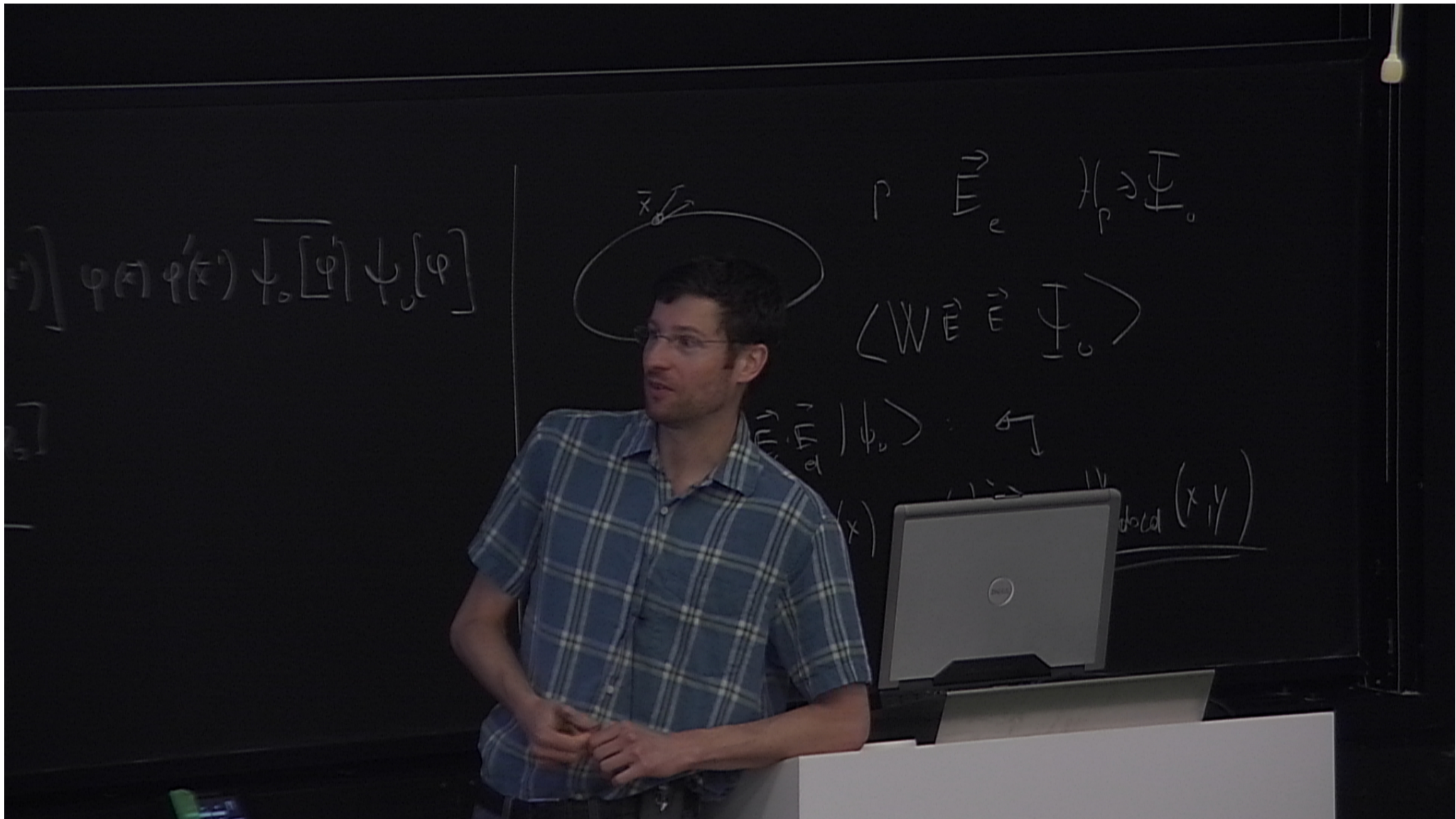


Title: Explorations in Numerical Relativity - Lecture 14

Date: Apr 20, 2012 11:30 AM

URL: <http://pirsa.org/12040056>

Abstract:





# Outline

- Reminder: why we need AMR, and properties of the solutions that dictate the particular “flavor” of AMR that is adequate
- Berger & Oliger style AMR
  - ideal for hyperbolic wavelike equations, and certain classes of problems in GR
  - extensions for coupled hyperbolic/elliptic systems
- PAMR/AMRD
  - infrastructure for implementing B&O AMR on clusters (using MPI)

# AMR

- Adaptive Mesh Refinement is a technique to make the solution of discrete PDEs more *efficient* for *certain classes* of problem
  - there is a wide range of relevant length scales in the problem, yet the smallest length scales are relatively isolated and not volume filling
  - not known a-priori where the small length scales will develop, or it will be too difficult/cumbersome to construct a non-uniform mesh to efficiently resolve the small length scales
  - computationally too expensive to solve the problem on a single uniform mesh



# AMR

- AMR allows for solution of such classes of problems by covering the domain with a *mesh hierarchy*, where high resolution meshes are only added where needed to resolve small length scale features
  - NOTE: AMR is *not* a technique to increase the accuracy of a solution; in fact, the AMR solution can never be more accurate than a unigrid solution with resolution corresponding to that of the finest AMR mesh
  - furthermore, AMR generically creates unwanted high-frequency solution components (“noise”) at refinement boundaries, and though this can be controlled and made small, it is usually quite challenging to get very high accuracy solutions with AMR
  - Think of AMR as a tool to *get* an answer to a computationally challenging problem in the first place; worry about the  $n^{\text{th}}$  digit later

# Why would AMR be beneficial in GR?

- In many astrophysical scenarios where GR is important and numerical solution is needed, in particular binary compact object mergers and gravitational collapse, there is a clean hierarchy of a modest range of metric length scales that need to be resolved
  - compact object radius  $\rightarrow$  near field zone (10's of gravitational radii)  $\rightarrow$  far field zone (100's gravitational radii)
- *in the strong-field regime* small length-scales are isolated (one or two compact objects) and not volume filling
  - however not always the case in GR, e.g. generic cosmological singularities
- *in the strong-field regime* temporal scales are commensurate with spatial scales; i.e. rapid temporal variation of the metric is typically confined to correspondingly small spatial length scales
- the equations are non-linear, and in many cases we will not *a-priori* know where/when refinement will be needed
- maximum causal speed of propagation (1 !)
- *in the weak-field regime* gravitational wave propagation is the feature of interest
  - this *will* be volume filling, and though the temporal scale for variations is always the same, the spatial scales will reflect the relevant scales of the source at the time of emission

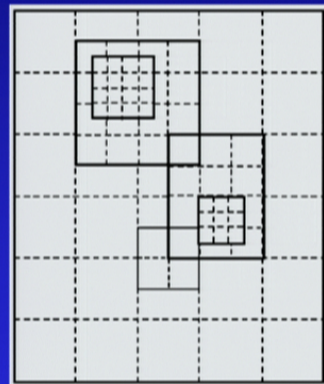


# Implications for an AMR algorithm

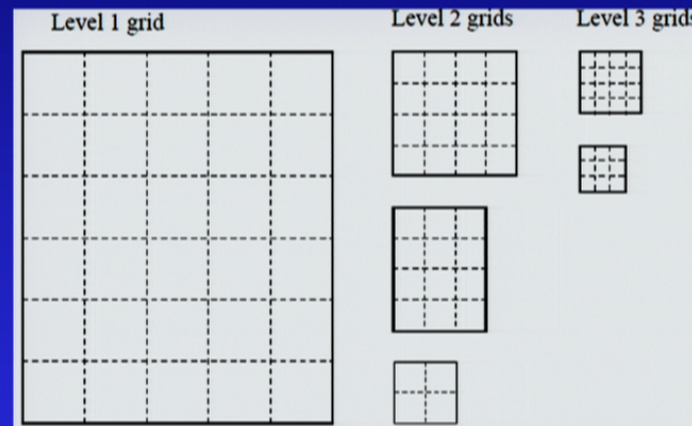
- The preceding properties suggest
  - it is *not* important to have sophisticated grid structures that can efficiently track features with complicated shapes; rather simple “aligned box-in-box” type strategies will be adequate
  - it *is* however important for the algorithm to provide a mechanism to automatically generate the hierarchy as evolution proceeds (i.e. “adaptive”!)
  - it *is* important to use an algorithm that maintains the same CFL factor everywhere in the domain; i.e. need *time-subcycling*
  - AMR by itself, regardless of how sophisticated the algorithm, will *not* help in tracking gravitational wave emission out to large radii with high accuracy ... other technology will be needed to overcome this if it becomes an issue (though in a binary black hole merger the shortest GW wavelength  $\sim 5$  gravitational radii --- not *too* small):
    - changing the spatial coordinates to more efficiently represent the wave structure; e.g. spherical polar coordinates, as the angular structure in the wave will not change by much far from the source, and could efficiently be represented with a relatively small set of multipoles
    - changing the slicing to be asymptotically null, to “quickly” propagate the waves to large radii from the source
- In all then, a simplified version of the original Berger and Oliger AMR algorithm (JCP 53, 1984) is ideal for our purposes
  - though some modifications needed if elliptic equations are solved during evolution

# Berger and Oliger AMR

- (simplified and extended) Berger and Oliger AMR, as implemented in AMRD [FP & Choptuik, JCP 218,2006]
  - computational domain covered by a hierarchy of *independent uniform rectangular meshes*, where *higher resolution child meshes* are *aligned with and entirely contained within coarser resolution parent meshes*



*mesh hierarchy on  
computational domain*



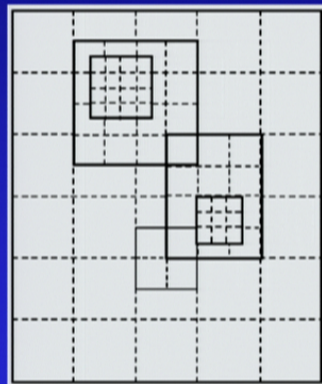
*memory map of grids in hierarchy*

- original algorithm allowed for child meshes to be rotated relative to the parent mesh

