
Patch-Based Adaptive Mesh Refinement for Hydrodynamics

Joint Russian-American Five-Laboratory Conference on
Computational Mathematics/Physics

Vienna, Austria

June 19–23 2005



Richard Pember, Jeffrey Greenough

AX Division

Ben Liu, Ilya Lomov

Earth Science Division

June 16–20, 2005

This work was performed under the auspices of the U.S. Department of Energy by the University of California
Lawrence Livermore National Laboratory under Contract No. W-7405-Eng-48.

Lawrence Livermore National Laboratory, P.O. Box 808, Livermore, CA 94551–0808

Overview



- **“AMR 101” for single fluid gas dynamics**
- **Issues for multimaterial, material strength, other physics**
- **Assorted results**



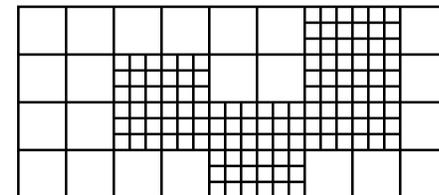
Why AMR?

- **Goal: to focus computational resources on regions of the domain required for accuracy**
- **Why do that?**
 - **Hopefully, realize CPU savings**
 - **Hopefully, realize memory savings**

In this talk we focus on Berger-Oliger-Colella style adaptive mesh refinement



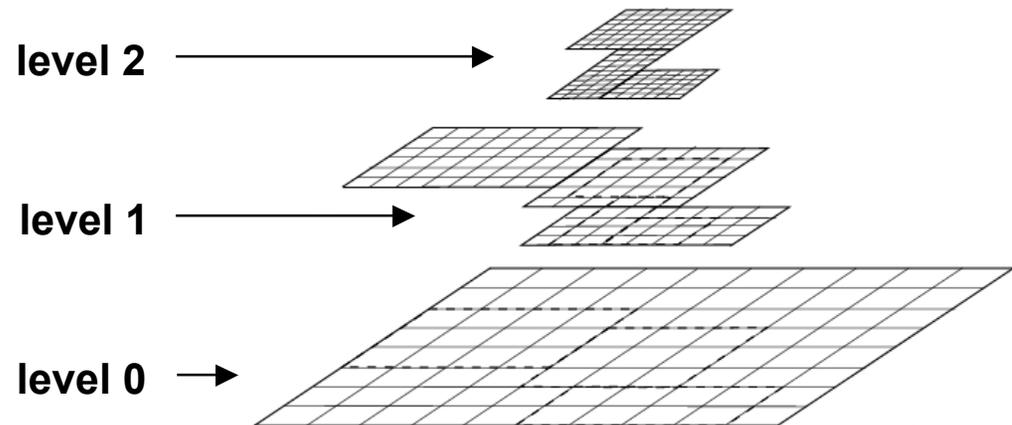
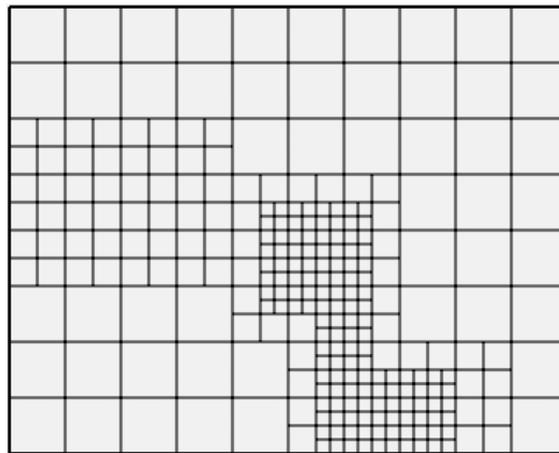
- Features of Berger-Oliger-Colella approach
 - Locally refine patches of the domain in space and time
 - Each patch is a logically rectangular structured grid
 - Patches are “properly nested”
 - Patch consists of high “error” zones grouped along with some (but not many) low-error zones
 - Grids are dynamically created and destroyed to allow for changing features of unsteady flow
 - Patches vary in size spatially and temporally
 - Subcycling in time (recursive time step) is possible, not necessary
- For the purpose of this talk
 - Physically rectangular grids
 - Cell-centered variables
 - Single level time advance is explicit, direct Eulerian, discretely conservative, 2nd order accurate, structured grid
 - structured grid advance achieved through use of ghost zones
 - 2nd order accurate → linear interpolations are sufficient
 - Subcycling in time



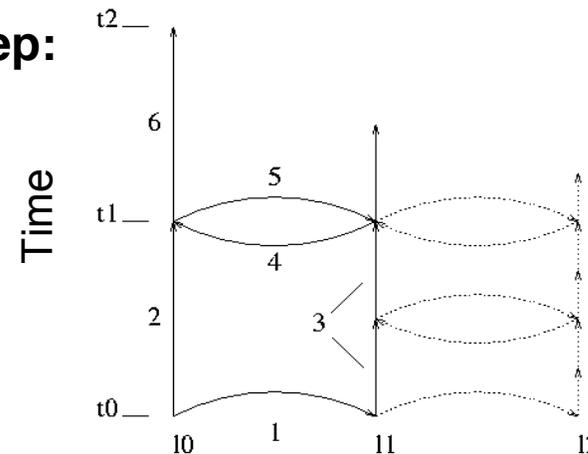
Berger-Oliger-Colella AMR uses dynamic hierarchy of meshes and recursive time step



