Adaptive Mesh Refinement for multi-fluid simulations of the processes in the solar atmosphere

Angel de Vicente, Elena Khomenko, Manuel Luna and David Martín Belda

Instituto de Astrofísica de Canarias
Universidad de La Laguna

June 20, 2013

Outline

1. Overview of Adaptive Mesh Refinement (AMR)

2. Paramesh

3. Paramesh use cases

Description of structured AMR

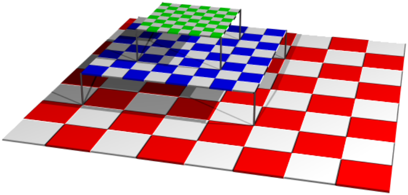


Figure: Graphical representation of AMR refinement levels (from http://www.lsc.phy.cam.ac.uk/research/amr.shtml)

- Method for dynamically (de)refining the grid resolution.
- Allows simulations with large length and time-scales.
- Increased computational savings over a static grid approach.
- Increased storage savings over a static grid approach.
- Eeach mesh can be advanced independently using its own time step.
- Strict conservation by matching fluxes at grid boundaries.
- Easily parallelizable due to the logically rectangular patches.

Description of structured AMR

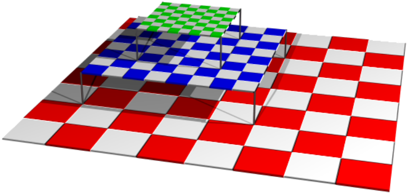


Figure: Graphical representation of AMR refinement levels (from http://www.lsc.phy.cam.ac.uk/research/amr.shtml)

- Method for dynamically (de)refining the grid resolution.
- Allows simulations with large length and time-scales.
- Increased computational savings over a static grid approach.
- Increased storage savings over a static grid approach.
- Eeach mesh can be advanced independently using its own time step.
- Strict conservation by matching fluxes at grid boundaries.
- Easily parallelizable due to the logically rectangular patches.

Description of structured AMR

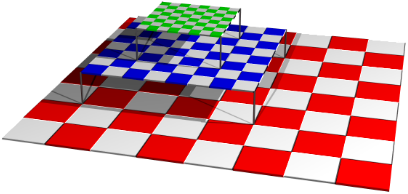


Figure: Graphical representation of AMR refinement levels (from http://www.lsc.phy.cam.ac.uk/research/amr.shtml)

- Method for dynamically (de)refining the grid resolution.
- Allows simulations with large length and time-scales.
- Increased computational savings over a static grid approach.
- Increased storage savings over a static grid approach.
- Eeach mesh can be advanced independently using its own time step.
- Strict conservation by matching fluxes at grid boundaries.
- Easily parallelizable due to the logically rectangular patches.

Description of structured AMR

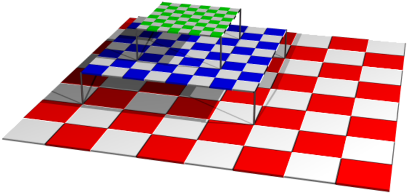


Figure: Graphical representation of AMR refinement levels (from http://www.lsc.phy.cam.ac.uk/research/amr.shtml)

- Method for dynamically (de)refining the grid resolution.
- Allows simulations with large length and time-scales.
- Increased computational savings over a static grid approach.
- Increased storage savings over a static grid approach.
- Eeach mesh can be advanced independently using its own time step.
- Strict conservation by matching fluxes at grid boundaries.
- Easily parallelizable due to the logically rectangular patches.

Description of structured AMR

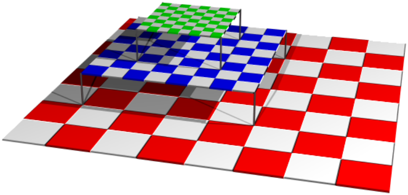


Figure: Graphical representation of AMR refinement levels (from http://www.lsc.phy.cam.ac.uk/research/amr.shtml)

- Method for dynamically (de)refining the grid resolution.
- Allows simulations with large length and time-scales.
- Increased computational savings over a static grid approach.
- Increased storage savings over a static grid approach.
- Eeach mesh can be advanced independently using its own time step.
- Strict conservation by matching fluxes at grid boundaries.
- Easily parallelizable due to the logically rectangular patches.

