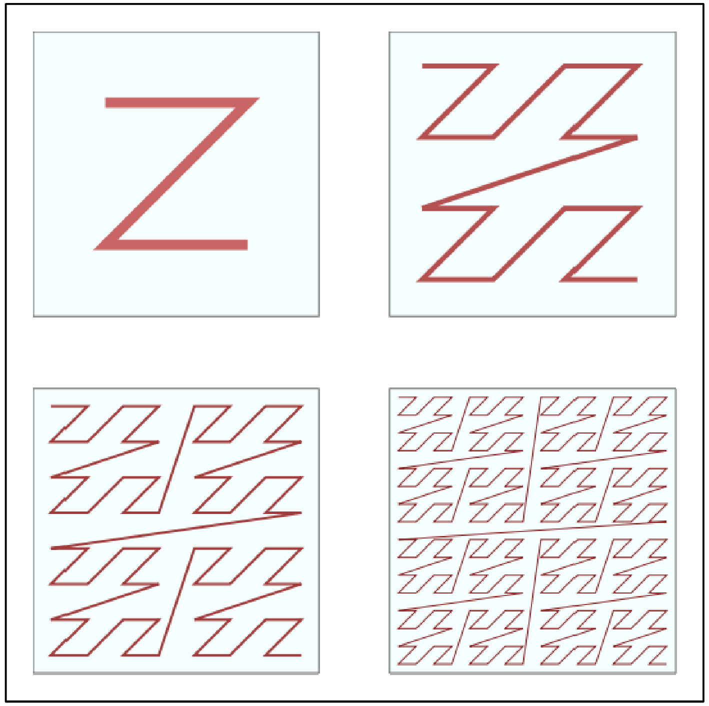


From Paramesh User Guide

- Each block has a fixed number of grid points
- Each block can be divided into 2^{ndim} sub-blocks
- Blocks are distributed between processes minimizing communications (see next slides)

An Example:
- 6 x 4 grid is created on each block
- The numbers assigned to each block designate the blocks location in the quad-tree
- The numbers assigned to each block designate the blocks location in the quad-tree

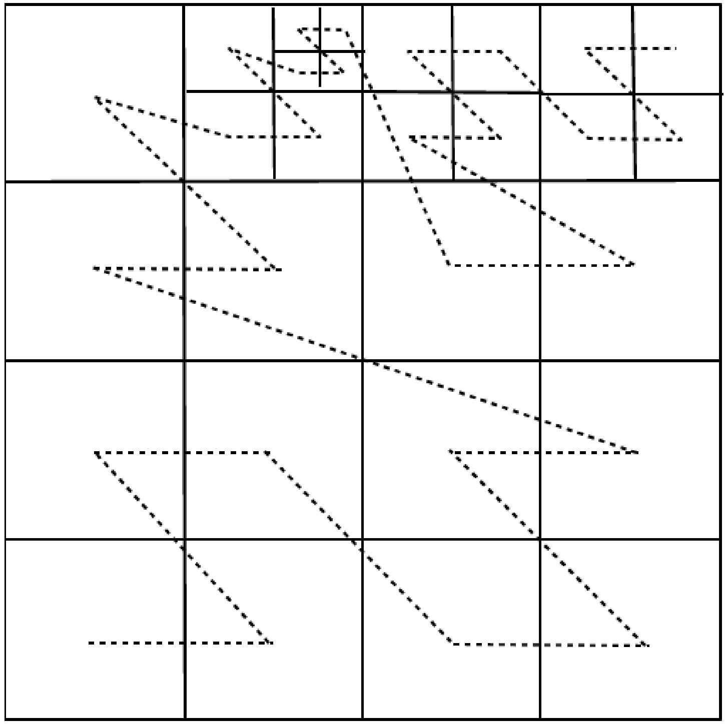
Block ordering

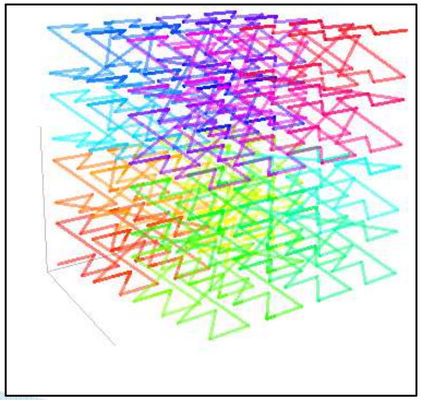


•Usually, the most used block ordering algorithm is Morton (or Z) ordering.

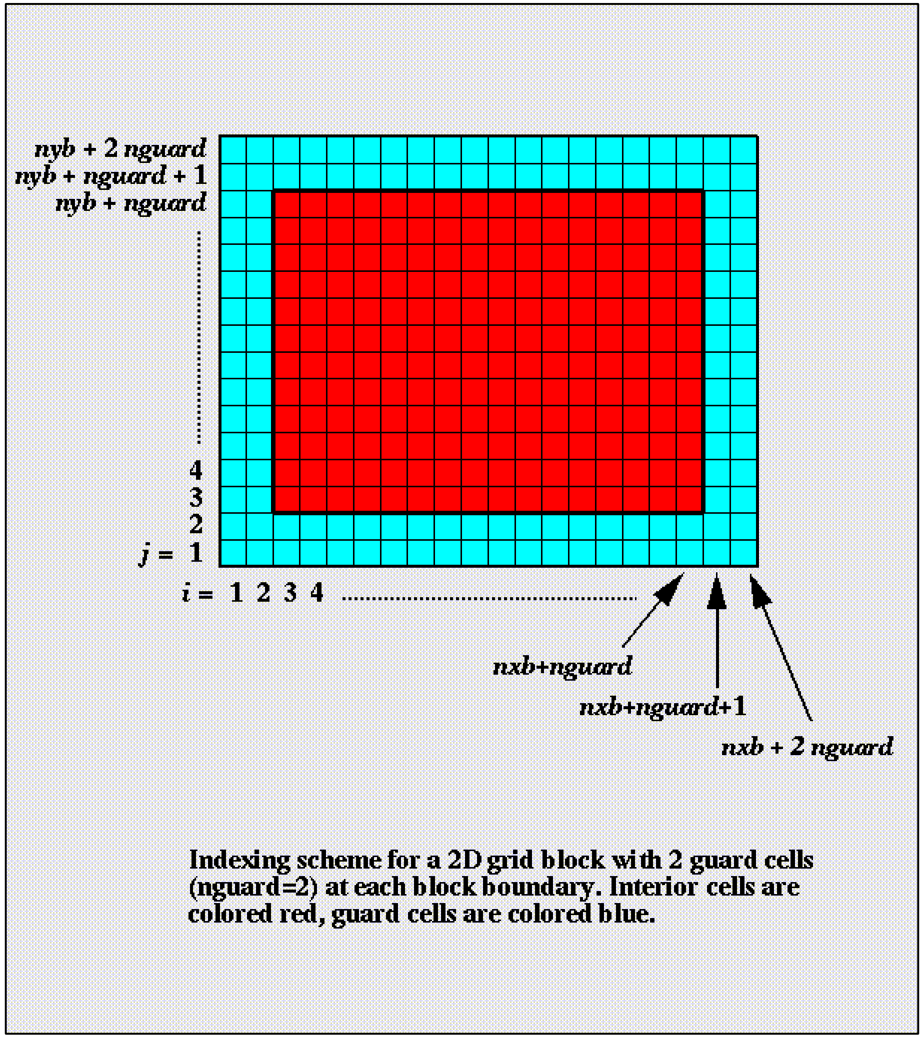
•It is particularly useful in order to:

- •Optimize the usage of cache memory;
- •Optimize ghost cells communications between process (see next slide);





Block Structure

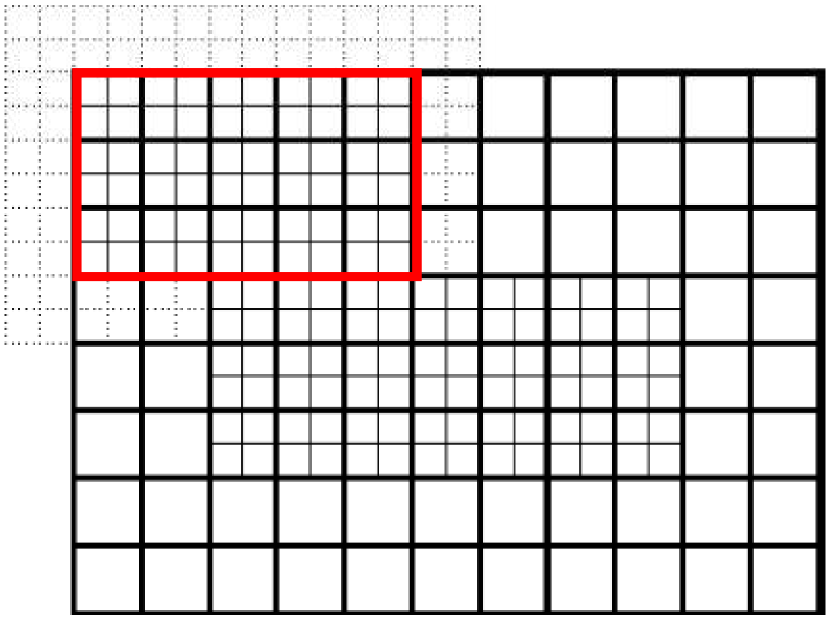


From Paramesh User Guide

Usually, each block is composed by:
•standard cells
• ghost cells
In Fortran, the indexes starts with 1 and ends with $N_{(X \text{ or } Y \text{ or } Z)}$ + 2*(number of ghost cells)
In C, the indexes starts whit 0 and ends $N_{(X \text{ or } Y \text{ or } Z)}$ + 2*(number of ghost cells) -1

Passing Ghost Cells

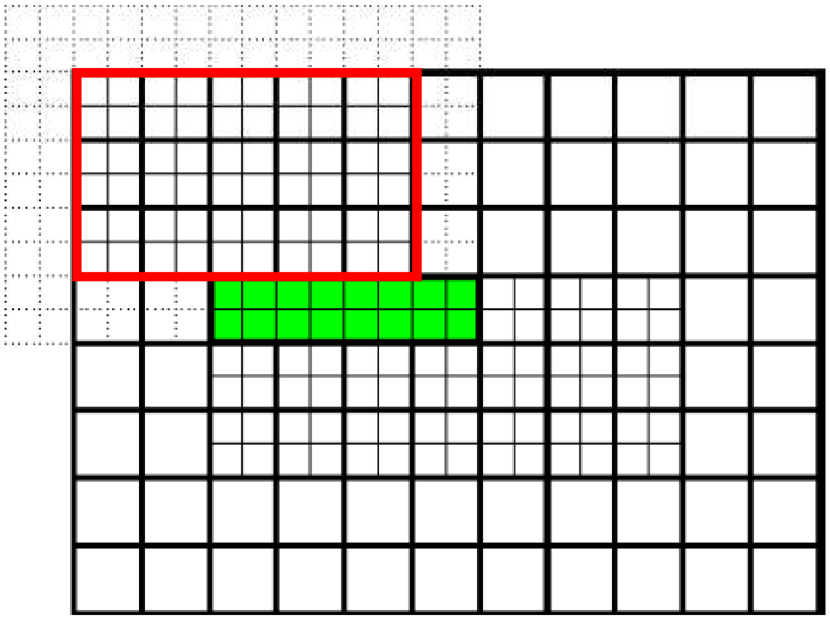
- ghost zones values need
  to be filled before integration;



Courtesy of Dr. Andrea Mignone, University of Turin

Passing Ghost Cells

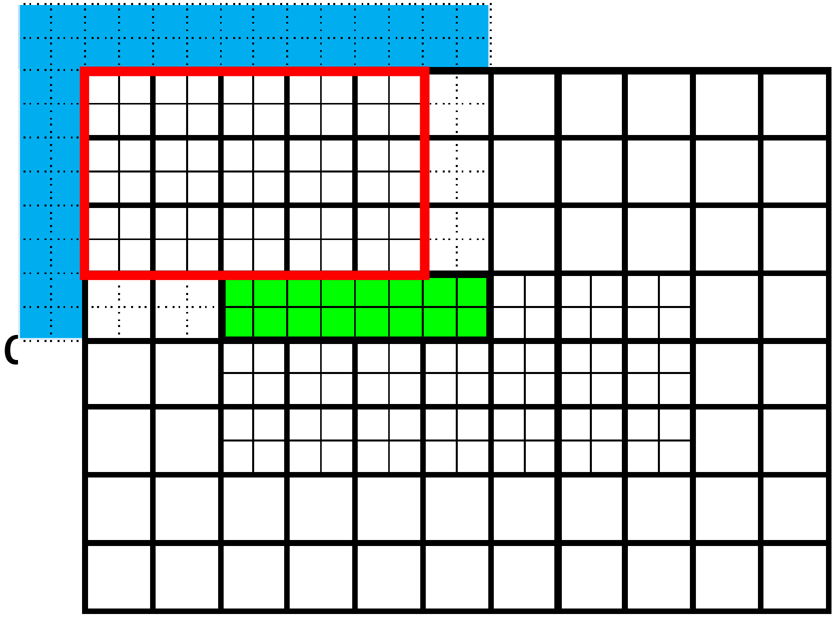
- ghost zones values need to be filled before integration;
- Patches at the same level are synchrhonized.