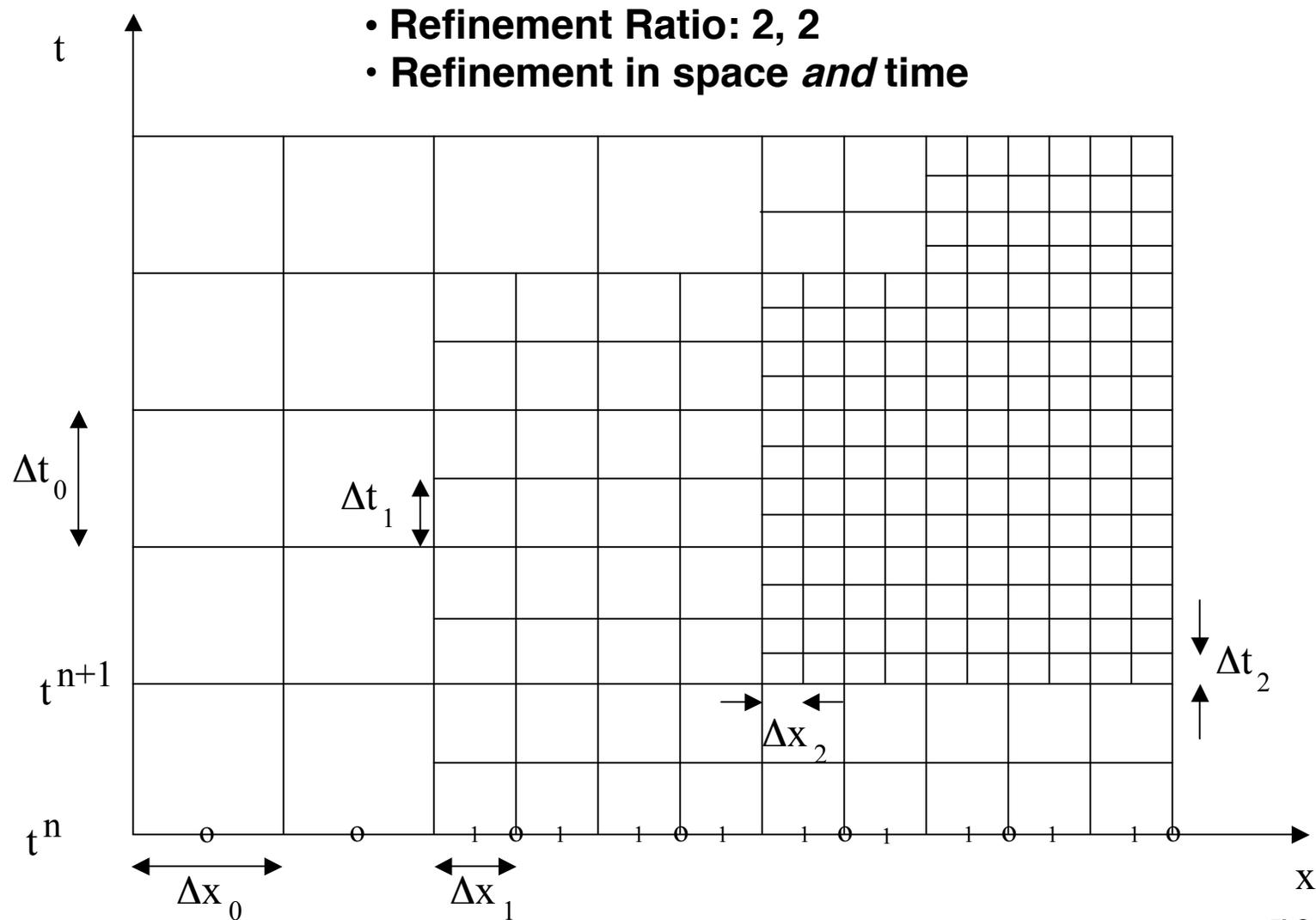
- Spatial refinement



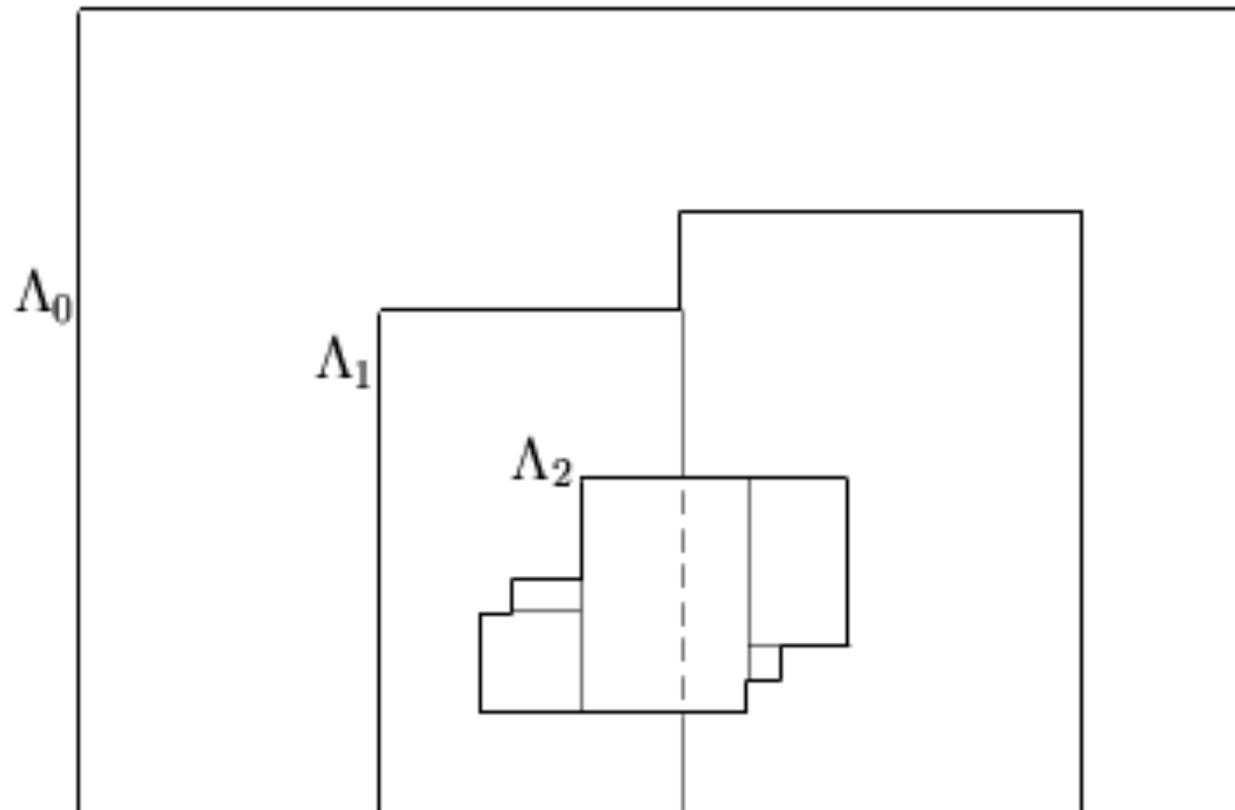
- (Recursive) coarse level time step:
 - Advance coarse level
 - Advance fine level
 - Synchronize levels
 - Regrid current and all finer levels



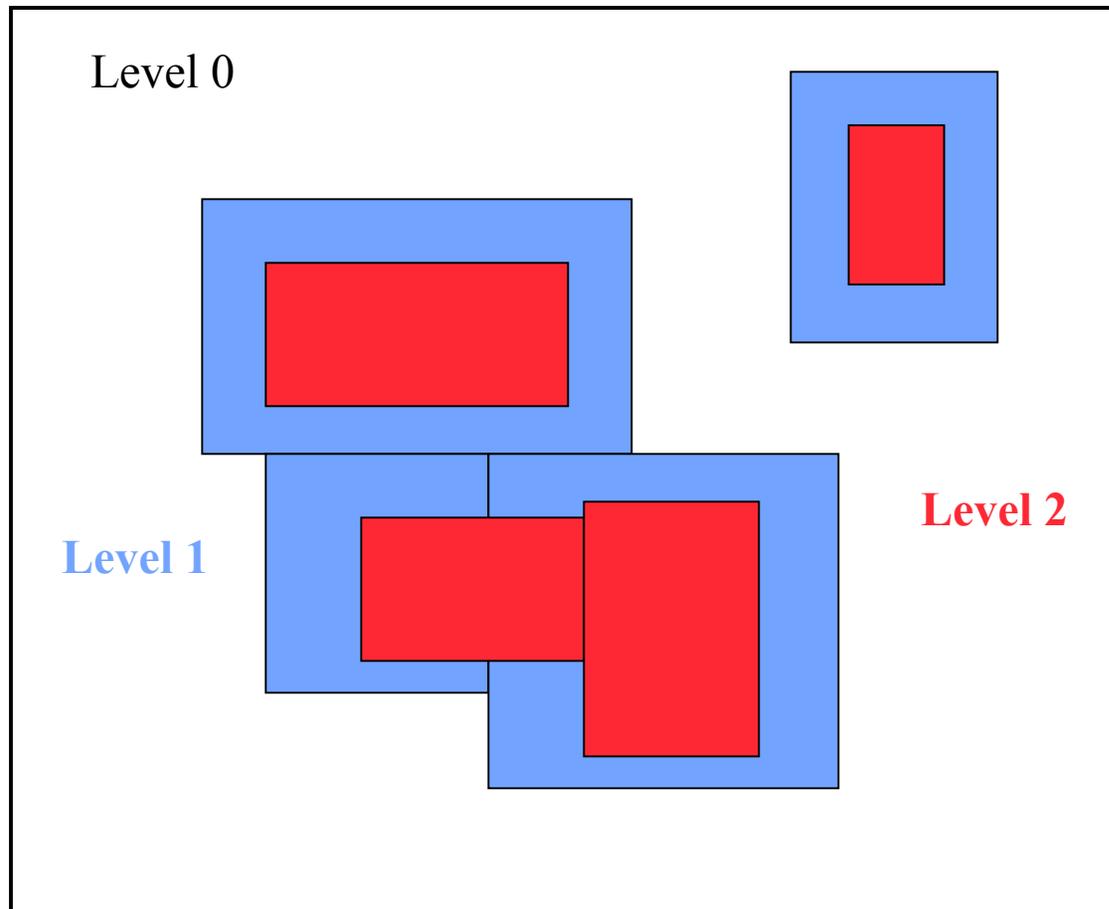
Recursive AMR Timestepping



“Proper nesting” of patches is enforced



“Proper nesting” of patches is enforced

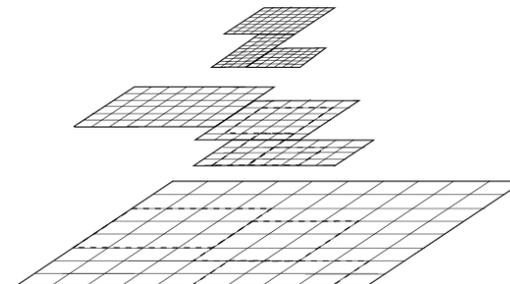


- Base Grid is Level 0 and covers all of computational domain
- Locally refine spatially to create a new level. Tag structures, errors, etc. for refinement.
- Finer levels strictly contained in next coarser level -> proper nesting
- Solution on a level (union of grids) maintained as a fundamental object



Why variable sized, patch-based AMR?

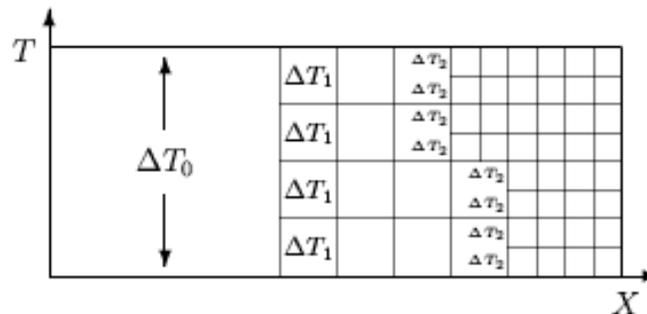
- Some alternatives
 - Cell-by-cell refinement (e.g., Rage)
 - Advantages:
 - minimal memory
 - simple data structures (quadtree, octtree)
 - grid generation is simpler (no clustering required)
 - Possible disadvantages (my guess)
 - higher communication costs
 - more “irregular” zones
 - Fixed patch size (e.g., Grace)
 - Advantage: simplicity, cheap communication
 - Possible disadvantage (my guess)
 - memory inefficient: ratio of refined low error zones higher than patch-based or cell-by-cell
- Really, why?
 - Original reason: vectorization on Crays
 - Justification for not changing:
 - Patches are a convenient unit for distribution of problem among processors
 - Note: *not* domain decomposition
 - Parent/children grids not forced to be on same processor



General form of recursive time step for an explicit conservative scheme



- find level l time step
 - subcycling: use $\min(\Delta t_l, \Delta t_{l-1}/r_l)$
 - not subcycling: use $\min_k \Delta t_k$
- advance level l a single time step
 - BC's from $l - 1$ as needed (interpolated in time and space if necessary)
 - ignore $l + 1$ and higher
- advance level $l + 1$ to the same time
 - multiple time steps if subcycling
 - BC's from l as needed
- synchronize level l and $l + 1$
 - on overlap, redefine level l solution by average of level $l + 1$ solution
 - in level l cells just outside level $l + 1$, modify level l solution to account for “flux” mismatch
- if time to regrid, redefine levels l and higher
 - recursively interpolate from level $l - 1$ as needed