Description of structured AMR

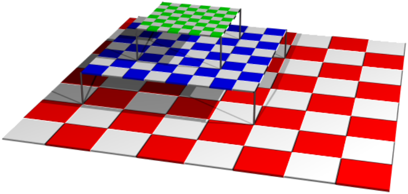


Figure: Graphical representation of AMR refinement levels (from http://www.lsc.phy.cam.ac.uk/research/amr.shtml)

- Method for dynamically (de)refining the grid resolution.
- Allows simulations with large length and time-scales.
- Increased computational savings over a static grid approach.
- Increased storage savings over a static grid approach.
- Eeach mesh can be advanced independently using its own time step.
- Strict conservation by matching fluxes at grid boundaries.
- Easily parallelizable due to the logically rectangular patches.

Description of structured AMR

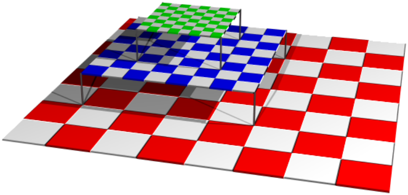


Figure: Graphical representation of AMR refinement levels (from http://www.lsc.phy.cam.ac.uk/research/amr.shtml)

- Method for dynamically (de)refining the grid resolution.
- Allows simulations with large length and time-scales.
- Increased computational savings over a static grid approach.
- Increased storage savings over a static grid approach.
- Eeach mesh can be advanced independently using its own time step.
- Strict conservation by matching fluxes at grid boundaries.
- Easily parallelizable due to the logically rectangular patches.

Some existing AMR packages

- Many non-generic AMR packages: Pluto, Enzo, Uintah, RAMSES, NIRVANA, A-MAZE, FLASH, AstroBEAR, ...
- Some generic structured AMR packages

Table: Generic AMR packages

Package name	Prog. language	Actively developed?
AMRClaw	F77,F90, Python	Last: January 2013 (Part of Clawpack)
ForestClaw	F77,F90, Python, C	In development, not yet available.
AMROC	C++	Last: November 2004
Boxlib	C, Fortran90	Last: November 2012
Chombo	C++	Last: March 2012
Paramesh	Fortran90	Last: March 2008. Support stopped.
SAMRAI	C++	Last: January 2013

  - Freely available (to run and to visualize output)
  - 2D and 3D.
  - Multi-stepping
  - Parallel (OpenMP and/or MPI)

Some existing AMR packages

- Many non-generic AMR packages: Pluto, Enzo, Uintah, RAMSES, NIRVANA, A-MAZE, FLASH, AstroBEAR, . . .
- Some generic structured AMR packages

Table: Generic AMR packages

Package name	Prog. language	Actively developed?
AMRClaw	F77,F90, Python	Last: January 2013 (Part of Clawpack)
ForestClaw	F77,F90, Python, C	In development, not yet available.
AMROC	C++	Last: November 2004
Boxlib	C, Fortran90	Last: November 2012
Chombo	C++	Last: March 2012
Paramesh	Fortran90	Last: March 2008. Support stopped.
SAMRAI	C++	Last: January 2013

- Freely available (to run and to visualize output)
- 2D and 3D.
- Multi-stepping
- Parallel (OpenMP and/or MPI)

Some existing AMR packages

- Many non-generic AMR packages: Pluto, Enzo, Uintah, RAMSES, NIRVANA, A-MAZE, FLASH, AstroBEAR, ...
- Some generic structured AMR packages

Table: Generic AMR packages

Package name	Prog. language	Actively developed?
AMRClaw	F77,F90, Python	Last: January 2013 (Part of Clawpack)
ForestClaw	F77,F90, Python, C	In development, not yet available.
AMROC	C++	Last: November 2004
Boxlib	C, Fortran90	Last: November 2012
Chombo	C++	Last: March 2012
Paramesh	Fortran90	Last: March 2008. Support stopped.
SAMRAI	C++	Last: January 2013