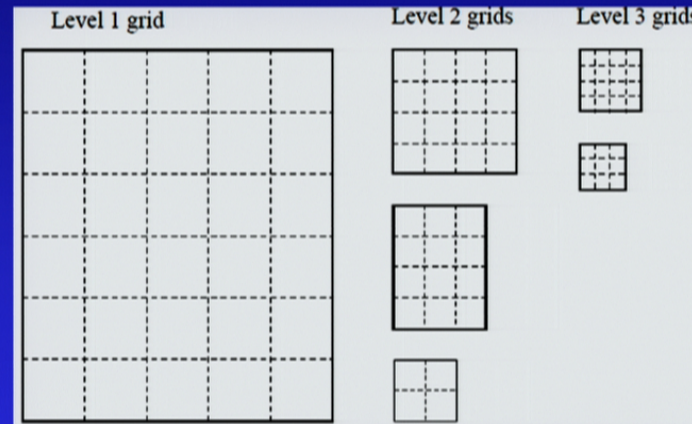


# Berger and Oliger AMR

- (simplified and extended) Berger and Oliger AMR, as implemented in AMRD [FP & Choptuik, JCP 218,2006]
  - computational domain covered by a hierarchy of *independent uniform rectangular meshes*, where *higher resolution child meshes* are aligned *with and entirely contained within coarser resolution parent meshes*



*mesh hierarchy on  
computational domain*



*memory map of grids in hierarchy*

- original algorithm allowed for child meshes to be rotated relative to the parent mesh

# Berger and Oliger AMR

- recursive time stepping algorithm, so refinements occur in space and time (example in a few slides)
  - a single *unigrid* time step is taken on a parent level *before*  $\rho_t$  (temporal refinement ratio) *unigrid* time steps are taken on the child level
    - this ordering is *crucial* to set boundary conditions for interior equations, in particular the elliptics
      - though alternative strategies are possible for purely hyperbolic systems with explicit time integration, or certain classes of linear elliptic PDEs driven by conserved sources [Lehner et al, CQG 23 (2006) S421-S446]
  - allows the AMR technology to be implemented independently of the particulars or details of the numerics used to solve them, and conversely shields the user from AMR implementation details
- after  $\rho_t$  steps on the child grid, when the parent and child are in sync again, solution from the child region is *injected* into the overlapping region of the parent level, so that the most accurate solution available at a point is propagated to all levels of the hierarchy containing that point
- gives near- $O(N)$  (optimal) solution of the PDEs



# Berger and Oliger AMR

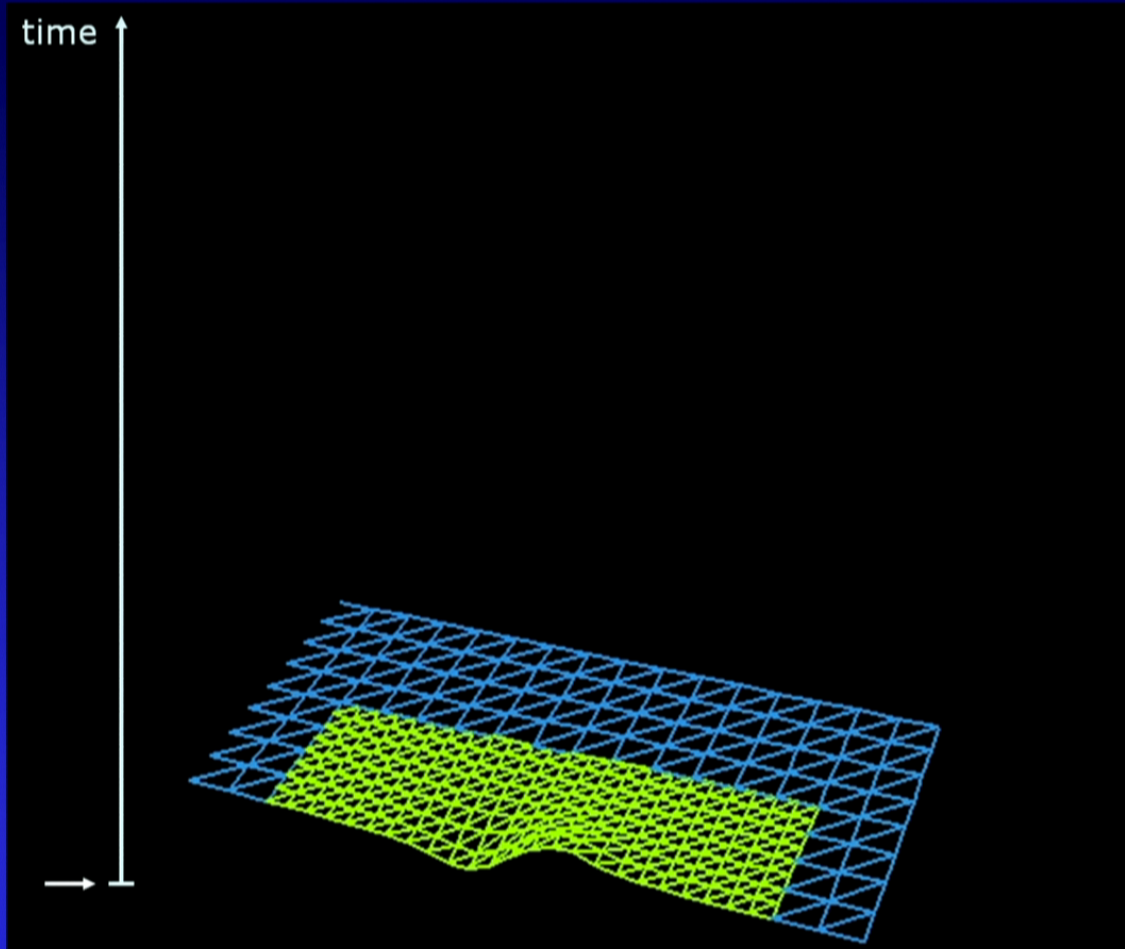
- hierarchy construction driven by truncation error (TE) estimates
  - the B&O proposal to compute this was to periodically make a 2:1 (say) coarsened version of a level in the hierarchy, evolve the two meshes independently for a short time (typically 1 coarse level time step), then  $\Delta / \lambda$  Richardson, subtract the two solutions to give the TE estimate
  - here, use a “self-shadow” hierarchy to obviate the need to duplicate levels
    - due to the recursive nature of the algorithm, just before the fine-to-coarse level injection phase, information to compute TE estimates is naturally available
    - to make this work, simply need to “boot-strap” the procedure by requiring that the coarsest level always be fully refined
    - negligible additional cost ... just choose mesh parameters so that the first refined level is the desired “coarsest” level

# Berger and Oliger AMR

- need to alter the algorithm to incorporate elliptic PDEs
  - for hyperbolic equations, a poorly resolved interior region of a coarse level will not adversely affect the solution on the parts of the level that are locally of the finest resolution, as the “junk” from the under-resolved region does not have more than 1 time step to propagate to the exterior before it is replaced with finer grid solutions
  - the above does *not* hold for elliptic equations. To deal with elliptics, in a nutshell, modify the algorithm as follows:
    - when *descending* the tree in the recursive time-stepping algorithm, evolve hyperbolics one step, using an extrapolated solution of the variables satisfied by elliptic equations
      - getting stable extrapolation is a bit tricky
    - when *ascending* the tree, post injection, solve the elliptics over the entire sub-hierarchy that is in sync with the given coarse level