Explicit conservative scheme example



- Cast equation in conservation form, for example

$$\frac{\partial U}{\partial t} + \frac{\partial F(U)}{\partial x} + \frac{\partial G(U)}{\partial y} = 0$$

where

$$U = \begin{pmatrix} \rho \\ \rho u \\ \rho v \\ \rho E \end{pmatrix} \quad F(U) = \begin{pmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ \rho uE + pu \end{pmatrix} \quad G(U) = \begin{pmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ \rho vE + pv \end{pmatrix}$$

and $E = e + 1/2 (u^2 + v^2)$

Explicit conservative scheme example (con'd)



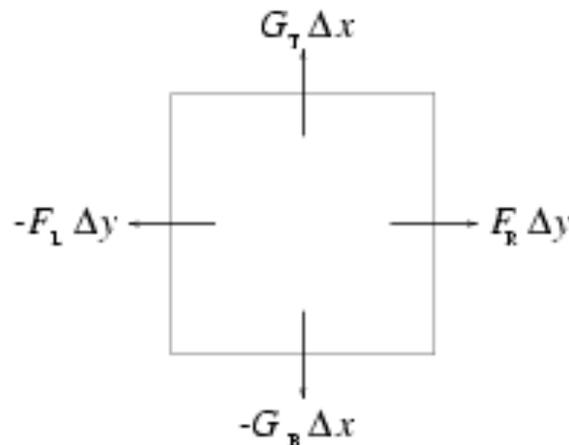
- finite difference scheme \equiv approximate $\nabla \cdot \vec{F}$ ($\vec{F} = (F, G)$)

- recall:

$$\int \nabla \cdot \vec{F} dV = \int \vec{F} \cdot \vec{n} dA$$

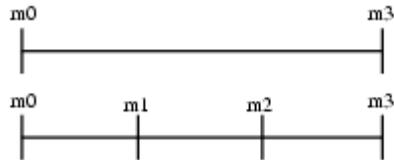
- use divergence theorem

$$\nabla \cdot \vec{F} \approx \frac{1}{\Delta x \Delta y} \sum_i \vec{F}_i \cdot \vec{n}_i A_i, \quad i = L, R, T, B$$



- update is $U^{n+1} = U^n - \Delta t \nabla \cdot \vec{F}$

Conservative interpolation uses limited slopes and volume and mass coordinates



- mass coordinates

- mass of coarse cell: $m_3 - m_0$

- mass of i -th fine cell: $m_i - m_{i-1}$

- mass center of coarse cell

$$M^c = \frac{m_0 + m_3}{2}$$

- mass centers of fine cells

$$M_i^f = \frac{m_{i-1} + m_i}{2}$$

- interpolation

$$q_i^f = q^c + \delta q (M_i^f - M_c)$$

- conservative:

$$\sum (m_{k+1} - m_k) \left(\frac{m_{k+1} + m_k}{2} - \frac{m_3 + m_0}{2} \right) = 0$$

- volume coordinates: similar

- higher dimensions: apply iteratively to slabs, strips, cells

- use

- volume coordinates for ρ

- mass coordinates for u, v, E

- *Note: this scheme is not conservative interpolation via advection*

- *It's just good, old linear (conservative) interpolation*

Volume coordinate interpolation for a spatially rectangular grid is just vanilla linear interpolation



(1) Define limited central difference approximations

$$\left(\Delta_x U^{L-1}\right)_{i,j} / \Delta x_{L-1}, \left(\Delta_y U^{L-1}\right)_{i,j} / \Delta y_{L-1},$$

to the spatial derivatives

$$\partial U^{L-1} / \partial x, \partial U^{L-1} / \partial y$$

at x_i, y_j, z_k of the coarse grid data

(2) Define the fine grid data by the following:

$$\begin{aligned} U_{l,m}^L &= U_{i,j}^{L-1} + (x_l - x_i) \left(\Delta_x U^{L-1}\right)_{i,j} / \Delta x_{L-1} \\ &\quad + (y_m - y_j) \left(\Delta_y U^{L-1}\right)_{i,j} / \Delta y_{L-1} \end{aligned}$$

Coarsening is just volume or mass weighted averaging



$$\rho_{i,j}^L = \sum_{l=r_L i}^{r_L(i+1)-1} \sum_{m=r_L j}^{r_L(j+1)-1} \frac{\Delta x_{L+1} \Delta y_{L+1} \rho_{l,m}^{L+1}}{\Delta x_L \Delta y_L}.$$

$$\rho_{i,j}^L q_{i,j}^L = \sum_{l=r_L i}^{r_L(i+1)-1} \sum_{m=r_L j}^{r_L(j+1)-1} \frac{\Delta x_{L+1} \Delta y_{L+1} \rho_{l,m}^{L+1} q_{l,m}^{L+1}}{\Delta x_L \Delta y_L}.$$

for $q = u, v, E$

Flux correction changes values of coarse cells that share a face with a fine cell but are not covered by fine cells



- suppose coarse grid cell i, j, k at level L shares its right x -cell face with a level $L + 1$ grid boundary.

- define the flux correction $\delta F_{i,j,k}^L$ by

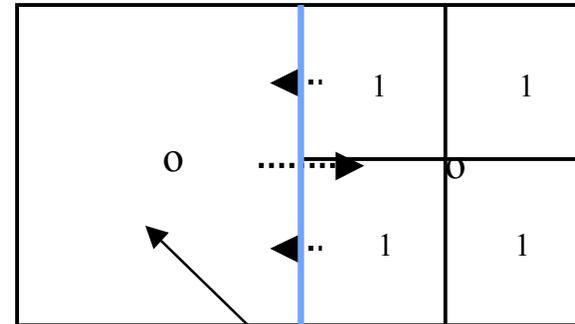
$$\delta F_{i,j,k}^L = \Delta t \Delta y_L \Delta z_L F_{i+1/2,j,k}^L -$$

$$\sum_{J=1, r_L}^{r_L(j+1)-1} \sum_{m=r_L j}^{r_L(k+1)-1} \sum_{n=r_L k} \left(\frac{\Delta t}{r_L} \Delta y_{L+1} \Delta z_{L+1} F_{l-1/2,m,n}^{L+1,J} \right).$$

- update solution in cell i, j, k is then updated by

$$U_{i,j,k}^L = U_{i,j,k}^L + \frac{\delta F_{i,j,k}^L}{\Delta x_L \Delta y_L \Delta z_L}.$$

- the update is equivalent to repeating the integration of the coarse cell using the sum of the fine grid fluxes to update the cell instead of the coarse grid flux.
- correction for other three cases defined similarly



This zone is being modified

Regridding level L determines new grids at level L and greater



- For ilev = finest_level-1, L, ilev—
 - Tag high error cells in level ilev to be refined
 - High error determined by
 - Richardson extrapolation
 - estimate error as difference between coarsen and advance and advance and coarsen
 - Feature detection (gradients, interface, temperature, ...)
 - Other
 - Tag zones in a buffer around high error zones so that high error zones remain (hopefully) refined between regrid steps
 - Buffer width = 1 zone * (regrid interval)
 - If ilev < finest_level-1, project tagged cells from level ilev+1 onto ilev
 - Ensures proper nesting
 - *Group cells into clusters*
 - Fit smallest possible rectangle around each cluster
 - Break rectangles into manageable sizes (< max_nx X max_ny)
 - Create new fine grids (trivial)
 - Generate find grid data
 - Copy on intersect from old fine grids
 - Otherwise, conservatively interpolate from underlying coarse grid

Clustering algorithm is essentially a smart bisection algorithm

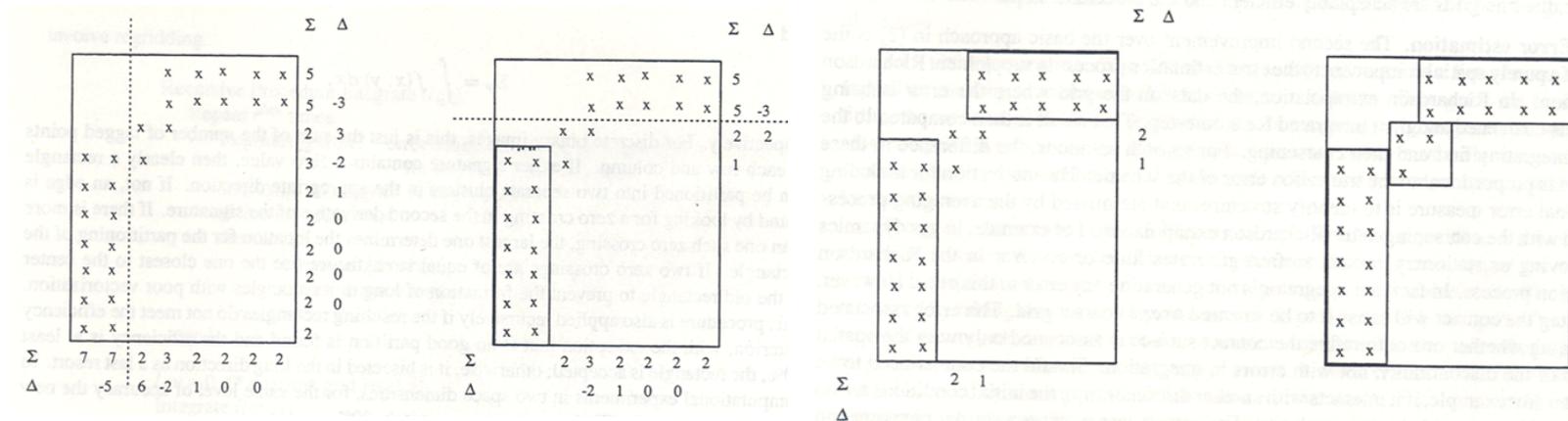


- horizontal and vertical signatures of Σ_x and Σ_y of $f(x, y)$ are defined

$$\Sigma_x = \int_y f(x, y) dy \quad \Sigma_y = \int_x f(x, y) dx$$

- for $f(x, y) = 1$, if tagged, 0 otherwise, Σ_x and Σ_y are number of tagged zones in each row and column
- edges detected by zeroes of signatures and second derivatives
- apply recursively until efficiency criterion met (e.g., want .7 of zones in rectangle to be tagged)
- if no good partition
 - if efficiency $> .5$, accept
 - else bisect