Courtesy of Dr. Andrea Mignone, University of Turin

Passing Ghost Cells

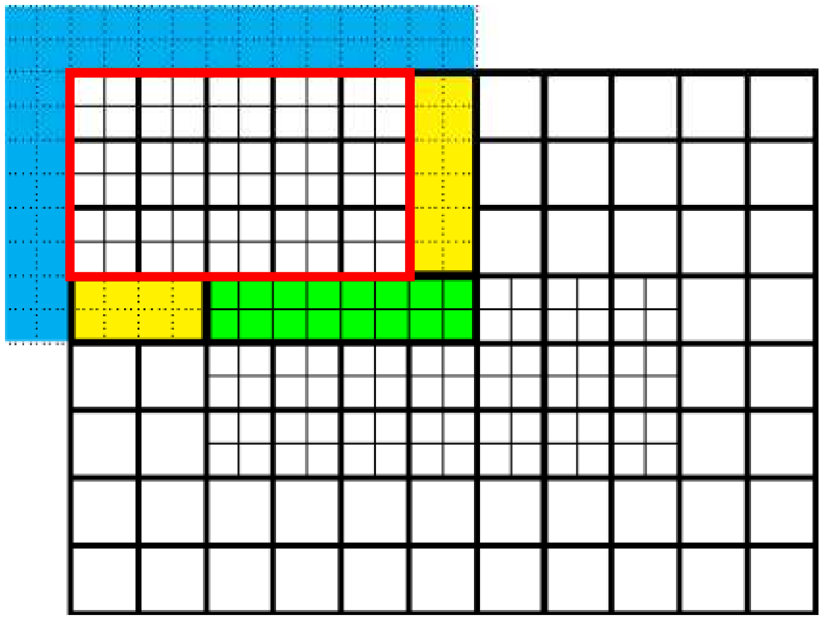
- ghost zones values need to be filled before integration;
- Patches at the same level are syncrhonized;
- Physical boundaries are imposed externally;



Courtesy of Dr. Andrea Mignone, University of Turin

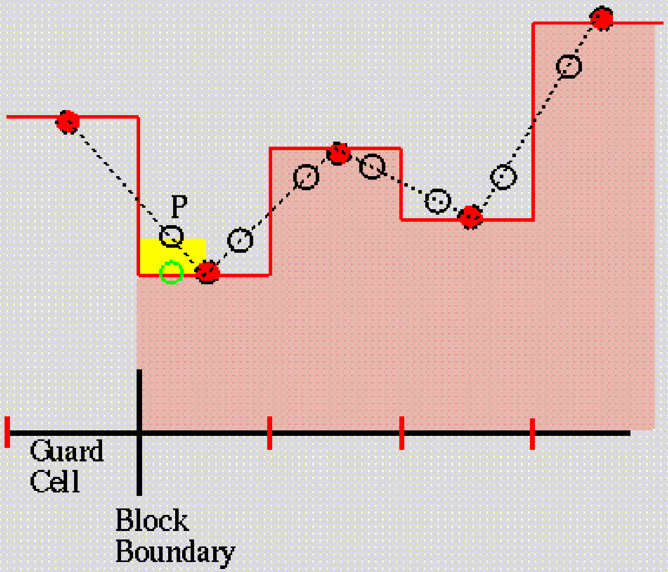
Passing Ghost Cells

- ghost zones values need to be filled before integration;
- Patches at the same level are synchronized;
- Physical boundaries are imposed externally;
- Fine-Coarse and Coarse-Fine interface need interpolation / averaging
- Integration proceeds as for the single-grid case



Courtesy of Dr. Andrea Mignone, University of Turin

Ghost cells communications



Non–conservative linear interpolation. The pink histogram and red circles indicate data on the parent grid block. The dashed line and empty circles show how linear interpolation would prolong this data onto a child grid with an even number of grid cells. The yellow area indicates the misconservation. Conservative interpolation replaces the point labeled P with the green circle so that the area under the curve is now the same as the pink shaded area.

From Paramesh User Guide

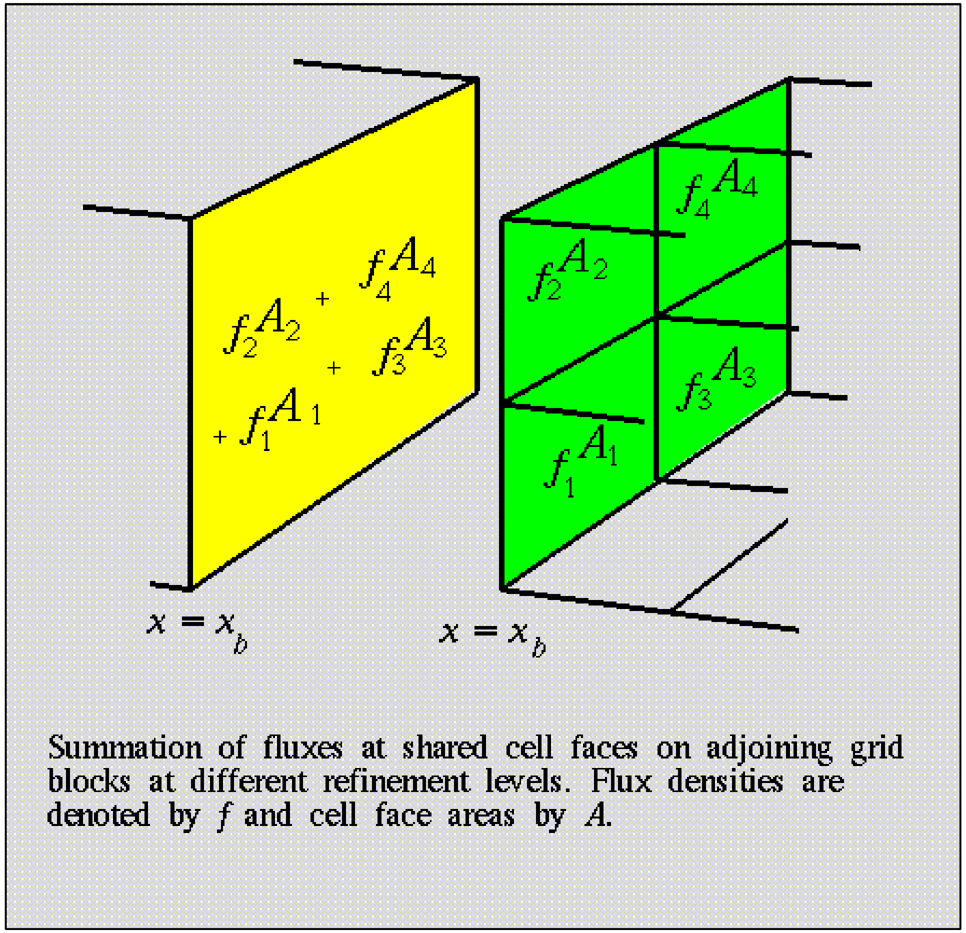
When we pass the ghost cells to the adjoining blocks, if these blocks have different resolutions we must modify the data.

The most simple (and used) method is the interpolation method:
- If we must pass the ghost cells to a block with higher resolution we can use the linear interpolation to artificially increase the resolution.
- If we must pass the ghost cells to a block with lower resolution we can average the data in order to have the same resolution.

Pros:
- Easy to implement
- It is possible to use many different kind of interpolation (linear, quadratic, and so on) increasing precision

Cons:
- Non-conservative

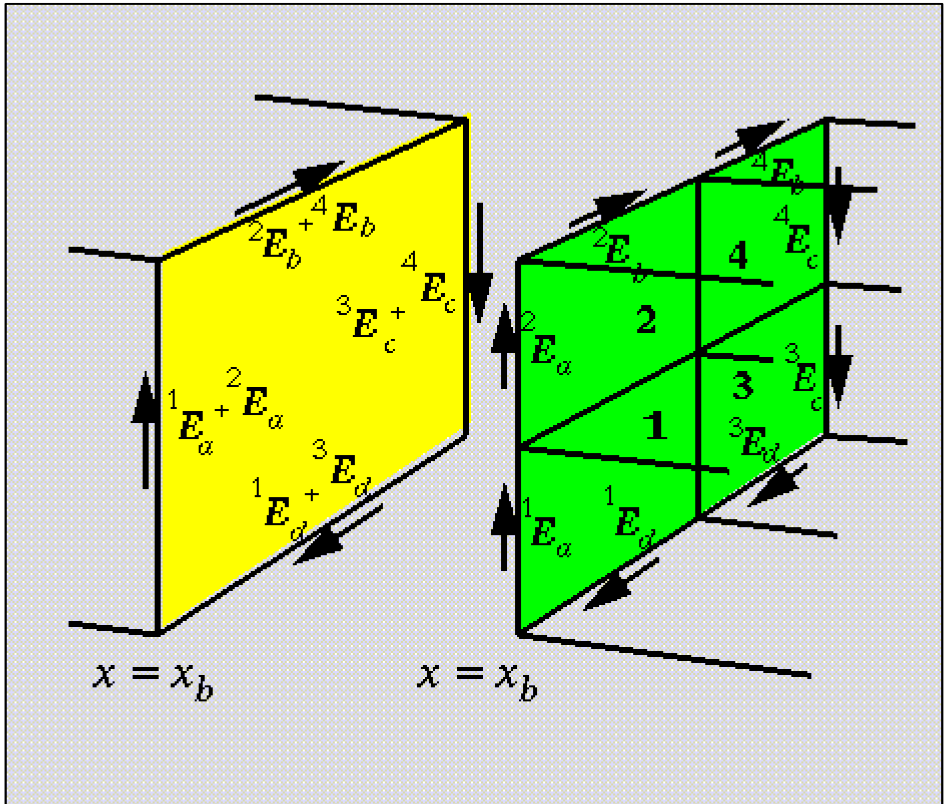
Passing ghost cells to neighbors blocks



From Paramesh User Guide

Flux conservation:
It is possible to ensure flux conservation after the interpolation checking the equation:
$f_1A_1+f_2A_2+f_3A_3+f_4A_4=F_{Tot}A_{Tot}$

Passing ghost cells to neighbors blocks



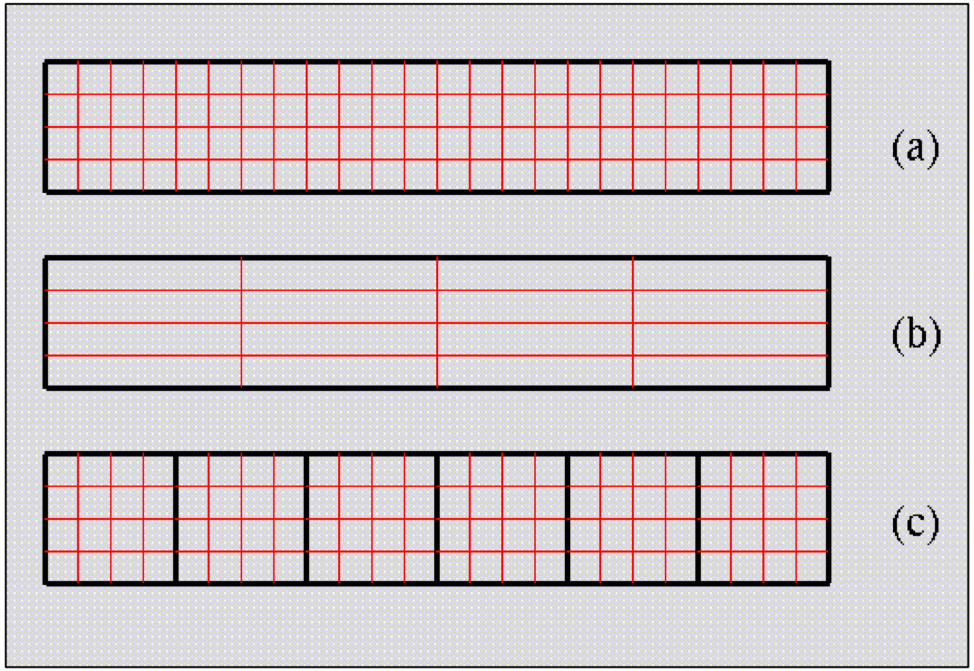
Edge based data on the shared cell face of adjoining grid blocks of different refinement level.

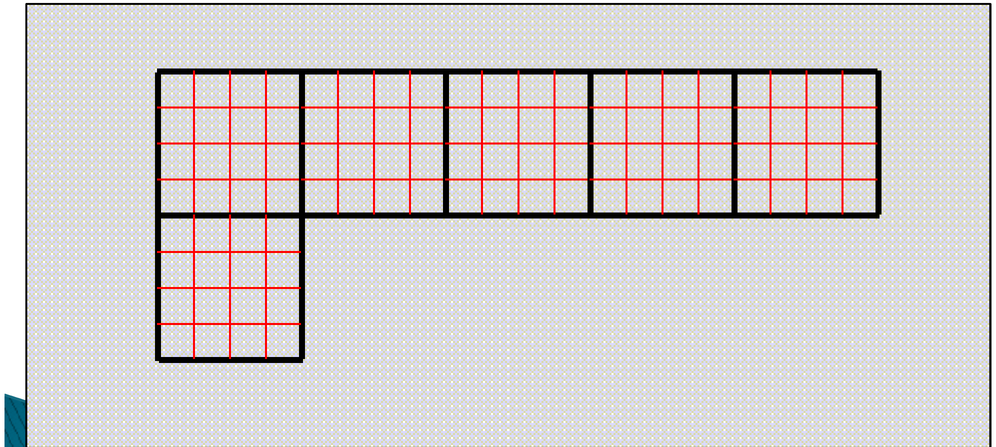
From Paramesh User Guide

Circulation integral control:
It is possible also to check the value of some physical quantity at the edges of the cells

NOTE: Both these three methods are usable in order to change the resolution of the blocks.

