

# AMR on Structured Meshes and PARAMESH

Peter MacNeice (Drexel University)

[pmacneic@pop900.gsfc.nasa.gov](mailto:pmacneic@pop900.gsfc.nasa.gov)

Room W205

Test calculation – mach 3 wind tunnel  
flow over a step (Courtesy of Univ. of  
Chicago, FLASH group)

Test calculation with grid block  
outlines (Courtesy of University of  
Chicago, FLASH Group)

# Overview

- Why AMR ?
- Different AMR approaches
- Block-based AMR
- Introduction to PARAMESH

## When is AMR appropriate ?

AMR is appropriate for problems that exhibit regions of fine scale structure surrounded by large regions where the solution is smooth.

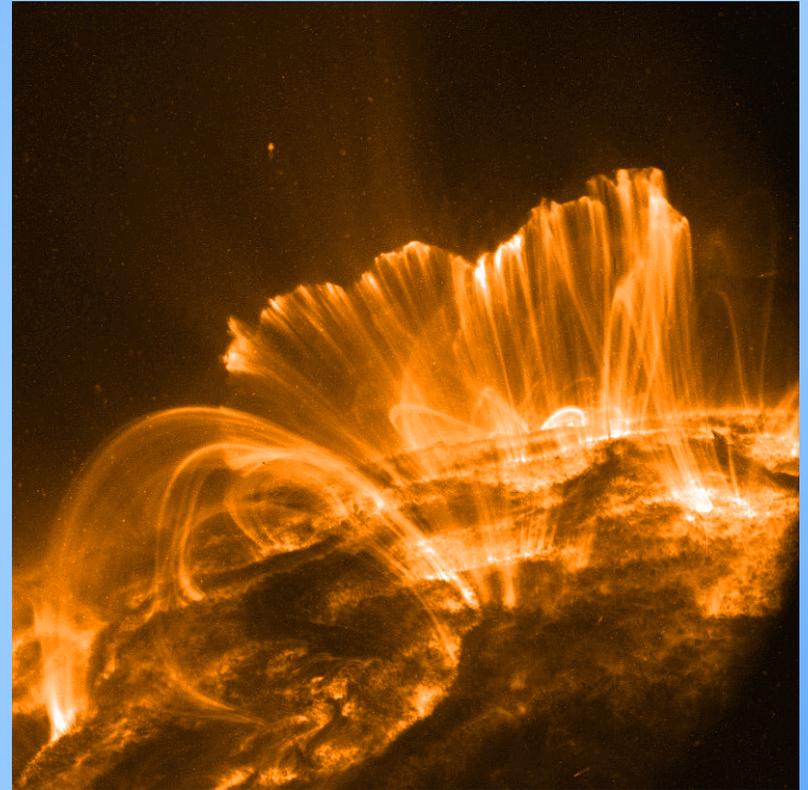
- Hydro or MHD with shocks and/or interfaces
- Self gravitating flows
- Complex engineering geometries
- Combustion

# Why do we want to do this?

Many science applications demand very large dynamic ranges in spatial resolution.

eg. Modeling the influence of Coronal Mass Ejections on the interplanetary medium and the Earth's magnetosphere.

(large CME energy equivalent to a 10km asteroid colliding with earth at  $30\text{km.s}^{-1}$ , or 15 billion Hiroshima bombs)



CMEs - coronagraph view

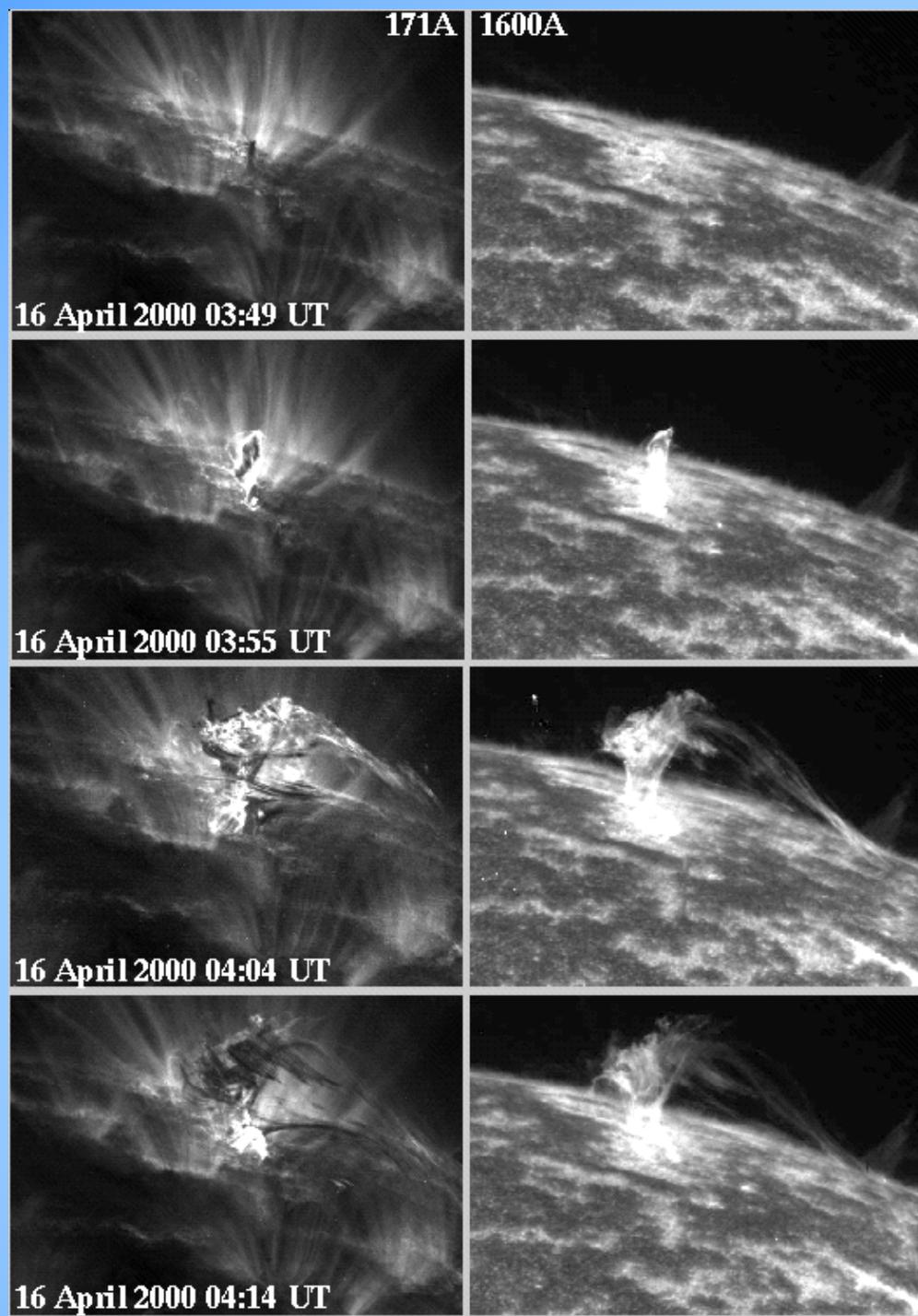
(Courtesy of SOHO consortium. SOHO is a project of international cooperation between NASA and ESA.)

Active Regions

High Resolution view

(Courtesy of TRACE instrument team)

# Filament Eruption at start of a coronal mass ejection



Images courtesy of SOHO.

# More TRACE Images

Trace movie 1 : activity on disk

Trace movie 2 : filament eruption on limb

# AMR approaches

Fixed number of mesh cells

- Lagrangian
- Rezoning

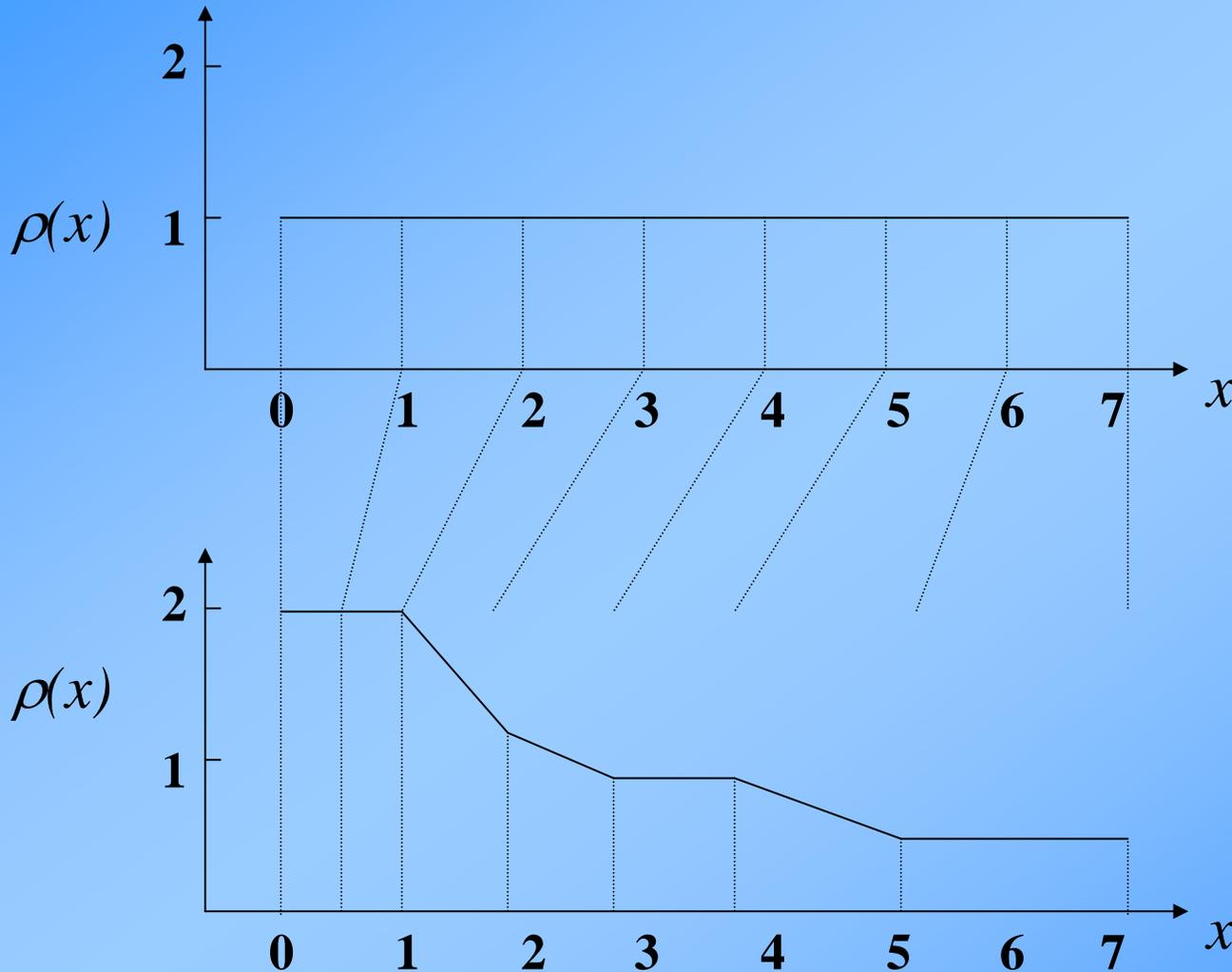
Variable number of mesh cells

- Unstructured finite element
- Structured cell-by-cell
- Structured sub-grid blocks

Composite meshes

# Lagrangian Methods

Grid moves with the fluid flow.



**Grid moves in a way such that the integral of mass in each cell remains constant.**

# Lagrangian Methods

## Advantages

- Lagrangian schemes are often less dissipative than Eulerian schemes for hydro.
- Fixed memory size.
- Conservation laws are easy to enforce.

## Disadvantages

- Refinement control is somewhat clumsy. Needs modification if anything other than mass flow is to control refinement.
- Mesh can get hopelessly tangled in 2 and 3D. A variant called the ‘free lagrange method’ is designed to solve this problem.
- Good for front tracking problems, but has problems with multiple interacting fronts.

# Rezoning (penalty methods)

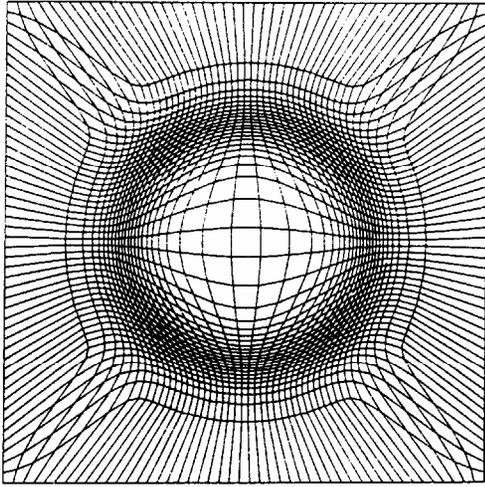
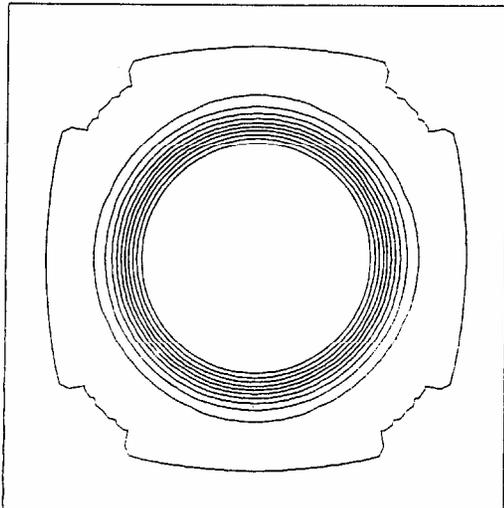


FIGURE 12

Adjust a fixed number of grid cells using some weighting function to specify relative mesh cell sizes.

Eg. Brackbill and Saltzman (1982) – Variational approach : have to solve a set of Euler equations which define the coordinate transformation



# Rezoning (penalty methods)

Transformation  $(x, y) \rightarrow (\xi, \eta)$  with Jacobian

$$J = \frac{\partial x}{\partial \xi} \frac{\partial y}{\partial \eta} - \frac{\partial x}{\partial \eta} \frac{\partial y}{\partial \xi}$$

Choose

$$\Delta \xi = \Delta \eta = 1$$

Smoothness of mapping:

$$I_s = \int_D (\nabla \xi)^2 + (\nabla \eta)^2 dV$$

Orthogonality of mapping:

$$I_o = \int_D (\nabla \xi \cdot \nabla \eta)^2 J^3 dV$$

Volume weighting using a weighting function  $w(x, y)$  :

$$I_w = \int_D w J dV$$

Choose a mapping which minimizes

$$I = I_s + \lambda_o I_o + \lambda_w I_w$$

Mapping is the solution to the Euler equations for this variational problem.

$$a_1 x_{\xi\xi} + b_1 x_{\xi\eta} + c_1 x_{\eta\eta} + d_1 y_{\xi\xi} + e_1 y_{\xi\eta} + f_1 y_{\eta\eta} = S_1$$

$$a_2 x_{\xi\xi} + b_2 x_{\xi\eta} + c_2 x_{\eta\eta} + d_2 y_{\xi\xi} + e_2 y_{\xi\eta} + f_2 y_{\eta\eta} = S_2$$

For example

$$a_1 = (y_\xi^2 + y_\eta^2)(x_\eta^2 + y_\eta^2)/J^3 + \lambda_o x_\eta^2 + \lambda_w 2w y_\eta^2$$

$$x_\xi = \frac{\partial x}{\partial \xi}$$

# Rezoning (penalty methods)

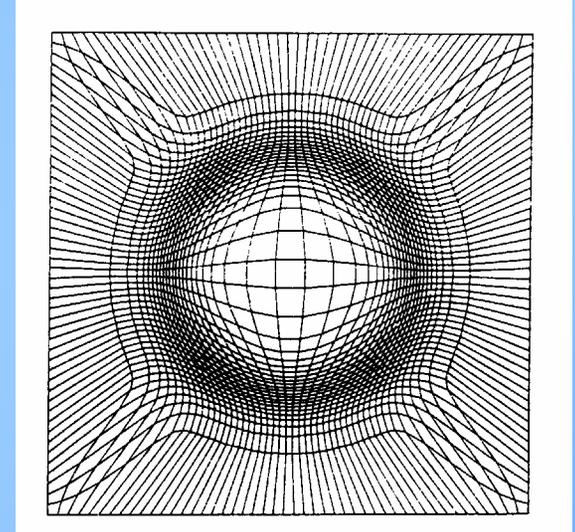
## Advantages

- Fixed memory size.
- Good for fitting grid to complex boundary shapes.
- Good cache line reuse.
- Conservation laws are easy to enforce.

# Rezoning (penalty methods) contd.

## Disadvantages

- Difficult to maintain smooth grids (eg rapidly varying cell sizes, anisotropic cell shapes).
- May need to solve an additional elliptic equation(s) to determine the new grid.
- Accuracy concerns associated with shock-capturing schemes for strong shocks where mesh cell sizes change too quickly.
- Time variation of the mesh is non-trivial.



# Unstructured Finite Element

Credit –Myers,  
Baptista and Priest

Tsunami modeling  
due to subduction  
event along the  
NW coast.

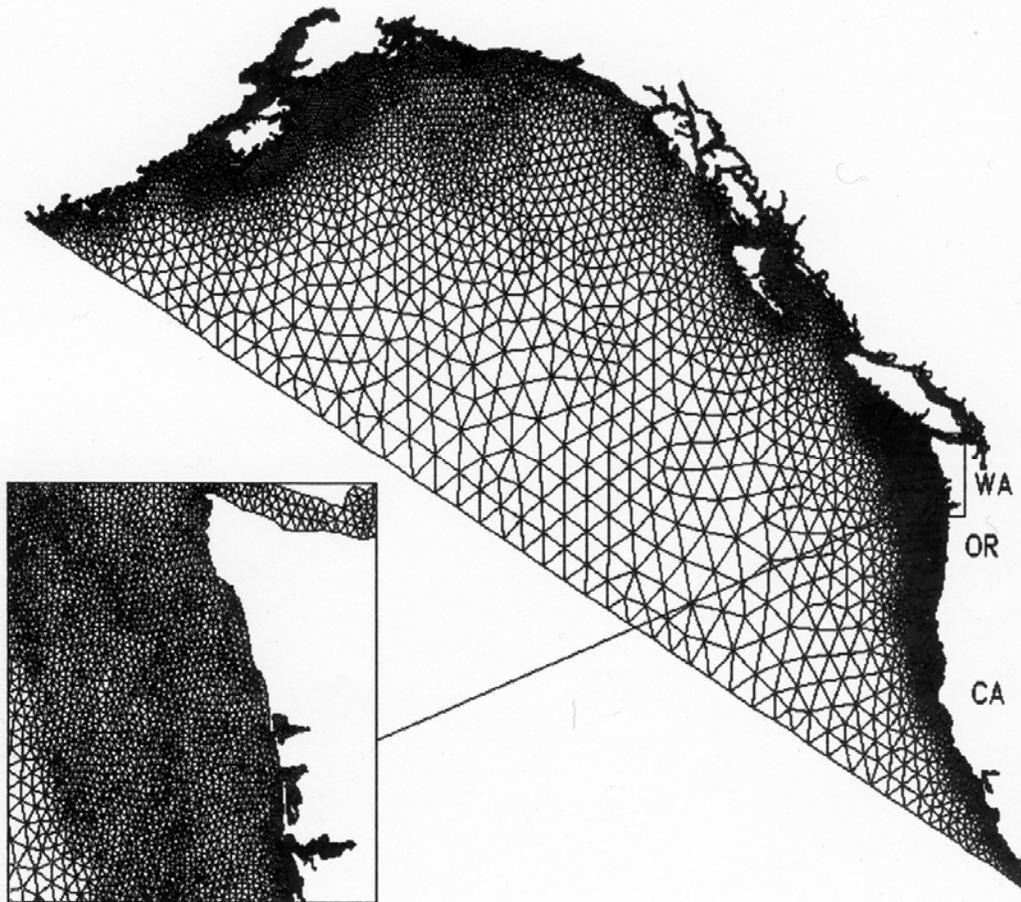


Figure 5a. Finite element grid 1.