Example from Bell, Berger, Saltzman, Welcome, SIAM J. Sci. Comput., Vol. 15, pp 127–138, 1994



Issues for multimaterial and material strength



- **Interface tracking**
 - Always refine the interface: done
 - Otherwise, issues are
 - Formulation of flux correction step
 - Derefinement, re-refinement issue: loss of information
 - “state-of-the-art”
 - Always refine the interface, or
 - Refine interface when no longer quiescent
 - Do not derefine
 - Note: flux correction not needed because no “flux” across coarse-fine boundaries at material interface
- **Material strength**
 - History variables
 - Strain tensors
 - “state-of-the-art” for both: don’t do anything special
 - Flux correction: distortional deformation tensor treated w/ non-divergence formulation
 - Alternative (Miller & Colella): full conservative treatment



Issues for other physics

- **0-dimensional physics: no issue**
- **Explicit, “hydro-like” physics: no issue**
- **Non-local physics (diffusion, transport)**
 - **Synchronization step involves non-local (parabolic, elliptic) solves**
 - **Computational and development expense**
 - **Alternate solution: rework time step**
 - **Hierarchy time step:**
 - **Hydro on all levels**
 - **Diffusion on all levels (via a multilevel coupled solve)**
 - ...
 - **Advantages**
 - **avoid synchronization step**
 - **allows for possibility of using existing unstructured grid package**
 - **Disadvantage**
 - **no subcycling: use a single (fine level) time step everywhere**
 - **need to translate between two data structures**

Development Issues for other physics

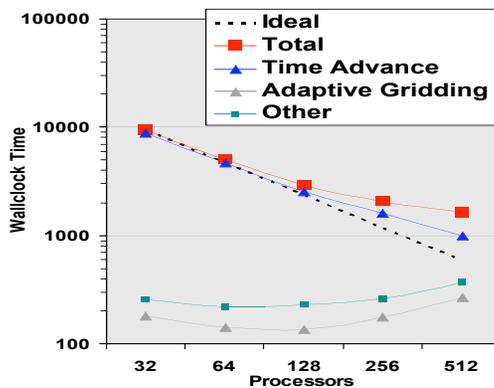
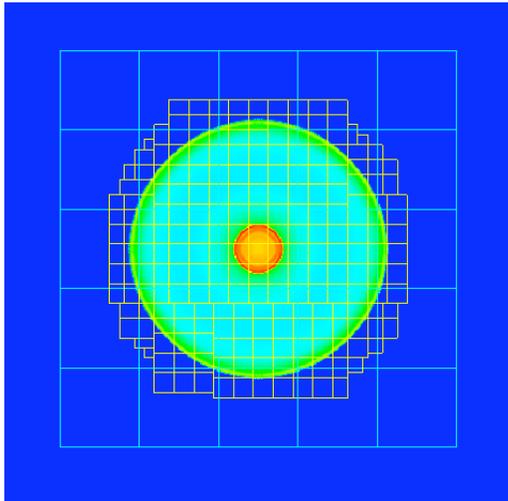


- Approach to date: mostly “write from scratch”
 - Except: many AMR developers use an infrastructure:
 - Boxlib, SAMRAI, Chombo, AMRCLAW, ChomboCLAW, Overture, AMRita, AMROC, GrACE (others?)
- Ideally, need approach that uses existing code and leaves existing code alone (as much as possible)
- ALE-AMR team in CASC is exploring “AMR-izing” CALE in a non-intrusive manner using SAMRAI
 - Currently have an AMR built on CALE for multimaterial Lagrange or Euler, strength, HE burn
 - Uses “co-routine”-like idea
 - Coroutines can be *simulated* using threads with explicit scheduling (threads as software engineering device)
 - Development overhead:
 - O(100) (of 225k) lines of code in orig sources
 - O(10k) new, SAMRAI related lines of “separate” code
 - Much work to be done

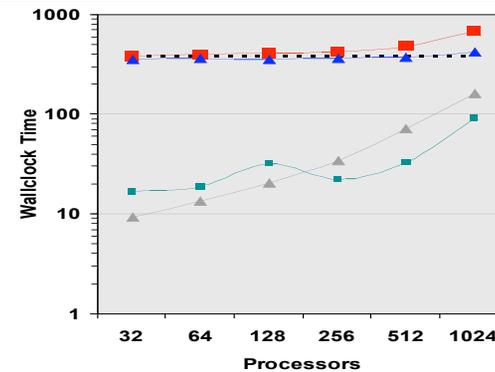
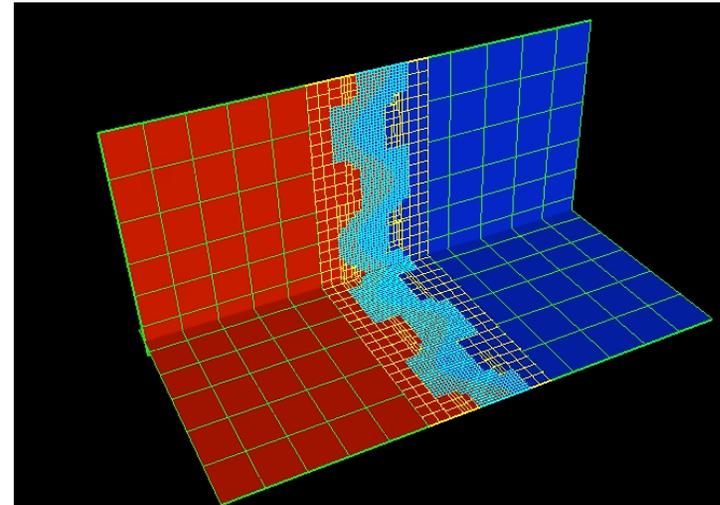
Example using SAMRAI shows both decent scaling and low AMR overhead



Non-scaled benchmark
4 level Sedov Problem
ASCI IBM Blue Pacific



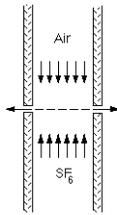
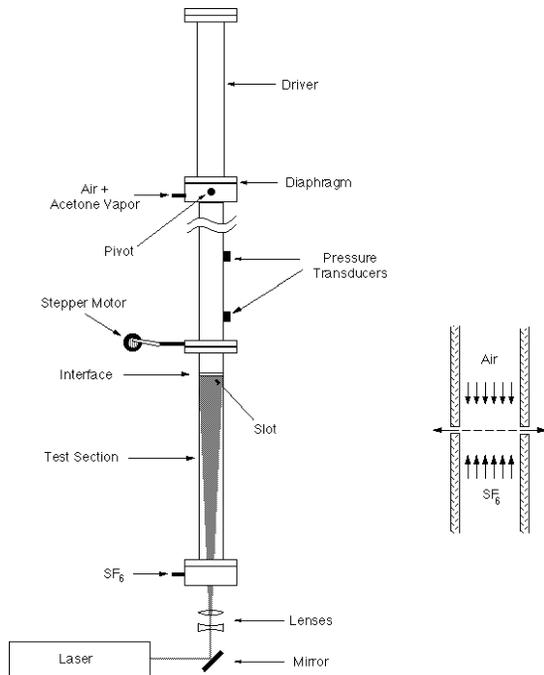
Scaled benchmark
3 level linear advection
Linux MCR Cluster



Raptor (Boxlib) computation of “re-shock” at $M=1.3$, Air/SF₆

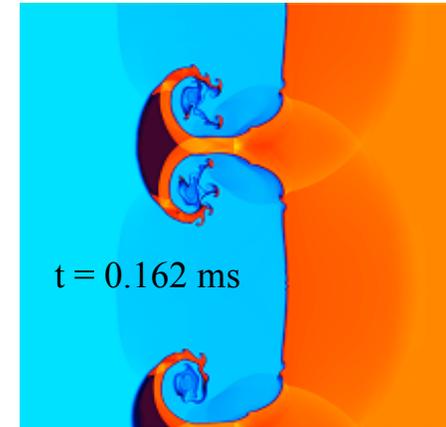
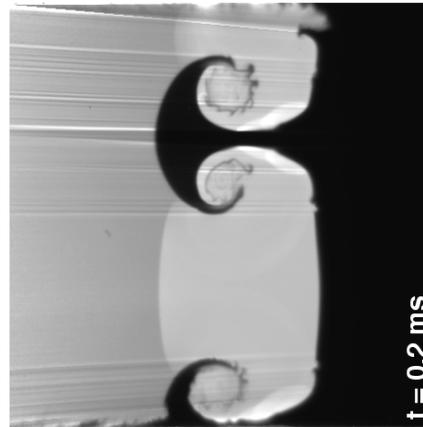