Particular Geometries





From [Paramesh User Guide]

When we have a non symmetric computational domain many different approach can be used. For a rectangular domain:

•We can have different number of points per block on x and y directions (dx = dy)

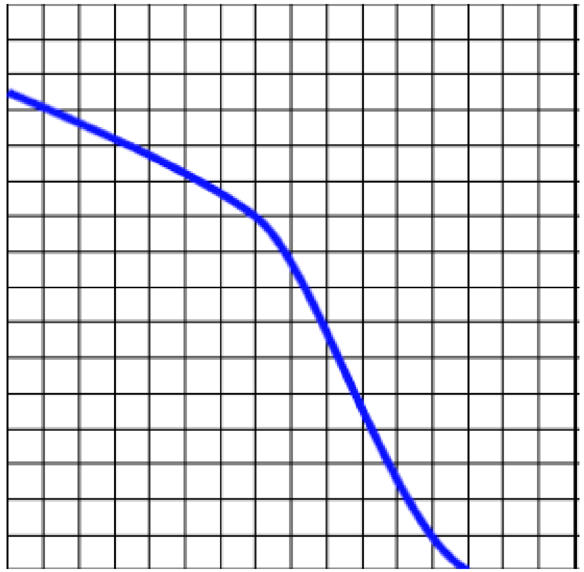
•We can have different number of points on x and y directions (dx ≠ dy)

•We can use more blocks on the x directions , and 1 block on x direction (same resolution on x and y, and more parallelizable)

If we have more complicate computational domains, we can always use more blocks in order to fully cover the whole domain.

How to refine

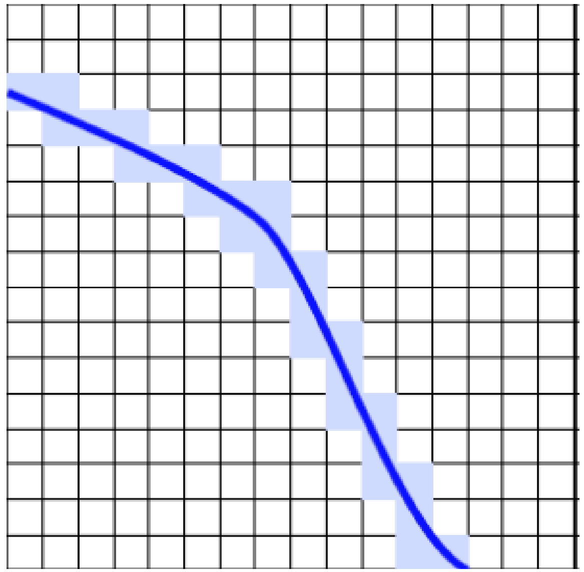
- fill data, level 0



Courtesy of Dr. Andrea Mignone, University of Turin

How to refine

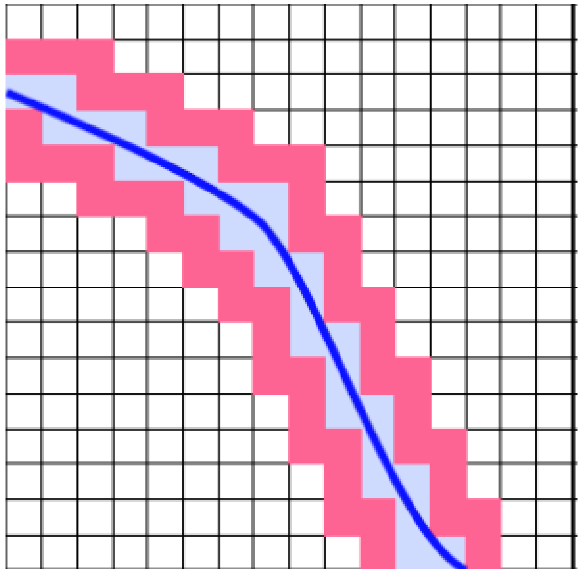
- fill data, level 0
- find where refinement is needed;



Courtesy of Dr. Andrea Mignone, University of Turin

How to refine

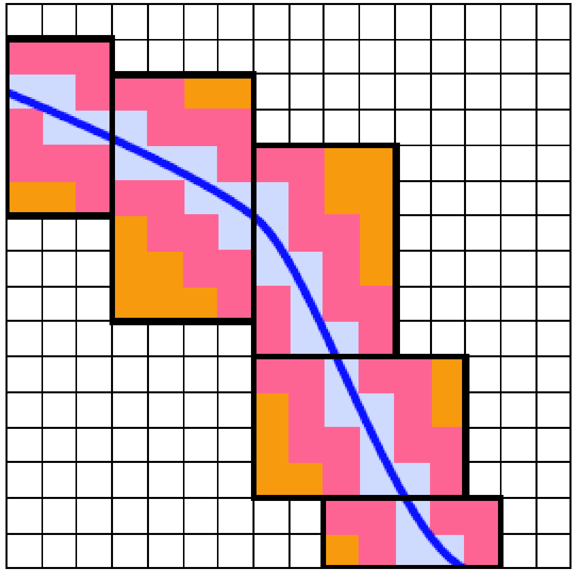
- fill data, level 0
- find where refinement is needed;
- group cells into patches according to the "grid efficiency"



Courtesy of Dr. Andrea Mignone, University of Turin

How to refine

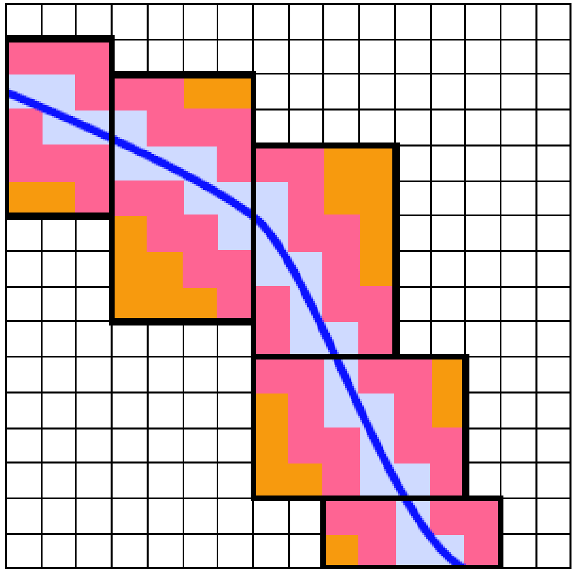
- fill data, level 0
- find where refinement is needed;
- group cells into patches according to the "grid efficiency"
- refine and ensure proper nesting



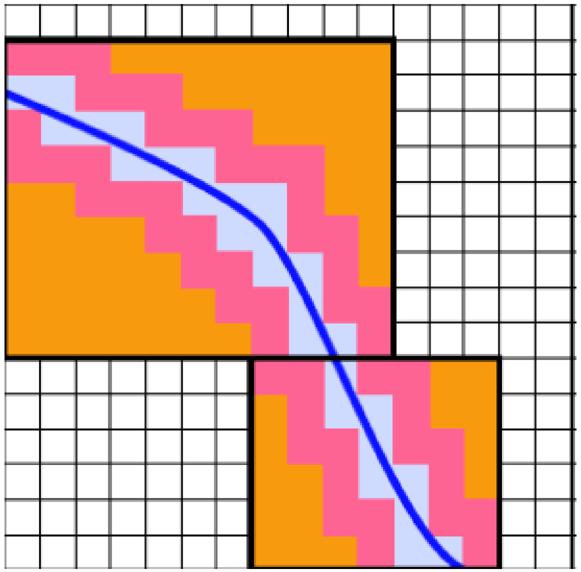
Courtesy of Dr. Andrea Mignone, University of Turin

How to refine

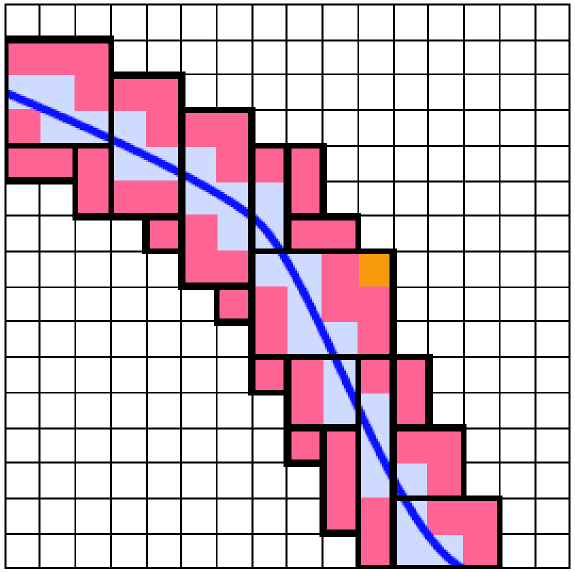
- fill data, level 0
- find where refinement is needed;
- group cells into patches according to the “grid efficiency”
- refine and ensure proper nesting



efficiency = 0.7



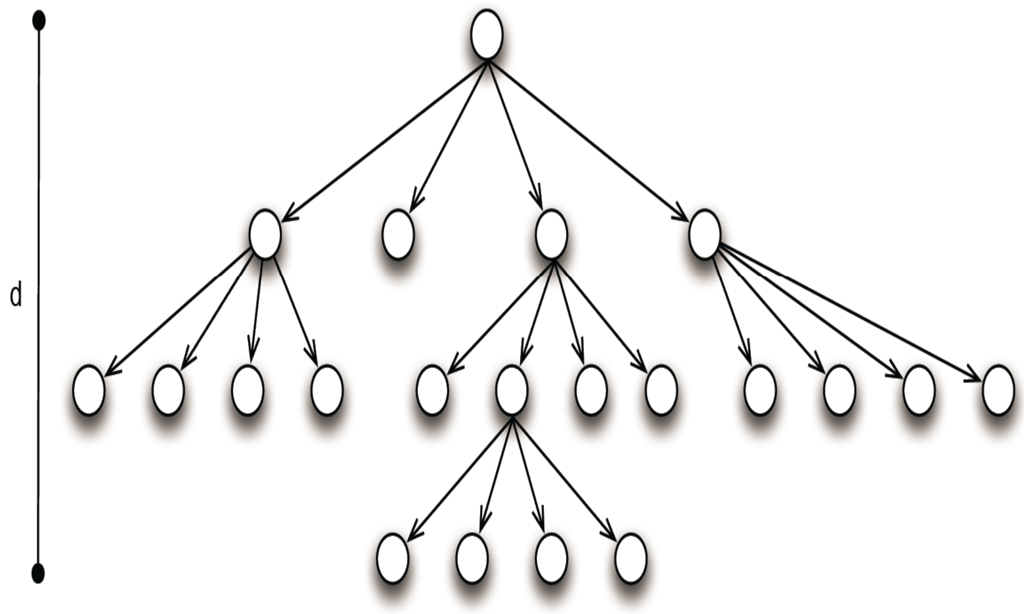
efficiency = 0.5



efficiency = 0.9

Courtesy of Dr. Andrea Mignone, University of Turin

Little more background on AMR



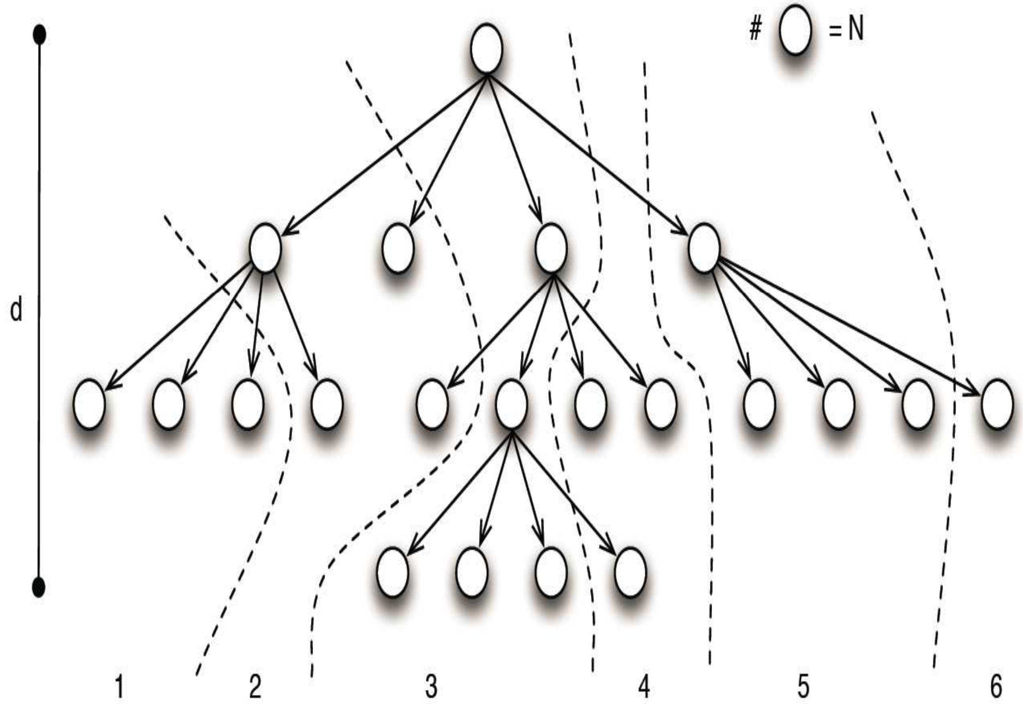
Refinement structure can be represented using a quad-tree (2D)/ oct-tree (3D)

An important condition in AMR

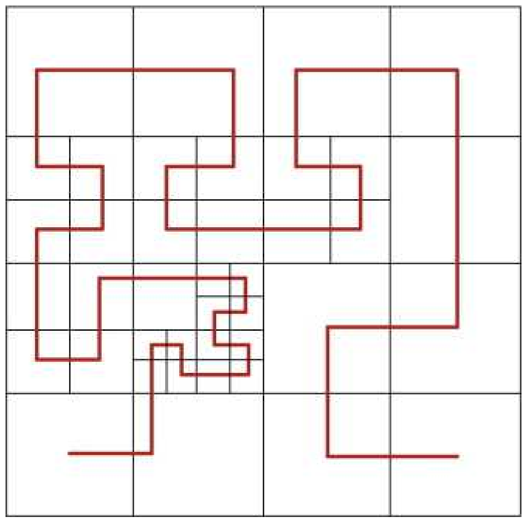
Refinement levels of neighboring blocks differ by ±1

Note: This is generally true, but Chombo library allow more than 1 refinement level discrepancy.