# Unstructured Finite Element

## Advantages

- Most flexible refinement control.
- Best approach for complex boundary geometries.
- Conservation laws are easy to enforce.
- AMR doesn't modify the basic data-structures.

## Disadvantages

- Relatively slow in execution because of high levels of indirect data referencing.
- Requires significant memory for element and node locations and interconnectivity arrays.
- Generating 'good' grids is a difficult art.

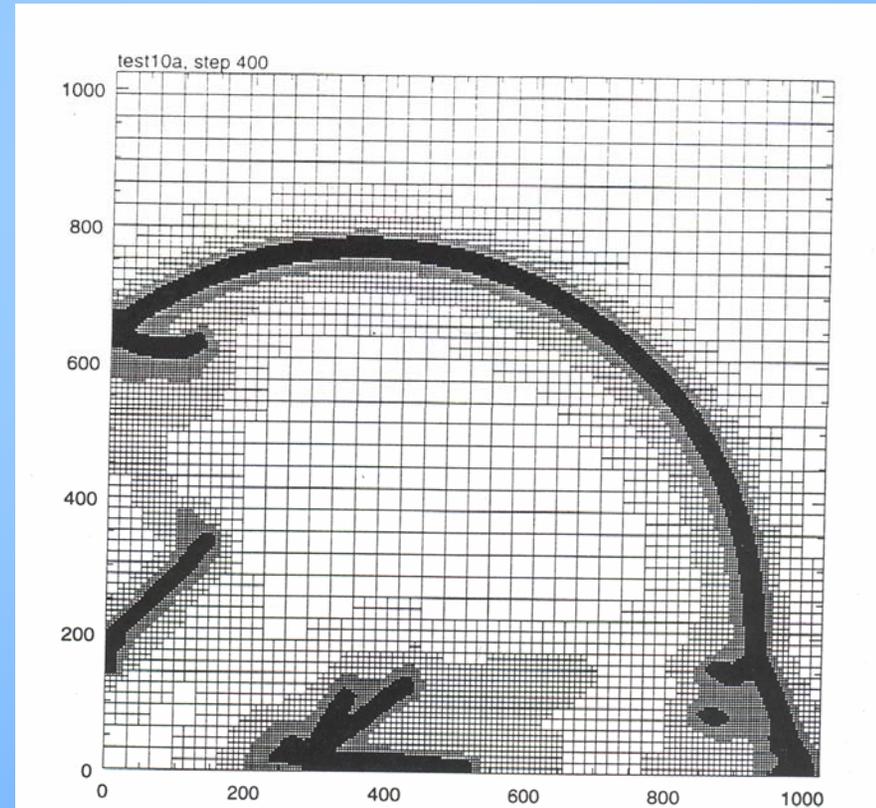
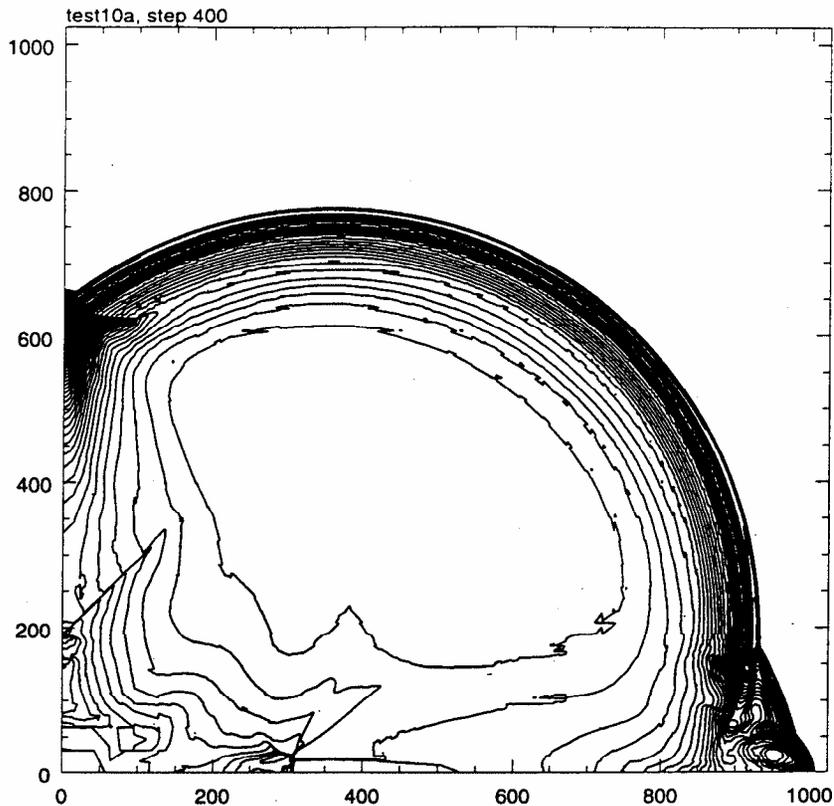
# Structured cell-by-cell

Equivalent to the unstructured FEM approach except that grid cells are rectangular in coordinate space, and you have to deal with hanging nodes.

eg Cart3D

<http://people.nas.nasa.gov/~aftosmis/cart3d>

<http://www.icemcfd.co.in/products/cart3d/cart3d.php3>



Credit - Khoklov

# Structured cell-by-cell

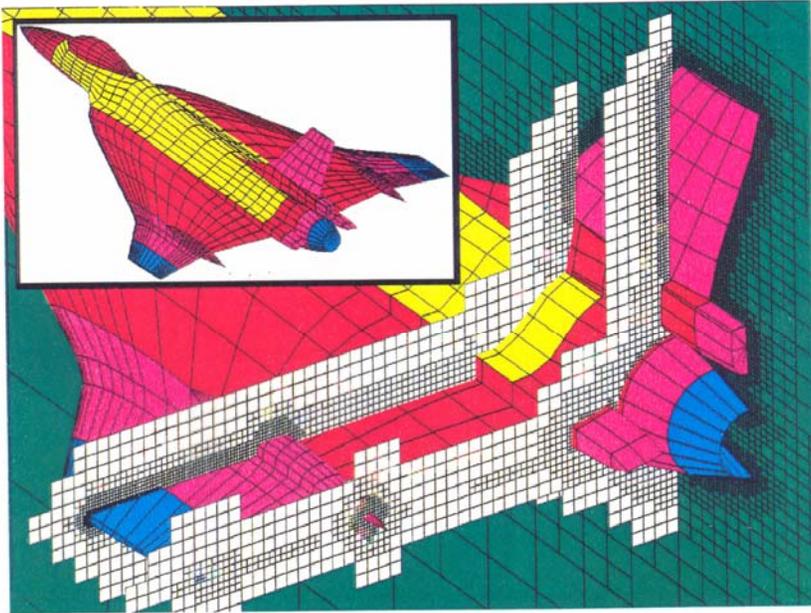
## Advantages

- Flexible and efficient refinement patterns.
- Low memory overhead.

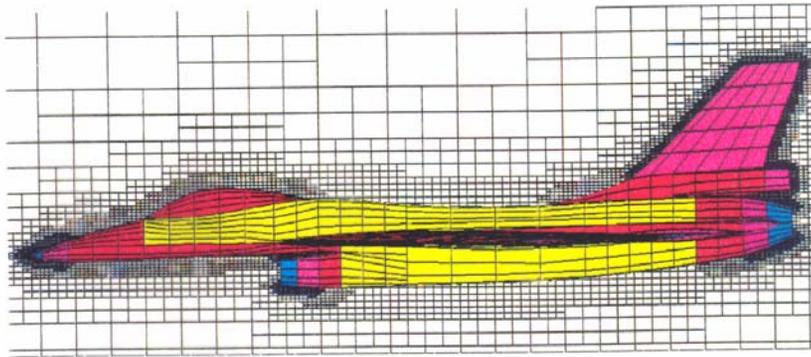
## Disadvantages

- Expensive general interpolation formulae or use of expensive non-uniform stencils for derivatives of required accuracy.
- Large tree data-structure to keep track of cells and their neighbors.
- Conservation laws require additional work at refinement jumps.
- Hard to deal with complex boundary shapes.

# Structured cell-by-cell



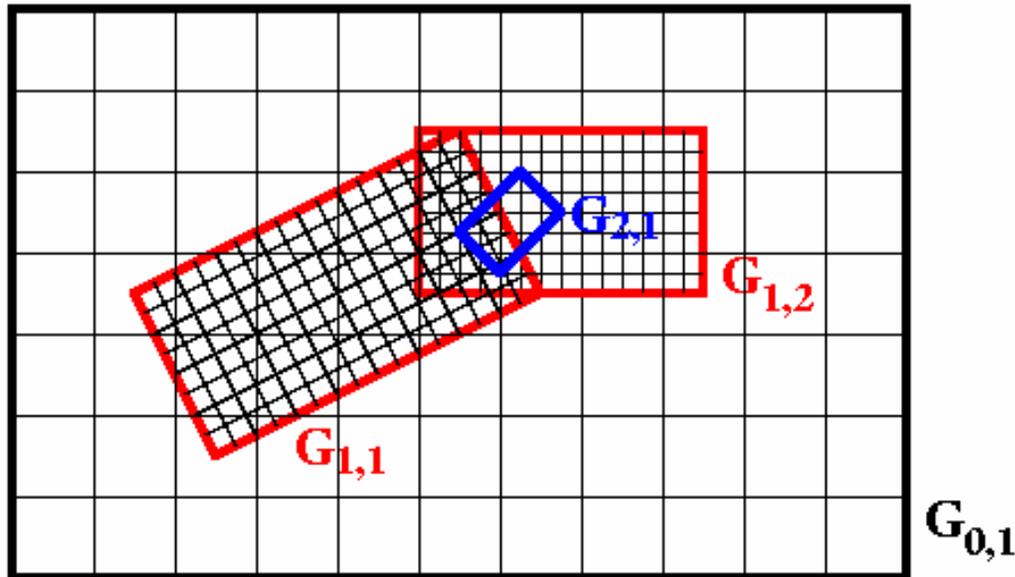
Credit – Berger and Melton



# Structured Sub-grids

Grid is a combination of ‘logically cartesian’ sub-grids at various refinement levels, overlaid so as to cover the computational domain with the desired resolution.

Pioneers of this approach – Berger and Oliger (1984) and Berger and Colella (1989), for hydrodynamic applications.

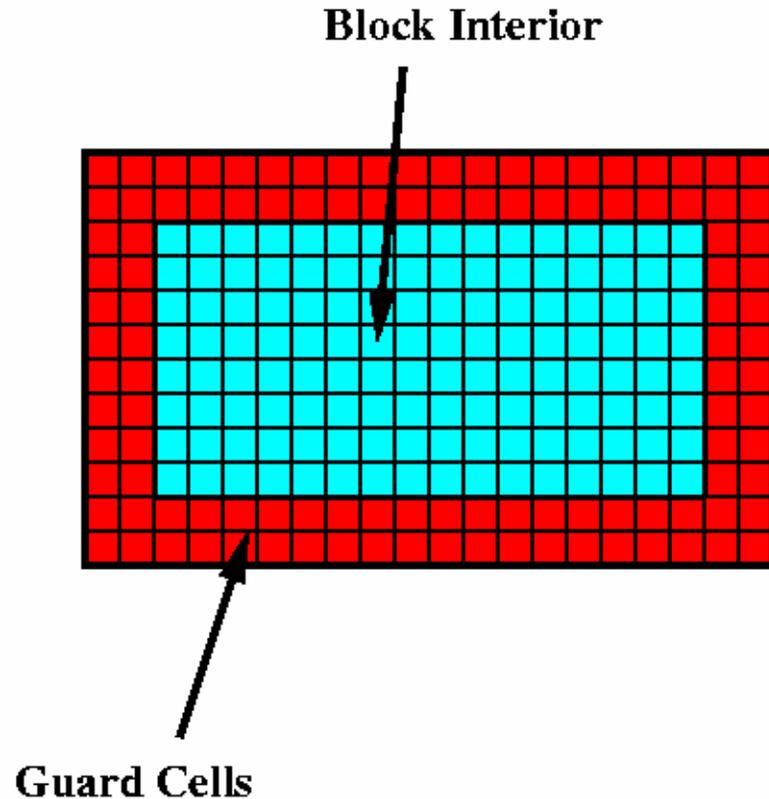


Uses a modified tree data-structure to manage the grids.

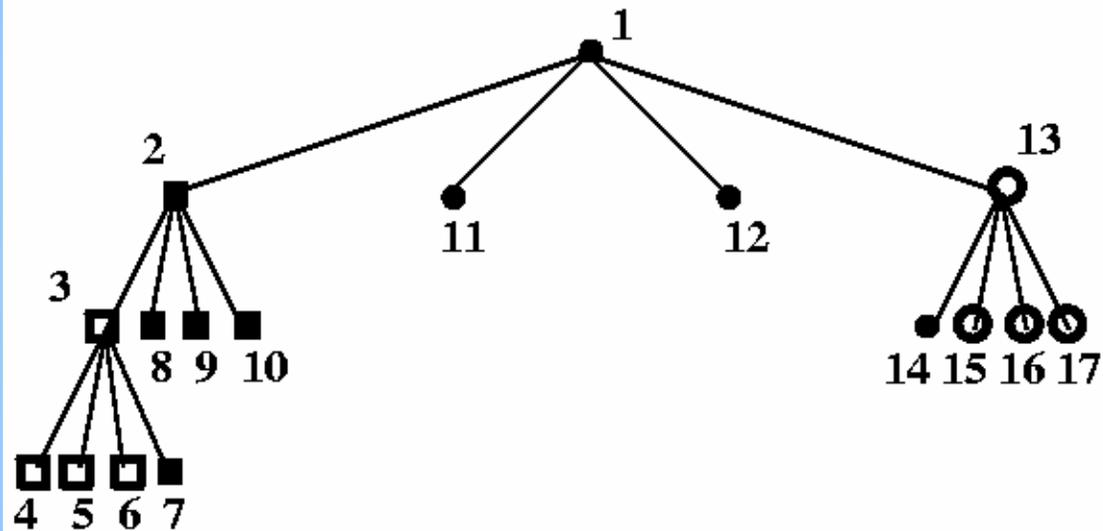
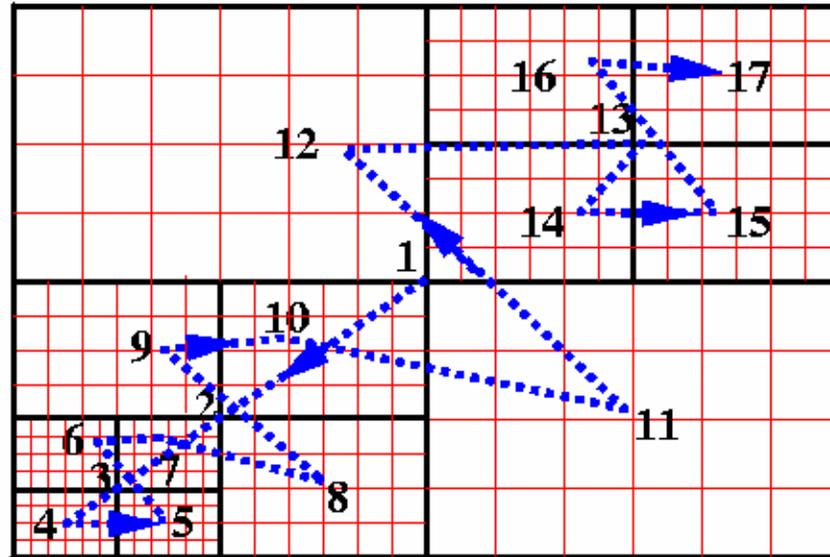
Refinements must be properly nested.

# Structured Sub-grids

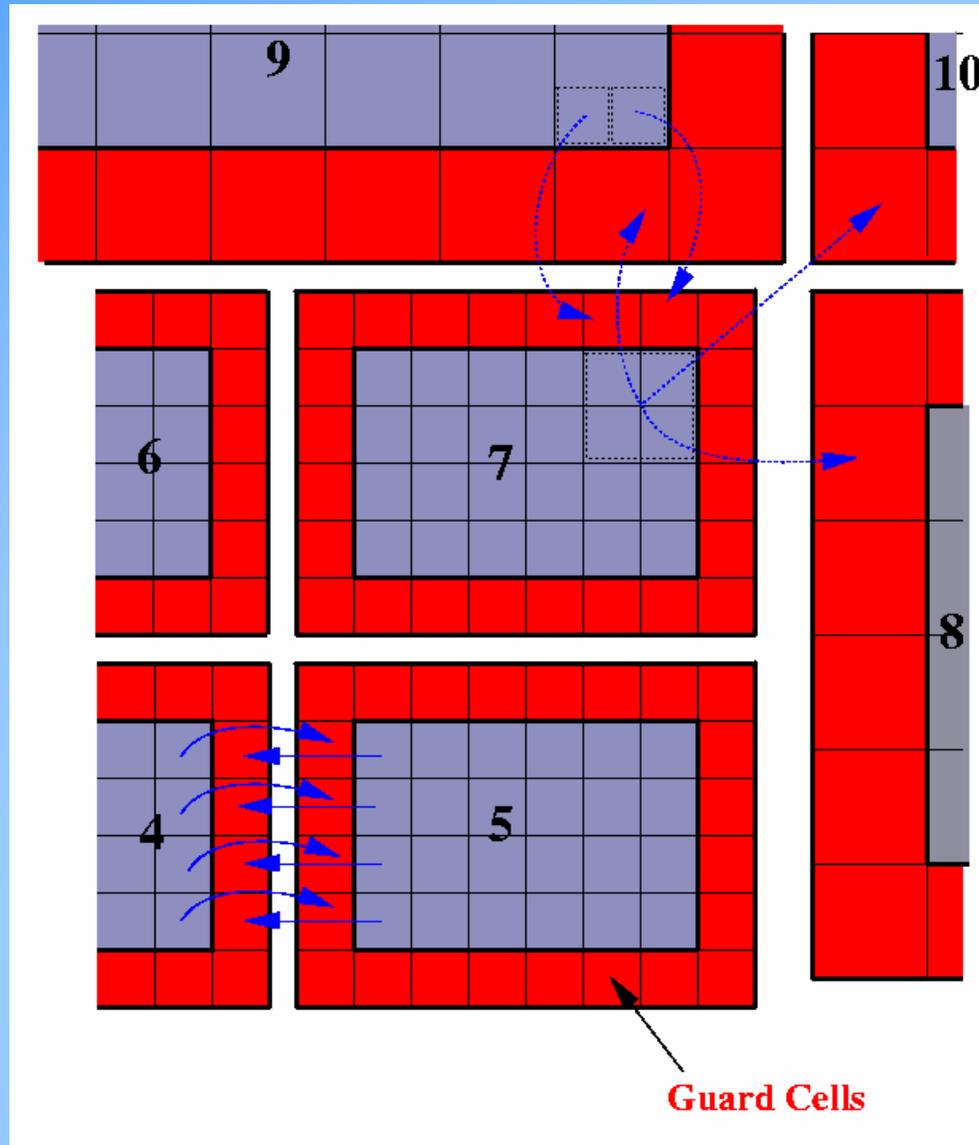
Sub-grid blocks are surrounded by layer(s) of 'guardcells' which are filled with data from neighboring blocks.