Vertical Shocktube at the University of Arizona (Prof. Jeff Jacobs)



AMR is well-suited for computing reshock

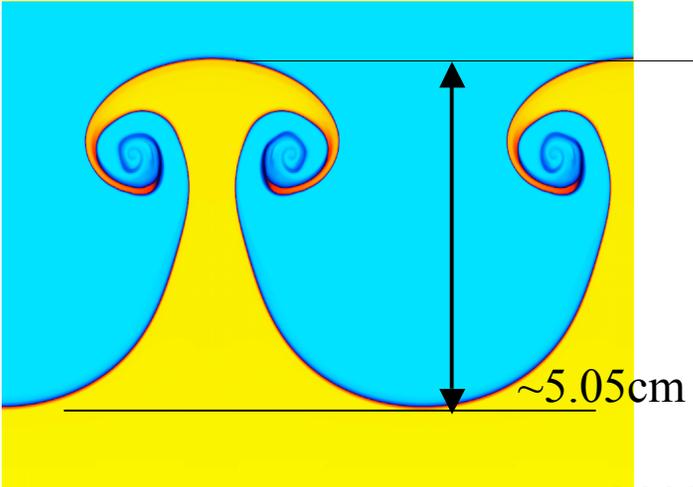
- Ignore small time offset due to experimental false bottom



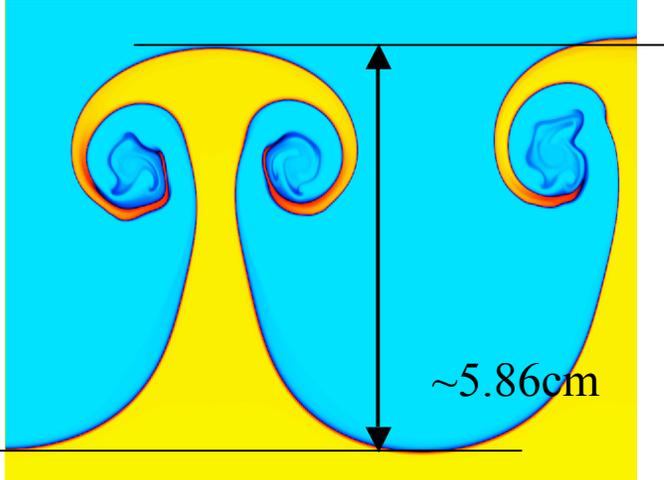
M=1.2 RM Initial Amplitude Sensitivity Study with Raptor



$t = 6.276 \text{ msec}, a = 0.221$



$t = 6.276 \text{ msec}, a = 0.3$



$t = 6.003 \text{ msec}, a = \text{unknown}$

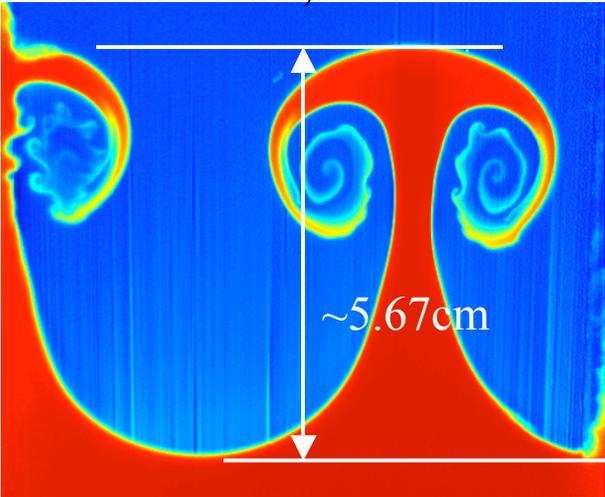


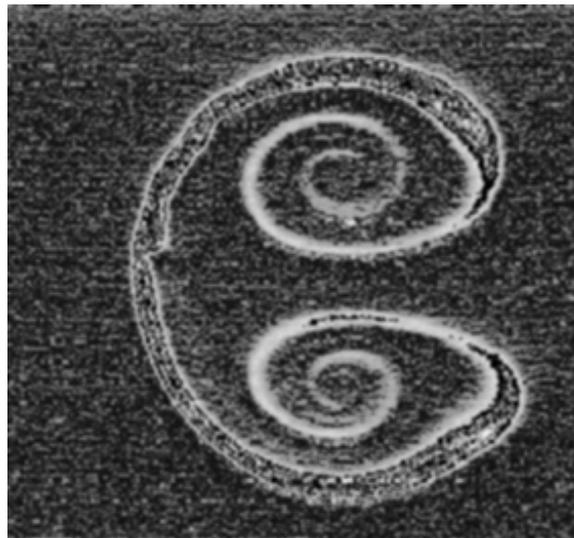
Figure courtesy of Prof. Jeff Jacobs, Univ. of Arizona

Shock SF₆ Jet Interaction Expt. Jacobs, Phys. Fluids A, vol. 5, no. 9, 1993

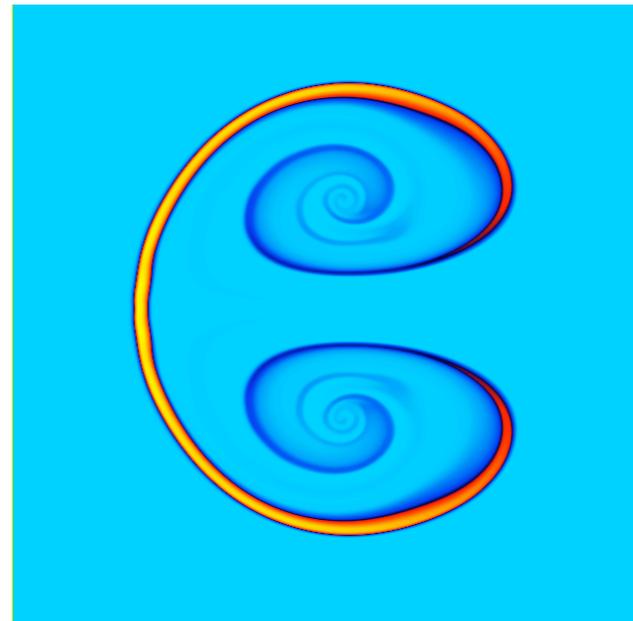


- Drive a weak shock wave, $M=1.095$, through a cylindrical column of SF₆
- Use PLIF (planar laser induced fluorescence) to visualize the cross-section

Experiment



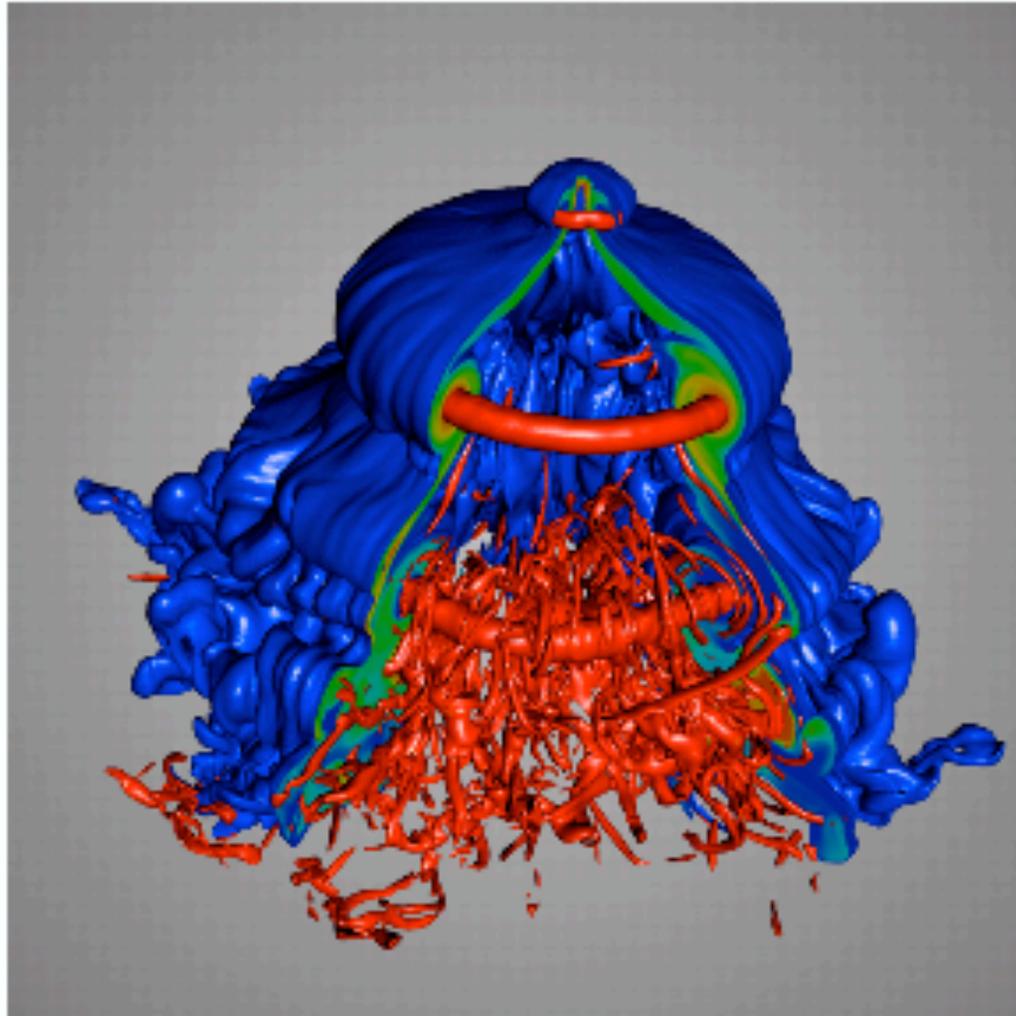
Raptor Simulation



3D Raptor calculation of shock-bubble interaction took 7 hours on ALC Linux cluster



- Mach 2.88 shock impinges on an argon bubble
- R_{120}
- $\sim 200 \mu\text{m}$ zoning
- red: vorticity mag.
- blue: argon conc.
- cutaway: soap film conc.