- Freely available (to run and to visualize output)
- 2D and 3D.
- Multi-stepping
- Parallel (OpenMP and/or MPI)

Some existing AMR packages

- Many non-generic AMR packages: Pluto, Enzo, Uintah, RAMSES, NIRVANA, A-MAZE, FLASH, AstroBEAR, ...
- Some generic structured AMR packages

Table: Generic AMR packages

Package name	Prog. language	Actively developed?
AMRClaw	F77,F90, Python	Last: January 2013 (Part of Clawpack)
ForestClaw	F77,F90, Python, C	In development, not yet available.
AMROC	C++	Last: November 2004
Boxlib	C, Fortran90	Last: November 2012
Chombo	C++	Last: March 2012
Paramesh	Fortran90	Last: March 2008. Support stopped.
SAMRAI	C++	Last: January 2013

- Freely available (to run and to visualize output)
- 2D and 3D.
- Multi-stepping
- Parallel (OpenMP and/or MPI)

Some existing AMR packages

- Many non-generic AMR packages: Pluto, Enzo, Uintah, RAMSES, NIRVANA, A-MAZE, FLASH, AstroBEAR, . . .
- Some generic structured AMR packages

Table: Generic AMR packages

Package name	Prog. language	Actively developed?
AMRClaw	F77,F90, Python	Last: January 2013 (Part of Clawpack)
ForestClaw	F77,F90, Python, C	In development, not yet available.
AMROC	C++	Last: November 2004
Boxlib	C, Fortran90	Last: November 2012
Chombo	C++	Last: March 2012
Paramesh	Fortran90	Last: March 2008. Support stopped.
SAMRAI	C++	Last: January 2013

- Freely available (to run and to visualize output)
- 2D and 3D.
- Multi-stepping
- Parallel (OpenMP and/or MPI)

Some existing AMR packages

- Many non-generic AMR packages: Pluto, Enzo, Uintah, RAMSES, NIRVANA, A-MAZE, FLASH, AstroBEAR, . . .
- Some generic structured AMR packages

Table: Generic AMR packages

Package name	Prog. language	Actively developed?
AMRClaw	F77,F90, Python	Last: January 2013 (Part of Clawpack)
ForestClaw	F77,F90, Python, C	In development, not yet available.
AMROC	C++	Last: November 2004
Boxlib	C, Fortran90	Last: November 2012
Chombo	C++	Last: March 2012
Paramesh	Fortran90	Last: March 2008. Support stopped.
SAMRAI	C++	Last: January 2013

- Freely available (to run and to visualize output)
- 2D and 3D.
- Multi-stepping
- Parallel (OpenMP and/or MPI)

Paramesh

Main characteristics

- Parallel (MPI) Adaptive Mesh Refinement (Fortran 90 and C)
- Multidimensional
- Structured Grid Blocks
- Portable
- User Tunable Load Balancing
- Support for Conservation Laws
- Distribution contains source code
- Builds upon user's existing codes
- Useful webpage with tutorial, examples, users' and reference guides
- I/O to Chombo HDF5 files

Paramesh

Main characteristics

- Parallel (MPI) Adaptive Mesh Refinement (Fortran 90 and C)
- Multidimensional
- Structured Grid Blocks
- Portable
- User Tunable Load Balancing
- Support for Conservation Laws
- Distribution contains source code
- Builds upon user's existing codes
- Useful webpage with tutorial, examples, users' and reference guides
- I/O to Chombo HDF5 files

Paramesh

Main characteristics

- Parallel (MPI) Adaptive Mesh Refinement (Fortran 90 and C)
- Multidimensional
- Structured Grid Blocks
- Portable
- User Tunable Load Balancing
- Support for Conservation Laws
- Distribution contains source code
- Builds upon user's existing codes
- Useful webpage with tutorial, examples, users' and reference guides
- I/O to Chombo HDF5 files

Paramesh

Main characteristics

- Parallel (MPI) Adaptive Mesh Refinement (Fortran 90 and C)
- Multidimensional
- Structured Grid Blocks
- Portable
- User Tunable Load Balancing
- Support for Conservation Laws
- Distribution contains source code
- Builds upon user's existing codes
- Useful webpage with tutorial, examples, users' and reference guides
- I/O to Chombo HDF5 files