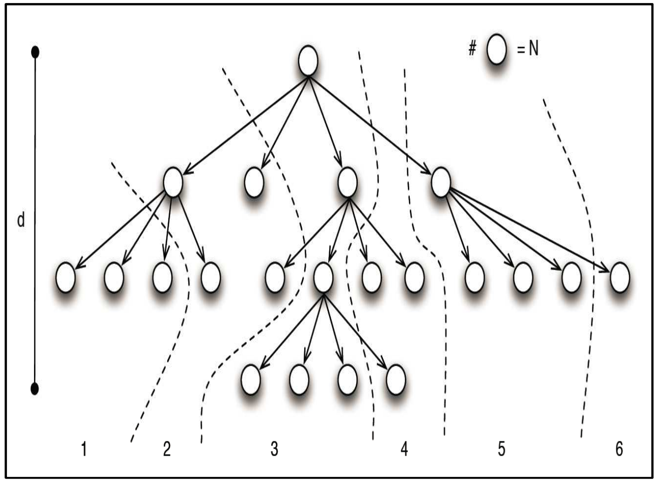
Traditional Approach - Parallel Implementations



- A set of blocks assigned to a process
- Use space-filling curves for load balancing



Traditional Approach - Disadvantages



- Adaptive mesh restructuring:
  - Tree metadata replicated on each process
    - ✓Required memory increases with # of cores
    - ✓Memory can became a problem if we use more than 10^5 cores (and more than 10^6 boxes)
  - Level-by-level restructuring
    - Ripple propagation
    - Step needed to propagate restructuring ∝ level of refinement (d)

- Load Balancing
  - Memory needed ∝ Number of blocks used
  - Time needed ∝ Number of blocks used

- Currently for 3D problems with less than 10^6 boxes standard AMR library scales up to few tens of thousands of cores
- This is a serious problem considering that next generation supercomputers will require the use of many hundreds of thousands of cores

Improving AMR: Possible strategies

1. Compress tree metadata
   - Already implemented in the last versions of CHOMBO, PARAMESH and SAMRAI libraries
2. Rewrite the algorithm for coarse-fine interpolation in order to minimize communications
   - Already implemented in the last versions of CHOMBO, PARAMESH libraries
   - Using these first two methods it is possible to scale up to 2×10^5 cores using 10^7 grid cells
3. Use a distributed memory version for tree metadata
   - Currently Langer at al are working on the implementation of this algorithm on CHARM++

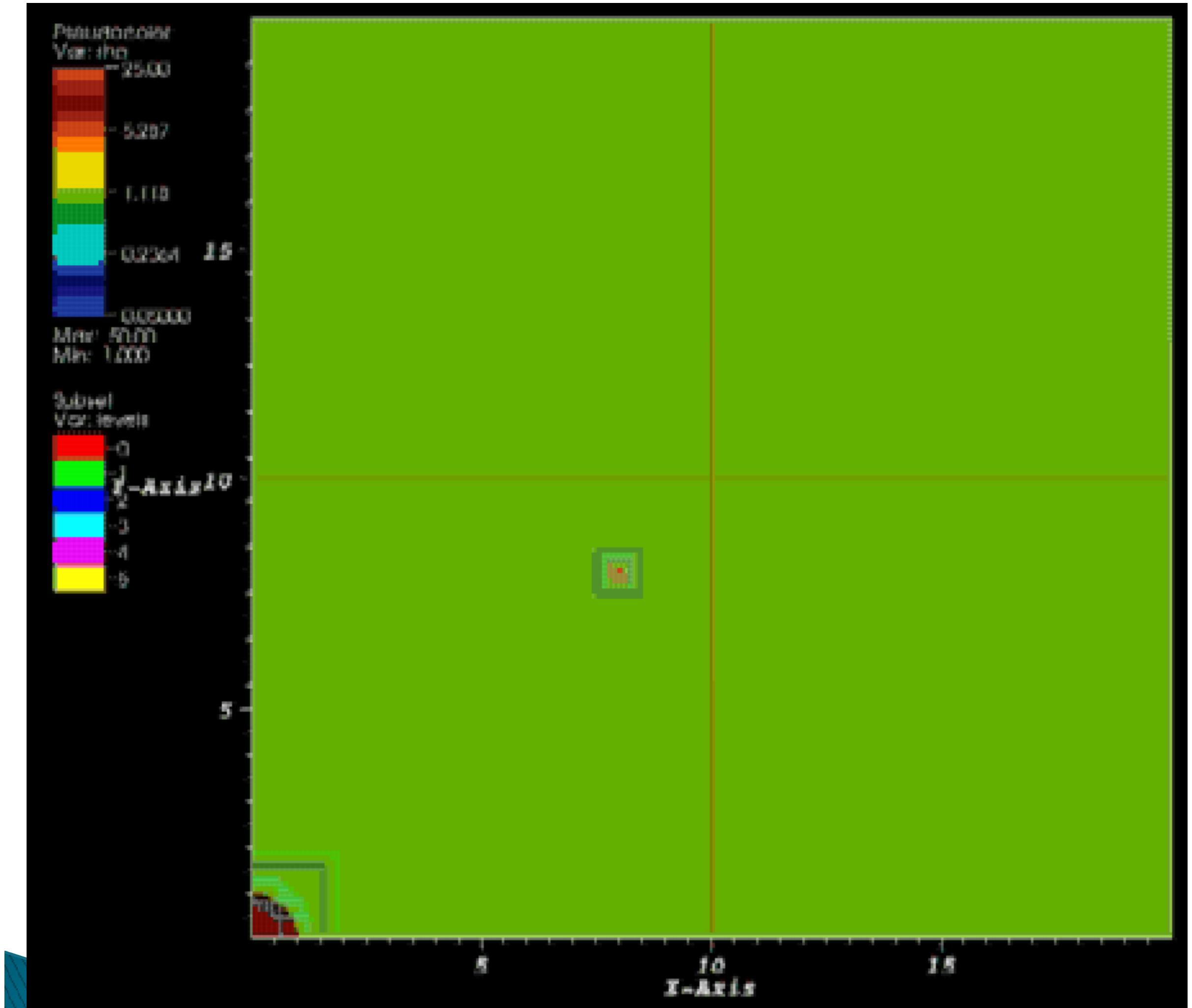
Some additional information about PARAMESH

- Written in Fortran 90
- Easy to implement on a existing code
- Support many geometries (Cartesian, cylindrical, spherical, from 1D to 3D)
- Refinement levels of neighbouring blocks differ by ±1
- Compatible with hdf5 format
- Some simple routine are already written by the authors of the library in order to save the data and the grid structure into Fortran binary format, and hdf5 format.
- Easy visualization of the results using many external programs (e.g. visit)

Some additional information about CHOMBO

- •Written in C
- •Easy to implement on a existing code
- •Support many geometries (Cartesian, cylindrical, spherical, from 2D to 3D)
- •Compatible with hdf5 format
- •Easy visualization of the results using many external programs (e.g. visit)

Example: 2D Blast Wave



Courtesy of Dr. Andrea Mignone, University of Turin

__Problem:__ Blast Wave – Cloud Interaction

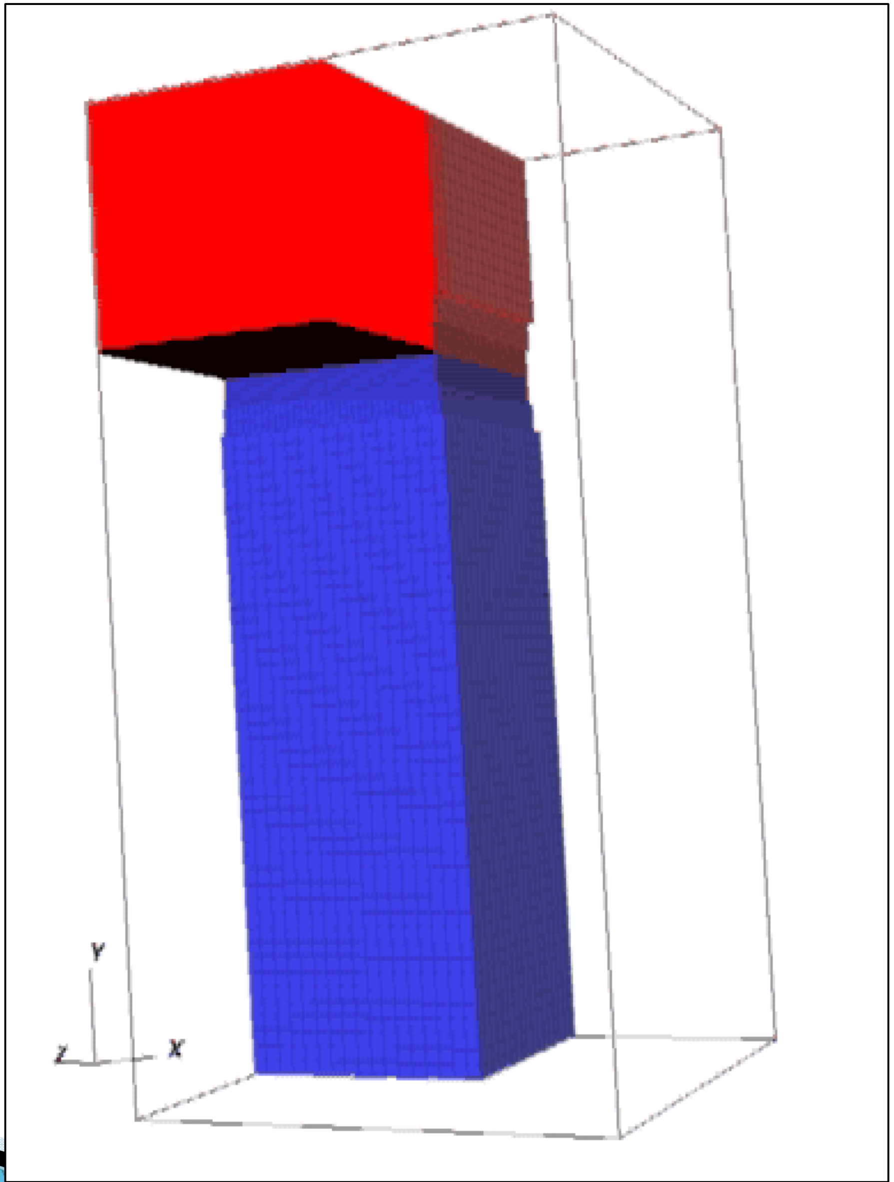
__Base Grid:__ 128x128

__Levels of Refinement:__ 5 (eq. 4096x4096)

__Method:__ Unsplit PPM

__Code:__ PLUTO + Chombo Lib

Example: 3D Rayleigh-Taylor



Problem:
Rayleigh Taylor

Base Grid:
32x64x32

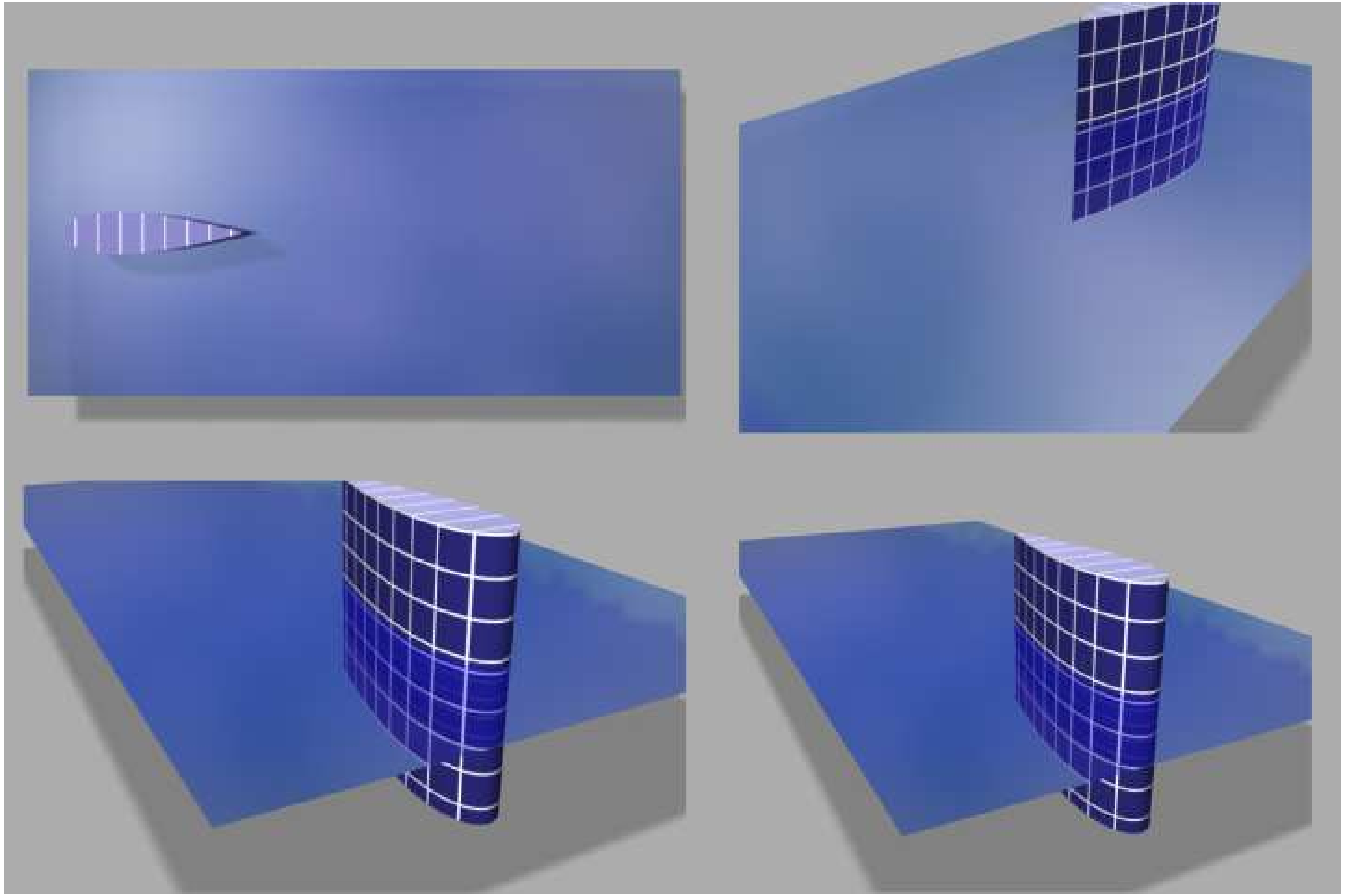
Levels of Refinement:
2 (eq. 128x256x128)

Method:
Unsplit PPM

Code:
PLUTO + Chombo Lib

Courtesy of Dr. Andrea Mignone, University of Turin

Example: 3D INCOMPRESSIBLE FLUID FLOW – Breaking waves due to a ship's hull.