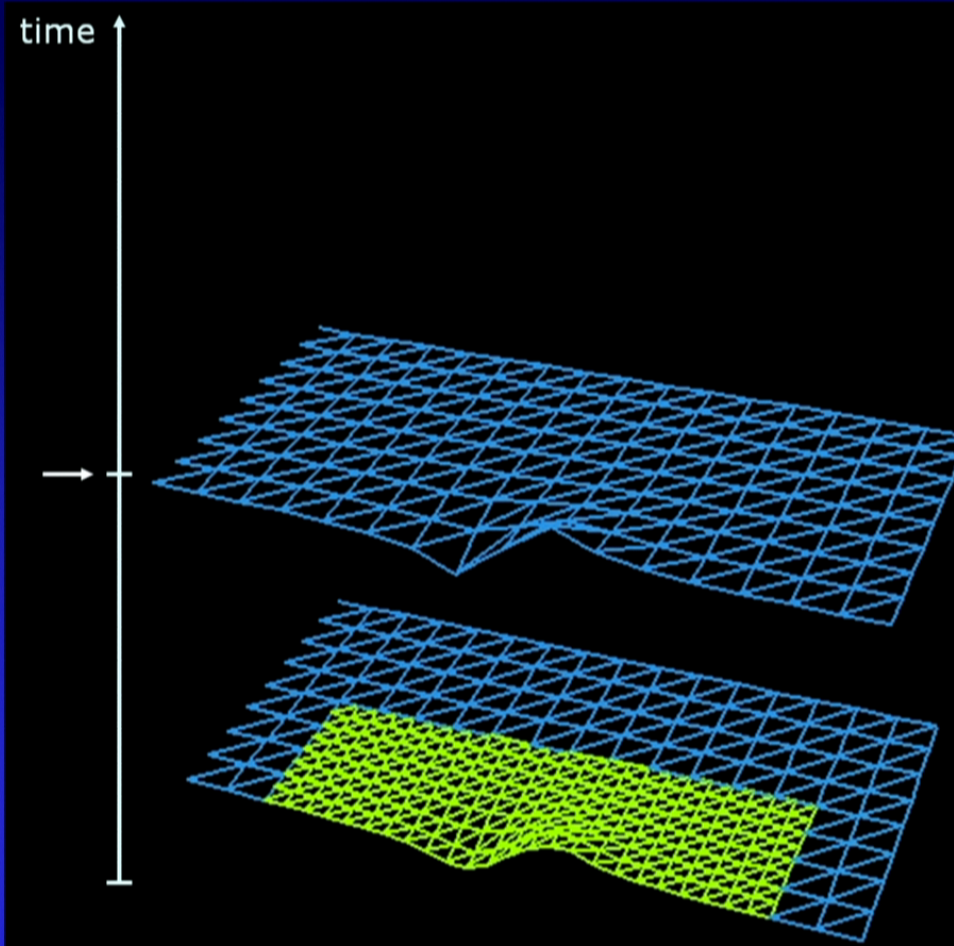
# B&O AMR Example



Initial  
hierarchy

2 Levels  
 $\rho_{sp} = \rho_t = 2:1$

# B&O AMR Example



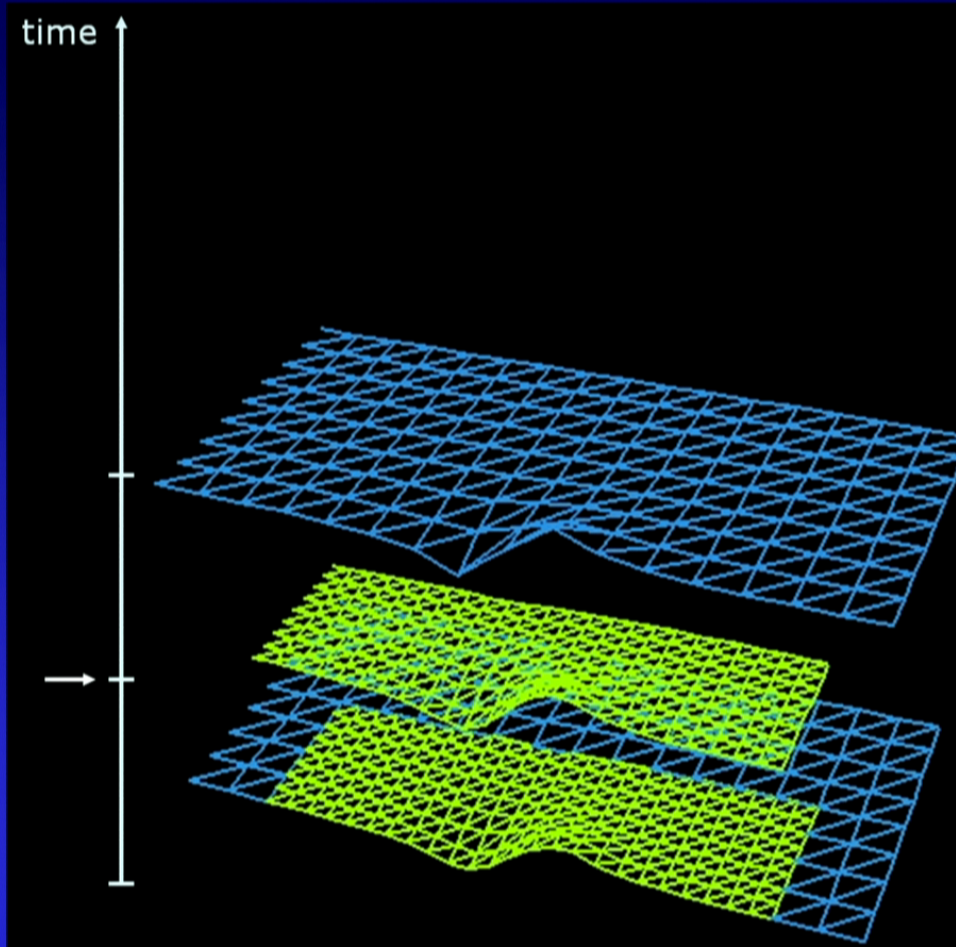
Level 1  
time step

- *evolve hyperbolics on level 1*
- *extrapolate elliptics on level 1 from past time levels*

2 Levels  
 $\rho_{\text{sp}} = \rho_t = 2:1$



# B&O AMR Example

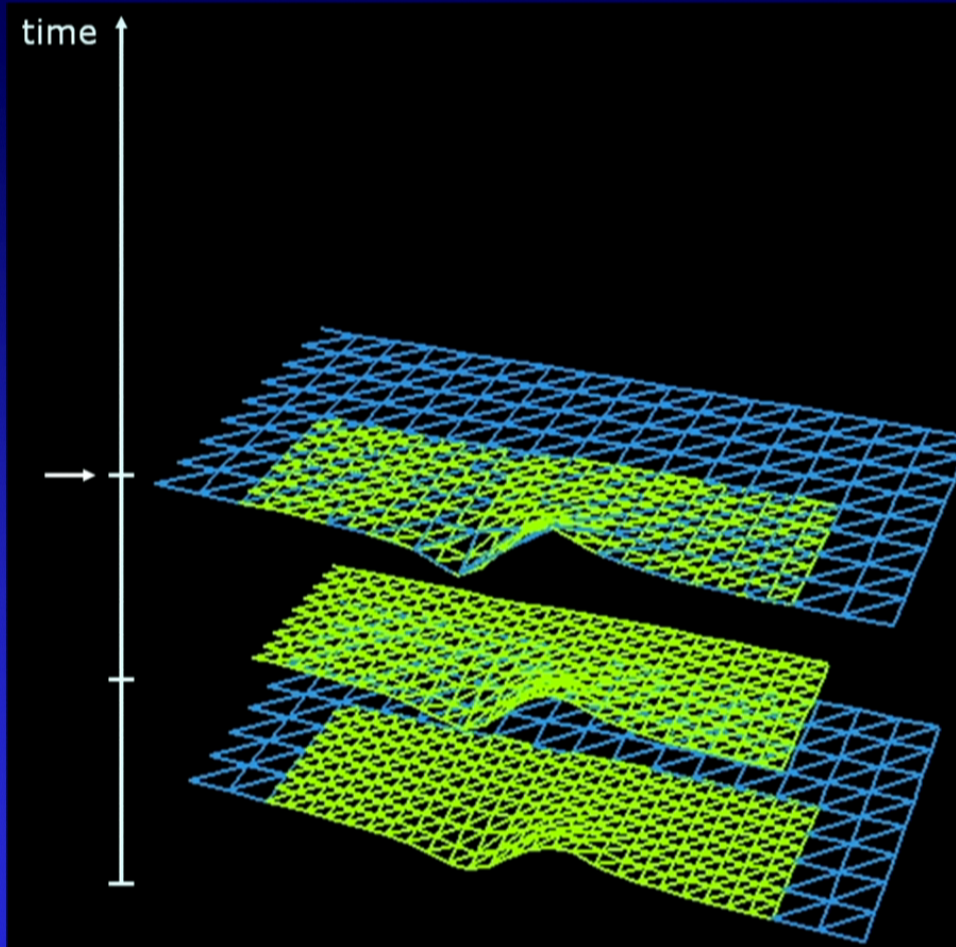


## Level 2 time step

- *evolve hyperbolics on level 2 using interpolated boundary conditions*
- *solve elliptics on level 2 using extrapolated boundary conditions*

2 Levels  
 $\rho_{sp} = \rho_t = 2:1$

# B&O AMR Example



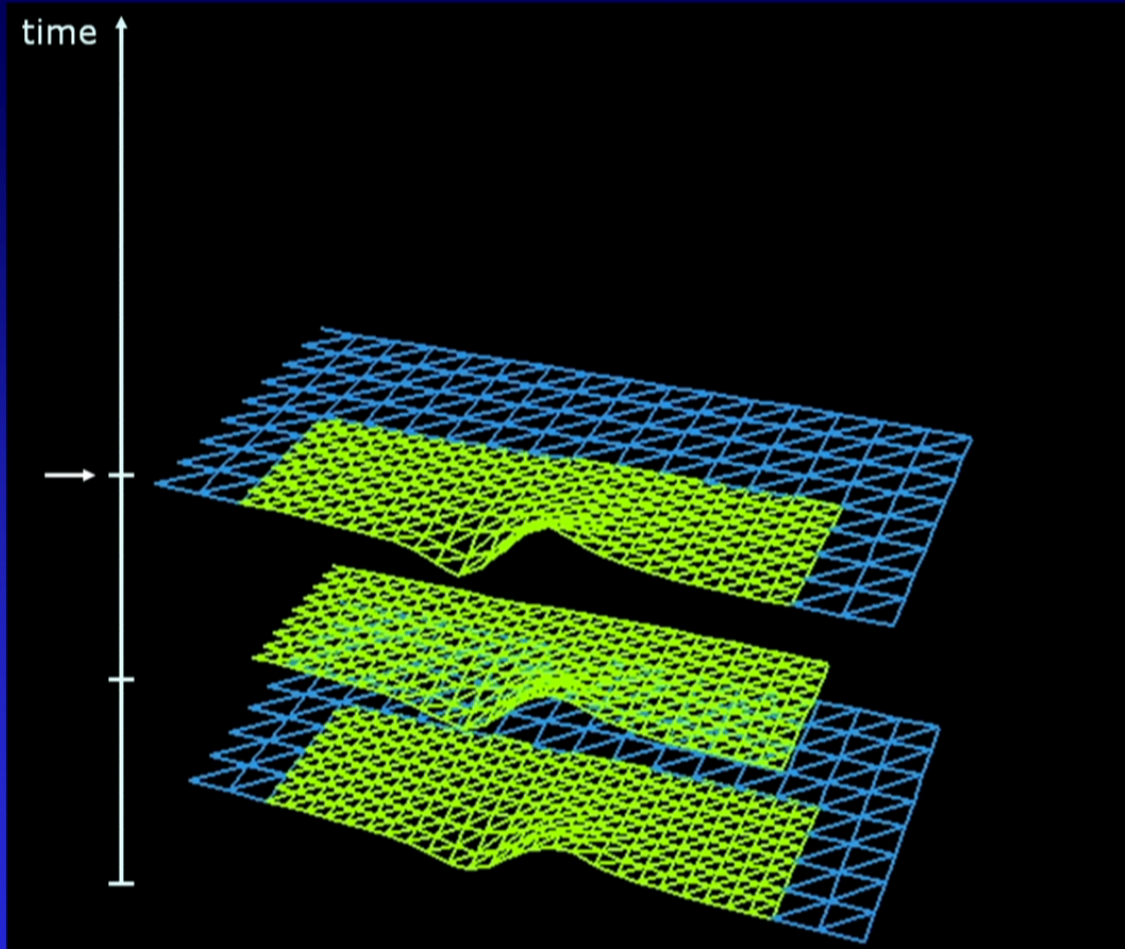
## Level 2 time step

- *evolve hyperbolics on level 2 using interpolated boundary conditions*
- *solve elliptics on level 2 using extrapolated boundary conditions*