# Structured Sub-grids



# Structured Sub-grids



# Structured Sub-grids (contd)

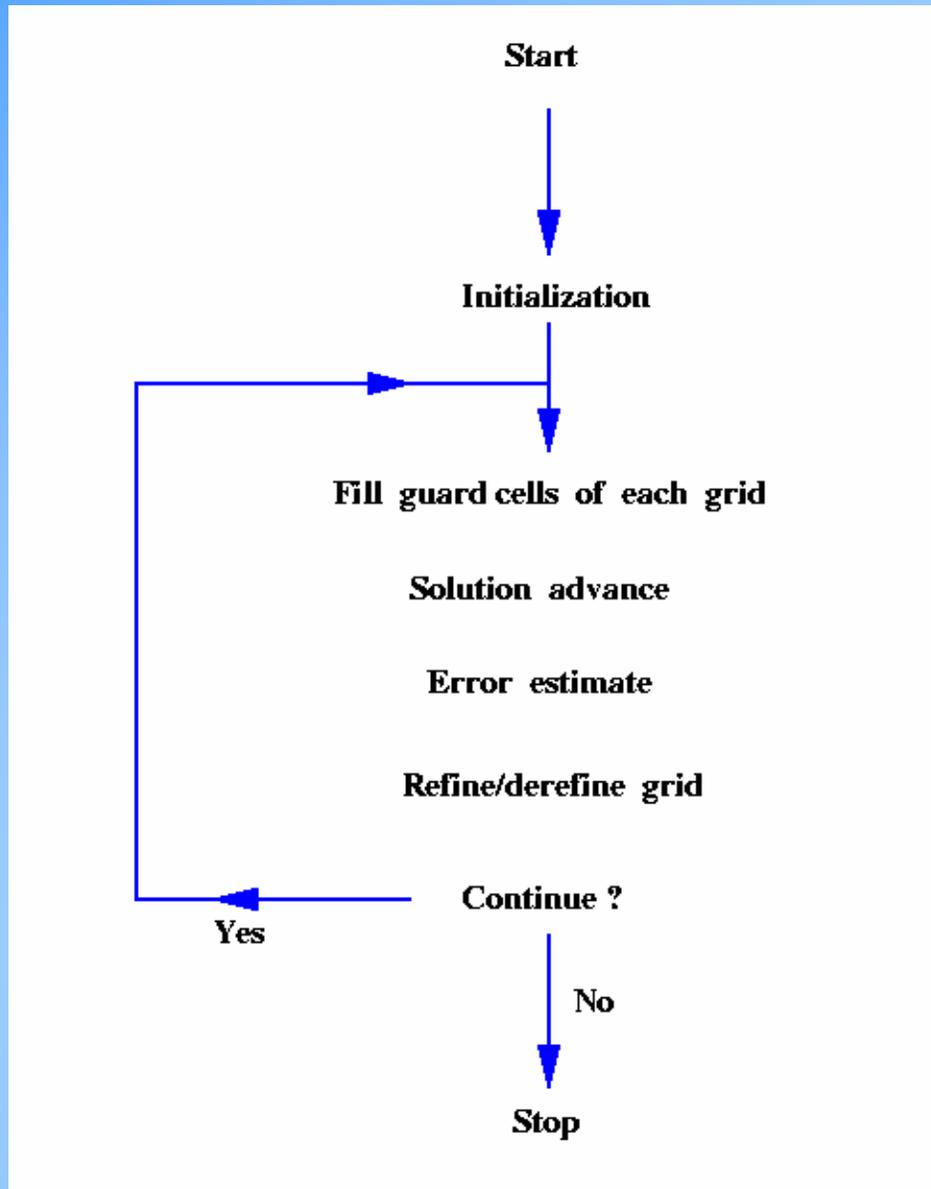
## Advantages

- Regular sub-grids produce efficient code.
- May enable reuse of most of a legacy code designed for uniform mesh.
- Convergence properties of algorithms are better understood on uniform meshes.
- Shock capturing schemes work better on uniform meshes.

## Disadvantages

- Hard to deal with complex boundary shapes.
- Memory overhead associated with guardcells.
- Conservation laws require additional work at refinement jumps.

# Structured Sub-grids (contd)



# Structured Sub-grids (contd)

## Components of Block Structured AMR Codes

### Data Structures

- Solution
- Mesh

### Parallel Computing

- Load balancing
- Solution coherence

### Computational Tasks

- Guardcell filling
- Prolongation (coarse to fine)
- Restriction (fine to coarse)
- Error estimation
- Mesh rebuilding
- Supporting conservation law

# Structured Sub-grids (contd)

Berger and Olinger scheme.

- Supports different size sub-grids.
- Allows sub-grids to be rotated with respect to the coordinate axes (removed in Berger and Colella).
- Allows user to choose different integer refinement ratios (usually choose 4).
- Refines integration timestep along with grid spacing.
- Can combine sub-grids at the same refinement level if it would be more efficient.
- Advances the solution on all grids.
- Restricts finest grid solution by conservative averaging onto underlying coarse grids at every synchronization time.
- Does convergence test to estimate error.

# Structured Sub-grids (contd)

## Numerical Analysis Issues

- Accuracy
- Stability

Block structured grid is like a regular structured grid, except that the locations where the grid resolution jumps, act as internal free boundaries.

All the critical design issues are associated with how we handle information at these refinement jumps.

# Structured Sub-grids (contd)

## Stability

Stability analyses are few and far between, and for very restrictive algorithm designs.

Eg Berger (Ph.d. thesis) – Lax-Wendroff

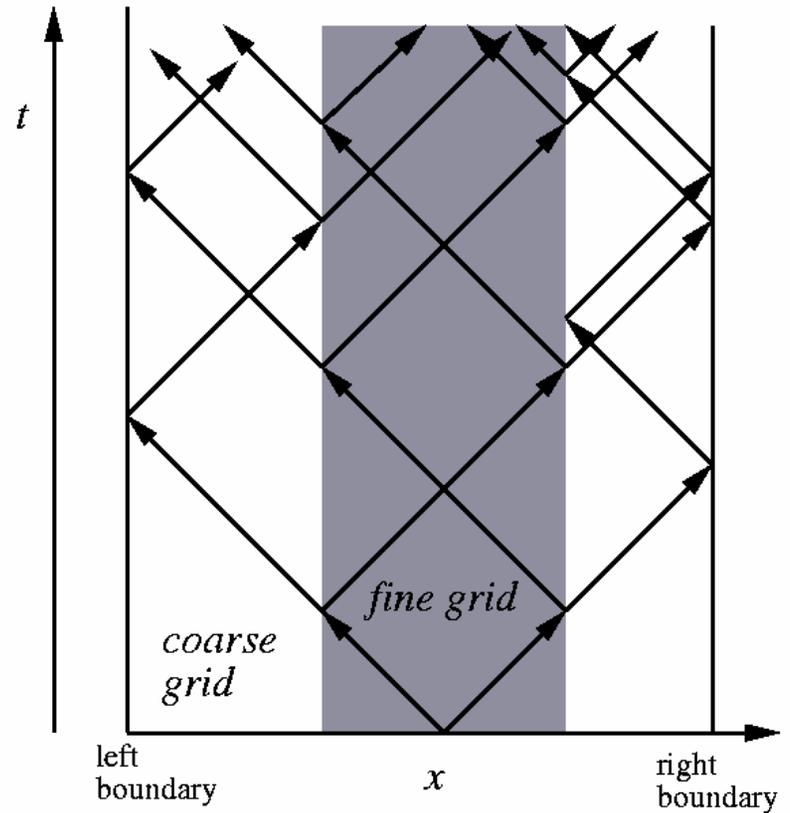
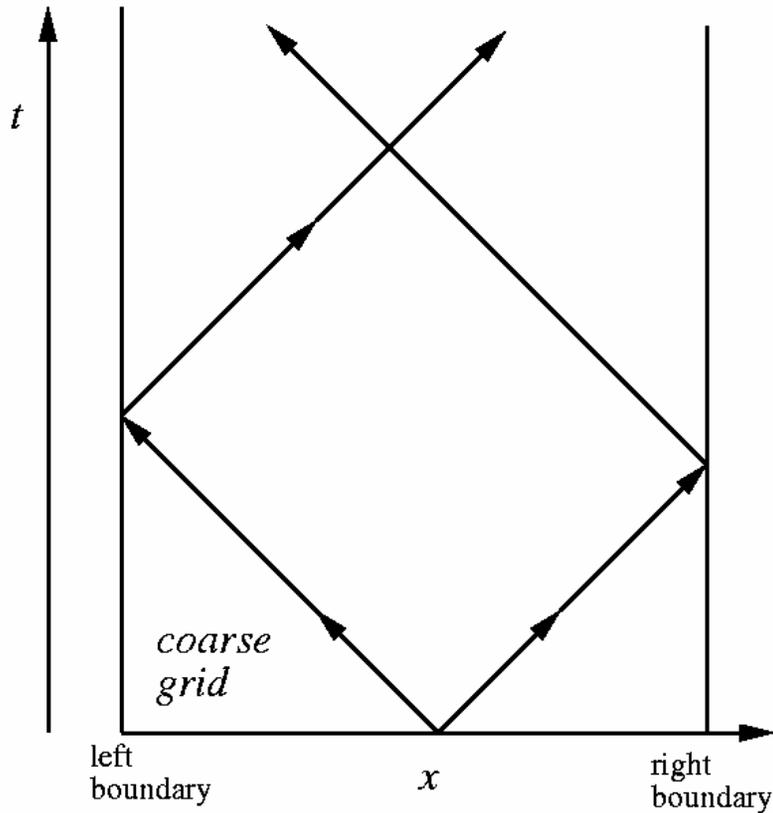
Trefethen (1985) - leapfrog

Trefethen considers block-structured AMR as a multi-interface model.

“Any multi-interface model potentially admits trapped numerical waves that reflect back and forth repeatedly from one interface to another. If the reflections cause amplification, this can lead to unbounded growth of numerical solutions. The factors that control this are: magnitude of the reflection coefficients, which is the source of growth; dissipation of waves as they travel between interfaces, which is a source of attenuation; and travel time between interfaces, which determines how frequently any reflection circuit that causes growth is repeated.”

# Structured Sub-grids (contd)

Wave reflections from refinement jumps



# Structured Sub-grids (contd)

**Stability is enhanced by :**

- Large sub-grid blocks
- Dissipative algorithm
- Small reflection coefficients.

**Reflection coefficients are influenced by :**

- The size of the refinement jump
- The interpolation schemes used to compute data in guardcells at refinement discontinuities.

In reality, stability analyses are only useful as a crude general guide. Stability for complex algorithms is established by coding them up and trying them.

For example, extensive experience has established that for standard interpolation formulae, upwinded hydro schemes are stable on these AMR grids, as long as you respect the normal Courant condition timestep limits.

# Structured Sub-grids (contd)

## Accuracy at refinement jumps

Refinement jumps reduce the order of the differencing scheme by at best 1 order.

Example 1 : 1D algorithm needing first derivative.

Example 2 : Divergence of face-centered fluxes in a conservative 2D algorithm.

Example 3 : Laplacian in 1D.

# Structured Sub-grids (contd)

## Example 1

Consider a spatially 2nd order accurate scheme.

Suppose our algorithm depends on  $\partial f/\partial x$ .

Use of centered differencing and 1 guardcell at external boundaries maintains 2nd order accuracy on a uniform grid.

$$\frac{\partial f^N}{\partial x} = \frac{f^{N+1} - f^{N-1}}{2\Delta x} + \mathcal{O}(\Delta x^2).$$

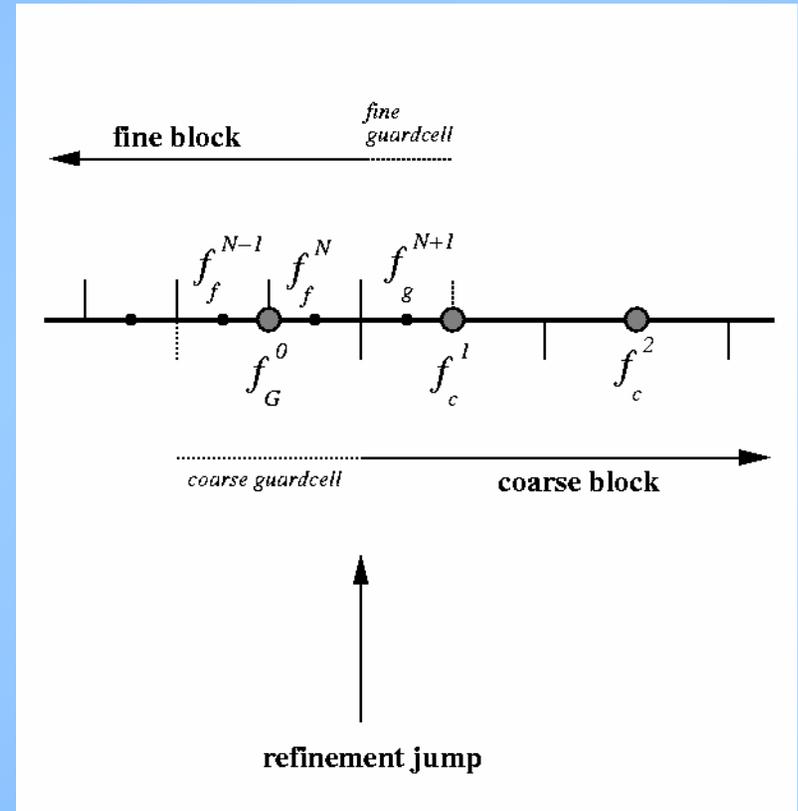
But at refinement jump  $f^{N+1} \equiv f_G$  must come from guardcell values resulting from interpolation.

Linear Interpolation.

$$f_G = \frac{1}{2}(f^{N-1} + f^N) + \mathcal{O}(\Delta x^2)$$

$$f_g = \frac{1}{3}(f^N + 2f^1) + \mathcal{O}(\Delta x^2)$$

RESULT - Reduction in order of accuracy by 1 order!



# Structured Sub-grids (contd)

## Example 2

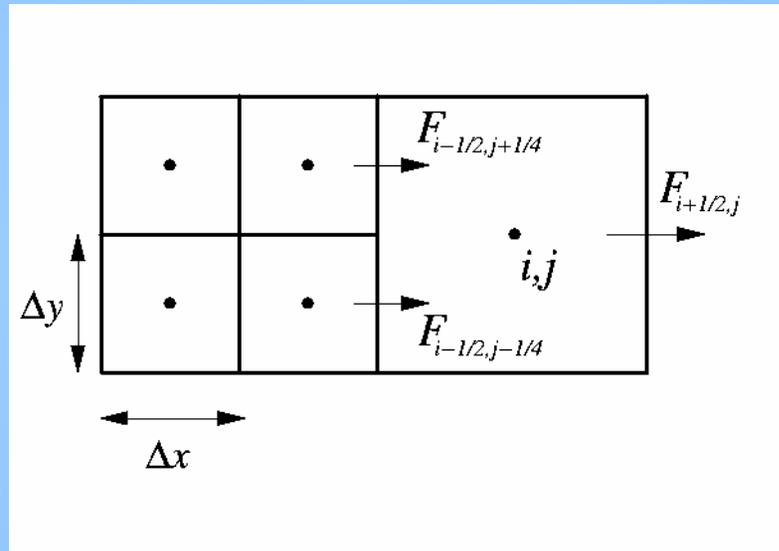
The divergence operator for conservative hydro.

Solving a conservation equation

$$\frac{\partial U}{\partial t} + \nabla \cdot F = 0.$$

Evaluate  $\partial F / \partial x$  for the coarse cell.

$$\begin{aligned} \frac{\partial F}{\partial x} &= \frac{1}{\Delta x} \left[ F_{i+1/2,j} - \frac{1}{2}(F_{i+1/2,j-1/4} + F_{i+1/2,j+1/4}) \right] + \mathcal{O}(\Delta x^2) \\ &= \frac{1}{\Delta x} \left[ F_{i+1/2,j} - \left( F_{i-1/2,j} + \mathcal{O}(\Delta y^2) \right) \right] + \mathcal{O}(\Delta x^2) \\ &= \frac{1}{\Delta x} \left[ F_{i+1/2,j} - F_{i-1/2,j} \right] + \mathcal{O}\left(\frac{\Delta y^2}{\Delta x}\right) + \mathcal{O}(\Delta x^2) \end{aligned}$$



# Structured Sub-grids (contd)

## Example 3

Consider second order approximation for Laplacian.

$$\frac{\partial^2 \phi^N}{\partial x^2} = \frac{\phi_f^{N+1} - 2\phi_f^N + 2\phi_f^{N-1}}{\Delta x_f^2} + \mathcal{O}(\Delta x_f^2).$$

But  $\phi_f^{N+1}$  must come from interpolation.

If we use linear interpolation then

$$\phi_f^{N+1} \equiv \phi_g = \frac{1}{3}(\phi_f^N + 2\phi_c^1) + \mathcal{O}(\Delta x_f^2)$$

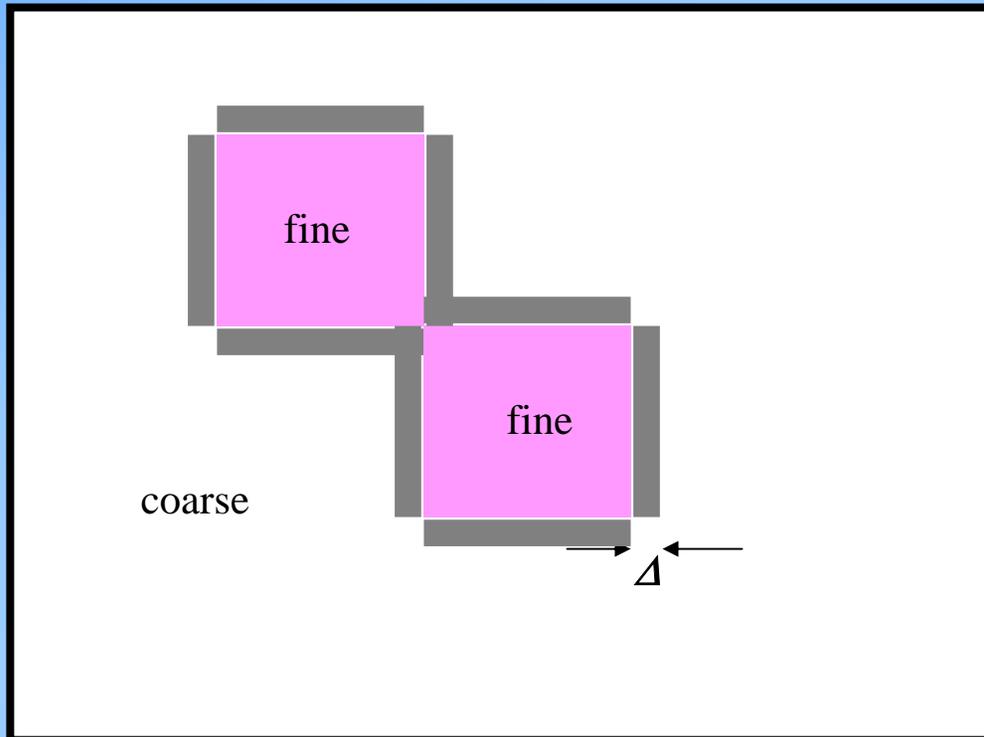
So

$$\frac{\partial^2 \phi^N}{\partial x^2} = \frac{(\phi_f^N + 2\phi_c^1)/3 - 2\phi_f^N + 2\phi_f^{N-1}}{\Delta x_f^2} + \mathcal{O}(\Delta x_f^0).$$

# Structured Sub-grids (contd)

## Accuracy :

Loss of accuracy is not a problem at refinement jumps if we lose no more than one order because the loss of accuracy occurs in a volume which is of width  $\Delta$ , and so it will not dominate the global error measure.