Paramesh

Main characteristics

- Parallel (MPI) Adaptive Mesh Refinement (Fortran 90 and C)
- Multidimensional
- Structured Grid Blocks
- Portable
- User Tunable Load Balancing
- Support for Conservation Laws
- Distribution contains source code
- Builds upon user's existing codes
- Useful webpage with tutorial, examples, users' and reference guides
- I/O to Chombo HDF5 files

Paramesh

Main characteristics

- Parallel (MPI) Adaptive Mesh Refinement (Fortran 90 and C)
- Multidimensional
- Structured Grid Blocks
- Portable
- User Tunable Load Balancing
- Support for Conservation Laws
- Distribution contains source code
- Builds upon user's existing codes
- Useful webpage with tutorial, examples, users' and reference guides
- I/O to Chombo HDF5 files

Paramesh

Main characteristics

- Parallel (MPI) Adaptive Mesh Refinement (Fortran 90 and C)
- Multidimensional
- Structured Grid Blocks
- Portable
- User Tunable Load Balancing
- Support for Conservation Laws
- Distribution contains source code
- Builds upon user's existing codes
- Useful webpage with tutorial, examples, users' and reference guides
- I/O to Chombo HDF5 files

Paramesh

Main characteristics

- Parallel (MPI) Adaptive Mesh Refinement (Fortran 90 and C)
- Multidimensional
- Structured Grid Blocks
- Portable
- User Tunable Load Balancing
- Support for Conservation Laws
- Distribution contains source code
- Builds upon user's existing codes
- Useful webpage with tutorial, examples, users' and reference guides
- I/O to Chombo HDF5 files

Paramesh

Main characteristics

- Parallel (MPI) Adaptive Mesh Refinement (Fortran 90 and C)
- Multidimensional
- Structured Grid Blocks
- Portable
- User Tunable Load Balancing
- Support for Conservation Laws
- Distribution contains source code
- Builds upon user's existing codes
- Useful webpage with tutorial, examples, users' and reference guides
- I/O to Chombo HDF5 files

Paramesh
Main characteristics

- Parallel (MPI) Adaptive Mesh Refinement (Fortran 90 and C)
- Multidimensional
- Structured Grid Blocks
- Portable
- User Tunable Load Balancing
- Support for Conservation Laws
- Distribution contains source code
- Builds upon user's existing codes
- Useful webpage with tutorial, examples, users' and reference guides
- I/O to Chombo HDF5 files

Paramesh

Basic usage

Define the models dimensionality, properties of each grid block, etc. (amr_runtime_parameters)

```
50                          ! maxblocks
2                           ! ndim
0                           ! l2p5d
124                         ! nxb
124                         ! nyb
1                           ! nzb
164                         ! nvar
[...]
5                           ! nguard
[...]
2                           ! nboundaries
.true.                      ! diagonals
.true.                      ! amr_error_checking
```

Paramesh

Basic usage (2)

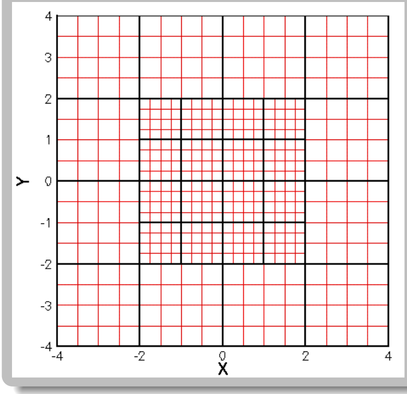
Set up initial grid (sizes, locations, neighbours, boundaries)

```
lrefine_max = 10           ! finest refinement level allowed
lrefine_min = 6            ! coarsest refinement level allowed

if(mype.eq.0.) then
    bnd_box(1,1,1) = 0.    ! lower x bound of block 1
    bsize(1,1) = bnd_box(2,1,1) - bnd_box(1,1,1)
    coord(1,1) = .5*(bnd_box(2,1,1) - bnd_box(1,1,1))
    nodetype(1) = 1        ! identify block 1 as a leaf block
    lrefine(1) = 1         ! set refinement level of block 1
    neigh(1:2,1,1) = -21   ! left boundary condition
    lnblocks = 1           ! no. of blocks on this processor
 endif

! x boundaries
boundary_box(1,2:3,1:2) = -1.e30 ! effectively this is -infinity
boundary_index(1)       = -21
```