*NOTE: at this moment we have all the information we need to compute a truncation error estimate for the solution at level 2*

2 Levels  
 $\rho_{\text{sp}} = \rho_t = 2:1$

# B&O AMR Example



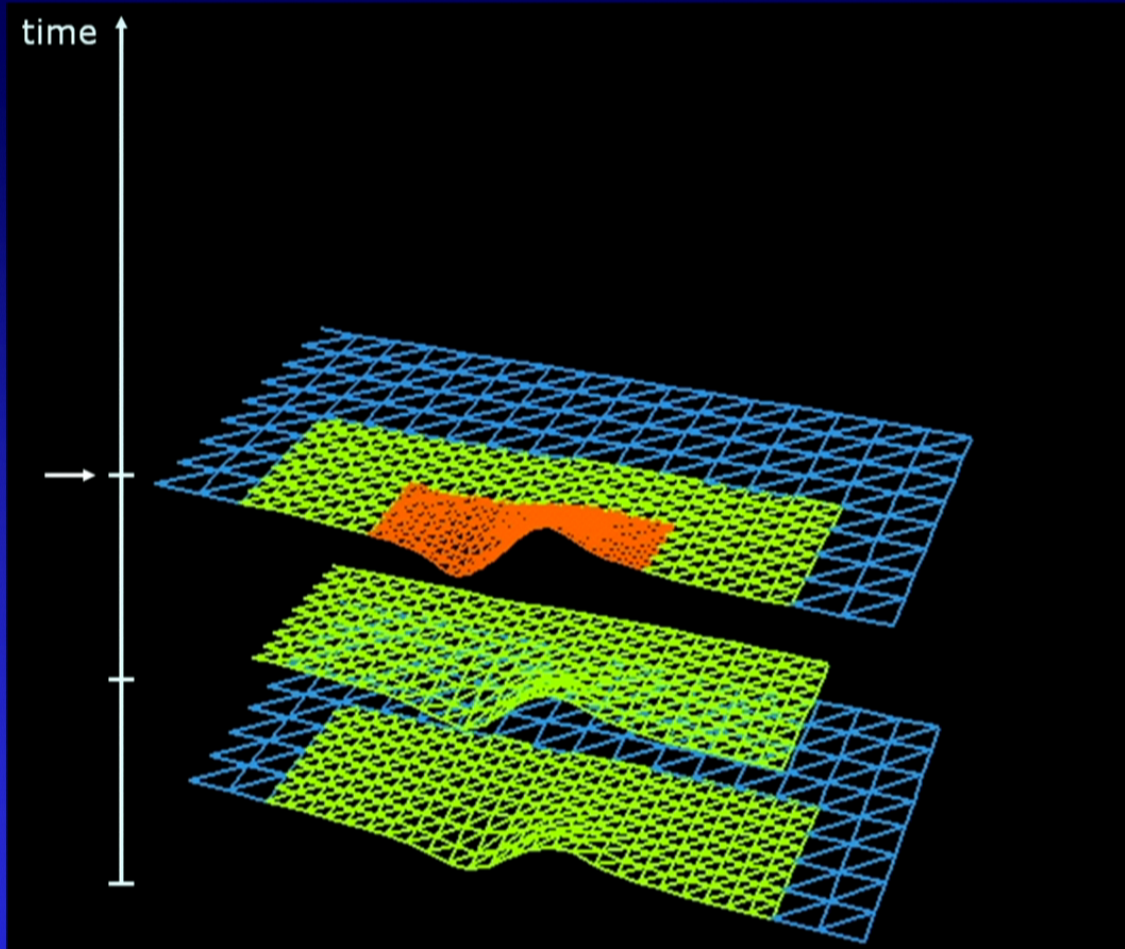
Inject from  
level 2 to 1

- *re-solve elliptics  
over the levels 2 & 1*

2 Levels  
 $\rho_{\text{sp}} = \rho_t = 2:1$



# B&O AMR Example

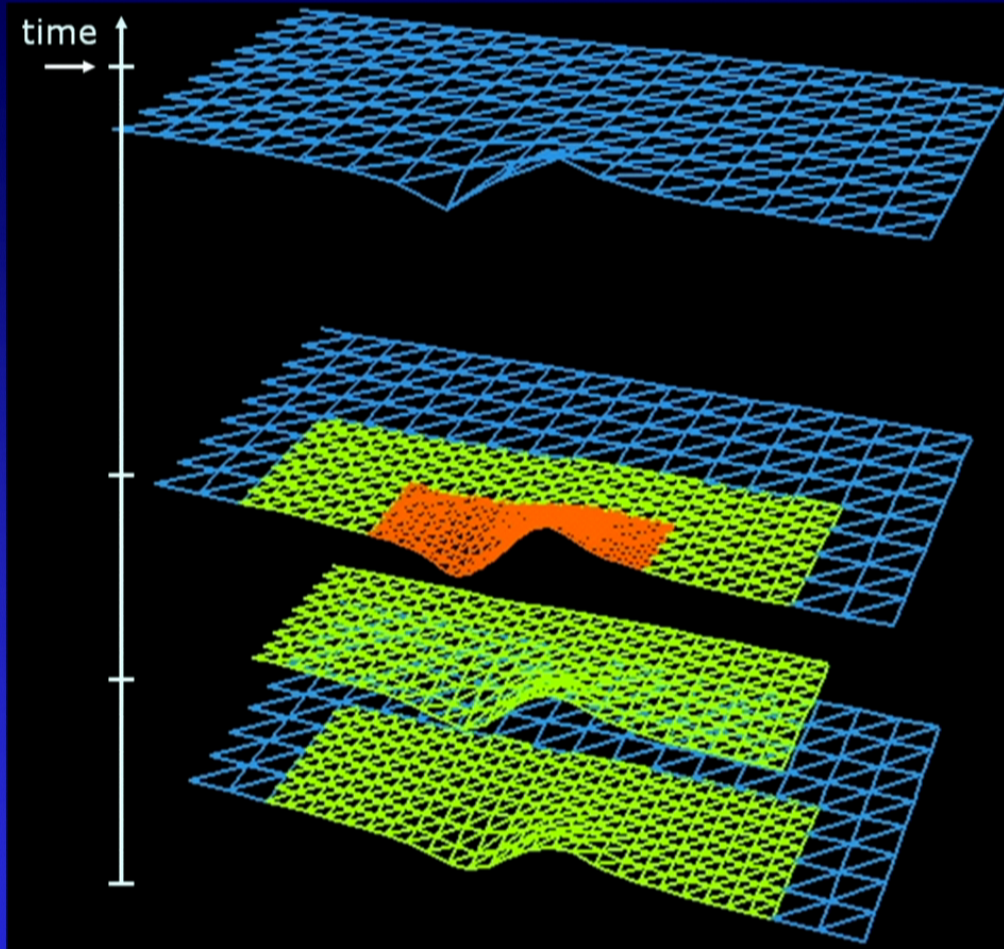


Regrid

3 Levels  
 $\rho_{sp} = \rho_t = 2:1$



# B&O AMR Example

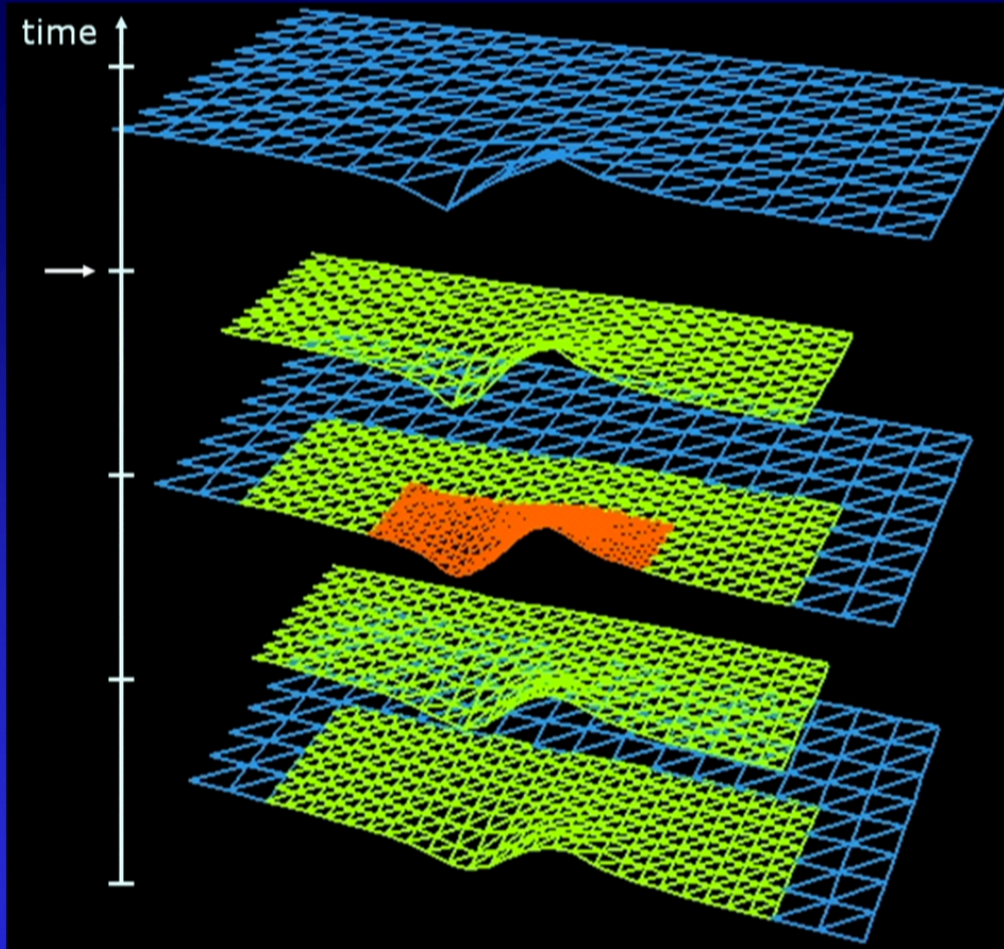


Level 1  
time step

- evolve hyperbolics on level 1
- extrapolate elliptics on level 1 from past time levels