# Structured Sub-grids (contd)

## Accuracy at refinement jumps – Linear Wave equations

Example, a wave equation in 1D.

Want a spatially 2nd order accurate scheme.

$$\frac{\partial^2 \phi}{\partial t^2} = \frac{\partial^2 \phi}{\partial x^2}$$

or

$$\begin{aligned}\frac{\partial \phi}{\partial t} &= \Pi \\ \frac{\partial \Pi}{\partial t} &= \nabla \cdot \nabla \phi\end{aligned}$$

← Divergence of flux

Use of centered differencing and 1 guardcell at external boundaries maintains 2nd order accuracy on a uniform grid.

$$\frac{\partial^2 \phi^N}{\partial x^2} = \frac{\phi_f^{N+1} - 2\phi_f^N + 2\phi_f^{N-1}}{\Delta x_f^2} + \mathcal{O}(\Delta x_f^2).$$

At refinement jump  $\phi_f^{N+1} \equiv \phi_g$ .

# Structured Sub-grids (contd)

Consider 3 different guardcell filling choices.

Linear Interpolation.

$$\phi_G = \frac{1}{2}(\phi_f^{N-1} + \phi_f^N) + \mathcal{O}(\Delta x_f^2)$$

$$\phi_g = \frac{1}{3}(\phi_f^N + 2\phi_c^1) + \mathcal{O}(\Delta x_f^2)$$

Indirect Linear Interpolation.

$$\phi_G = \frac{1}{2}(\phi_f^{N-1} + \phi_f^N) + \mathcal{O}(\Delta x_f^2)$$

$$\phi_g = \frac{1}{4}(\phi_G + 3\phi_c^1)$$

$$= \frac{1}{8}(\phi_f^{N-1} + \phi_f^N + 6\phi_c^1) + \mathcal{O}(\Delta x_f^2)$$

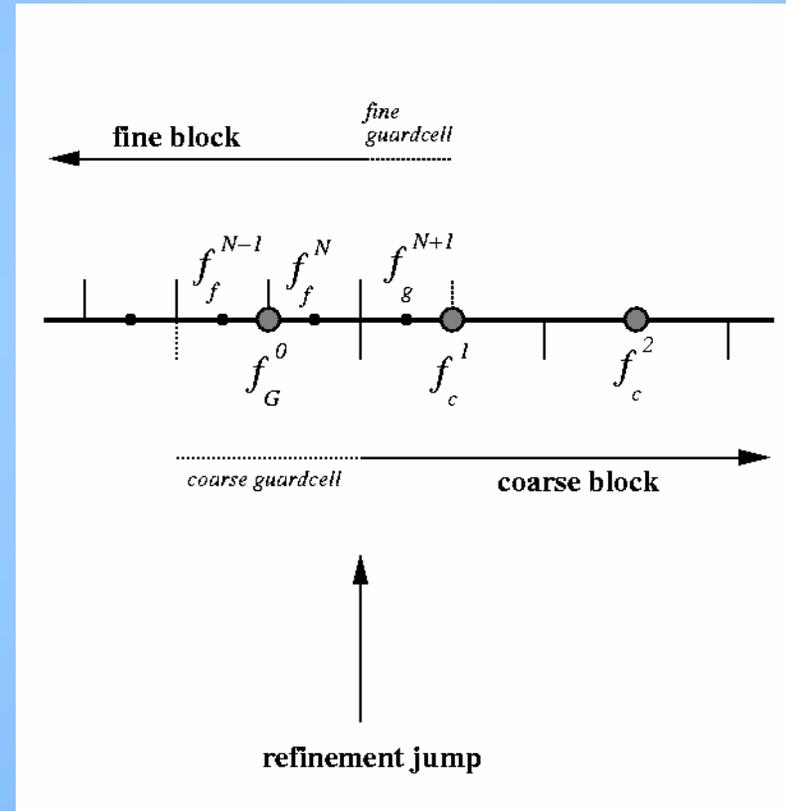
Direct Quadratic Interpolation.

$$\phi_g = \frac{1}{15}(-3\phi_f^{N-1} + 10\phi_f^N + 8\phi_c^1) + \mathcal{O}(\Delta x_f^3)$$

$$\frac{\phi_c^1 - \phi_G}{\Delta x_c} = \frac{\phi_g - \phi_f^N}{\Delta x_f}$$

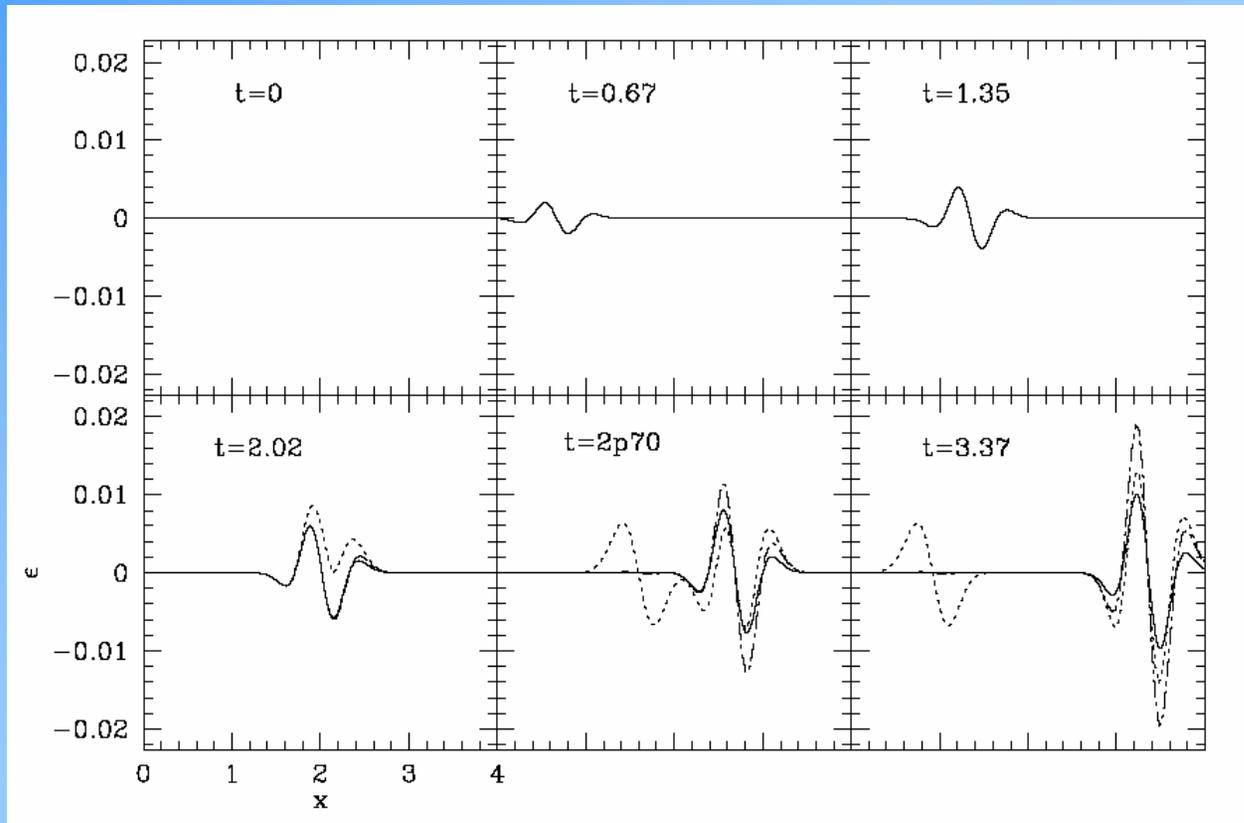
$$\phi_G = \frac{1}{15}(6\phi_f^{N-1} + 10\phi_f^N - \phi_c^1) + \mathcal{O}(\Delta x_f^3)$$

← Flux matching



# Structured Sub-grids (contd)

## Wave reflection at a refinement jump

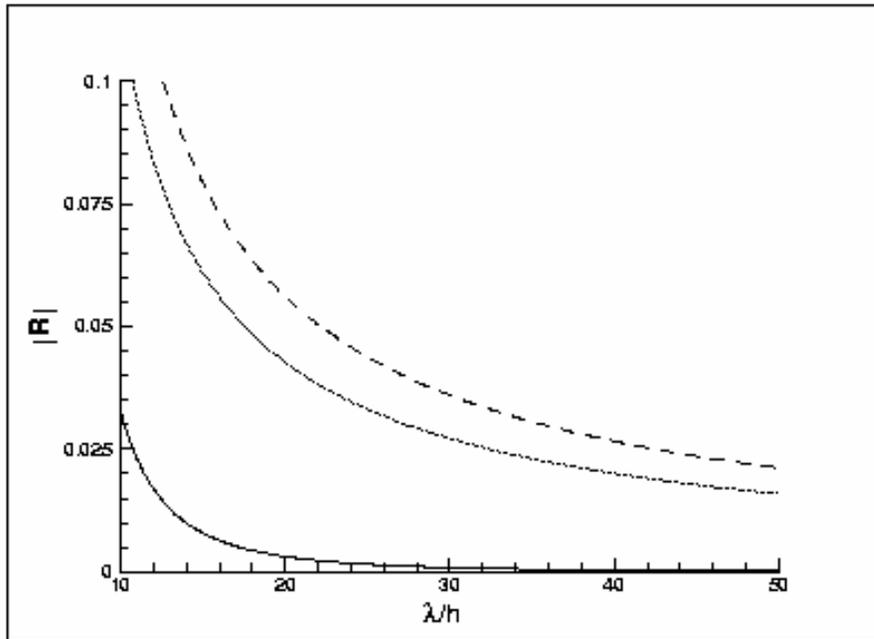


Choi et al (2004)

Absolute error  $\varepsilon$  – refinement jump at  $x = 2.1$

# Structured Sub-grids (contd)

Reflection coefficients for different guardcell filling strategies.



(Courtesy of Joan Centrella and Dale Choi)

# Structured Sub-grids (contd)

## ACCURACY AT REFINEMENT JUMPS - ELLIPTIC EQUATIONS

How do we solve an elliptic equation on a multi-level grid?

Solve  $\nabla^2 \psi^c = g^c$  on the coarse grid,  $\Omega^c$ .

Solve  $\nabla^2 \psi^f = g^f$  on the fine grid,  $\Omega^f$ , using coarse grid values to interpolate into fine grid guardcells.

Doesn't work!!! You get coarse grid accuracy even on the fine grid.

Must add the following Neumann gradient matching constraint on the solution at the interface  $\Omega^{c/f}$ , between the fine and coarse grid.

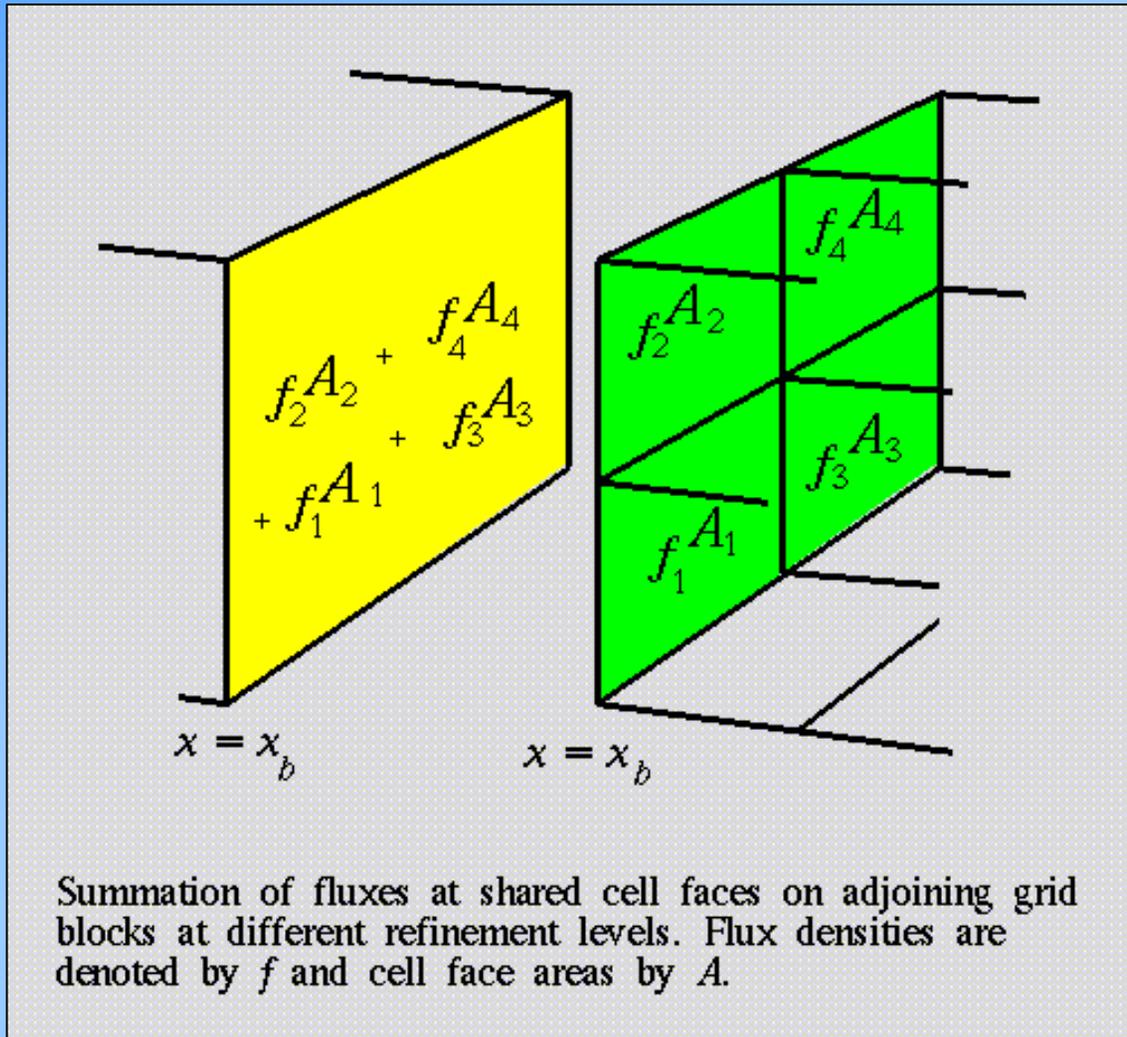
Flux matching



$$\begin{aligned}\psi^f - \psi^c &= 0 \\ \frac{\partial \psi^f}{\partial n} - \frac{\partial \psi^c}{\partial n} &= 0\end{aligned}$$

# Structured Sub-grids (contd)

Compressible Hydro - conservation is critical



# Structured Sub-grids (contd)

## Accuracy :

**Elliptic equations** – key is to ensure gradients are continuous across the refinement jumps.

## **Wave equations** –

- Difference based `fluxes' - key is to ensure gradients are continuous across the refinement jumps, which is equivalent to using quadratic interpolation.
- Godonov Hydro - key is to ensure consistent fluxes across refinement jumps.

# Structured Sub-grids (contd)

## Accuracy Summary

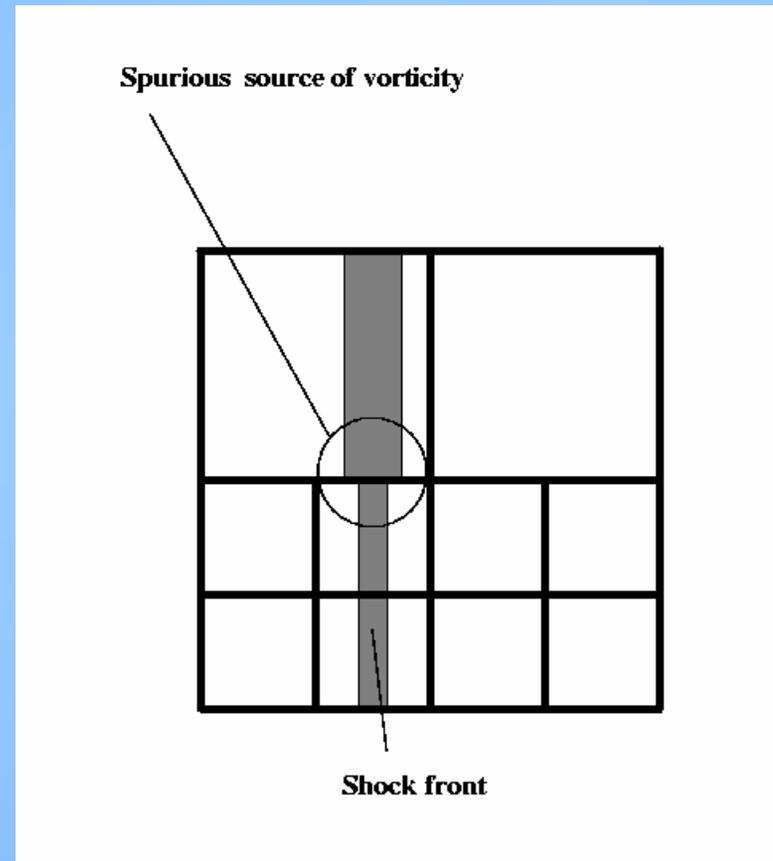
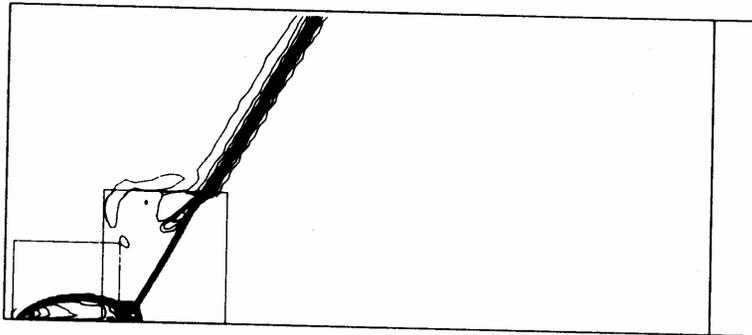
Common Theme - Key to accuracy is to ensure that 'flux-like' quantities are continuous across refinement jumps.

# Structured Sub-grids (contd)

Things **NOT** to do!