From Paramesh website. The movie is courtesy of Douglas Dommeruth (SAIC).

Thank you for attention

SuperComputing Applications and Innovation

An introduction to Adaptive Mesh Refinement (AMR) - Part 2: A very short tutorial about PARAMESH

HPC Numerical Libraries
26–28 April 2016
CINECA – Casalecchio di Reno (BO)

Massimiliano Guarrasi- m.guarrasi@cineca.it
Super Computing Applications and Innovation Department

Step 1: How to install

- Downolad the source code from: http://downloads.sourceforge.net/project/paramesh/PARAMESH/paramesh_4.1/paramesh_4.1.tar.gz
  - On PICO use the *wget* command
- Uncompress the source files:
  - *tar -xzvf paramesh_4.1.tar.gz*
- Enter in the main source directory:
  - *cd paramesh_4.1*
- Load MPI module:
  - *module load autoload intelmpi/5.1.1--binary*
- Edit the Makefile.gnu file:
  - *kate Makefile.gnu*
  - Comment row 51 (NAG specific compilation commands)
  - Uncomment row 58 (Intel specific compilation commands)
  - Save and close the file
- Compile the source code:
  - *gmake –f Makefile.gnu*

Step 2: Our problem

Solve

$$\frac{d}{dt}U(x,y,t) = \kappa\left(\frac{\partial^2}{\partial x^2} - \frac{\partial^2}{\partial y^2}\right)U(x,y,t),$$

in the domain

$$x| \leq 4, |y| \leq 4$$

with

$$\kappa - 1,$$

with timestep

$$\Delta t = \frac{1}{10}\frac{\min(\Delta x^2, \Delta y^2)}{\kappa}$$

and initial conditions

$$U(x,y,0) = \begin{cases} 10.0 & \text{if } |x| \leq 1 \text{ and } |y \leq 1 \\ 1.0 & \text{otherwise} \end{cases}$$

•Our numerical scheme (4-pt centered second order accurate difference method):

*U(i,j,t+dt) = U(i,j,t) + dt * A / (dx*dx)*

*A = U(i+1,j,t) + U(i-1,j,t) + U(i,j+1,t) + U(i,j-1,t) - 4*U(i,j,t)*

Step 3: Create the files

- Preliminary steps:
  - Create a subdirectory inside PARAMESH main directory (*AMRDIR* from now) named *your_tutorial*
  - Copy the file *AMRDIR/templates/amr_main_prog_template.F90* into the current directory and rename it *tutorial.F90*
  - Copy the file *AMRDIR/templates/amr_1blk_bcset_template.F90* into the current directory and rename it *amr_1blk_bcset.F90*

mguarra1@node066:~/PARAMESH/test/paramesh_4.1/your_tutorial

File Edit View Search Terminal Help

```
mguarra1@node066.pico:[paramesh_4.1]$
mguarra1@node066.pico:[paramesh_4.1]$
mguarra1@node066.pico:[paramesh_4.1]$
mguarra1@node066.pico:[paramesh_4.1]$
mguarra1@node066.pico:[paramesh_4.1]$
mguarra1@node066.pico:[paramesh_4.1]$
mguarra1@node066.pico:[paramesh_4.1]$
mguarra1@node066.pico:[paramesh_4.1]$
mguarra1@node066.pico:[paramesh_4.1]$
mguarra1@node066.pico:[paramesh_4.1]$
mguarra1@node066.pico:[paramesh_4.1]$
mguarra1@node066.pico:[paramesh_4.1]$
mguarra1@node066.pico:[paramesh_4.1]$
mguarra1@node066.pico:[paramesh_4.1]$ mkdir your_tutorial
mguarra1@node066.pico:[paramesh_4.1]$ cd your_tutorial/
mguarra1@node066.pico:[your_tutorial]$ cp ../templates/amr_main_prog_template.F90 tutorial.F90
mguarra1@node066.pico:[your_tutorial]$ cp ../templates/amr_1blk_bcset_template.F90 amr_1blk_bcset.F90
mguarra1@node066.pico:[your_tutorial]$ ls
amr_1blk_bcset.F90  tutorial.F90
mguarra1@node066.pico:[your_tutorial]$
```

Step 4: Modify paramesh_preprocessor.fh (old version)

- Edit the header file paramesh_preprocessor.fh
  - cd to *AMRDIR/headers*
  - Edit *paramesh_preprocessor.fh*
    - If you want to use double precision then define REAL8:

      #define REAL8
  - Comment out the following preprocessor definitions (none of these features will be used in this example):

    !#define VAR_DT
    !#define PRED_CORR
    !#define EMPTY_CELLS
  - Define the preprocessor variable DIAGONALS (used only during the test phase in this case):

    #define DIAGONALS
- Set the model dimensionality to 2 by setting

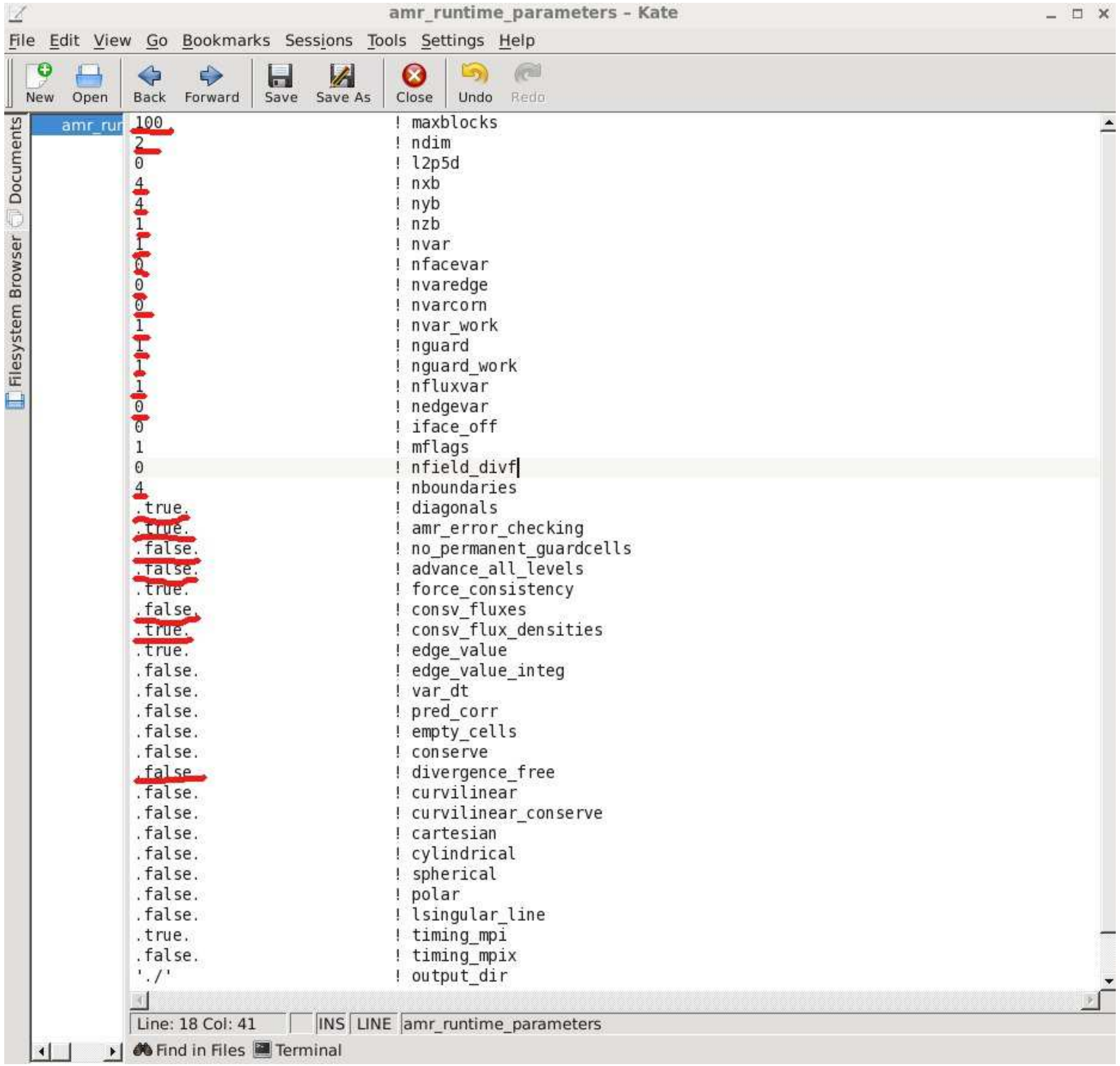
  #define N_DIM 2
- Leave CURVILINEAR undefined since we are using cartesian coordinates in the tutorial.
- Comment out the following preprocessor definitions since none of these features will be used in this example.

  !#define NO_PERMANENT_GUARDCELLS
  !#define ADVANCE_ALL_LEVELS
- Make the following definitions to set up the case we want to run. In order, these settings establish the grid blocks as 4x4, allow up to 100 blocks on each processor, establish 1 cell centered variable and 0 cell-face-centered variables, 0 edge-dentered variables, 0 corner-centered variables, and set 1 layer of guard cells at each block boundary.

```
#define NX_B 4
#define NY_B 4
#define MAX_BLOCKS 100
#define N_GUARD_CELLS 1
#define N_GUARD_CELLS_WORK 1
#define N_VAR 1
#define N_FACEVAR 0
#define N_VAR_EDGE 0
#define N_VAR_CORN 0
#define N_VAR_WORK 1
#define N_FLUX_VAR 1
#define N_EDGE_VAR 0
```

Step 4: Modify paramesh_preprocessor.fh and amr_runtime parameter

- Edit the header file paramesh_preprocessor.fh
  - cd to *AMRDIR/headers*
  - Edit *paramesh_preprocessor.fh*
    - If you want to use double precision then define REAL8:
      - #define REAL8
- Edit the amr_runtime_parameter file
  - cd to *AMRDIR/*
  - *Copy* amr_runtime_parameter into your_tutorial directory
  - Edit the file in the your_tutorial directory following this example:



Step 5: Create the makefile

- Copy the *AMRDIR/templates/Makefile.gnu_template* file into *your_directory*
- Edit the file:
    - Modify the macro definition MAIN to:
        main := tutorial.F90
    - Modify the macro definition SOURCES to:
        sources := amr_1blk_bcset.F90
    - Define the CMD macro to be tutor, ie:
        CMD = tutor
- cd back on AMRDIR
- Copy *Makefile.gnu* into *make_tutor*.
- Edit the file:
    - Replace the character string 'User_applic' with 'your_tutorial', wherever it appears.

Step 6: Modify the program template

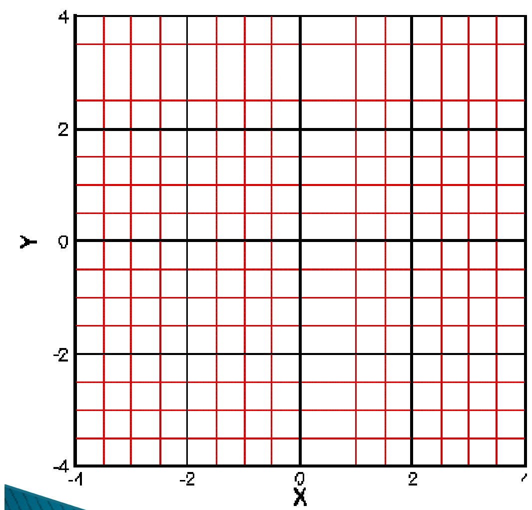
- Edit the file tutorial.f90:
  - The file is divided into a sequence of numbered sections. Comment out all executable lines in sections 4, 5 and 6.
- Edit the file amr_1blk_bcset.F90:
  - Uncomment the line:

    `! if(ibc.eq. ???? ) then`
  - and its corresponding *endif*.
  - Change the ???? in the if statement to any integer less than or equal to -20
  - Uncomment the line:

    `! unk1(:,i,j,k,idest) = ???? !<<<<< USER EDIT`
  - and replace the right hand side of this line with 0.0