3 Levels  
 $\rho_{\text{sp}} = \rho_t = 2:1$

# B&O AMR Example



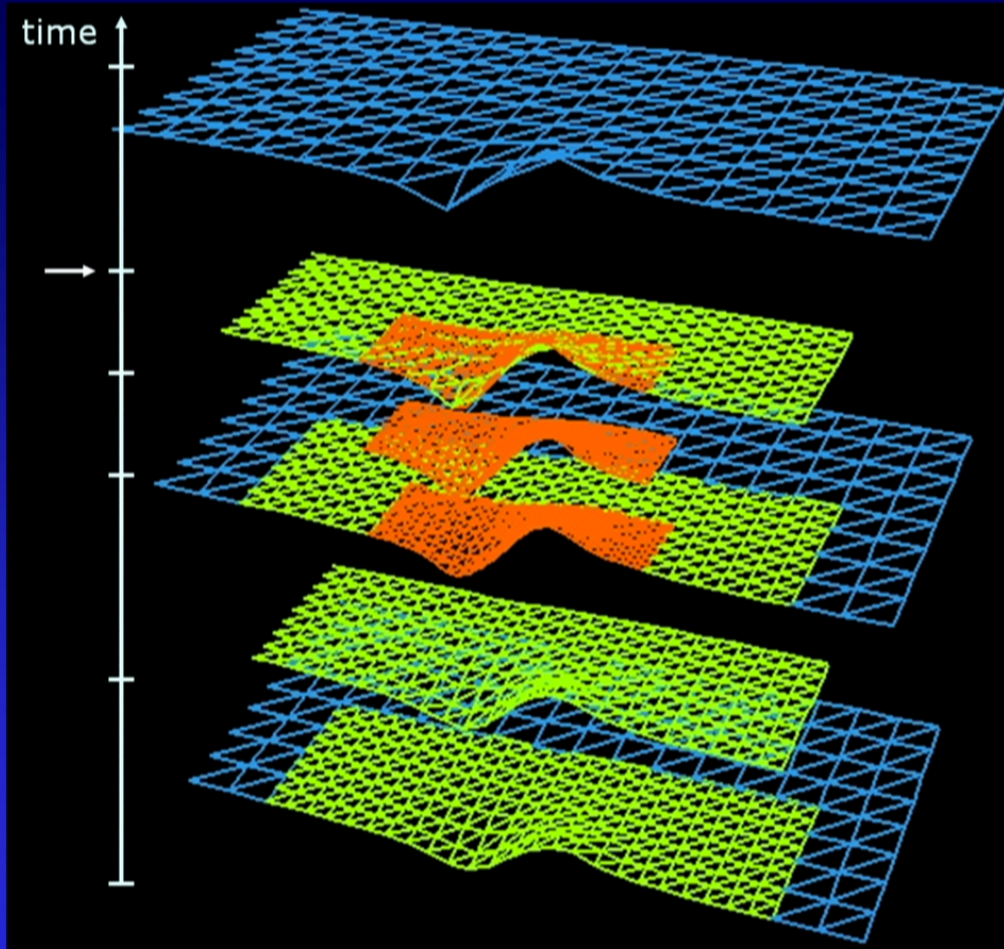
## Level 2 time step

- evolve hyperbolics on level 2 using interpolated boundary conditions
- extrapolate elliptics on level 2 from past times levels

3 Levels  
 $\rho_{sp} = \rho_t = 2:1$



# B&O AMR Example

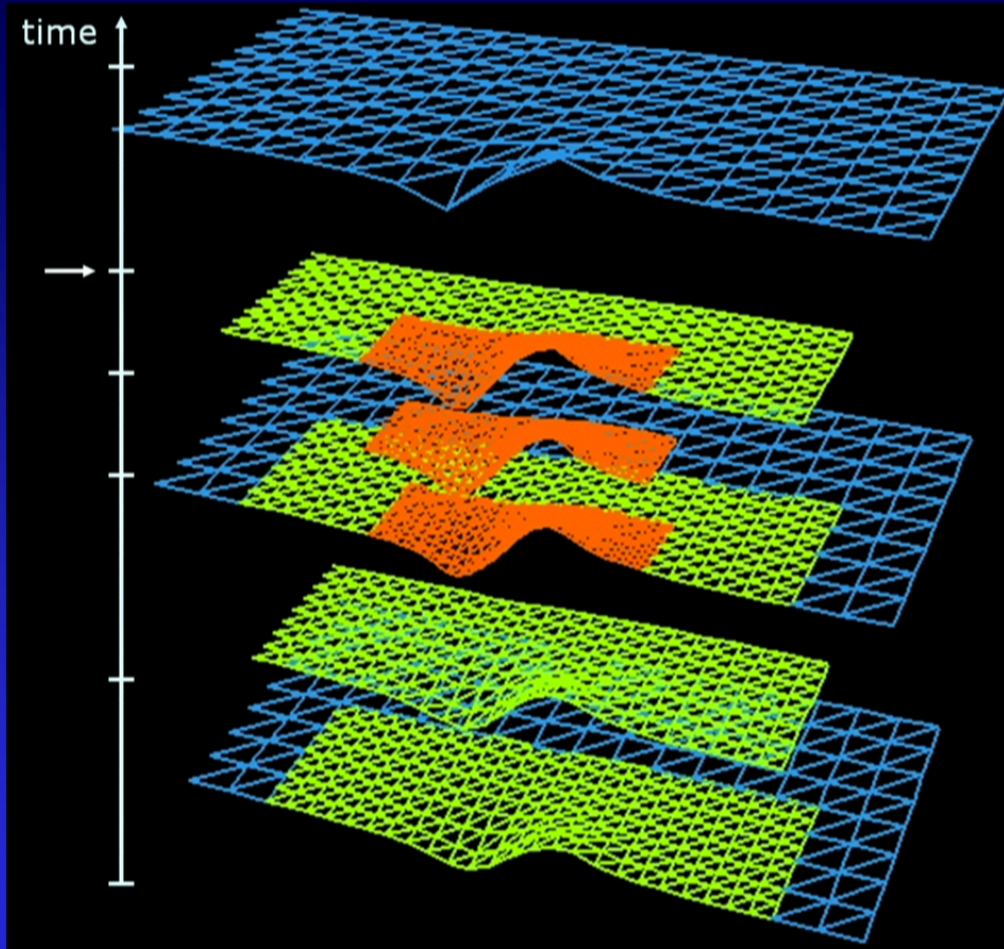


## Level 3 time step

- evolve hyperbolics on level 3 using interpolated boundary conditions
- solve elliptics on level 3 using extrapolated boundary conditions

3 Levels  
 $\rho_{\text{sp}} = \rho_t = 2:1$

# B&O AMR Example



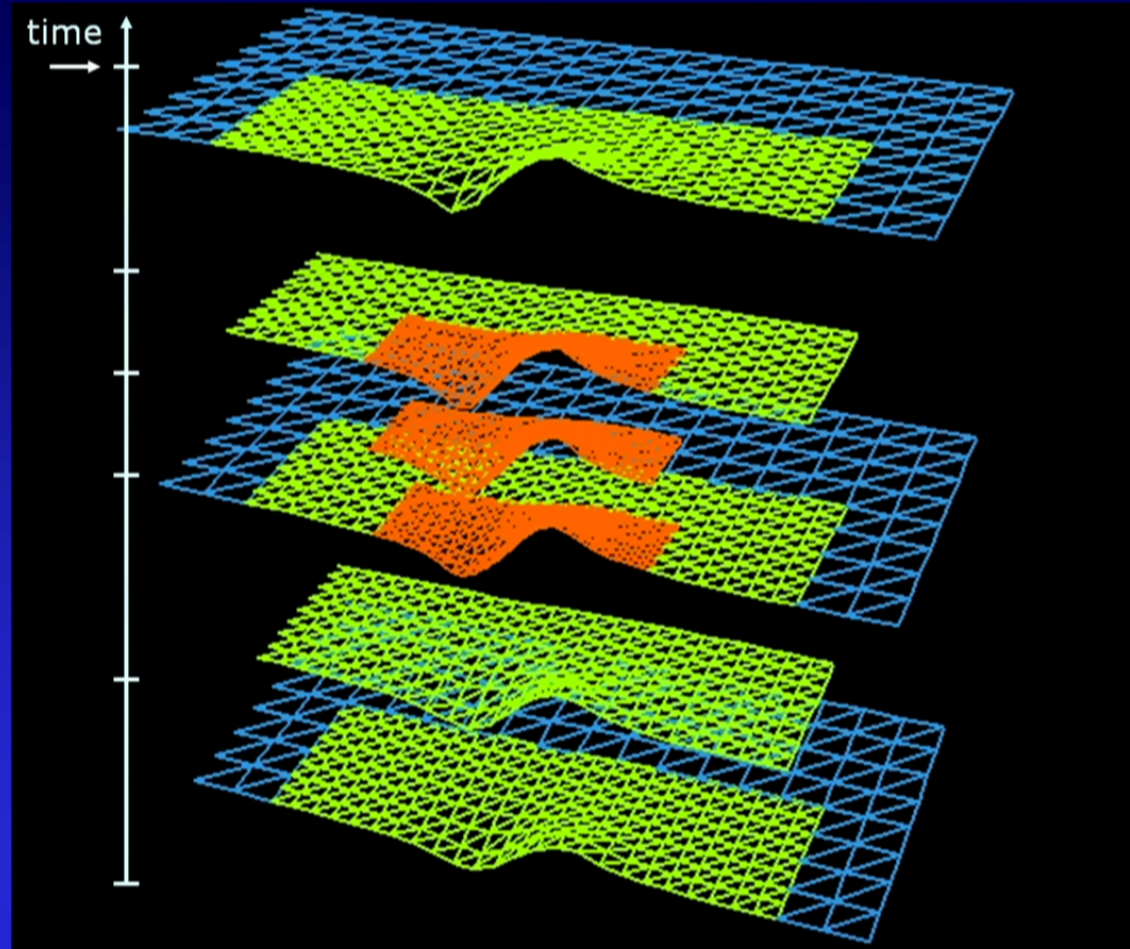
Inject from  
level 3 to 2

- *re-solve elliptics  
over the levels 3 & 2,  
using extrapolated  
boundary conditions*