Paramesh

Basic usage (3)

Sample mesh



Mesh details

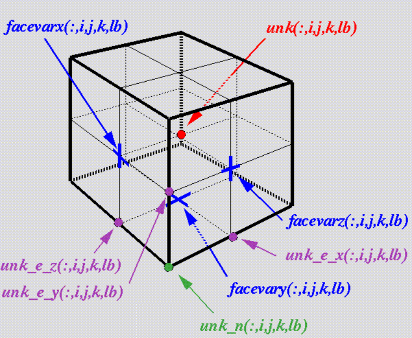
- All grid blocks identical logical structure
- Logically cartesian
- When refined, a block spawns 4 child blocks (2D), same area but twice spatial resolution
- Neighbour leaf blocks differ at most by one refinement level

Paramesh

Basic usage (4)

Use pointers in our code to link to Paramesh data structures

```
REAL(KIND=fp), DIMENSION(:,:), POINTER :: te, pe, rho
[. . .]
te => unk(1,:,:,1,block)
pe => unk(2,:,:,1,block)
rho => unk(3,:,:,1,block)
```



Paramesh

Basic usage (5)

Load initial solution and evolve

```
time = 0.
call initial_soln(mype)

do loop=1,ntsteps
    call amr_guardcell(mype,iopt,nlayers)

    ! advance and amr_test_refinement: user defined
    call advance(mype,dt,time,nprocs,loop)
    call amr_test_refinement(mype,lrefine_min,lrefine_max)

    call amr_refine_derefine
    call amr_prolong(mype,iopt,nlayers)
enddo

call amr_close()
```

Multi-grid example

Whole domain

Detail

Adaptive mesh example: Rayleigh-Taylor instability

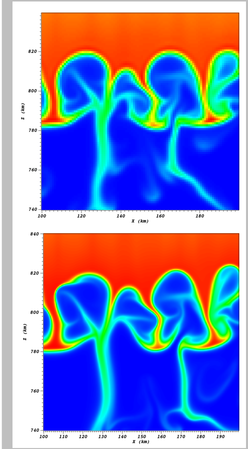
Whole domain

Detail

Adaptive mesh example: RTI

Mesh refinement dynamics

Grid dynamics

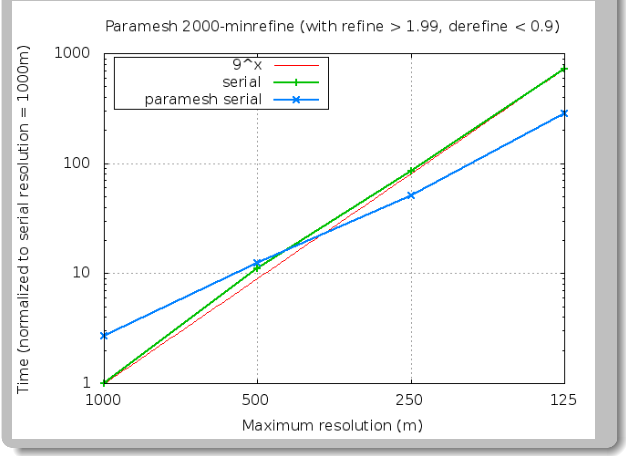


- It works! But is it worth it?