Step 7: Build & Run

- Build the executable:
  gmake -f make_tutor your_tutorial
- Run the executable:
  ./tutor
- If everything went according to plan you should have generated a short output listing which concludes with something equivalent to the following lines (the order in which the blocks are listed may vary slightly, from one machine to another):




```
mguarra1@node066:~/PARAMESH/test/paramesh_4.1/your_tutorial
File  Edit  View  Search  Terminal  Help
mguarra1@node066.pico:[your_tutorial]$ ./tutor
 Running on            1  processors
  iteration, no. not moved =            0           0
  iteration, no. not moved =            0           0
 pe / blk / blk-coords / blk-sizes
           0           1  0.000000000000000E+000  0.000000000000000E+000
   8.00000000000000        8.00000000000000
           0           2  -2.00000000000000       -2.00000000000000
   4.00000000000000        4.00000000000000
           0           3  -3.00000000000000       -3.00000000000000
   2.00000000000000        2.00000000000000
           0           4  -1.00000000000000       -3.00000000000000
   2.00000000000000        2.00000000000000
           0           5  -3.00000000000000       -1.00000000000000
   2.00000000000000        2.00000000000000
           0           6  -1.00000000000000       -1.00000000000000
   2.00000000000000        2.00000000000000
           0           7   2.00000000000000       -2.00000000000000
   4.00000000000000        4.00000000000000
           0           8   1.00000000000000       -3.00000000000000
   2.00000000000000        2.00000000000000
           0           9   3.00000000000000       -3.00000000000000
   2.00000000000000        2.00000000000000
           0          10   1.00000000000000       -1.00000000000000
   2.00000000000000        2.00000000000000
           0          11   3.00000000000000       -1.00000000000000
   2.00000000000000        2.00000000000000
           0          12  -2.00000000000000        2.00000000000000
   4.00000000000000        4.00000000000000
           0          13  -3.00000000000000        1.00000000000000
   2.00000000000000        2.00000000000000
           0          14  -1.00000000000000        1.00000000000000
   2.00000000000000        2.00000000000000
           0          15  -3.00000000000000        3.00000000000000
   2.00000000000000        2.00000000000000
           0          16  -1.00000000000000        3.00000000000000
   2.00000000000000        2.00000000000000
           0          17   2.00000000000000        2.00000000000000
   4.00000000000000        4.00000000000000
           0          18   1.00000000000000        1.00000000000000
   2.00000000000000        2.00000000000000
           0          19   3.00000000000000        1.00000000000000
   2.00000000000000        2.00000000000000
           0          20   1.00000000000000        3.00000000000000
   2.00000000000000        2.00000000000000
           0          21   3.00000000000000        3.00000000000000
   2.00000000000000        2.00000000000000
mguarra1@node066.pico:[your_tutorial]$
```

Step 8: Inizializing the solution

- copy the file *AMRDIR/templates/amr_initial_soln_template.F90* into the current directory and rename it *amr_initial_soln.F90*
- edit /your_tutorial/Makefile.gnu, adding *amr_initial_soln.F90* to the macro definition of source
- Edit *amr_initial_soln.F90*:
  - delete the lines *unk(1,i,j,k,lb) = ???* and *unk(2,i,j,k,lb) = ???* the 3 dotted lines that follow.
  - insert the following lines before the triply nested loop which sets values for unk:

```
dx = bsize(1,lb)/real(nxb)
dy = bsize(2,lb)/real(nyb)
```

  - replace the line *unk(1,i,j,k,lb) = ???* with the following segment:

```
unk(1,i,j,k,lb) = 1.0
xi = bnd_box(1,1,lb) + dx*(real(i-nguard0)-.5)
yi = bnd_box(1,2,lb) + dy*(real(j-nguard0)-.5)
if( abs(xi).lt.1.0 .and. abs(yi).lt.1.0) then
        unk(1,i,j,k,lb) = 10.0
endif
```

- Edit tutorial.F90:
  - uncommenting the call to amr_initial_soln, in SECTION 4.
  - insert the following write statements at the end of SECTION 4.

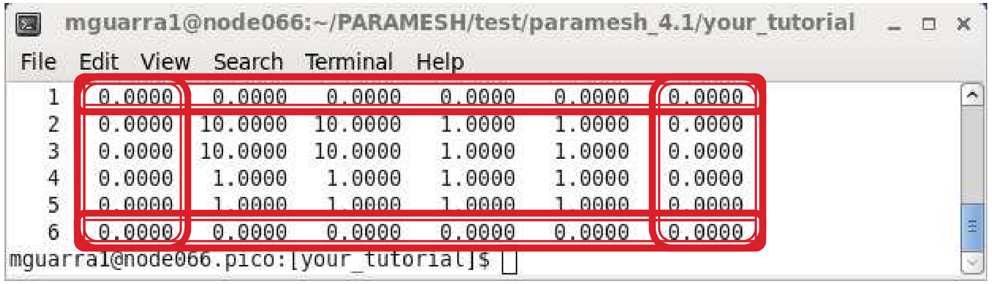
```
do lb=1,lnblocks
        if(coord(1,lb).eq.1.0.and.coord(2,lb).eq.1.0) then
                do j=1,nyb+2*nguard
                        write(*,50) j,(unk(1,i,j,1,lb),i=1,nxb+2*nguard)
                enddo
        endif
enddo
50      format(1x,i3,6(2x,f7.4))
```

Step 9: Build & Run

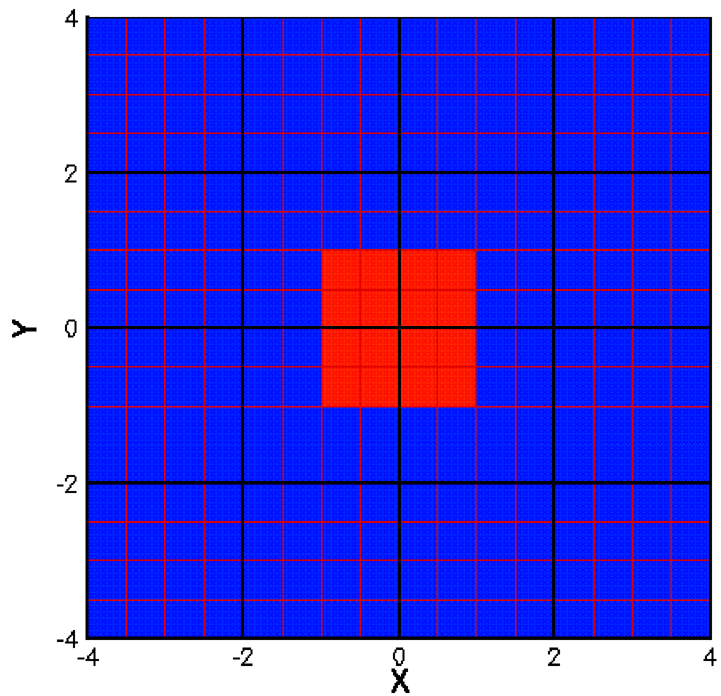
- Remake and run:

    cd AMRDIR
    gmake -f make_tutor your_tutorial
    cd your_tutorial
    ./tutor

- You have now initialized the solution array unk(1,:,:,:,:) on all the grid blocks of the initial grid. As proof, the last six lines of your output show the data values on the centered at (1.0,1.0). It should look like this:



- This block is located at 0 < x < 2 and 0 < y < 2. It straddles one corner of the high density region. Notice, the 4x4 block interior has been initialized with non-zero values and there is a layer of guard cells surrounding the block which are currently all set to 0.0.
- The complete initial state is shown here, with the block boundaries superimposed in black and the grid cells outlined in red:



Step 10: Filling Guardcells

- Edit the file tutorial.f90:
  - Uncomment the 3 executable lines in SECTION 5.
  - Move the output code fragment shown below from the end of SECTION 4 to the end of SECTION 5:

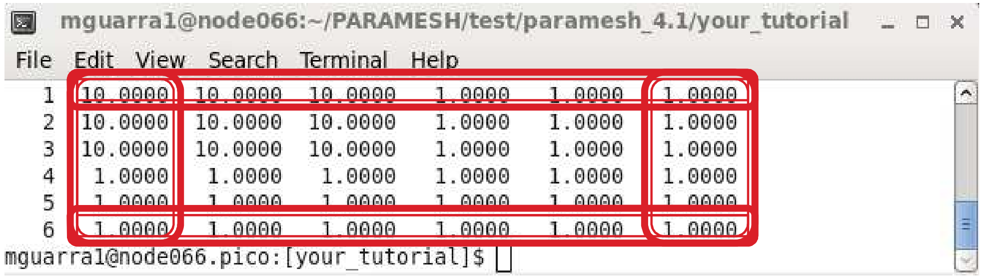
```
        do lb=1,lnblocks
                if(coord(1,lb).eq.1.0.and.coord(2,lb).eq.1.0) then
                do j=1,nyb+2*nguard
                                write(*,50) j,(unk(1,i,j,1,lb),i=1,nxb+2*nguard)
                        enddo
                endif
        enddo
 50   format(1x,i3,6(2x,f7.4))
```

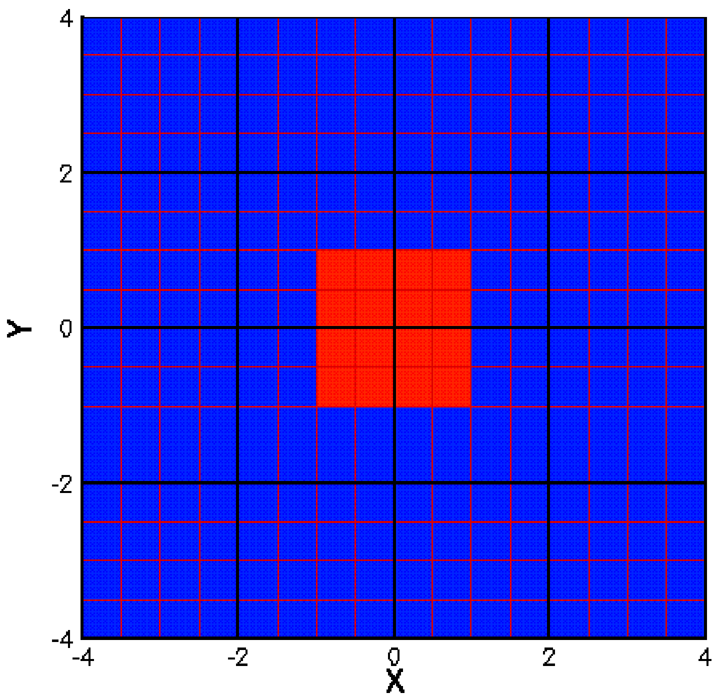
Step 11: Build & Run

- Remake and run:

  cd AMRDIR
  gmake -f make_tutor your_tutorial
  cd your_tutorial
  ./tutor

- Notice, the guard cell layer has been filled with the correct data from the neighboring blocks.





Step 12: Constructing a routine to test refinement levels - 1

- Copy *AMRDIR/templates/amr_test_refinement_template.F90* into the local directory and rename it *amr_test_refinement.F90*
- edit /your_tutorial/Makefile.gnu, adding *amr_test_refinement.F90* to the macro definition of source
- Edit *amr_test_refinement.F90*, commenting out the call to error_measure and immediately after the call inserting this simple error measure:

```
error(:,:,:) = 0.
do k=klw,kuw
        do j=jlw+1,juw-1
                do i=ilw+1,iuw-1
                        error1 = abs(work(i+1,j,k,lb,1)-work(i,j,k,lb,1))
                        error2 = abs(work(i-1,j,k,lb,1)-work(i,j,k,lb,1))
                        error3 = abs(work(i,j+1,k,lb,1)-work(i,j,k,lb,1))
                        error4 = abs(work(i,j-1,k,lb,1)-work(i,j,k,lb,1))
                        error_num = max( error1,error2,error3,error4 )
                        error_den = max( work(i,j,k,lb,1) ,work(i+1,j,k,lb,1), &
                                        work(i-1,j,k,lb,1),work(i,j+1,k,lb,1), &
                                        work(i,j-1,k,lb,1), 1.0e-6 )
                        error(i,j,k) = error_num/error_den
                enddo
        enddo
enddo
```

- Edit *tutorial.F90*
  - Uncomment the call to amr_restrict and the 2 lines preceding it (making sure that the 'if (.not.advance_all_levels) then' and corresponding 'endif' are also uncommented), and uncomment the calls to amr_test_refinement, amr_refine_derefine, amr_prolong and amr_guard cell in SECTION 6.
  - Change lrefine_max in SECTION 3 to allow 1 more level of refinement:

    ```
    lrefine_max = 4
    ```

- Continue…..

Step 12: Constructing a routine to test refinement levels - 2

- Continue in editing *tutorial.F90*:
  - Insert the following lines after the call to amr_test_refinement in SECTION 6:

```
if(mype.eq.0) write(*,*) ' pe blk refine derefine', &
                         ' curr.ref.level'
call MPI_BARRIER(MPI_COMM_WORLD, ierr)
do l=1,lnblocks
      write(*,51) mype,l,refine(l),derefine(l),lrefine(l)
enddo
51   format(1x,i3,2x,i3,2x,l8,2x,l8,10x,i3)
```

  - Insert the following lines at the end of SECTION 6:

```
if(mype.eq.0) write(*,*) 'pe / blk / blk-coords / blk-sizes'
call MPI_BARRIER(MPI_COMM_WORLD, ierr)
do l=1,lnblocks
      write(*,*) mype,l,(coord(i,l),i=1,ndim),(bsize(j,l),j=1,ndim)
enddo
```

- Set the number of guard cell layers for 'work' :
  - Edit *amr_runtime_parameters*, defining the variable:

```
1                    ! nguard_work
```

Step 13: Build & Run - 1

- Remake and run:

  cd AMRDIR
  gmake -f make_tutor your_tutorial
  cd your_tutorial
  ./tutor

- The changes that you have just made, analyzed the error estimate on each existing grid block, and marked some blocks for additional refinement. In your new output there will be a section looking like this (the order of lines may be slightly different) :

```
mguarra1@node066:~/PARAMESH/test/paramesh_4.1/you
File  Edit  View  Search  Terminal  Help
 pe blk refine derefine curr.ref.level
  0    1         F         F           1
  0    2         T         F           2
  0    3         F         T           3
  0    4         F         T           3
  0    5         F         T           3
  0    6         T         F           3
  0    7         T         F           2
  0    8         F         T           3
  0    9         F         T           3
  0   10         T         F           3
  0   11         F         T           3
  0   12         T         F           2
  0   13         F         T           3
  0   14         T         F           3
  0   15         F         T           3
  0   16         F         T           3
  0   17         T         F           2
  0   18         T         F           3
  0   19         F         T           3
  0   20         F         T           3
  0   21         F         T           3
 iteration, no. not moved =            0           7
 iteration, no. not moved =            1           0
pe / blk / blk-coords / blk-sizes
```

- This is telling you that the test in amr_test_refinement marked blocks 3, 6, 7, 10, 13, 14, 17, and 18 for further refinement.
- However blocks 3, 7, 13, and 17 are parent blocks at level 2 and so their refinement flags will be ignored.
- Blocks 6, 10, 14 and 18 will be refined. Notice also that blocks 2,4,5,8,9,11,12,15,16,19,20 and 21 have been marked for derefinement.
  - However each of these blocks has a sibling which has not been marked for derefinement (in fact all their siblings have been marked for refinement), and so these particular derefinement choices will be cancelled by PARAMESH.