3 Levels  
 $\rho_{\text{sp}} = \rho_t = 2:1$



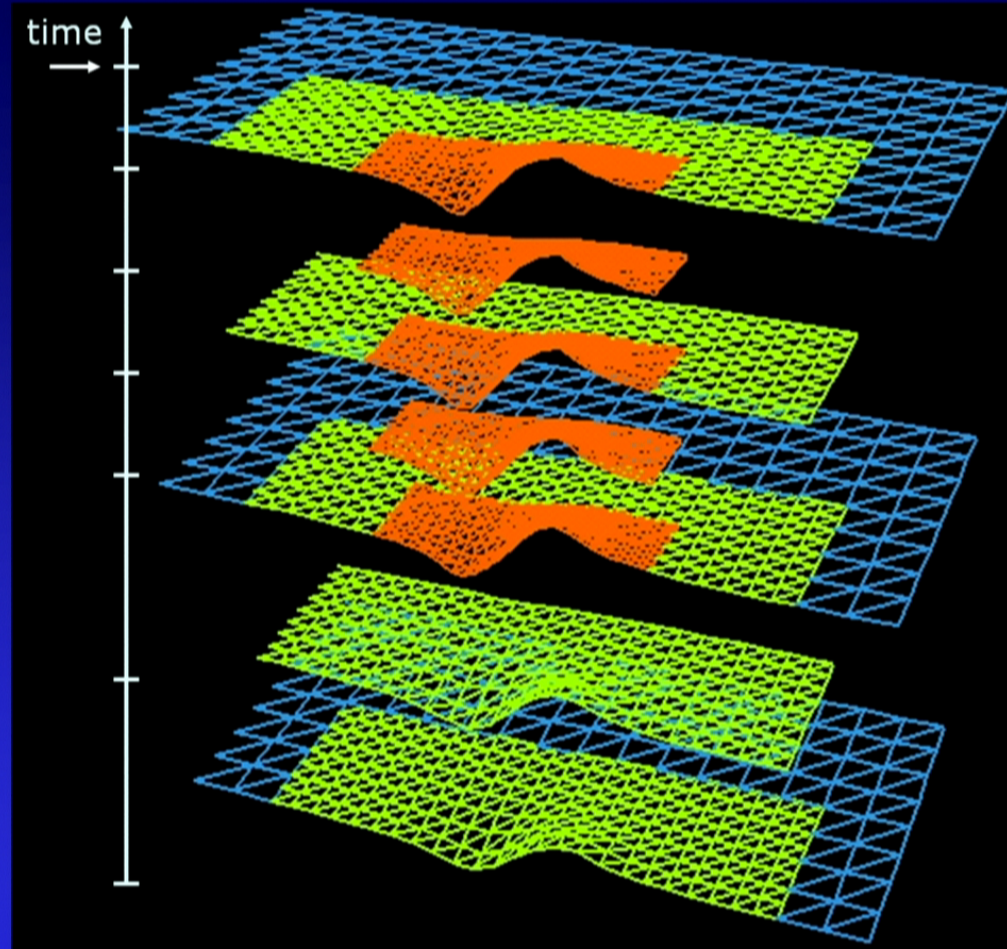
# B&O AMR Example



Inject from  
level 2 to 1

3 Levels  
 $\rho_{sp} = \rho_t = 2:1$

# B&O AMR Example



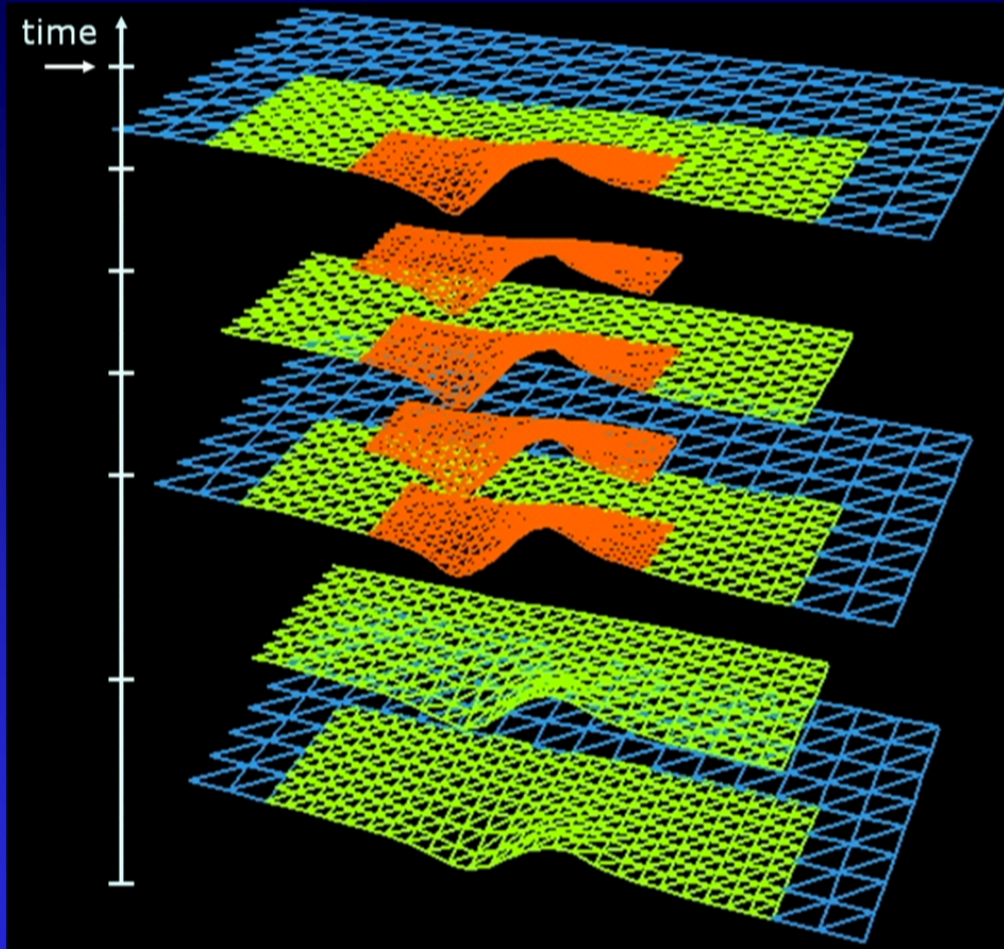
Inject from  
level 3 to 2  
to 1

- *re-solve elliptics  
over the levels 3, 2  
and 1*

3 Levels  
 $\rho_{\text{sp}} = \rho_t = 2:1$



# B&O AMR Example



Inject from  
level 3 to 2  
to 1

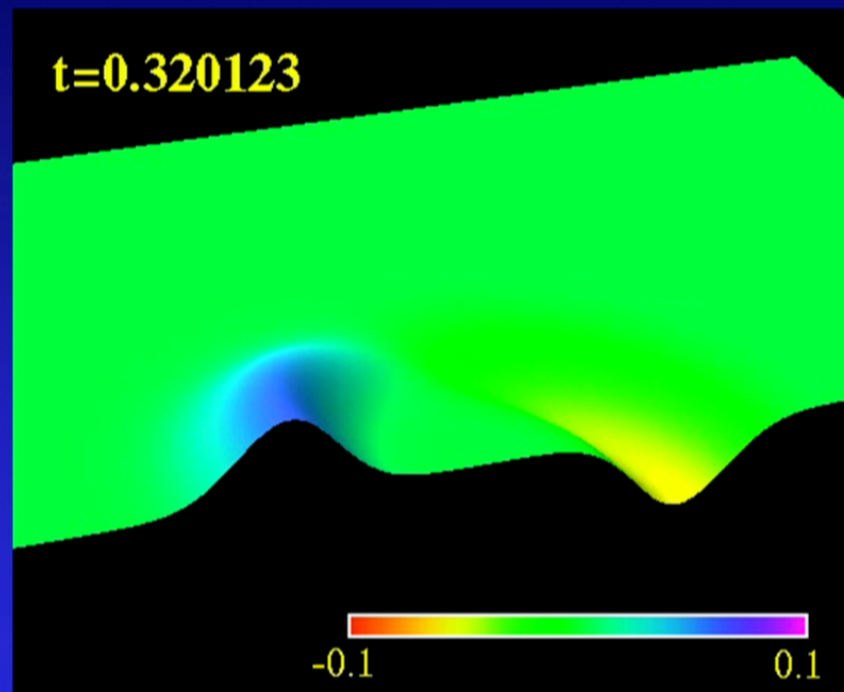
- *re-solve elliptics  
over the levels 3, 2  
and 1*

3 Levels  
 $\rho_{\text{sp}} = \rho_{\text{t}} = 2:1$

## Evolution of plane anti-symmetric initial data

*Initial data that is reflection anti-symmetric about  $z=0$  (a conserved symmetry)*

$p \sim p^*$   
(tuned to within  
1 part in  $10^{15}$ )