Don't place a refinement jump across a shock!

This will generate spurious numerical vorticity.



Credit – Berger and Colella

CT Summer School  
July, 2004

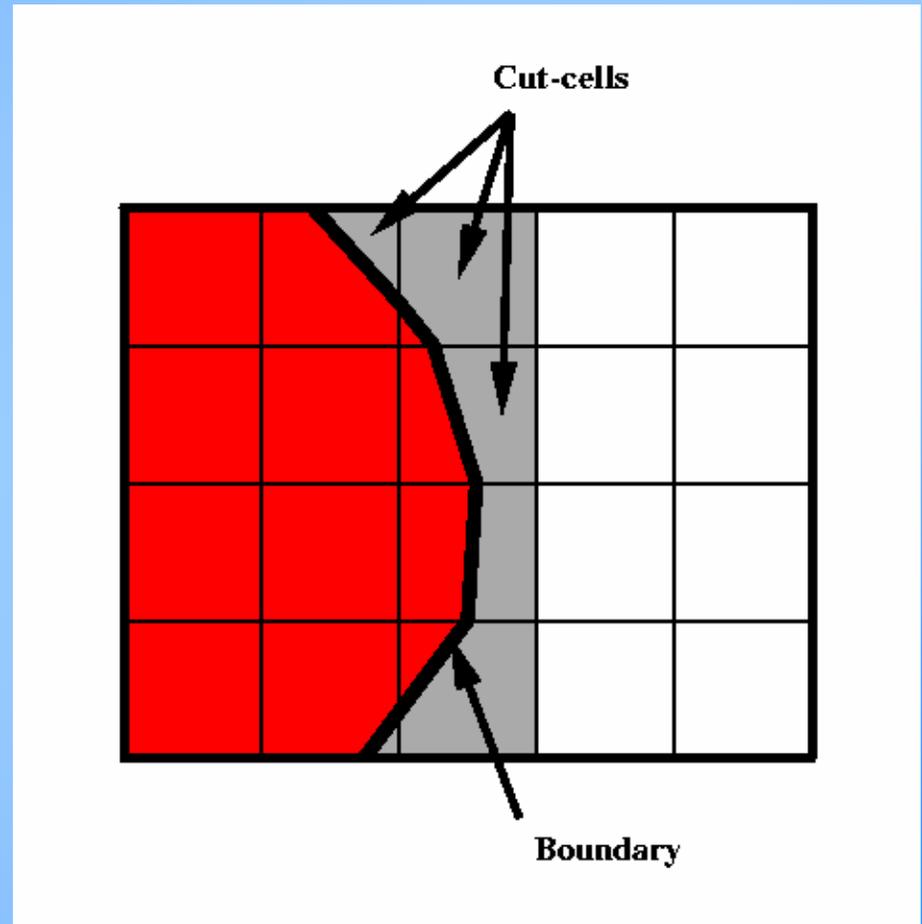
# Structured Sub-grids (contd)

## Problems with complex boundaries

How do we treat complex boundary shapes?

- Coordinate transformation
- Cut-cells

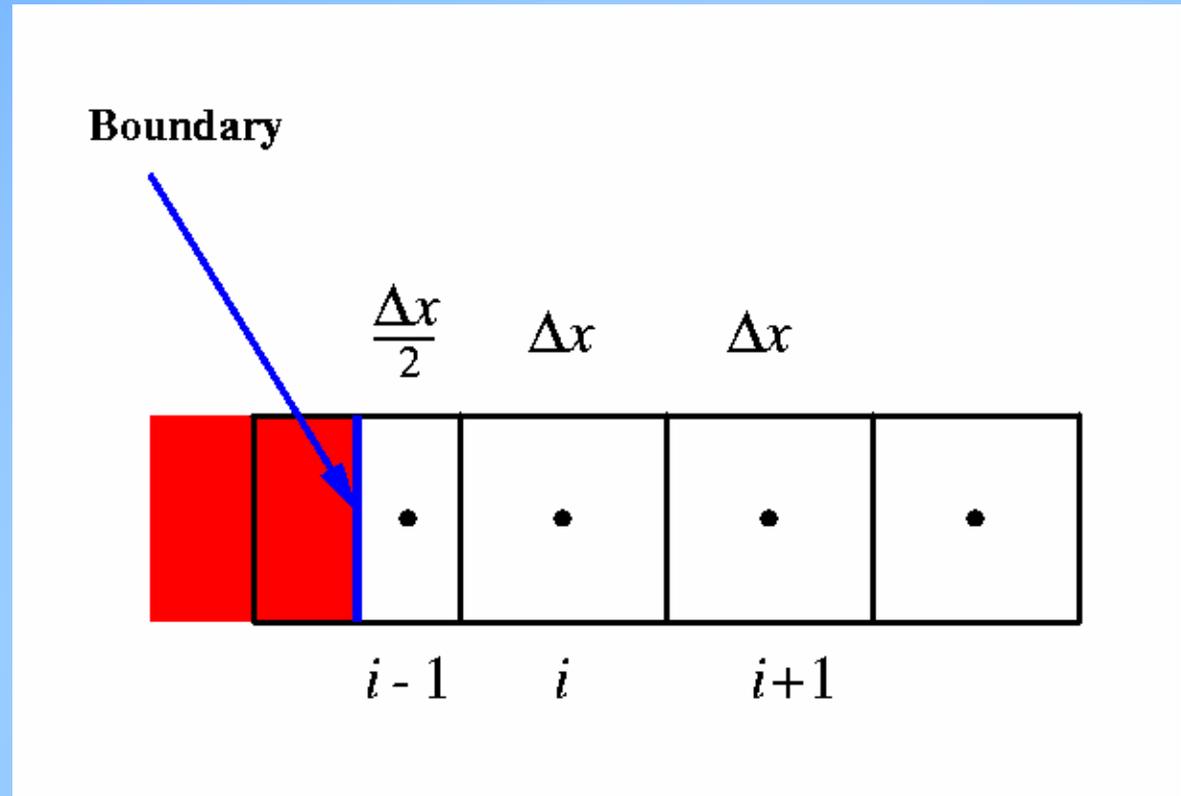
How do we maintain accuracy and stability, when some cells are tiny?



# Structured Sub-grids (contd)

## Accuracy near cut-cells

How is truncation error affected by the small cell at the boundary?



# Structured Sub-grids (contd)

Difference approximations must retain their accuracy even when tiny cells are next to regular sized cells.

eg.

Ignore fact that leftmost cell is cut in half.

$$\frac{du_i}{dx} = \frac{u_{i+1} - u_{i-1}}{2\Delta x} + \mathcal{O}(\Delta x^0)$$

Modify denominator for true distance between points  $i - 1$  and  $i + 1$ .

$$\frac{du_i}{dx} = \frac{u_{i+1} - u_{i-1}}{7\Delta x/4} + \frac{1}{8}\mathcal{O}(\Delta x^1)$$

Compute left and right slopes separately and then average.

$$\frac{du_i}{dx} = \frac{1}{2} \left( \frac{u_{i+1} - u_i}{\Delta x} + \frac{u_i - u_{i-1}}{3\Delta x/4} \right) + \frac{1}{16}\mathcal{O}(\Delta x^1)$$

Best approach is a weighted average of these slopes.

$$\frac{du_i}{dx} = \frac{4}{7} \left( \frac{3}{4} \left[ \frac{u_{i+1} - u_i}{\Delta x} \right] + \left[ \frac{u_i - u_{i-1}}{3\Delta x/4} \right] \right) \mathcal{O}(\Delta x^2)$$

**Accuracy near  
cut-cells**

# Structured Sub-grids (contd)

## Problems with complex boundaries

How is stability affected by the small cell at the boundary?

$$\left(\frac{\partial}{\partial t} + a\frac{\partial}{\partial x}\right)u = 0$$

Stability constraint on timestep is

$$\Delta t \leq \frac{\Delta x}{2a}$$

For a cell with very small  $\Delta x$ , timestep must be very small.

Two solutions:

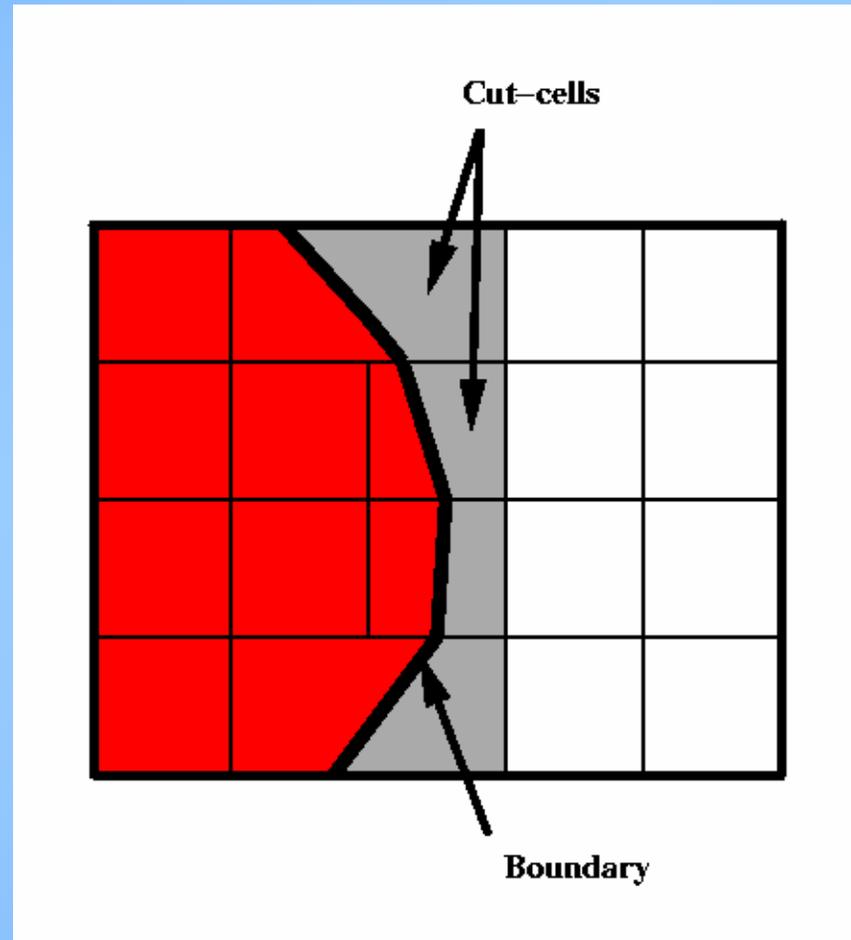
- Cell Merging - eliminates small cells
- Wave propagation approaches - apply scaled timestep  $\Delta t \Delta x_c / \Delta x$  to fluxes in cut-cells and redistribute remainder of 'true' fluxes between neighboring cells.

# Structured Sub-grids (contd)

## Problems with complex boundaries

Stability fix :

To avoid the Courant condition enforcing tiny timesteps, merge small cells.

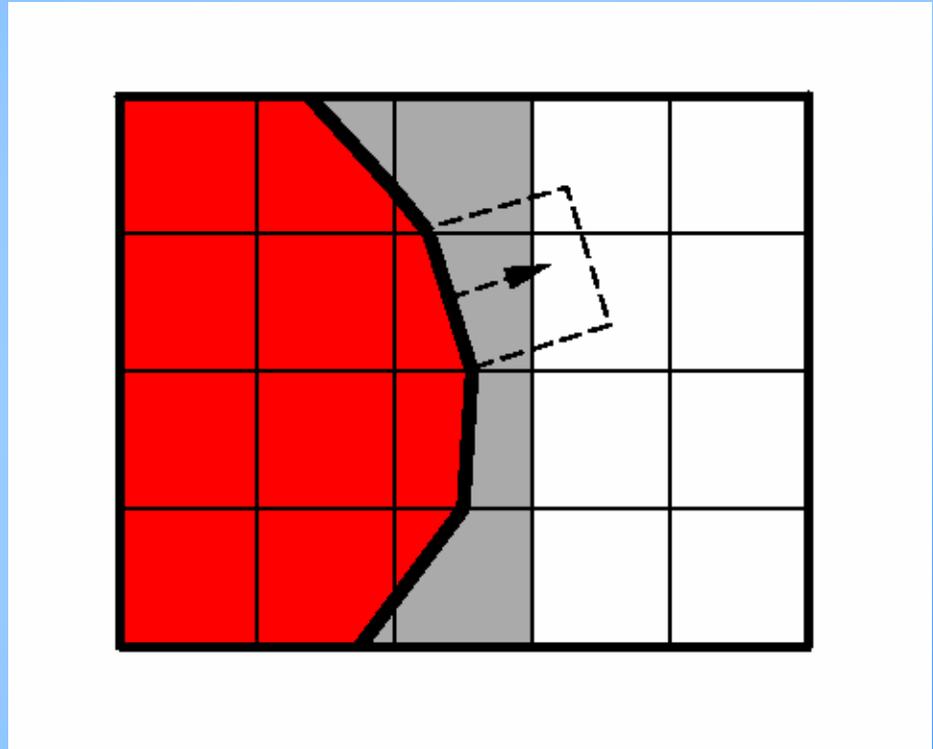


# Structured Sub-grids (contd)

## Problems with complex boundaries

Stability fix :

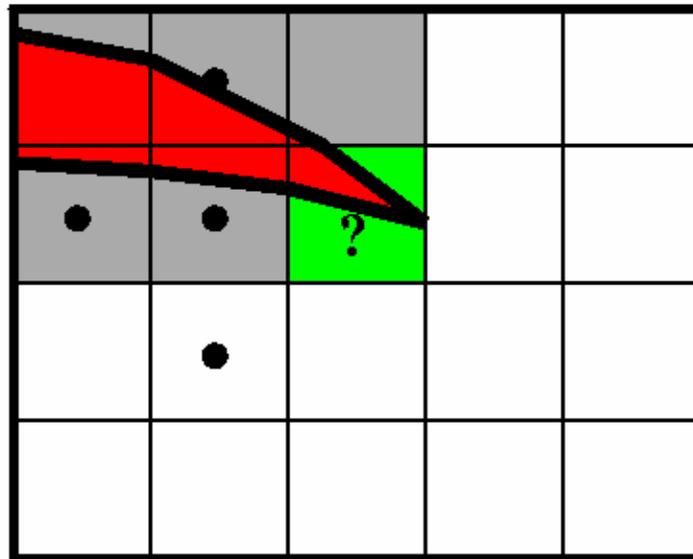
Redistribute flux reflected during a 'normal' timestep.



# Structured Sub-grids (contd)

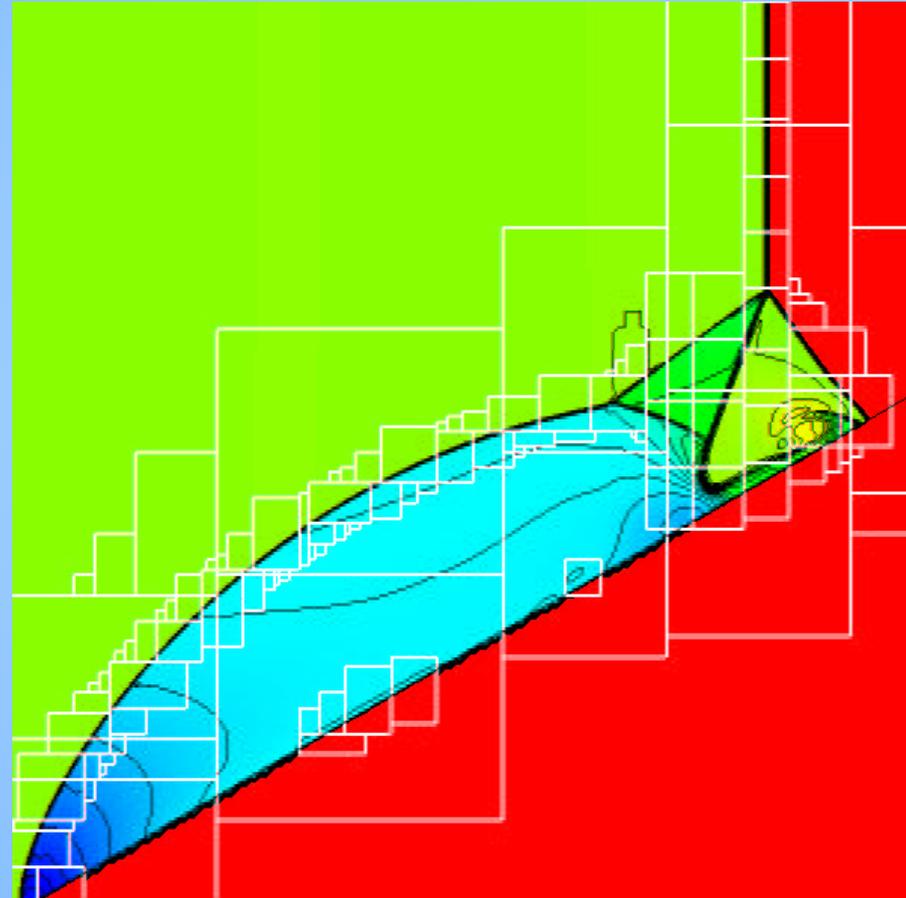
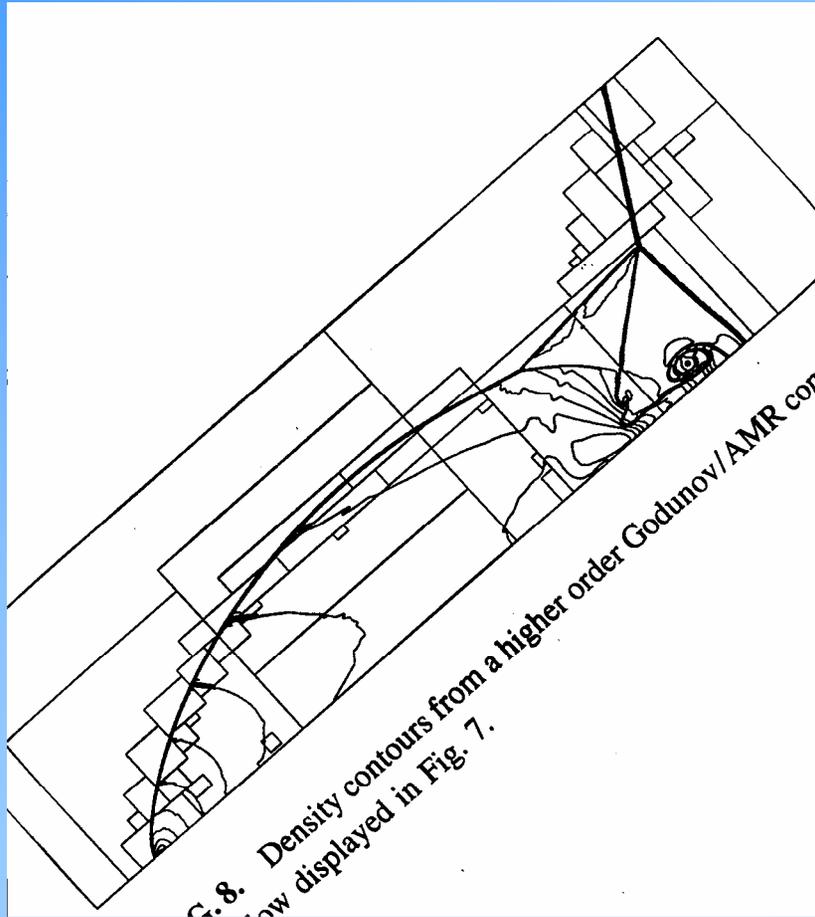
Problems with complex boundaries

## Double Valued Cells



# Structured Sub-grids (contd)

30 degree ramp – rotated(left) and with cut-cells (right)



Credit – Pember et al, Colella

CT Summer School  
July, 2004

# Bibliography (incomplete)

## **Basics**

Berger and Olinger, 1984, JCP, **53**, p.484

## **Stability**

Trefethen, 1983, JCP, **49**, p.199

Trefethen, 1985, Math.of Comp., **45**, p.172

Berger, 1985, Math. Of Comp., **45**, p.301

## **Compressible Hydro Applications**

Berger and Colella, 1989, JCP, **82**, p.64

Bell et al, 1994, J. Sci. Comput., **15**, p.127

## **Incompressible Solvers**

Minion, 1996, JCP, **127**, p.158

Thompson and Ferziger, 1989, JCP, **82**, p.94

## **Elliptic Solvers**

Balls and Colella, 2002, JCP, **180**, p.25

# Bibliography (incomplete)

## **Cut Cells**

Pember et al, 1995, JCP, **120**, p.278

Berger and LeVeque, 1990, Comp. Sys. In Engin., **1**, p.305

## **Gravitational Waves**

Choi et al, 2004, JCP, 193, p.398.