Step 13: Build & Run - 2

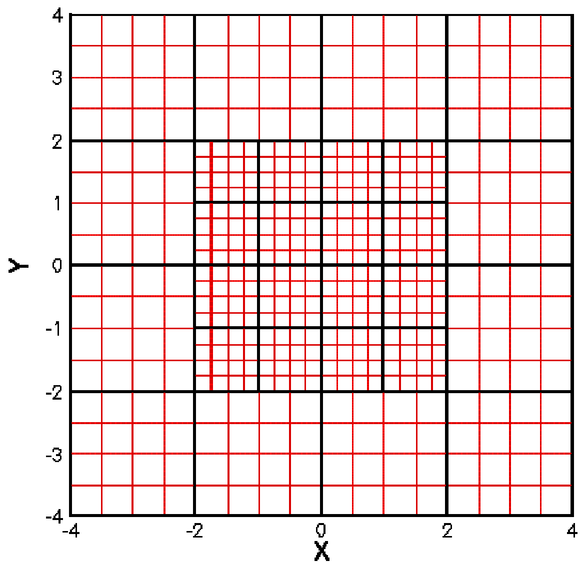
- This is the new positions of the grid blocks:

mguarra1@node066:~/PARAMESH/test/paramesh_4.1/your_tutorial

File Edit View Search Terminal Help

```
pe / blk / blk-coords / blk-sizes
           0           1  0.000000000000000E+000  0.000000000000000E+000   8.00000000000000        8.00000000000000
           0           2  -2.00000000000000       -2.00000000000000        4.00000000000000        4.00000000000000
           0           3  -3.00000000000000       -3.00000000000000        2.00000000000000        2.00000000000000
           0           4  -1.00000000000000       -3.00000000000000        2.00000000000000        2.00000000000000
           0           5  -3.00000000000000       -1.00000000000000        2.00000000000000        2.00000000000000
           0           6  -1.00000000000000       -1.00000000000000        2.00000000000000        2.00000000000000
           0           7  -1.50000000000000       -1.50000000000000        1.00000000000000        1.00000000000000
           0           8 -0.500000000000000       -1.50000000000000        1.00000000000000        1.00000000000000
           0           9  -1.50000000000000      -0.500000000000000        1.00000000000000        1.00000000000000
           0          10 -0.500000000000000      -0.500000000000000        1.00000000000000        1.00000000000000
           0          11   2.00000000000000       -2.00000000000000        4.00000000000000        4.00000000000000
           0          12   1.00000000000000       -3.00000000000000        2.00000000000000        2.00000000000000
           0          13   3.00000000000000       -3.00000000000000        2.00000000000000        2.00000000000000
           0          14   1.00000000000000       -1.00000000000000        2.00000000000000        2.00000000000000
           0          15  0.500000000000000       -1.50000000000000        1.00000000000000        1.00000000000000
           0          16   1.50000000000000       -1.50000000000000        1.00000000000000        1.00000000000000
           0          17  0.500000000000000      -0.500000000000000        1.00000000000000        1.00000000000000
           0          18   1.50000000000000      -0.500000000000000        1.00000000000000        1.00000000000000
           0          19   3.00000000000000       -1.00000000000000        2.00000000000000        2.00000000000000
           0          20  -2.00000000000000        2.00000000000000        4.00000000000000        4.00000000000000
           0          21  -3.00000000000000        1.00000000000000        2.00000000000000        2.00000000000000
           0          22  -1.00000000000000        1.00000000000000        2.00000000000000        2.00000000000000
           0          23  -1.50000000000000       0.500000000000000        1.00000000000000        1.00000000000000
           0          24 -0.500000000000000       0.500000000000000        1.00000000000000        1.00000000000000
           0          25  -1.50000000000000        1.50000000000000        1.00000000000000        1.00000000000000
           0          26 -0.500000000000000        1.50000000000000        1.00000000000000        1.00000000000000
           0          27  -3.00000000000000        3.00000000000000        2.00000000000000        2.00000000000000
           0          28  -1.00000000000000        3.00000000000000        2.00000000000000        2.00000000000000
           0          29   2.00000000000000        2.00000000000000        4.00000000000000        4.00000000000000
           0          30   1.00000000000000        1.00000000000000        2.00000000000000        2.00000000000000
           0          31  0.500000000000000       0.500000000000000        1.00000000000000        1.00000000000000
           0          32   1.50000000000000       0.500000000000000        1.00000000000000        1.00000000000000
           0          33  0.500000000000000        1.50000000000000        1.00000000000000        1.00000000000000
           0          34   1.50000000000000        1.50000000000000        1.00000000000000        1.00000000000000
           0          35   3.00000000000000        1.00000000000000        2.00000000000000        2.00000000000000
           0          36   1.00000000000000        3.00000000000000        2.00000000000000        2.00000000000000
           0          37   3.00000000000000        3.00000000000000        2.00000000000000        2.00000000000000
                                                                                               77,1          96%
```

- and this is the new structure:



Step 14: Create the routine to update the solution

- Copy the file ../teamplates/amr_initial_soln.F90 to advance_soln.F90
- Edit advance_soln.F90, making the following changes:
    - Change the subroutine statement to:

      ```
      subroutine advance_soln(mype,time,dt)
      ```
    - Make the same modification to the end statement

      ```
      •end subroutine advance_soln
      ```
    - Add the declarations:

      ```
      integer :: mype
      real :: time,dt
      real old_soln(il_bnd:iu_bnd,jl_bnd:ju_bnd,kl_bnd:ku_bnd)
      ```
        - making sure that they appear after the use statements.
    - Before the "! loop over leaf grid blocks" comment line insert the line:

      ```
      call amr_timestep(dt,dtmin,dtmax,mype)
      ```
    - Insert the following line immediately before the "! set values for unk" comment line:

      ```
      old_soln(:,:,:) = unk(1,:,:,:,lb)
      dx = bsize(1,lb)/real(nxb)
      ```
    - Replace the triply nested loop which updates 'unk' with the following lines:

      ```
      do k=kl_bnd+nguard*k3d,ku_bnd-nguard*k3d
      do j=jl_bnd+nguard*k2d,ju_bnd-nguard*k2d
      do i=il_bnd+nguard,iu_bnd-nguard

      unk(1,i,j,k,lb) = old_soln(i,j,k) + dt/(dx*dx)* ( &
      old_soln(i+1,j,k) + old_soln(i-1,j,k) + &
      old_soln(i,j+1,k) + old_soln(i,j-1,k) - &
      old_soln(i,j,k)*4.0 )

      enddo
      enddo
      enddo
      ```
    - Add the following lines before the return statement:

      ```
      time = time + dt
      ```

Step 15: Create the timestep routine

- Copy AMRDIR/templates/amr_timestep_template.F90 in to the current directory and rename it amr_timestep.F90
- Edit amr_timestep.F90, making the following changes:
  - Delete the lines declaring the real variables speed2, press and maxspeed.
  - Delete the line including the file pointers.fh
  - Delete the following lines inside the loop over grid blocks

    ```
    rho => unk(1,:,:,:,l)
    vx => unk(2,:,:,:,l)
    vy => unk(3,:,:,:,l)
    vz => unk(4,:,:,:,l)
    ```

  - Change the parameter statement defining courant to:

    ```
    real, parameter :: courant=.1, kappa=1.0
    ```

  - Replace all the lines in the section labeled 'users timestep calculation' with the following line:
    - dtl = courant*dx*dx/kappa

Step 16: Modify main program to call the advance_soln routine.

- Edit tutorial.F90 making the following changes:
  - Uncomment the lines setting minstp and maxstp.
  - Uncomment the do istep=... statement and the corresponding enddo statement.
  - Uncomment the call to advance_soln .F90 .
  - Delete the two blocks of output code which we inserted into SECTION 6 earlier.
  - Insert the following output code immediately after the call to advance_soln:

```
write(*,*) 'dt = ',dt
do lb=1,lnblocks
  if(coord(1,lb).eq.1.0.and.coord(2,lb).eq.1.0) then
    do j=1,nyb+2*nguard
      write(*,50) j,(unk(1,i,j,1,lb),i=1,nxb+2*nguard)
    enddo
  endif
enddo
```

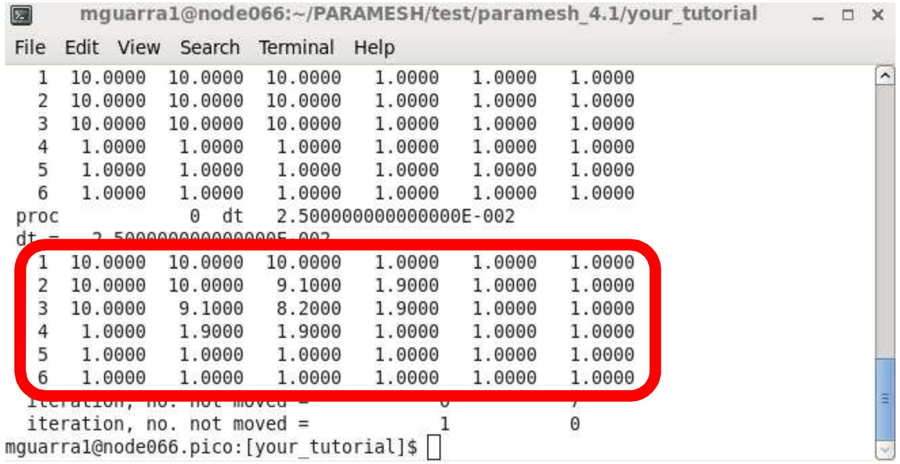
Step 17: Build & Run

- remake and rerun by typing:

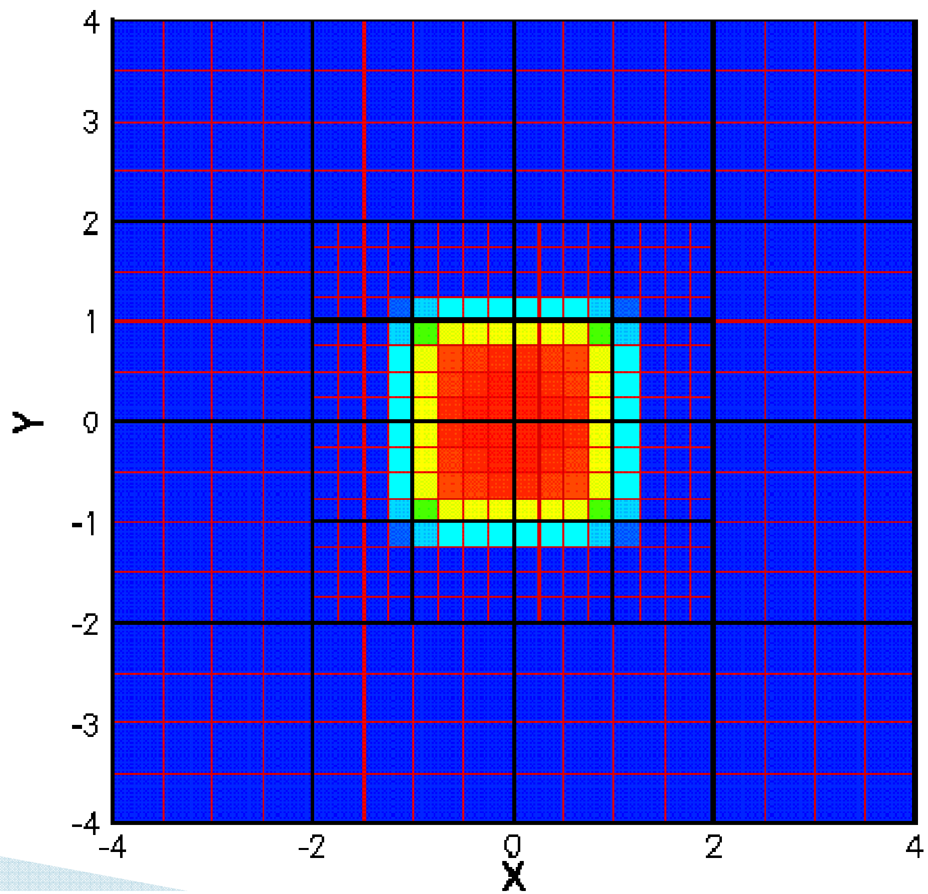
  gmake -f make_tutor your_tutorial .

  ./tutor

- You have now advanced the solution through 1 timestep, and the output section immediately after the call to advance_soln will show how the data on the block centered on (1.0,1.0) has been diffused. The data should look like this: :



- Note, at this point the cell interior (indeces 2-5 in both x and y) are correct, but the guardcells (indeces 1 and 6) have not yet been updated.
- After the solution has been advanced on the block interiors, we test the solution to see if refinement is required. In this case refinement is selected for the 4 blocks around the center of the domain. These are refined, and the solution is prolonged to the newly created blocks there. The complete updated solution after these steps is shown here.