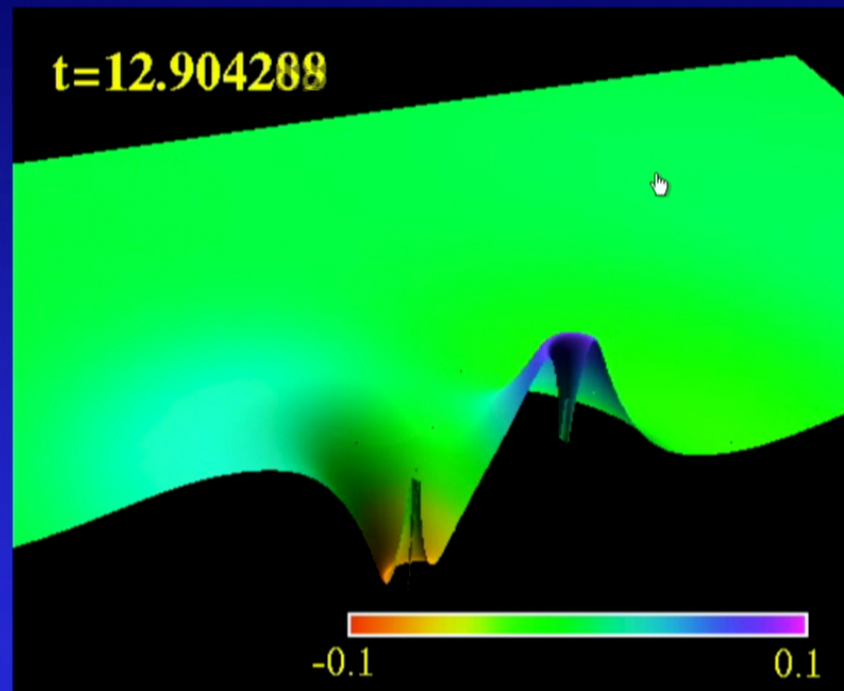
$\rho$   
 $z$



# Evolution of plane anti-symmetric initial data

*Initial data that is reflection anti-symmetric about  $z=0$  (a conserved symmetry)*

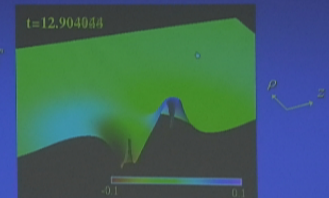
$p \sim p^*$   
(tuned to within  
1 part in  $10^{15}$ )



Evolution of plane anti-symmetric initial data

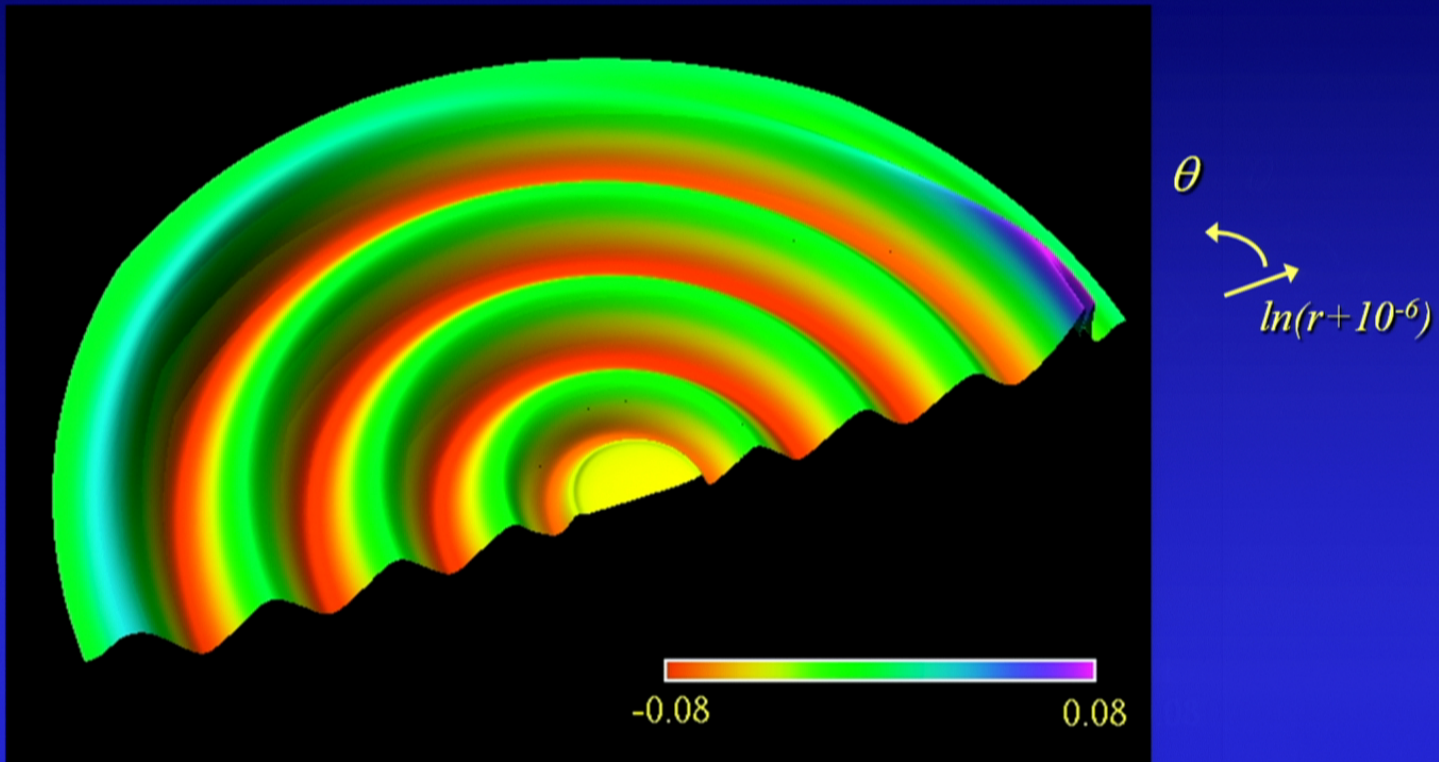
*Initial data that is reflection anti-symmetric about  $z=0$  (a conserved symmetry)*

$p \sim p^*$   
(tuned to within  
1 part in  $10^{15}$ )



## Evolution of plane anti-symmetric initial data

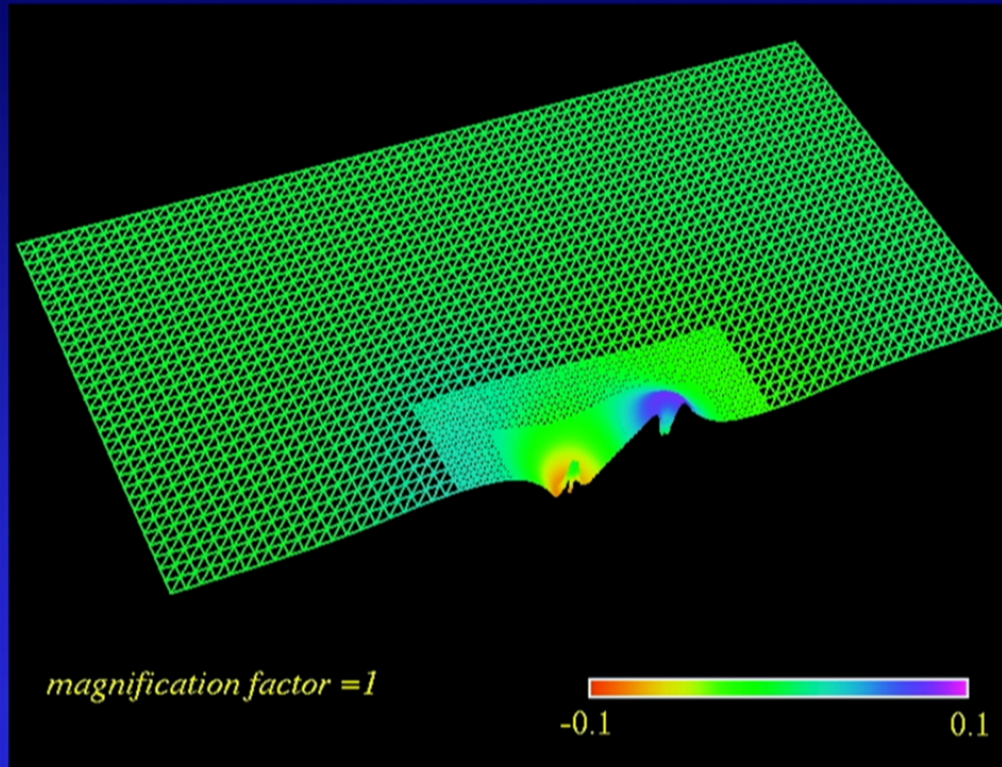
*Last frame from the previous animation, transformed to a logarithmic radial coordinate centered about the left most echoer*