# The Challenge

Adaptive mesh refinement is hard!

Parallel programming is hard!

# The Goal

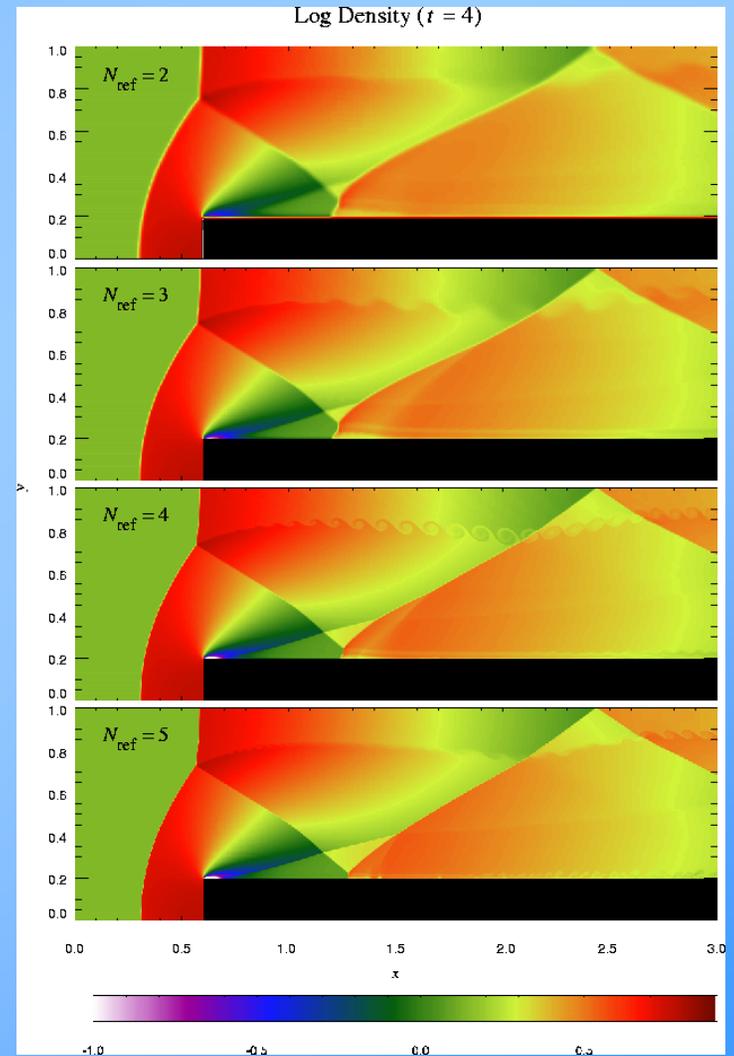
If parallel AMR is to become widely used, somebody needs to produce a toolkit which does the hard parts for a typical model developer.

# What Our Software Does!

Adds block adaptivity and parallelism to applications developed for structured grids.

Test calculation – mach 3 wind tunnel flow over a step (Courtesy of Univ. of Chicago, FLASH group)

Test calculation with grid block outlines (Courtesy of University of Chicago, FLASH Group)



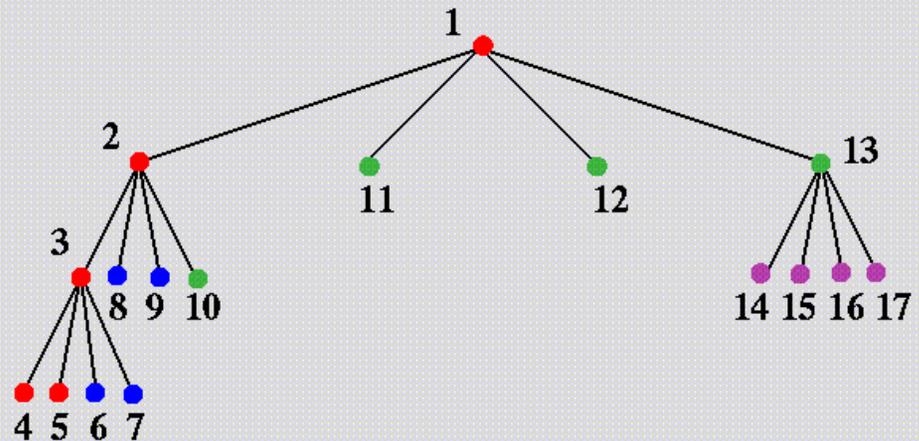
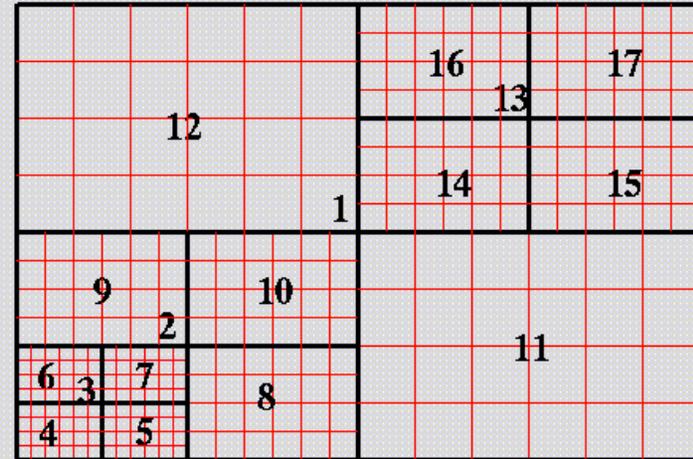
# Package Features

- Written in Fortran 90 – we distribute source code
- Multi-dimensional – works in 1,2, 2.5 and 3D
- Uses structured grid blocks
- Portable
- User tunable load balancing
- Builds upon user's existing code
- User's manual written in HTML
- Tutorial
- Support for conservation laws and solenoidal constraints of hydrodynamics and MHD

# Paramesh Design

- refinement by bisection of sub-grid blocks
- grid blocks organised into a tree datastructure
- distribution of sub-grid blocks across procs designed to minimize communication costs

Example of a simple grid distributed across 4 processors

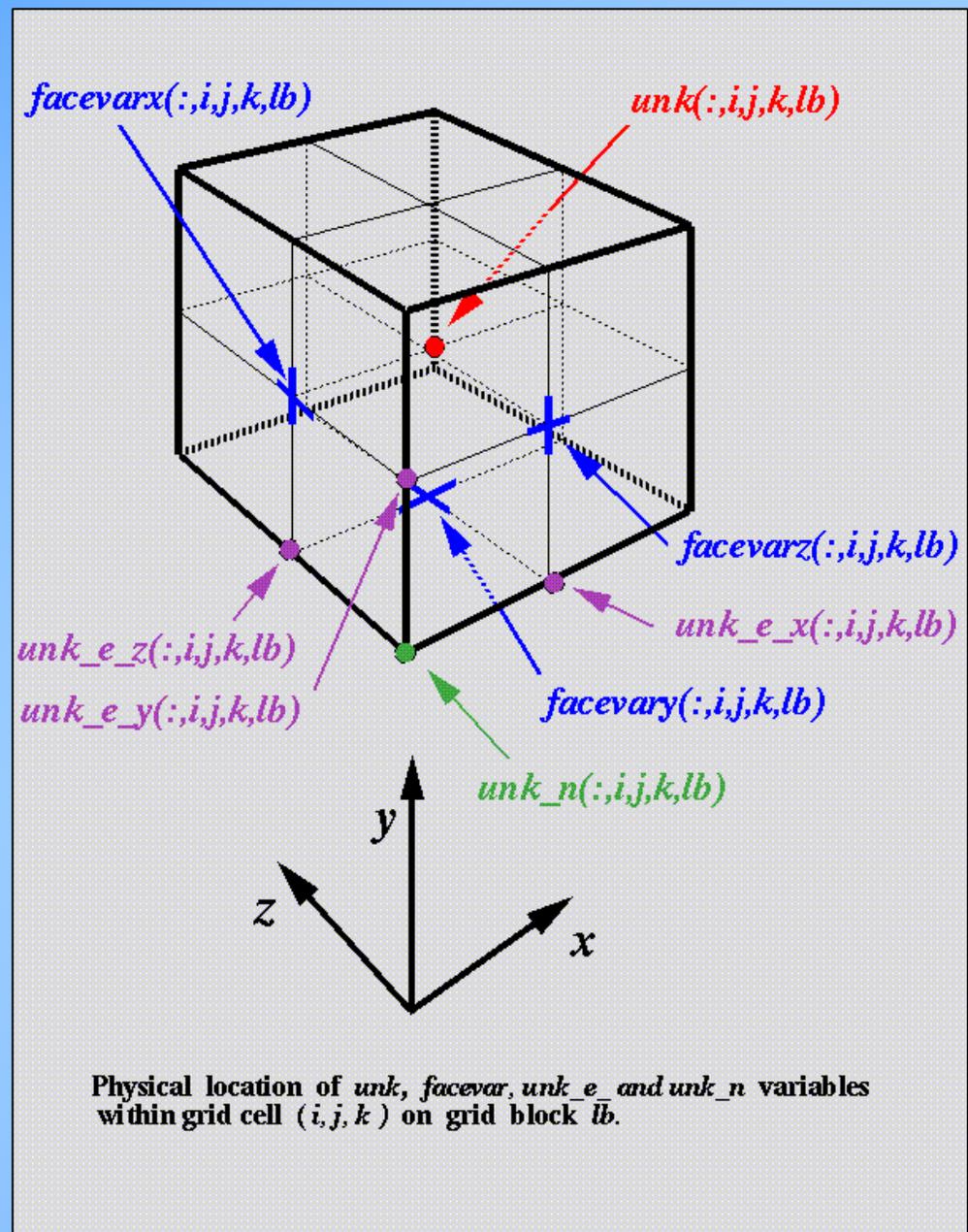


# Design Philosophy

- To provide the application developer with an effectively serial programming environment – they program for a generic sub-grid block
- To enable the user to modify the PARAMESH source code to suit their own individual needs – code is cleanly written with copious comments
- To enable the user to make use of their existing non-AMR code

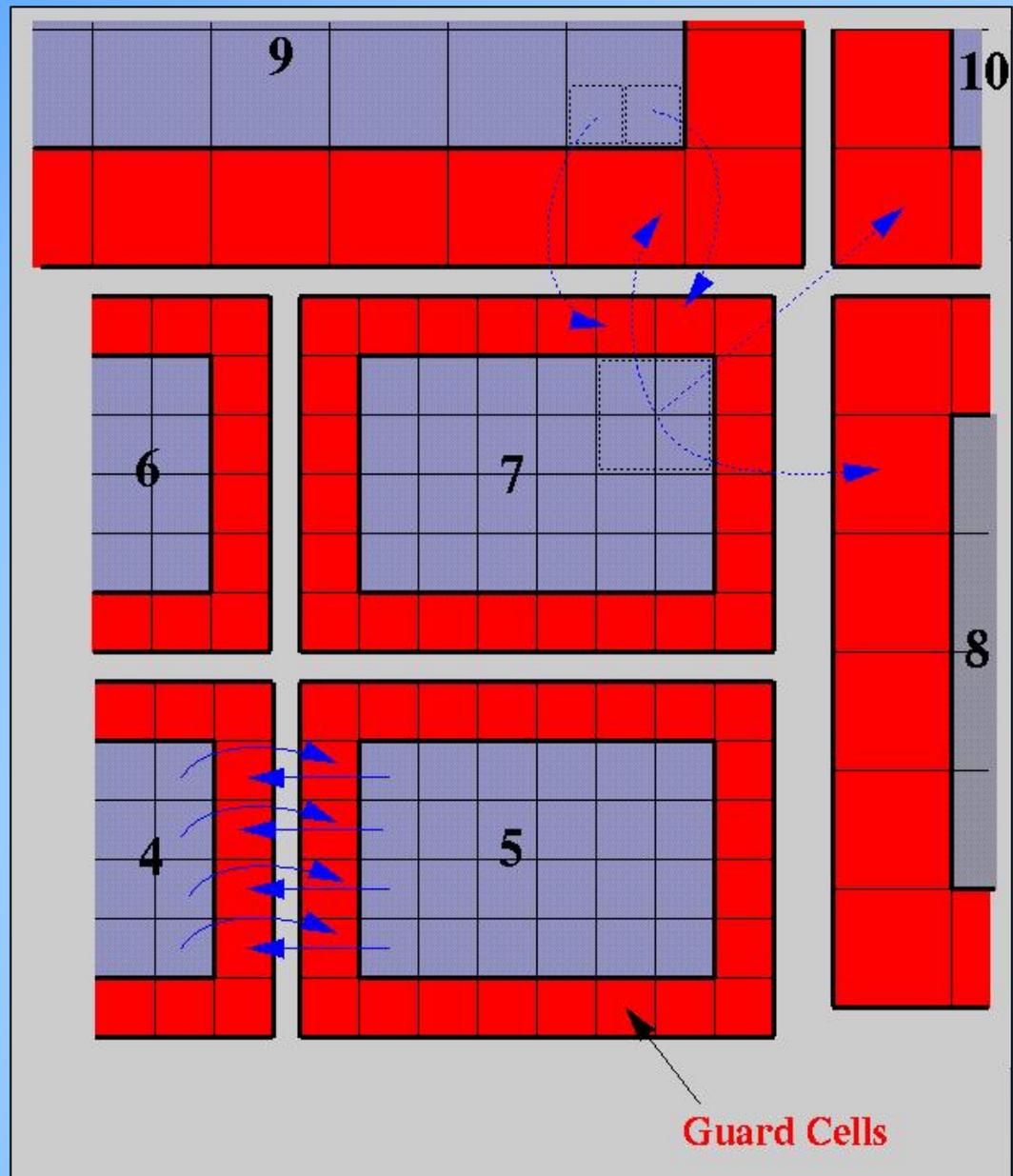
# Paramesh Design

Location of  
solution data  
types within a  
grid cell in 3D



# Paramesh Design

Sub-grid blocks are surrounded by layer(s) of 'guardcells' which are filled with data from neighboring blocks.

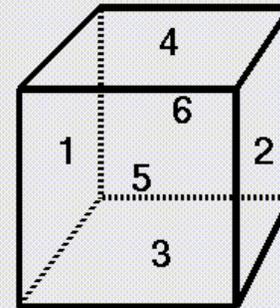
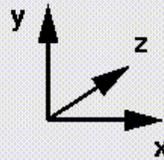


Example of some aspects of guardcell filling in 2D.

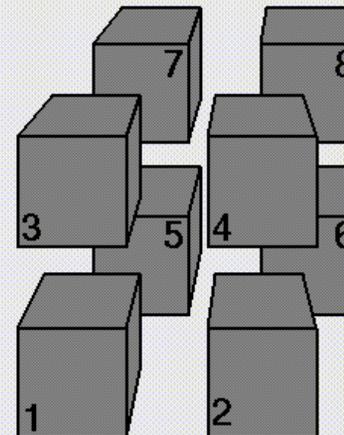
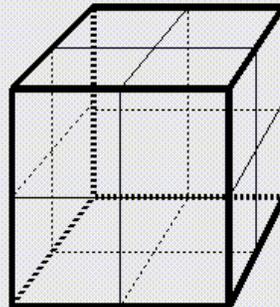
# Paramesh Design

Numbering  
scheme for  
block faces and  
children

Numbering scheme for  
block neighbors.

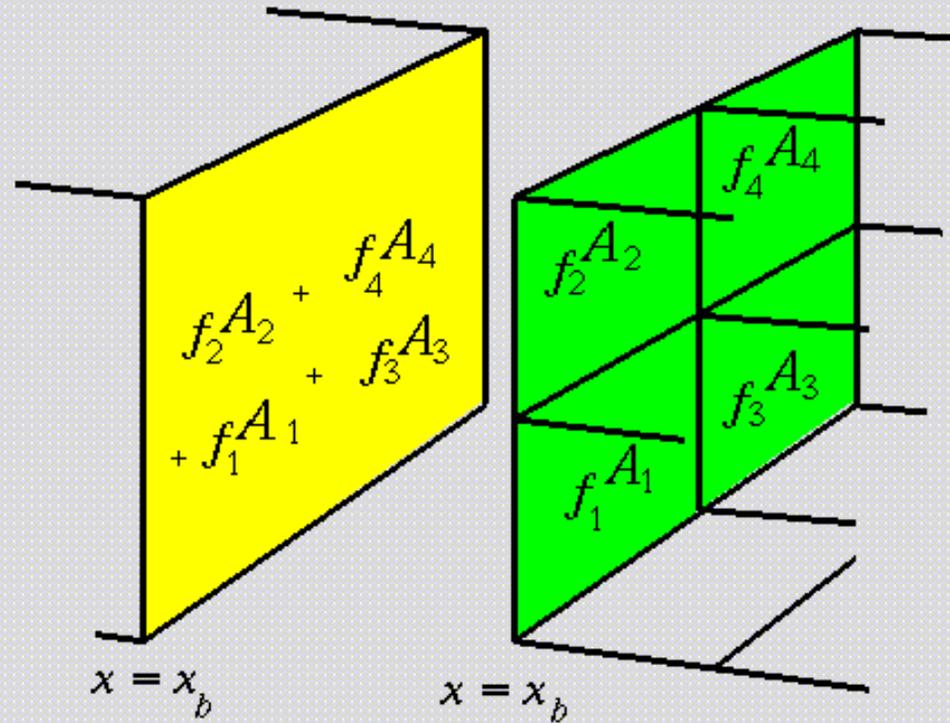


Numbering scheme for child blocks.