Step 18: Run for 250 timesteps - 1

- Edit tutorial.F90 making the following changes:
  - Set maxstp = 250
  - Remove the output statements immediately after the call to advance_soln in SECTION 6.
  - Insert the following line into SECTION 6 immediately before the enddo statement:

```
if(mype.eq.0) write(*,*) 'iteration ',istep, &
                     ' no of blocks = ',lnblocks
```

  - Insert the following statements immediately before the amr_close call:

```
if(mype.eq.0) write(*,*) 'pe / blk / blk-coords / blk-sizes'
call MPI_BARRIER(MPI_COMM_WORLD, ierr)
do l=1,lnblocks
        write(*,*) mype,l,(coord(i,l),i=1,ndim),(bsize(j,l),j=1,ndim)
enddo
```

- remake and rerun by typing:

```
gmake -f make_tutor your_tutorial .
./tutor
```

- In your output you will notice that before the first timestep we had 21 blocks, with uniform refinement at level 3 throughout the computational domain. On the first timestep (iteration 1) the 4 blocks at the center containing the high data values were refined, adding 16 blocks to make a total of 37. After the seventeenth timestep the solution has diffused outward so that all the outer level 3 blocks, except for those on the corners, are now all marked for refinement, adding another 32 child blocks at level 4, for a total of 69.

Step 18: Run for 250 timesteps - 2

- After 250 timesteps, we can see from the final block listing below that block number 55 is a leaf block located near the origin (it has coordinates x=0.5, y=0.5 ; the line order may be slightly different in your output).

mguarra1@node066:~/PARAMESH/test/paramesh_4.1/your_tutorial

File Edit View Search Terminal Help

```
proc            0  dt   6.25000000000000E-003
iteration        249  no of blocks =          69
proc            0  dt   6.25000000000000E-003
iteration        250  no of blocks =          69
pe / blk / blk-coords / blk-sizes
          0           1  0.00000000000000E+000  0.00000000000000E+000   8.0000000000000        8.0000000000000
          0           2  -2.0000000000000       -2.0000000000000        4.0000000000000        4.0000000000000
          0           3  -3.0000000000000       -3.0000000000000        2.0000000000000        2.0000000000000
          0           4  -1.0000000000000       -3.0000000000000        2.0000000000000        2.0000000000000
          0           5  -1.5000000000000       -3.5000000000000        1.0000000000000        1.0000000000000
          0           6 -0.50000000000000       -3.5000000000000        1.0000000000000        1.0000000000000
          0           7  -1.5000000000000       -2.5000000000000        1.0000000000000        1.0000000000000
          0           8 -0.50000000000000       -2.5000000000000        1.0000000000000        1.0000000000000
          0           9  -3.0000000000000       -1.0000000000000        2.0000000000000        2.0000000000000
          0          10  -3.5000000000000       -1.5000000000000        1.0000000000000        1.0000000000000
          0          11  -2.5000000000000       -1.5000000000000        1.0000000000000        1.0000000000000
          0          12  -3.5000000000000      -0.50000000000000        1.0000000000000        1.0000000000000
          0          13  -2.5000000000000      -0.50000000000000        1.0000000000000        1.0000000000000
          0          14  -1.0000000000000       -1.0000000000000        2.0000000000000        2.0000000000000
          0          15  -1.5000000000000       -1.5000000000000        1.0000000000000        1.0000000000000
          0          16 -0.50000000000000       -1.5000000000000        1.0000000000000        1.0000000000000
          0          17  -1.5000000000000      -0.50000000000000        1.0000000000000        1.0000000000000
          0          18 -0.50000000000000      -0.50000000000000        1.0000000000000        1.0000000000000
          0          19   2.0000000000000       -2.0000000000000        4.0000000000000        4.0000000000000
          0          20   1.0000000000000       -3.0000000000000        2.0000000000000        2.0000000000000
          0          21  0.50000000000000       -3.5000000000000        1.0000000000000        1.0000000000000
          0          22   1.5000000000000       -3.5000000000000        1.0000000000000        1.0000000000000
          0          23  0.50000000000000       -2.5000000000000        1.0000000000000        1.0000000000000
          0          24   1.5000000000000       -2.5000000000000        1.0000000000000        1.0000000000000
          0          25   3.0000000000000       -3.0000000000000        2.0000000000000        2.0000000000000
          0          26   1.0000000000000       -1.0000000000000        2.0000000000000        2.0000000000000
          0          27  0.50000000000000       -1.5000000000000        1.0000000000000        1.0000000000000
          0          28   1.5000000000000       -1.5000000000000        1.0000000000000        1.0000000000000
          0          29  0.50000000000000      -0.50000000000000        1.0000000000000        1.0000000000000
          0          30   1.5000000000000      -0.50000000000000        1.0000000000000        1.0000000000000
          0          31   3.0000000000000       -1.0000000000000        2.0000000000000        2.0000000000000
          0          32   2.5000000000000       -1.5000000000000        1.0000000000000        1.0000000000000
          0          33   3.5000000000000       -1.5000000000000        1.0000000000000        1.0000000000000
          0          34   2.5000000000000      -0.50000000000000        1.0000000000000        1.0000000000000
          0          35   3.5000000000000      -0.50000000000000        1.0000000000000        1.0000000000000
          0          36  -2.0000000000000        2.0000000000000        4.0000000000000        4.0000000000000
          0          37  -3.0000000000000        1.0000000000000        2.0000000000000        2.0000000000000
          0          38  -3.5000000000000       0.50000000000000        1.0000000000000        1.0000000000000
          0          39  -2.5000000000000       0.50000000000000        1.0000000000000        1.0000000000000
          0          40  -3.5000000000000        1.5000000000000        1.0000000000000        1.0000000000000
          0          41  -2.5000000000000        1.5000000000000        1.0000000000000        1.0000000000000
          0          42  -1.0000000000000        1.0000000000000        2.0000000000000        2.0000000000000
          0          43  -1.5000000000000       0.50000000000000        1.0000000000000        1.0000000000000
          0          44 -0.50000000000000       0.50000000000000        1.0000000000000        1.0000000000000
          0          45  -1.5000000000000        1.5000000000000        1.0000000000000        1.0000000000000
          0          46 -0.50000000000000        1.5000000000000        1.0000000000000        1.0000000000000
          0          47  -3.0000000000000        3.0000000000000        2.0000000000000        2.0000000000000
          0          48  -1.0000000000000        3.0000000000000        2.0000000000000        2.0000000000000
          0          49  -1.5000000000000        2.5000000000000        1.0000000000000        1.0000000000000
          0          50 -0.50000000000000        2.5000000000000        1.0000000000000        1.0000000000000
          0          51  -1.5000000000000        3.5000000000000        1.0000000000000        1.0000000000000
          0          52 -0.50000000000000        3.5000000000000        1.0000000000000        1.0000000000000
          0          53   2.0000000000000        2.0000000000000        4.0000000000000        4.0000000000000
          0          54   1.0000000000000        1.0000000000000        2.0000000000000        2.0000000000000
          0          55  0.50000000000000       0.50000000000000        1.0000000000000        1.0000000000000
          0          56   1.5000000000000       0.50000000000000        1.0000000000000        1.0000000000000
          0          57  0.50000000000000        1.5000000000000        1.0000000000000        1.0000000000000
          0          58   1.5000000000000        1.5000000000000        1.0000000000000        1.0000000000000
          0          59   3.0000000000000        1.0000000000000        2.0000000000000        2.0000000000000
          0          60   2.5000000000000       0.50000000000000        1.0000000000000        1.0000000000000
          0          61   3.5000000000000       0.50000000000000        1.0000000000000        1.0000000000000
          0          62   2.5000000000000        1.5000000000000        1.0000000000000        1.0000000000000
          0          63   3.5000000000000        1.5000000000000        1.0000000000000        1.0000000000000
          0          64   1.0000000000000        3.0000000000000        2.0000000000000        2.0000000000000
          0          65  0.50000000000000        2.5000000000000        1.0000000000000        1.0000000000000
          0          66   1.5000000000000        2.5000000000000        1.0000000000000        1.0000000000000
          0          67  0.50000000000000        3.5000000000000        1.0000000000000        1.0000000000000
          0          68   1.5000000000000        3.5000000000000        1.0000000000000        1.0000000000000
          0          69   3.0000000000000        3.0000000000000        2.0000000000000        2.0000000000000
-- INSERT --                                                                                   626,73        Bot
```

Step 19: Check solution

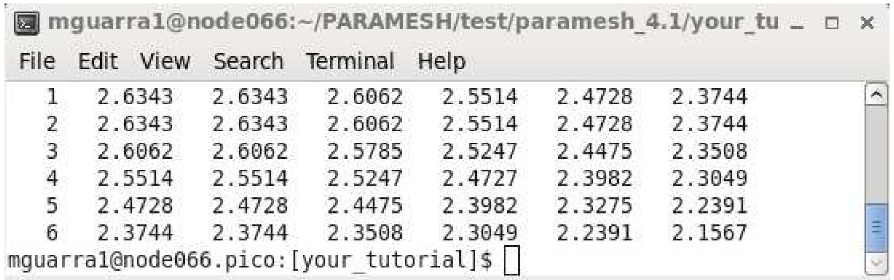
- Edit tutorial.F90, adding the following immediately before the call to amr_close, to show the solution on the block centered on (0.5,0.5):

```
do lb=1,lnblocks
   if(coord(1,lb).eq..5.and.coord(2,lb).eq..5) then
     do j=1,nyb+2*nguard
        write(*,50) j,(unk(1,i,j,1,lb),i=1,nxb+2*nguard)
     enddo
   endif
enddo
```

- remake and rerun by typing:

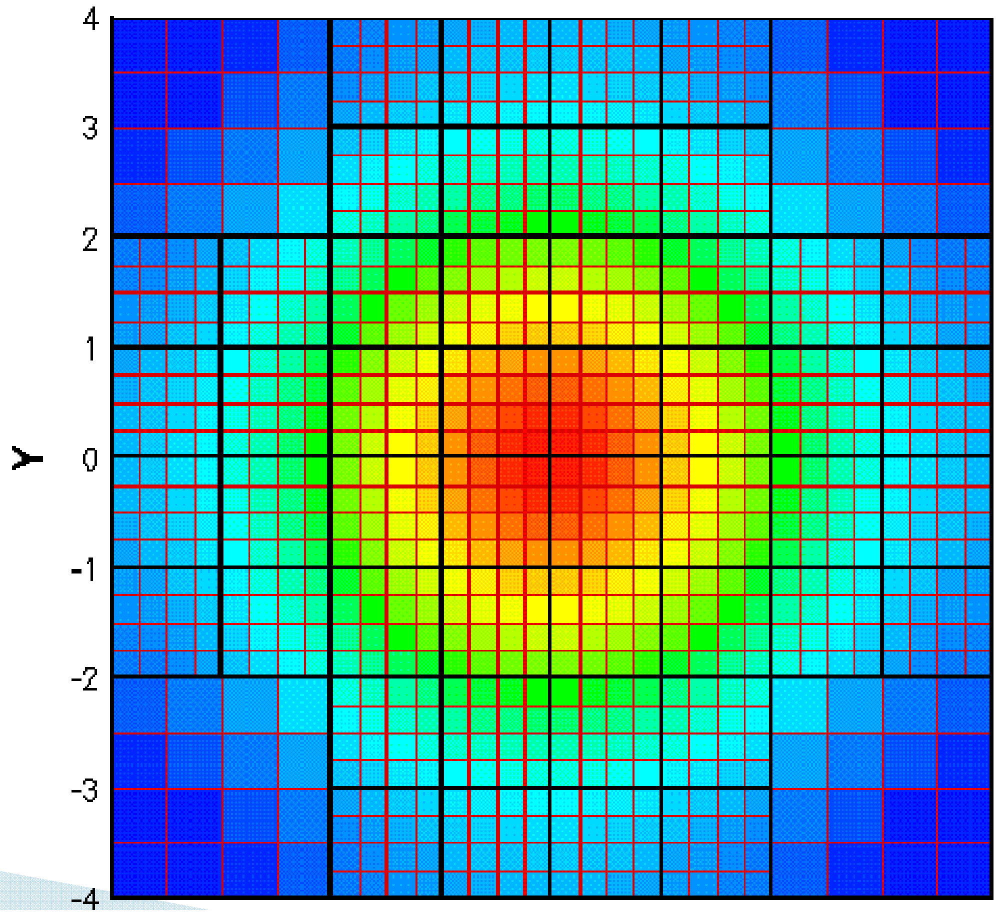
```
gmake -f make_tutor your_tutorial .
./tutor
```

- Your final lines of output should look like this:



```
mguarra1@node066:~/PARAMESH/test/paramesh_4.1/your_tu
File  Edit  View  Search  Terminal  Help
 1   2.6343   2.6343   2.6062   2.5514   2.4728   2.3744
 2   2.6343   2.6343   2.6062   2.5514   2.4728   2.3744
 3   2.6062   2.6062   2.5785   2.5247   2.4475   2.3508
 4   2.5514   2.5514   2.5247   2.4727   2.3982   2.3049
 5   2.4728   2.4728   2.4475   2.3982   2.3275   2.2391
 6   2.3744   2.3744   2.3508   2.3049   2.2391   2.1567
mguarra1@node066.pico:[your_tutorial]$
```

- The final solution is displayed here:



Step 20: Run in Parallel

- Create the file jobscript.sh
- Open the file
- Write:

mguarra1@node066:~/PARAMESH/test/param

File Edit View Search Terminal Help

```
#!/bin/bash
#PBS -l select=1:ncpus=2:mpiprocs=2
#PBS -l walltime=02:00
#PBS -A cin_staff

# goto launch directory
cd $PBS_O_WORKDIR

# load modules
module load profile/advanced
module load autoload intelmpi/5.1.1--binary
module list
# launch tutor over 1*2 = 2 cores
mpirun -np 2 ./tutor
```

1,1 All

- Type:

  qsub jobscript.sh

Thank you for your attention