## Aside: AMR grid hierarchy sample

*Last frame from the previous animation*

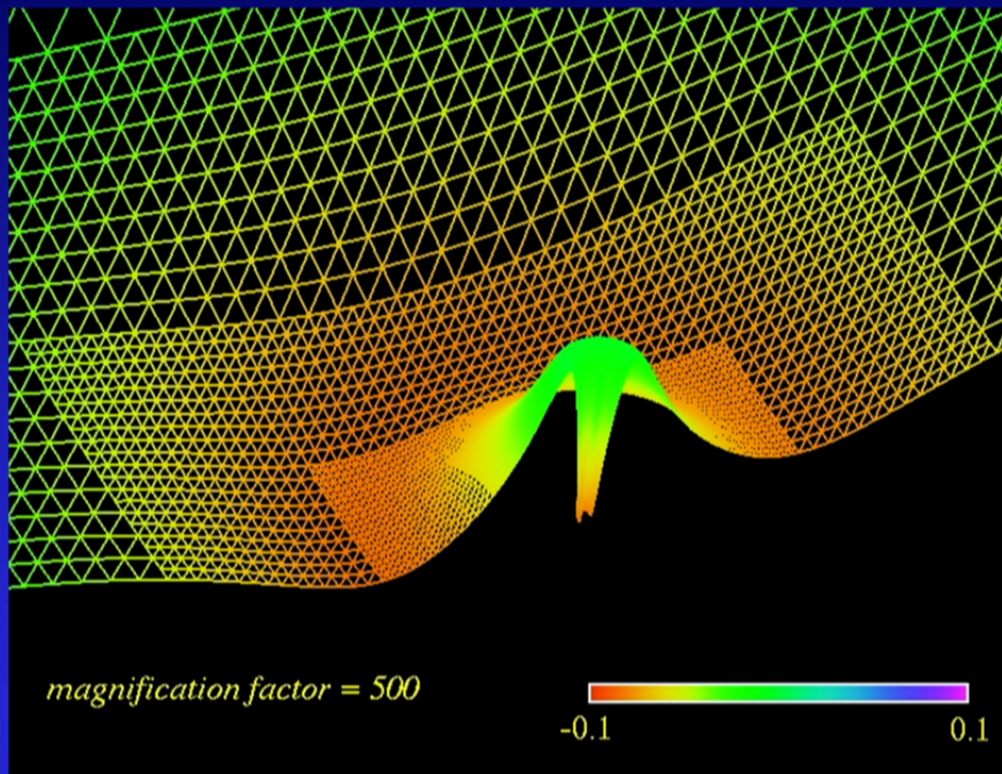


25, 2:1 refined levels  
(2:1 coarsened in  
figure)



## Aside: AMR grid hierarchy sample

*Last frame from the previous animation*



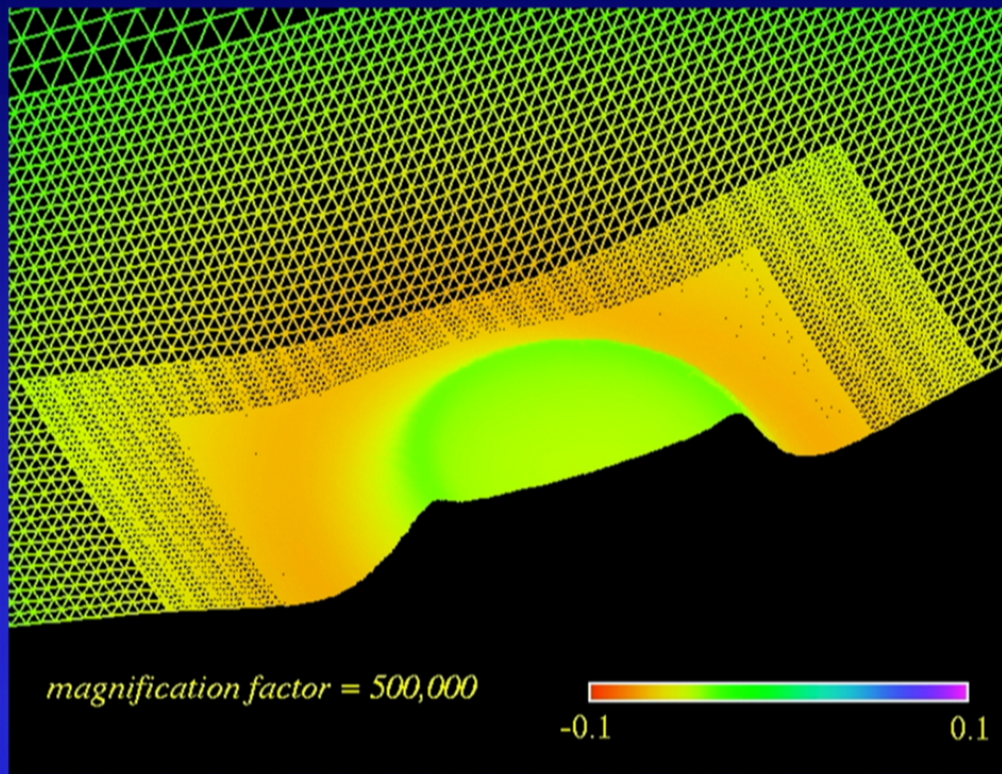
25, 2:1 refined levels  
(2:1 coarsened in  
figure)





## Aside: AMR grid hierarchy sample

*Last frame from the previous animation*



25, 2:1 refined levels  
(2:1 coarsened in  
figure)

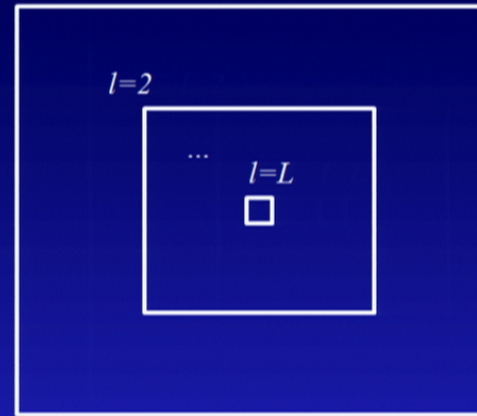


# Optimistically, what kind of speedup can we expect?

## ■ Imagine

- $d+1$  dimensional evolution
- the coarsest level has  $N^d$  points
- 2:1 spatial and temporal refinement ratio
- $L$  levels of refinement, with  $l=1$  the coarsest level, and  $l=L$  the finest
- take  $N$  steps on the coarsest level; hence will need  $N^{2(l-1)}$  on the level  $l$
- linear filling factor of  $1/2$
- the total run-time is proportional to the total number of grid points in space and time (i.e. an optimal evolution scheme is used), and the overhead in the AMR algorithm is negligible
- compare to a unigrid run at the resolution of the finest AMR level

$l=1$



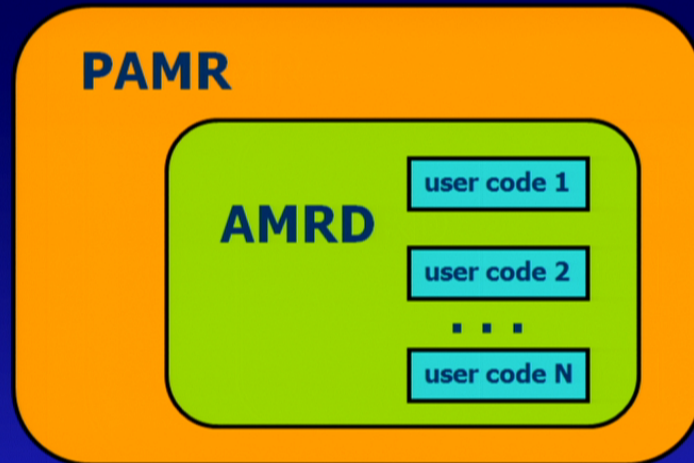
$$T_{AMR} = C \sum_{l=1}^L N^d N 2^{(l-1)} < C N^{d+1} 2^L$$

$$T_{UNIGRID} = C [N 2^{(L-1)}]^{d+1}$$

$$\frac{T_{UNIGRID}}{T_{AMR}} > 2^{d(L-1)-1}$$



# PAMR/AMRD



- *PAMR* (parallel adaptive mesh refinement) manages distributed B&O style grid hierarchies
  - *AMRD* (adaptive mesh refinement driver) implements the just-described version of B&O AMR, utilizing PAMR for hierarchy management
  - *User codes* designed as (in-principle) standalone unigrid/serial numerical solvers, and supply AMRD with a series of “hook functions” to incorporate them into the B&O algorithm
- 
- Reasons for this separation of functionality
    - from the point of view of a user writing a code to numerically solve a particular system of PDEs, AMR and parallel distribution are largely extraneous details
      - all the user should be aware of is the possibility that the code *could* be run in a parallel/adaptive environment, meaning grid boundaries could either be at the physical boundaries of the problem, or interior to the domain
        - in the latter case the user leaves the boundaries alone
    - The AMR driver does not need to know the details of how grids will be distributed in parallel, nor what equations the user will be solving on those grids
    - PAMR handles the non-local aspects of parallel grid distribution, and does not care what the underlying programs will do with the grids

# PAMR

- Takes care of most parallel grid distribution issues
  - support for 1,2 and 3D grids, with or without periodic boundaries
  - support for interwoven AMR/multigrid hierarchies
  - simple base application program interface (API)
    - `PAMR_compose_hierarchy()` : regrid function
    - `PAMR_sync()` : synchronize data across ghost zones
    - `PAMR_inject()` : fine-to-coarse level injection
    - `PAMR_interp()` : coarse-to-fine level interpolation
  - a complete set of data structure management API's, so that it can be called from fortran programs
  - supports vertex and cell-centered arrays



# AMRD

- Built on top of PAMR, hence “parallel ready”
- Implements a Berger and Oliger AMR algorithm, modified to support integrated solution of elliptic equations
- Provides a standard full approximation storage (FAS) adaptive multigrid algorithm
- User supplies a set of “hook functions” that are called by AMRD to perform the problem specific numerics
- Berger and Colella algorithm for conservative hydrodynamics (essentially applies flux corrections at AMR boundaries for conservative variables to preserve global conservation)

# A few final remarks

- For the Einstein equations, time taken to evaluate expressions dominates over other tasks, which helps guide coding priorities
  - 'locality' of the data less of an issue in load-balancing a parallel code (strategies designed to guarantee locality, such as space filling curves, may even have a negative impact on the performance)
  - algorithmic tasks (truncation error estimation, regridding, interpolation, injection, etc.) are essentially “free”
- Solving elliptic equations solved using FAS multigrid is optimal and *fast*
  - at *worst* a constant factor of 2-3 times slower per equation compared to a typical hyperbolic equation
  - for example, 2D axisymmetric gravitational collapse code solves 4 (3) hyperbolic equations and 3 (4) elliptic equations per time step; profiling indicated roughly 25-45% of the time is spent solving hyperbolics, 50-70% solving elliptics, with the remainder (usually  $\sim 5-10\%$ ) spent on miscellaneous functions in a typical simulation
- Code, including reference manuals and a couple of examples, can be downloaded from Matt Choptuik's web-page (google “Matt Choptuik”, or see links from my web-page)