# Paramesh Design

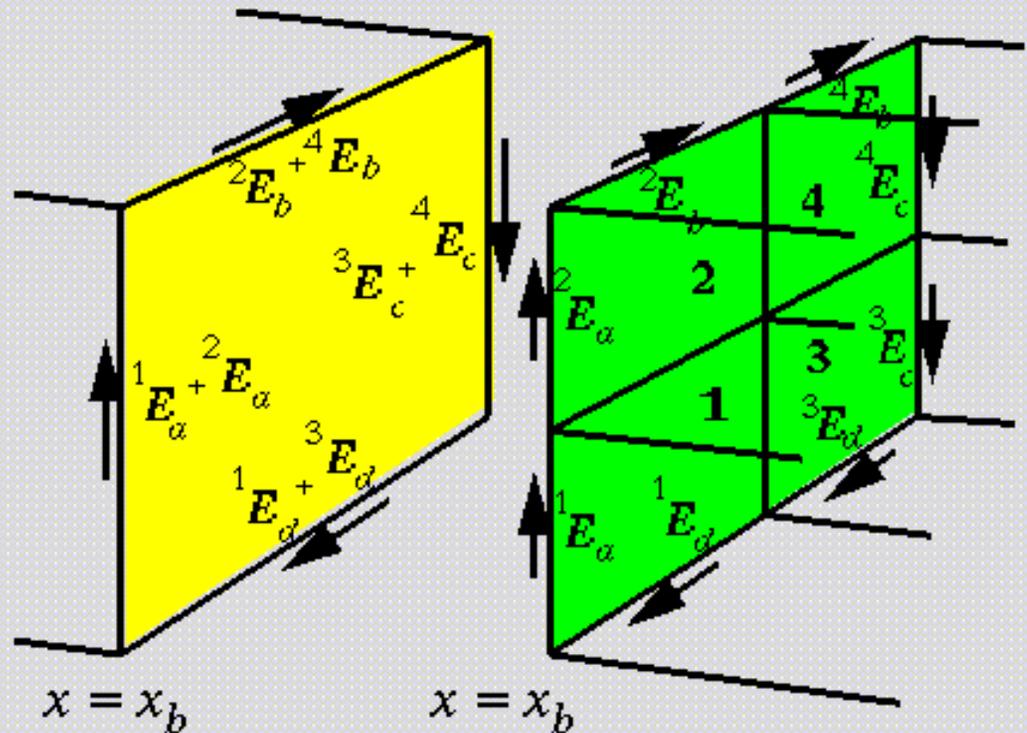
Managing fluxes at  
refinement jumps  
to support  
conservation  
constraints.



Summation of fluxes at shared cell faces on adjoining grid blocks at different refinement levels. Flux densities are denoted by  $f$  and cell face areas by  $A$ .

# Paramesh Design

Managing circulation  
integrals at refinement  
jump (relevant for  
MHD models)

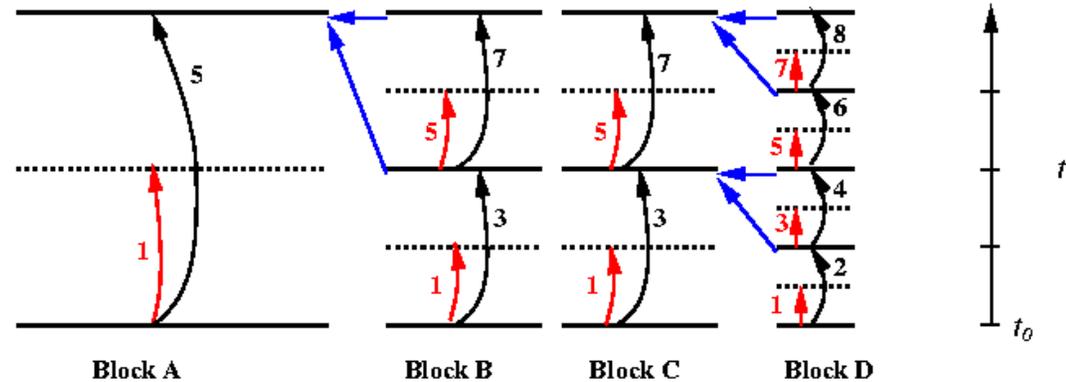


Edge based data on the shared cell face of adjoining  
grid blocks of different refinement level.

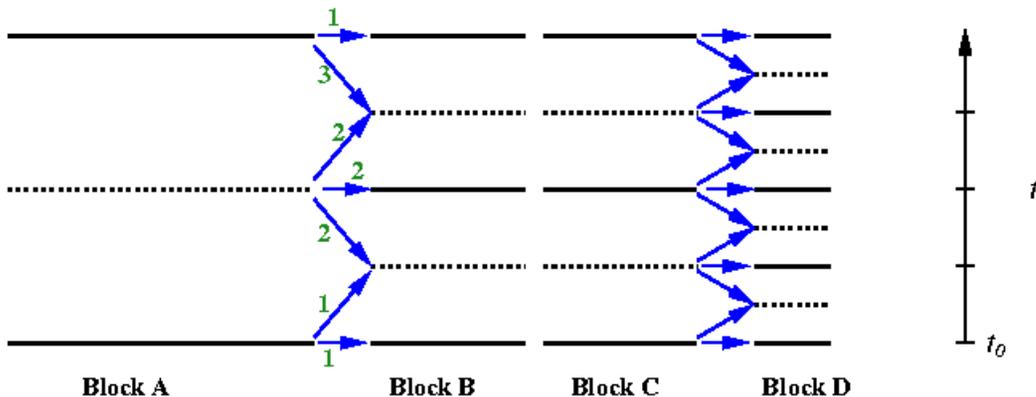
# Paramesh Design

Temporal adaptivity -  
Blocks at different  
refinement levels can use  
different timesteps

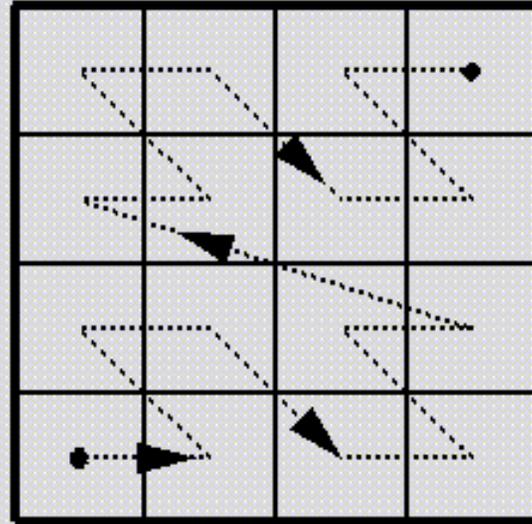
**Time Advance Sequence**  
For Predictor-Corrector algorithm.



**Guardcell Time-Averaging**  
For Predictor-Corrector algorithm.



# Paramesh Design

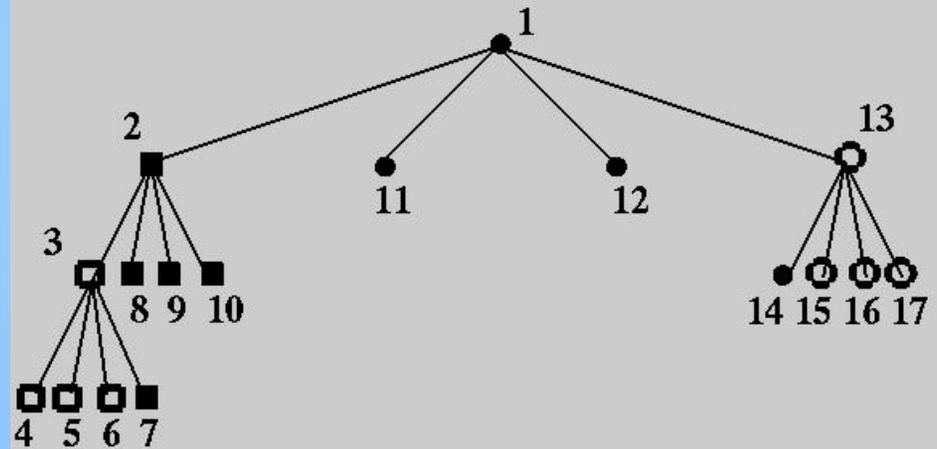
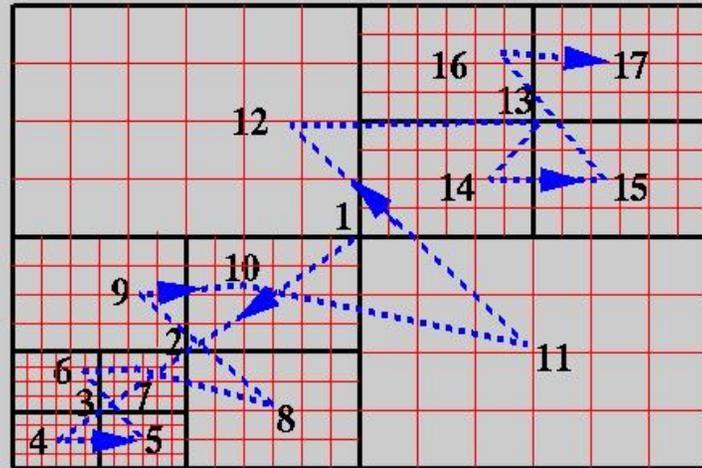


A simple 2D example of the use of a Peano-Hilbert space filling curve to order a list of grid blocks.

Sub-grid blocks distributed  
across processors to minimize  
communication costs

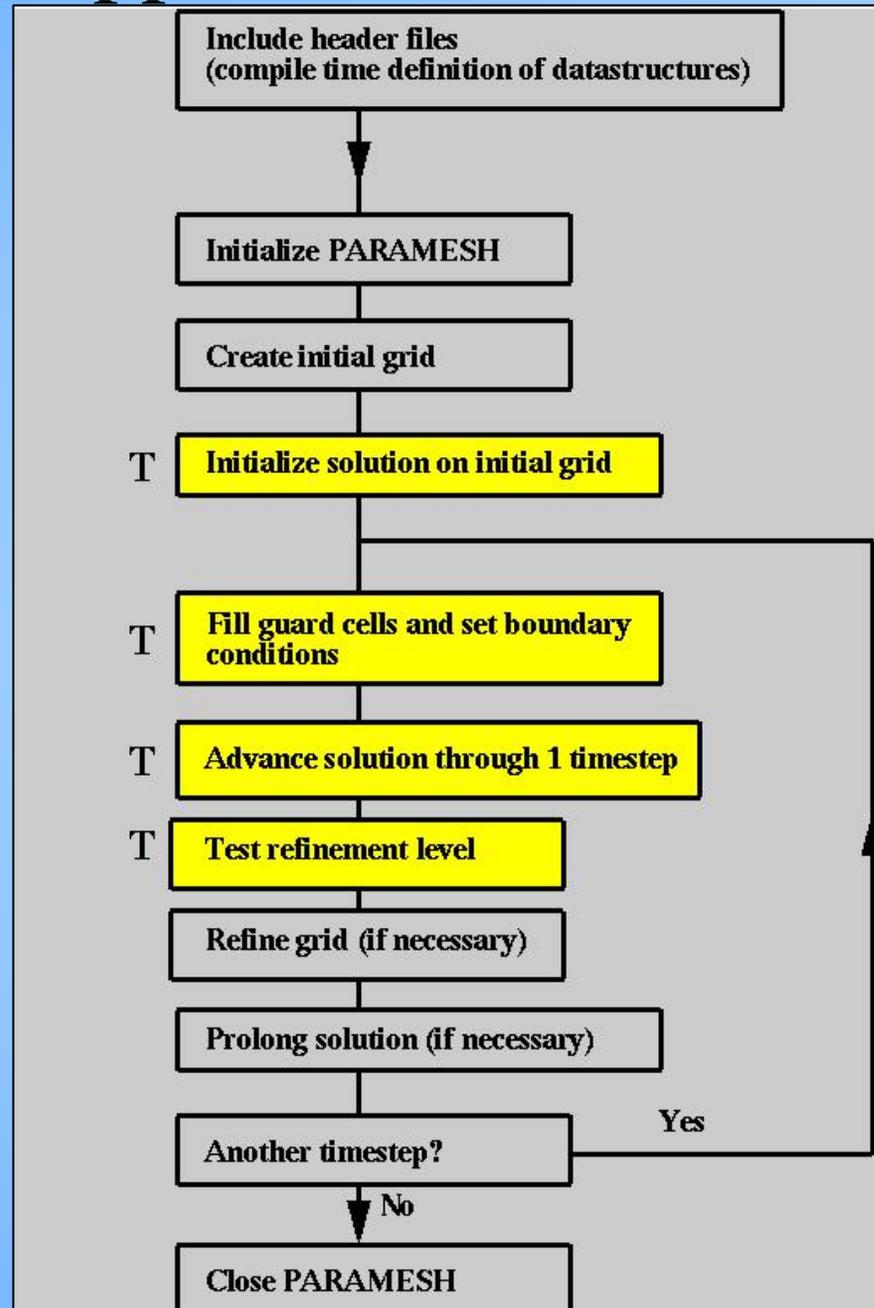
# Paramesh Design

Sub-grid blocks distributed  
across processors to  
minimize communication  
costs



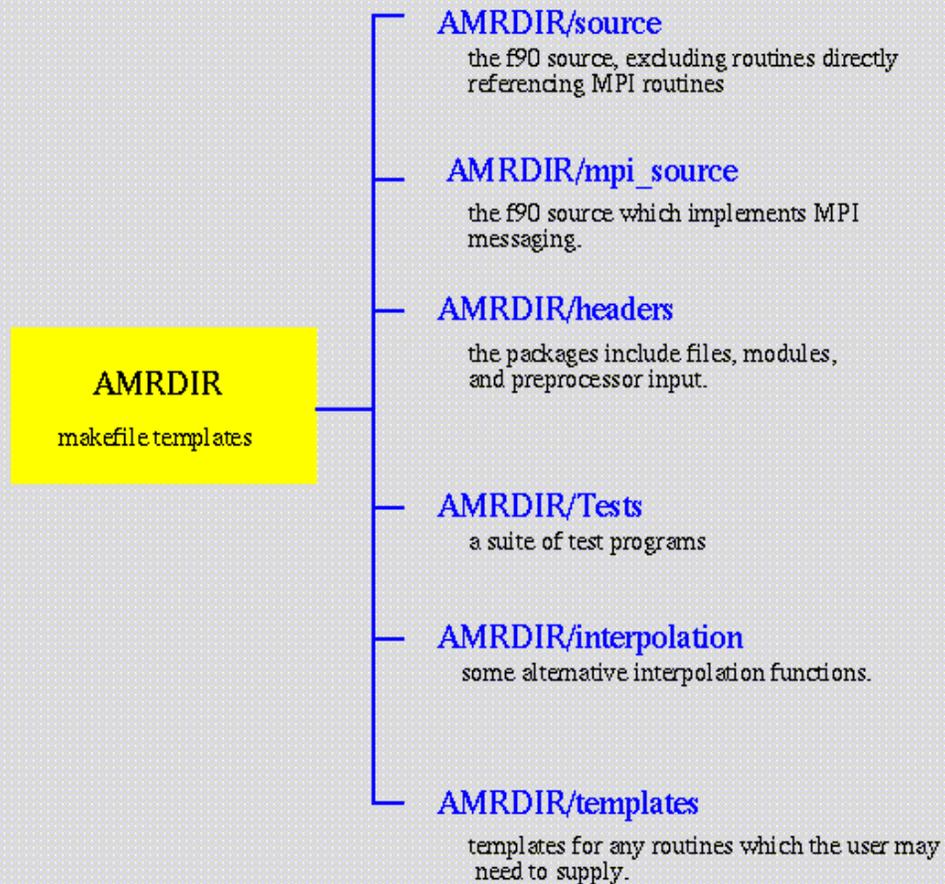
# Steps in Developing an application

1. Define model properties in pre-processor file
2. Build main program – template provided
3. Build routine to advance the solution on a sub-grid block through one timestep – template provided
4. Build a routine to compute the timestep on a sub-grid block – template provided
5. Build routine to set up initial state on a sub-grid block
6. Provide a routine defining how guardcells should be filled at external boundaries – template provided including examples of most common cases
7. Provide a function to determine if a sub-grid block needs to be refined or derefined



# Paramesh Distribution

## Directory Structure of The Package



# Portability

Interprocessor communications written in both SHMEM and MPI



Code runs on :

- T3E
- Beowolf Clusters
- SGIs (workstations, O2K, O3800)
- ASCI (Red, Blue Pacific, Nirvana, Frost)
- Compaq SC45
- Linux Workstations



# User's Manual

Written in  
HTML

Location Edit View Go Bookmarks Tools Settings Window Help

Location file:/home/macneice/Paramesh\_development/paramesh-doc/Users\_manual/amr.html

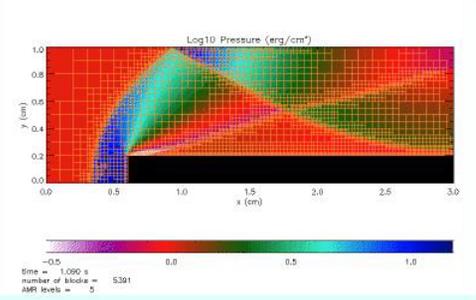
about:konqueror The Goddard Library Earth Science Technology Office Shine Group ASGI Center for As...rmonuclear Flashes Solar Physics Volu... images and movies

## PARAMESH V3.0

### Parallel Adaptive Mesh Refinement

Features :

- AMR
- Multidimensional
- Structured Grid Blocks
- Parallelized FORTRAN 90
- Portable
- Uses either Cray shmem library or MPI
- User Tunable Load Balancing
- Support for Conservation Laws
- Distribution contains source code
- Builds upon user's existing codes
- User's Manual written in HTML



Click on image above (and then click again on enlarged image) for a demo of a Shock wave passing over a step function, courtesy of the [Univ. of Chicago](#), [Flash Code group](#) (14Mb Quicktime Movie).

#### WHAT IS PARAMESH ?

PARAMESH is a package of Fortran 90 subroutines designed to provide an application developer with an easy route to extend an existing serial code which uses a logically cartesian structured mesh into a parallel code with adaptive mesh refinement(AMR).

Alternatively, in its simplest use, and with minimal effort, it can operate as a domain decomposition tool for users who want to parallelize their serial codes, but who do not wish to use adaptivity.

The package builds a hierarchy of sub-grids to cover the computational domain, with spatial resolution varying to satisfy the demands of the application. These sub-grid blocks form the nodes of a tree data-structure (quad-tree in 2D or oct-tree in 3D). Each grid block has a logically cartesian mesh.

The package supports 1, 2 and 3D models.

A description of Version 1.0 was been published in Computer Physics Communications, (2000), Volume 126, pages 330-354.