Niederhaus, Oakley, Anderson, Ranjan, Bonazza, and Greenough, Physics of Fluids, in review

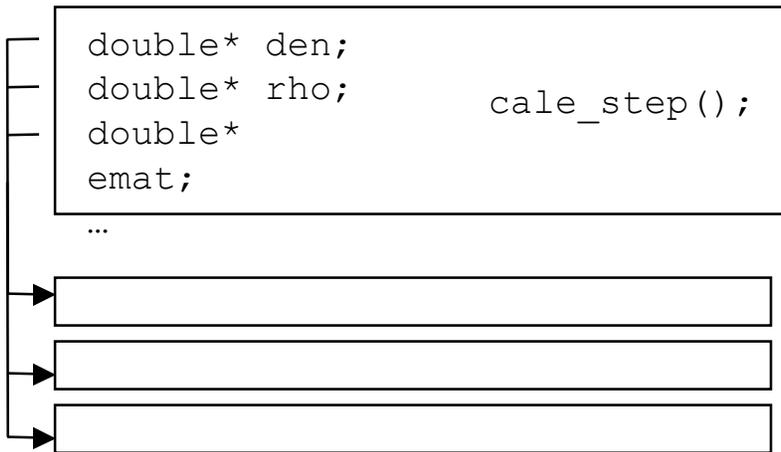
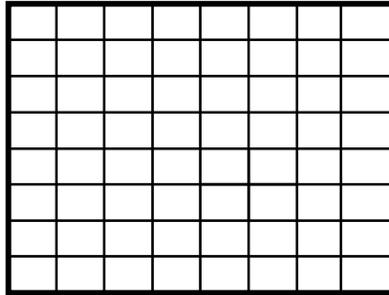
Basic Architecture

CALE

CALE-AMR



single block
structured grid
1 CPU

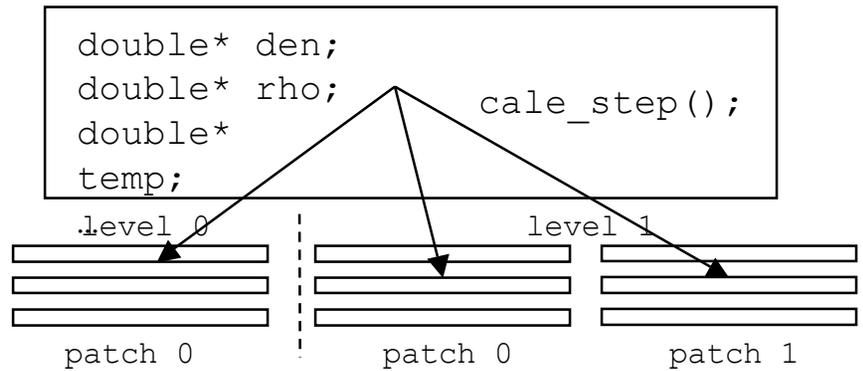
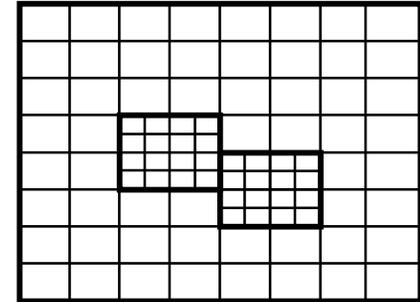


```

timestep loop {
  cale_step();
}

```

multiple patches
multiple levels
N CPU



```

timestep loop {
  communicate_bdry(); // <- SAMRAI
  patch loop {
    set_patch_environment();
    cale_step();
  }
}

```

CALE-internal data dependencies

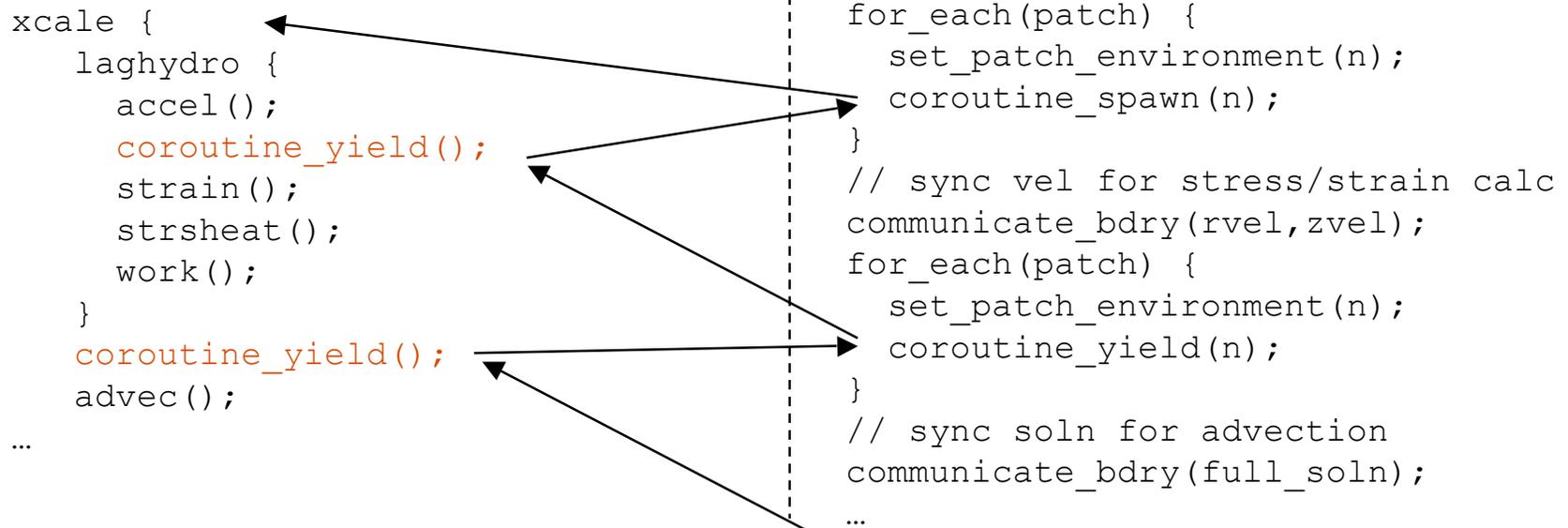


Drilling down into CALE's internals exposes a difficult problem:

```
timestep loop {  
  communicate_bdry(); // <- SAMRAI  
  patch loop {  
    set_patch_environment();  
    cale_step {  
      ... deeply nested with lots of stack-based state  
      compute_new_u();  
      communicate_new_u?(); // new u doesn't exist yet!  
      compute_q(new_u); // <- requires bdry new_u  
      ...  
    }  
  }  
}
```

- Knuth's "coroutines" (1963) neatly solve this problem
 - Permits "start" and "stop" at will without "losing where you are"
 - Needed for multiple patches on a processor
- Neither C nor C++ support coroutines
- Coroutines can be *simulated* using threads with explicit scheduling (threads as software engineering device)

Coroutine-enabled Synchronization

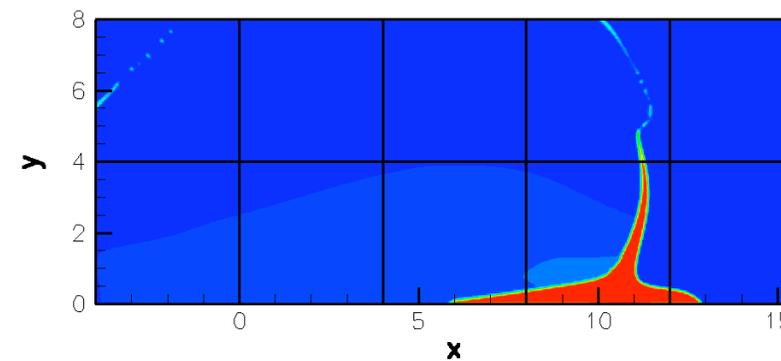
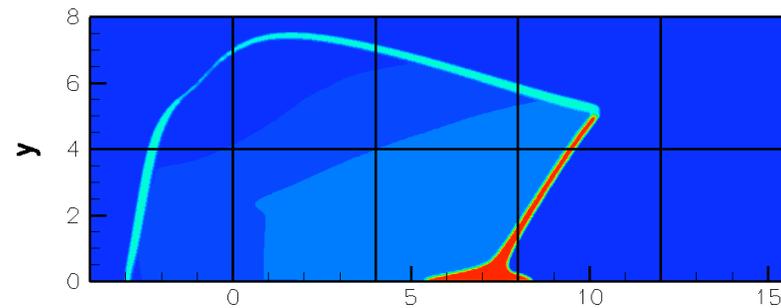
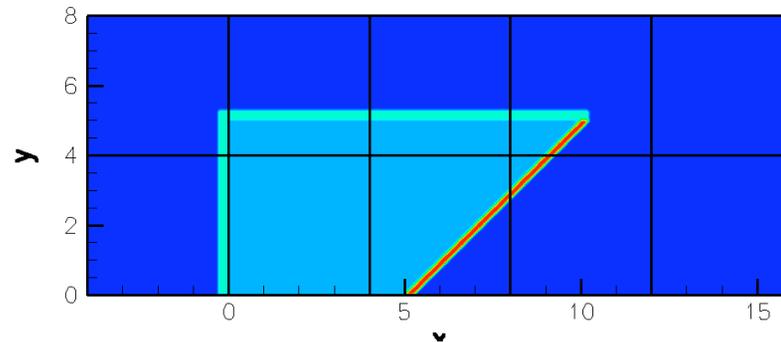


- Powerful device for synchronizing a once serial code
 - Minimally intrusive in existing source base
 - Fully general transfer of control
 - Lossless wrt execution environment
 - stack, temporaries, etc.