Adaptive mesh example: RTI

Paramesh computation overhead

Serial code vs. Paramesh (serial)



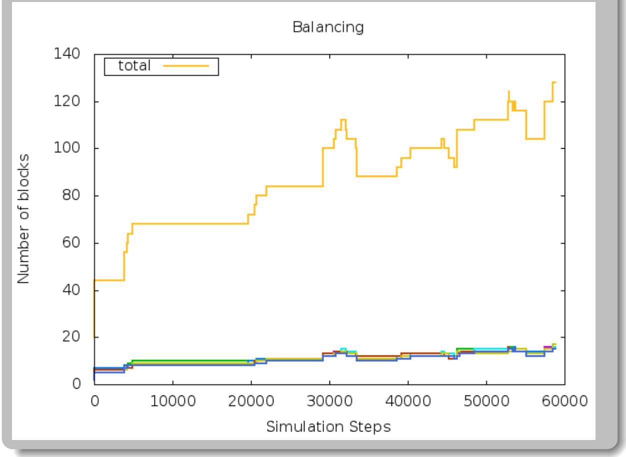
Notes

- Paramesh minimum levels of refinement: 2
- Performance will greatly depend on refinement details.

Adaptive mesh example: RTI

Grid blocks distribution

Blocks balance across 8 processors



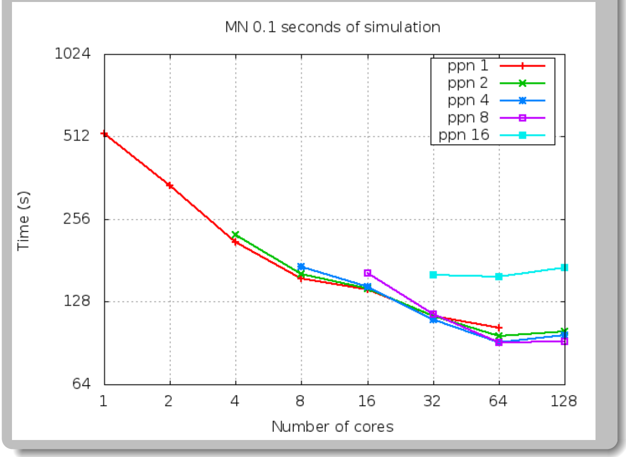
Notes

- Blocks are being created and destroyed dinamycally
- Number of blocks in each processor well balanced throughout

Adaptive mesh example: RTI

Parallel performance

Scalability in MN



Notes

- Poor scalability in single node due to cache misses (also present in serial and MPI versions)
- Poor scalability across nodes due to high communication requirements
- Cache misses have even bigger impact than network communication
- Parallel performance similar to that of own MPI version

On-going work

Variable Timestep

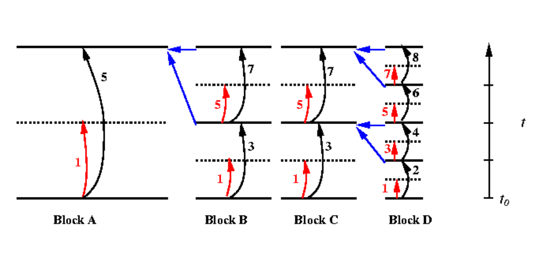


Figure: Variable timestep flow-chart (from http://www.physics.drexel.edu/~olson/paramesh-doc/Users_manual/amr_variable_dt.html)

Notes

- Need loads of extra storage
- Flow-chart can get quite complex
- Now not all blocks do the same work: block balance?

Conclusions

- Adding Paramesh not too difficult (specially if starting from scratch)
- CPU overhead quite large, so we need to be carefull when defining the refinement function if we are to get performance gains
- Memory overhead not too large for constant timestep, but quite large for variable timestep
- Minor drawback: blocks are not dinamycally allocated, but rather fixed in amr_runtime_parameters
- Paramesh has other features that we didn't discuss: divergenceless prolongation, conservation laws, checkpointing, ...

References

- Adaptive mesh refinement for hyperbolic partial differential equations - M. Berger, J. Oliger - J. Comp. Phy. 1984 http://www.sciencedirect.com/science/article/pii/0021999184900731
- Local adaptive mesh refinement for shock hydrodynamics - M. Berger, P. Colella - J. Comp. Phy. 1989 http://www.sciencedirect.com/science/article/pii/0021999189900351
- Paramesh V4.1 webpage: http://www.physics.drexel.edu/õlson/paramesh-doc/Users_manual/amr.html