#### INTRODUCTION

#### WHAT IS DIFFERENT IN V3.0

#### WHAT IS DIFFERENT IN V2.1

#### WHAT IS DIFFERENT IN V2.0

#### A SIMPLE WORKED EXAMPLE

#### A TUTORIAL

#### WHERE TO FIND SOURCE CODE

#### HOW TO INSTALL & COMPILE WITH PARAMESH

#### USERS GUIDE

#### REFERENCE GUIDE

#### PARAMESH GALLERY

#### PUBLISHED PAPERS USING PARAMESH

#### PLATFORMS SUPPORTED

#### DEVELOPMENT PLANS

#### KNOWN BUGS

#### FAQs

#### BUG REPORTS

#### POINT OF CONTACT

#### DISCLAIMER

#### CREDITS

---

#### HOW TO OBTAIN THE PARAMESH PACKAGE

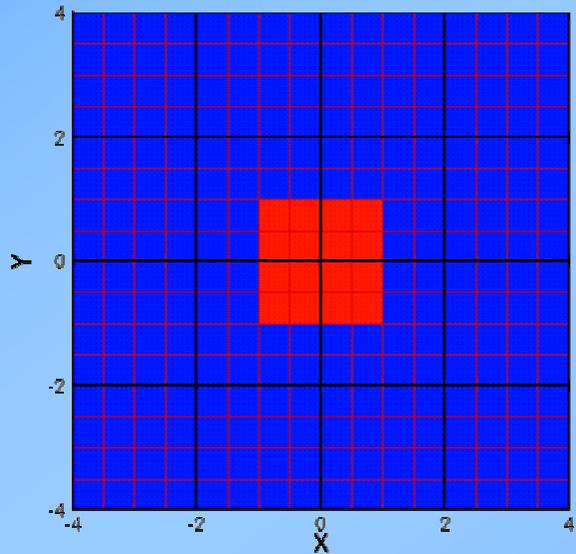
The PARAMESH package source can be downloaded directly ( [ftp file paramesh\\_v3.0.tar.gz](ftp://file.paramesh_v3.0.tar.gz)) or

Loading complete

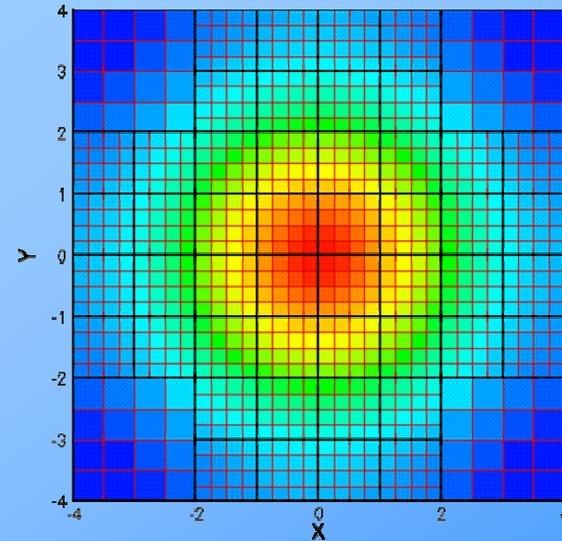
# User's Manual

Includes tutorial – solving 2D diffusion equation

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



Initial state



Final state

# Status

Released V1.0 – June 1999

Released V2.0 – May 2000

Released V3.0 – July 2003

Released V3.1 – November 2003

Current version V3.2 – March 2004

Posted CVS repository on in-house  
SourceForge-like site, called SourceMotel

# Requirements

F90 or F95 compiler, with CPP or FPP

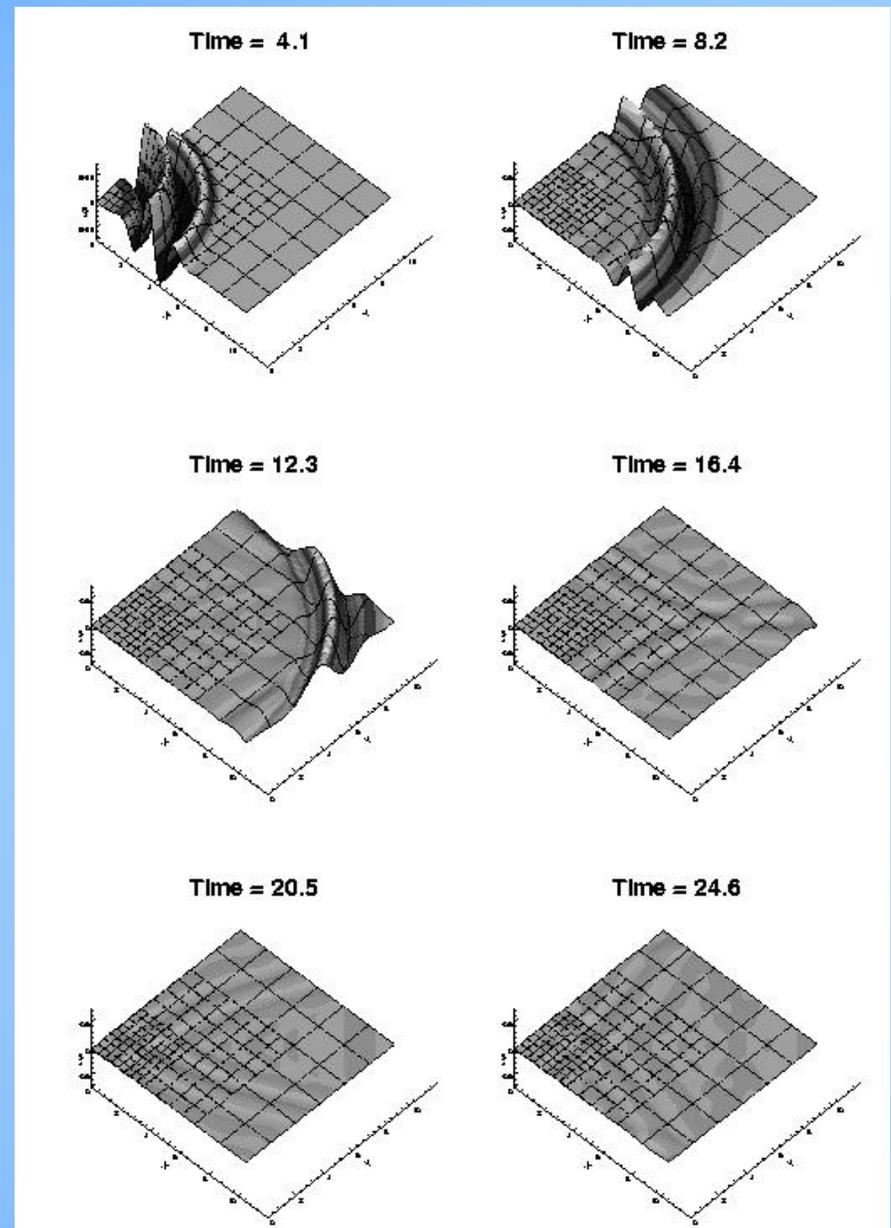
# Future Plans

- Add support for curvilinear coordinate systems (polar, cylindrical, spherical etc)
- Enable configuration as a callable library - done
- Broaden support for elliptic solvers

# Applications

General Relativity –  
gravitational waves generated  
by the interaction of massive  
compact objects (LISA project  
at NASA,GSFC)

Figure shows low noise  
propagation of wave across  
refinement discontinuity



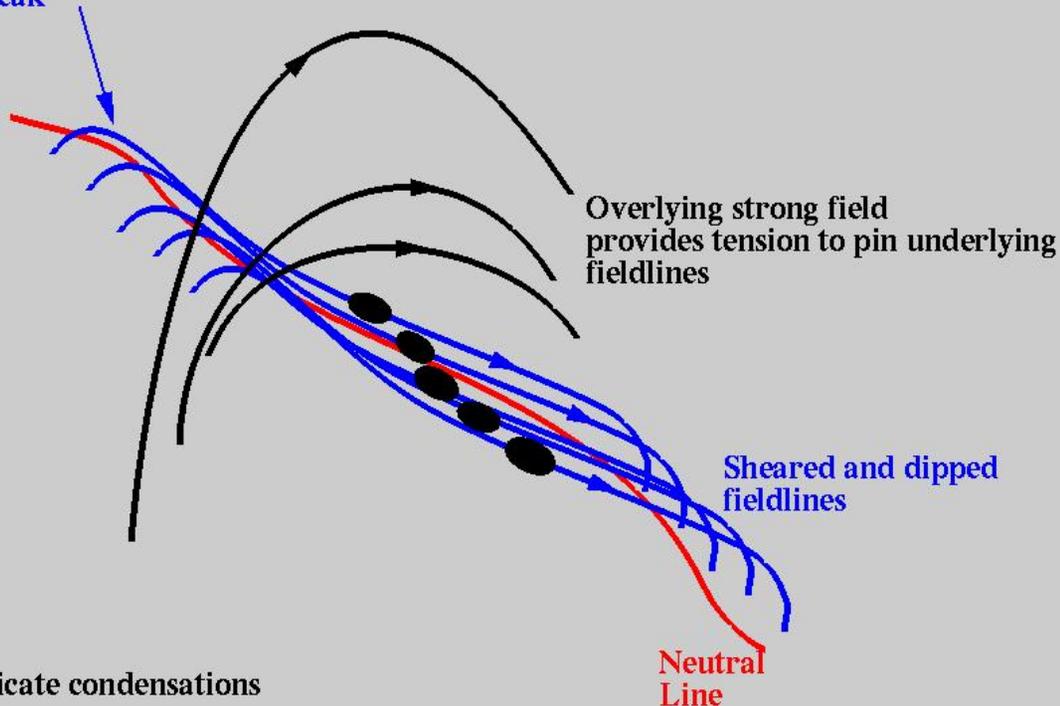
Credits : Drs. Dae-II Choi, Joan Centrella

CT Summer School  
July,2004

# Applications (contd)

## 1D Model of Formation of Coronal prominences (NASA GSFC, NRL)

Sheared underlying fieldlines can bulge up at ends where overlying field is weak



Formation of condensation in dipped flux tube

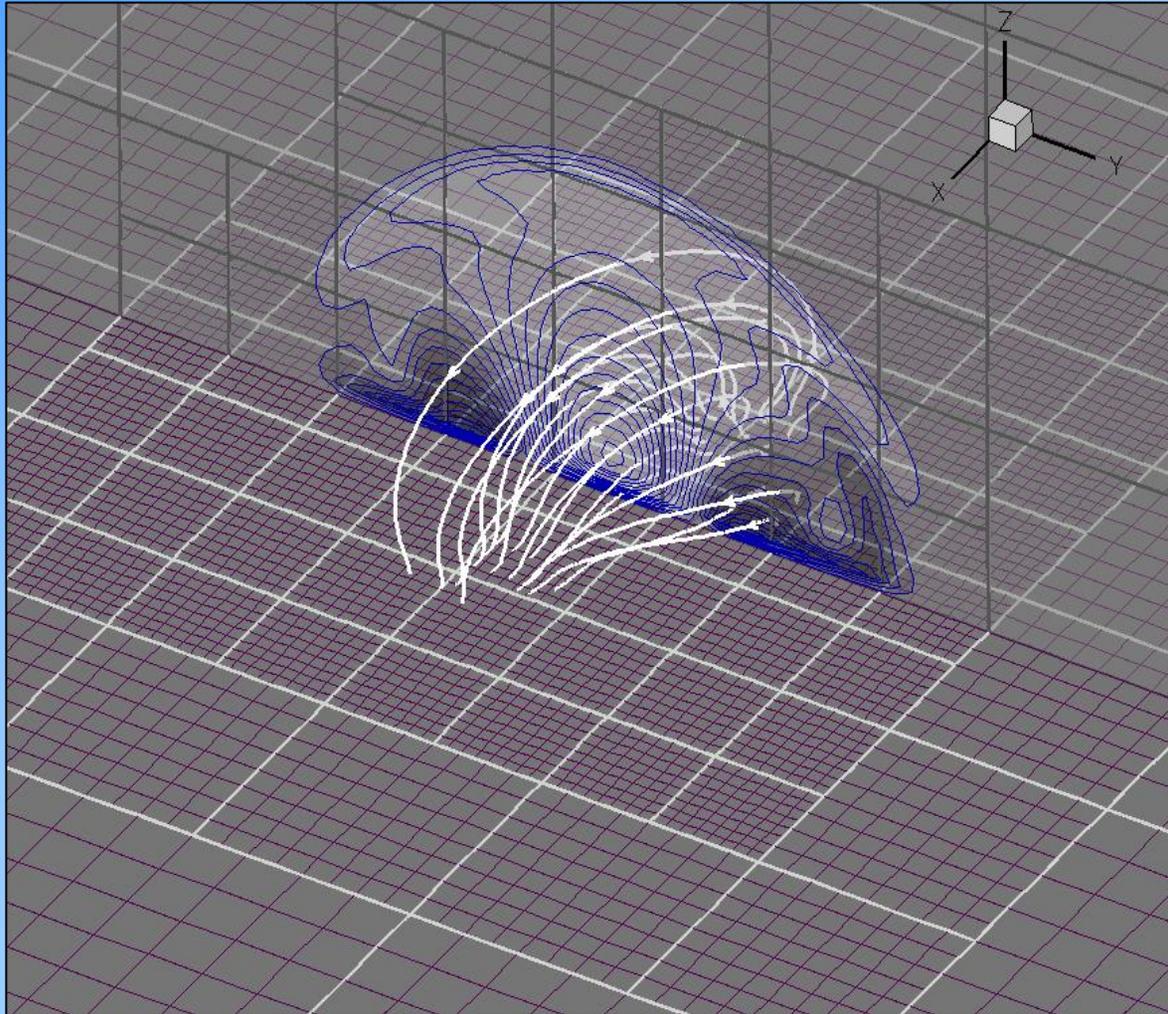
Expanded view of condensation

Credits : Drs. P MacNeice, S. Antiochos, D. Spicer

CT Summer School  
July, 2004

# Applications (contd)

Solar activity – magnetic field emergence (UC Berkeley)



Credits : Drs. W. Abbett and  
S. Ledvina

# Applications (contd)

Solar activity – active region migration into a coronal hole (Naval Research Lab.)

Movie 1 - topology

Credits : Drs. Spiro Antiochos and  
C.R.DeVore

Movie 2 - adaptivity

# Applications (contd)

Space Weather – initiation of CMEs and propagation in inner heliosphere (NCAR, NASA GSFC, NRL)

2D interacting CME flux ropes in interplanetary space

Same – with higher plasma beta

Credits : Dr. Dusan Odstrcil

# Applications (contd)

Space Weather – initiation of CMEs and propagation in inner heliosphere (NCAR, NASA GSFC, NRL)

‘Breakout’ model for CME initiation 2.5D

‘Breakout’ side view with grid

Credits : Drs. P. MacNeice, S. Antiochos, C.R. DeVore

# Applications (contd)

## Interaction of Comet with Solar Wind

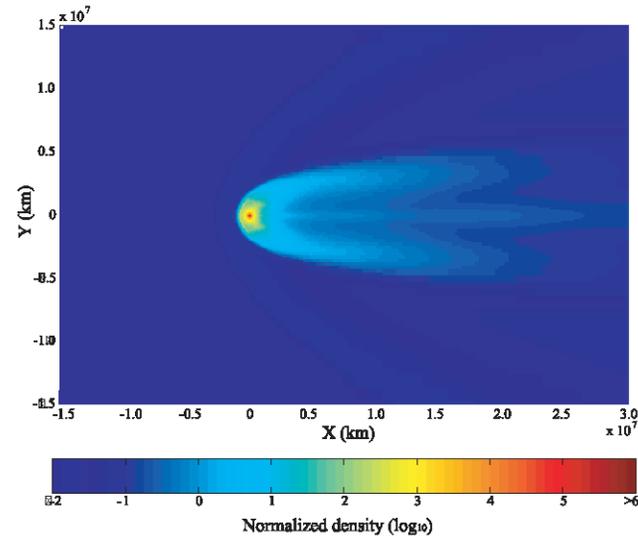
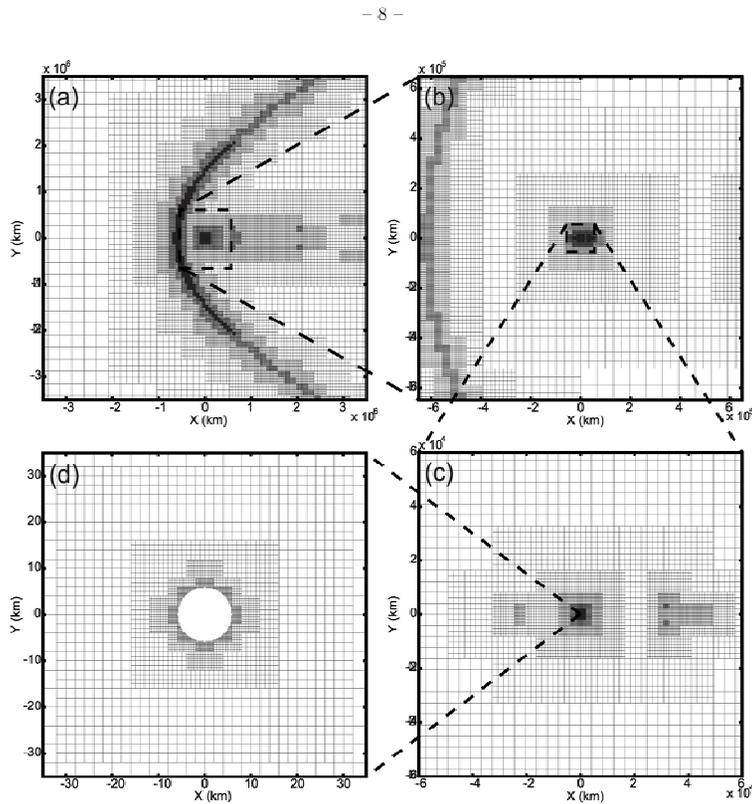
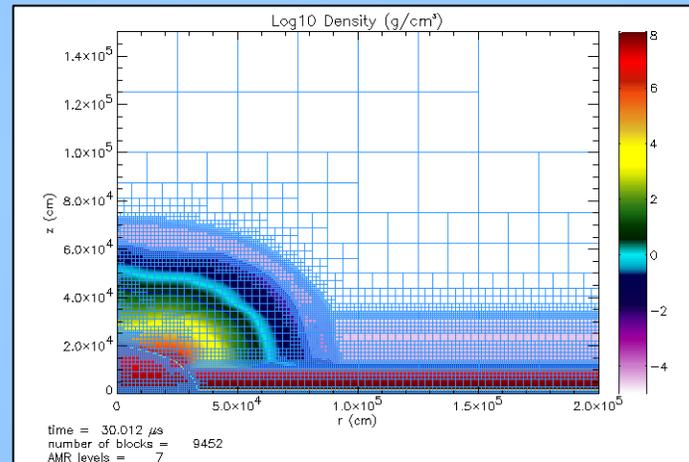
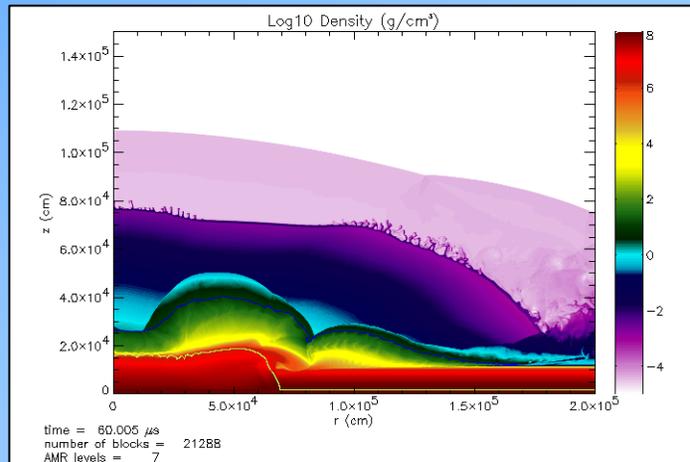


Fig. 7.— Neutral mass density for a Halley-like comet. The mass density is giving in units of mass density of the unperturbed solar wind.

# Applications (contd)

Astrophysics – nuclear burning in models of accretion onto neutron stars (University of Chicago, FLASH group and Mike Zingale, U. of Santa Cruz)



**FLASH Code won a Gordon Bell Prize at SC2000.**