Parallel Demonstration: Shaped Charge



- CALE Physics:
 - Eulerian hydro
 - Mixed materials
 - Al, Cu, HE, Air
 - Strength
 - HE Burn
- Ordinary CALE input deck
 - jet02
- 10 processors of mcr

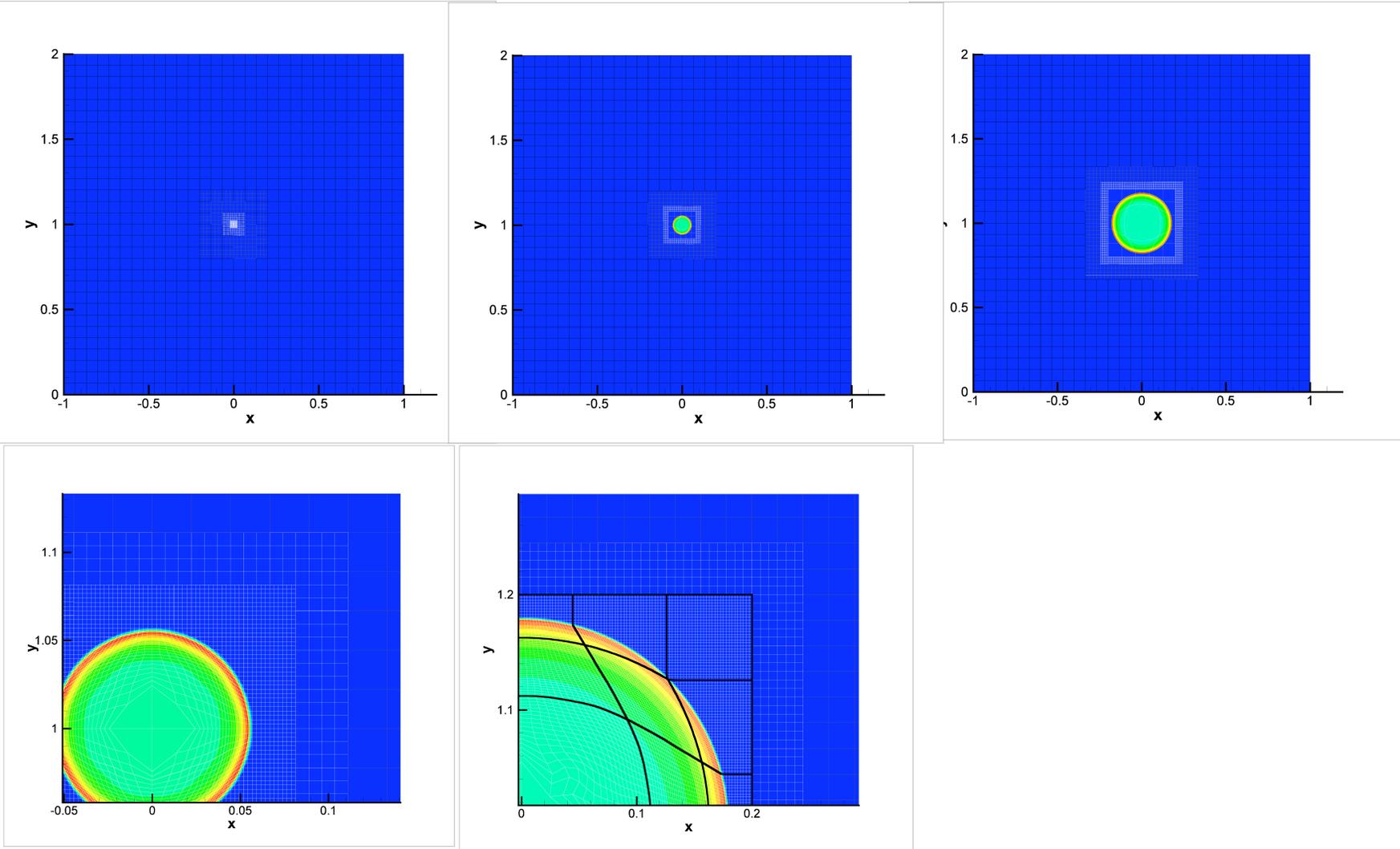




Parallel AMR Demonstration

- **CALE's Lagrangian hydro: simple blast wave**
 - **< 50 lines of modified CALE code**
- **Ordinary, unmodified CALE input deck**
 - **Augmented by SAMRAI input deck for parallel and AMR parameters**
- **16 processors (mcr)**
 - **dynamic load balancing at each regrid**
- **4 grid levels, ratio 3x3**
 - **Base grid 30x30**
 - **Effectively 810x810 at finest level**
- **Refinement criteria: 2nd differences of pressure**

Parallel AMR Demonstration

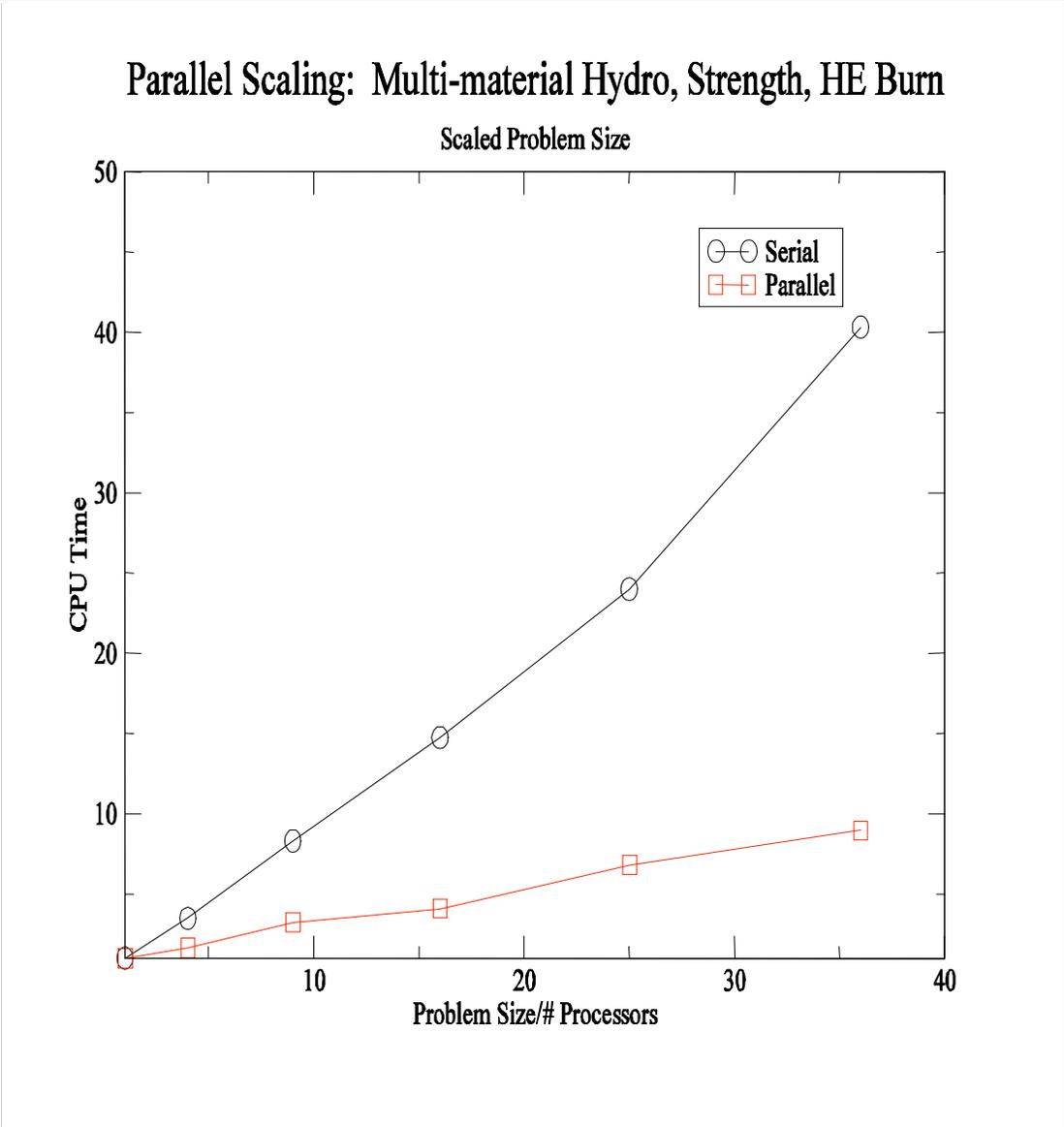




Performance: Overhead

- **Serial overhead: baseline check of library coupling**
 - Sedov problem: 650x325 for 50 steps
 - CALE: 30s, 97% physics
 - CALE-AMR: 31s
 - Context switch (coroutine) overhead is ~ 0
 - Memory: doubles at init time, otherwise ~ 0
- **Parallel overhead:**
 - Mixed zone communication requires care, some opt
 - No SAMRAI induced barriers to efficient comm.
- **Development overhead: $O(100)$ (of 225k) lines of code in orig sources**
 - For comm. synch points, fixing loops for ghost zones
 - $O(10k)$ new, SAMRAI related lines of “separate” code
- **Parallel speedup?**

Performance: Overhead





Current Status

- **Verified to full precision ('diff'able solution):**
 - In parallel:
 - Multi-material clean Lagrangian hydro
 - Mixed Lagrangian (t=0) hydro
- **Verified to "looks right":**
 - jet02 input deck (shaped charge)
 - Mixed material Eulerian (fixed mesh)
 - Strength
 - HE Burn
 - Remaining issue is 1st order advection at bdry of patches
- **AMR capability – spatial refinement**
 - Relatively simple interlevel operators
 - Clean Lagrangian hydro



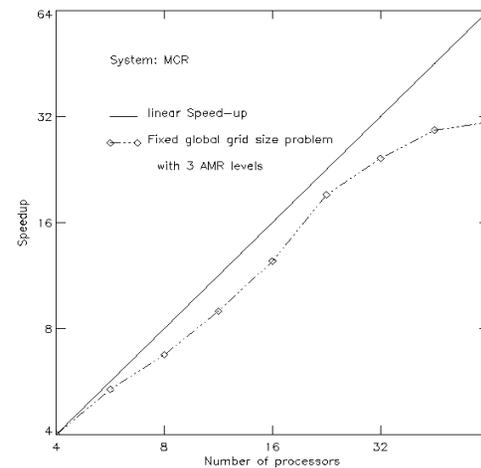
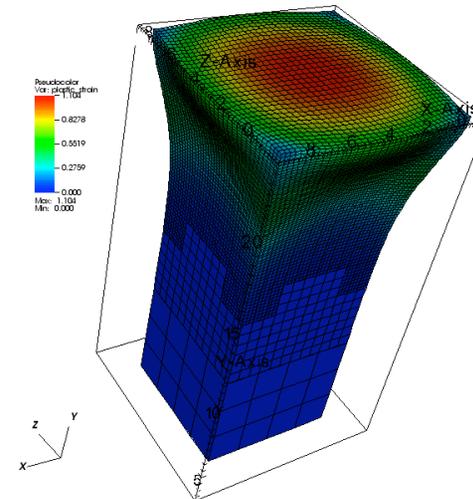
Current Limitations

- **No support yet for:**
 - **Parallel problem generation**
 - **ALE motion (Lagrange or Euler only)**
 - **Slide lines**
 - **Edits**
 - **Interactive graphics (?)**
 - **Other non-local physics: MHD, etc.**
- **On the other hand, many things are essentially “free”:**
 - **Equations of state**
 - **HE Burn**
 - **Sources**
 - **Local physics is generally trivial**
 - **VisIt for visualization**

Elastic-Plastic Materials



- Von Mises yield condition
- New AMR considerations:
 - Interlevel operators for tensor quantities
 - Interpolation/Coarsening
- physical considerations:
 - Interpolation near the elastic-plastic transition
 - Yield condition violation
 - Negative plastic strains
- Current research questions
 - Interpolation near the elastic-plastic transition
 - Yield condition violation
 - Negative plastic strains
- Integrating R. Becker's material modeling library



Design goals for interlevel transfer operators (coarsening and refinement)



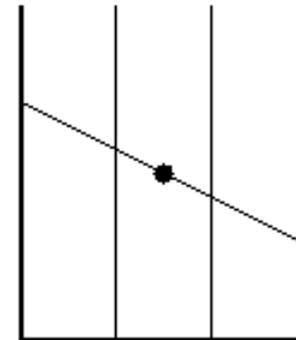
- **Freestream preservation of density, velocity, internal energy**
- **2nd order accuracy (linear reconstruction)**
- **Monotonicity**
- **Local conservation**
- **Exact inversion of refinement by coarsening**

Refinement uses linear interpolation of primitives and volume/mass coordinates



- Ensures
 - Conservation
 - Monotonicity and freestream preservation of primitives
- General 1-d forms:

$$\begin{aligned}\rho_k &= \rho_o + \frac{\Delta\rho_o}{V_o} \left(\bar{v}_i - \frac{1}{2}V_o \right) \\ u_k &= u_o + \frac{\Delta u_o}{\sum \hat{m}_i} \left(\hat{m}_i - \frac{1}{2} \sum \hat{m}_i \right) \\ e_k &= e_o + \frac{\Delta e_o}{M_o} \left(\bar{m}_i - \frac{1}{2}M_o \right)\end{aligned}$$

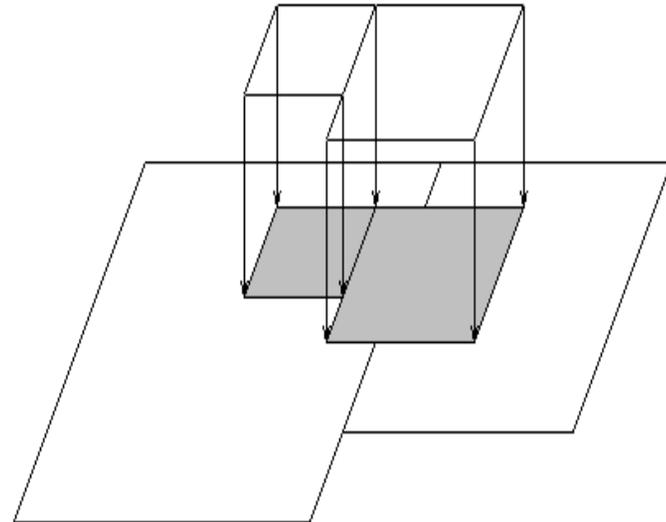


- Dimensionally split and unsplit extensions to 2,

Coarsening uses projection of nodes and simple volume/mass weighted averaging

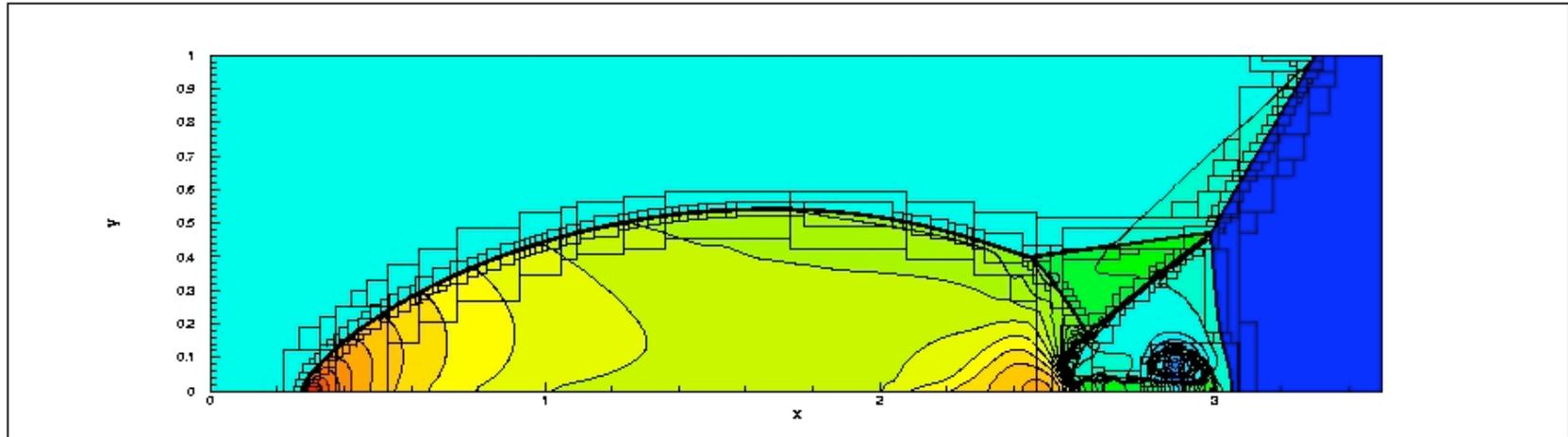


- Mesh is formed by selection of every r 'th mesh point
- Weighted averaging of flow variables constructed to identically invert refinement operator:



$$\rho_o = \frac{\sum \rho_i V_i}{\sum V_i}$$
$$u_o = \frac{\sum u_i \hat{m}_i}{\sum \hat{m}_i}$$
$$e_o = \frac{\sum e_i m_i}{\sum m_i}$$

Eulerian ALE-AMR calculation of Mach 10 double Mach reflection



- Shown: density at $t=0.21$ from 3 level ALE-AMR calculation of Mach 10 double Mach reflection

