Wave propagation algorithms in two dimensions, adaptive refinement and well-balancing

Donna Calhoun

Boise State University

PASI 2013
Valparaiso, Chile
January 2-13

Plan

- [x] How do these Riemann solvers make it into an actual code? Clawpack is based on solving Riemann problems.
- [x] Do we actually solve the non-linear problem at every grid cell interface? No! One can use approximate Riemann solvers.
- [x] How accurate are these methods? (second order, or high resolution with limiters)
- [ ] Well-balancing
- [ ] What do we do in two-dimensions?
- [ ] Adaptive mesh refinement?

Well-balanced schemes

Shallow water wave equations

$$h_t + (uh)_x = 0$$

$$(hu)_t + (hu^2 + \frac{1}{2}gh^2)_x = -ghB_x$$

In steady state, the bathymetry source term should balance the remaining terms on the left hand side. But a naive operator split approach to treating the source term will lead to oscillations about a steady state because the truncations errors in the different numerical methods do not cancel nicely.

F-wave approximate Riemann solver

Roe solver requires that we find a state $\widehat{q}$ between Q_i and Q_{i-1} that satisfies

$$f(Q_i) - f(Q_{i-1}) = A(\widehat{q})(Q_i - Q_{i-1})$$

This may not always be possible for some systems. In this case, one can decompose the jump in the flux difference directly into eigenvectors of the linearized system

$$f(Q_i) - f(Q_{i-1}) = \sum \beta^p r^p \equiv \sum_p \mathcal{Z}^p_{i-1/2}$$

where r^p are the eigenvectors of $\bar{A} = f'(\bar{q})$ for some $\bar{q}$ (not necessarily Roe averages).

F-wave approach, (Bale, LeVeque, Mitran, 2002)

Maintaining steady states

If we incorporate source terms directly into the splitting, we will get a *well-balanced* scheme.

$$f(Q_i) - f(Q_{i-1}) - \Delta x \Psi(q) = \sum \beta^p r^p \equiv \sum_p \mathcal{Z}^p_{i-1/2}$$

bathymetry

Steady states can be maintained exactly.

The conventional approach to handling source terms involves operator splitting, and can lead to spurious oscillations in the solution which may be on the order of the phenomena being investigated.

F-wave approach

Define "fluctuations"

$$\mathcal{A}^+ \Delta Q_{i-1/2} \equiv \sum_{\lambda^p > 0} \mathcal{Z}^p_{i-1/2}$$

$$\mathcal{A}^- \Delta Q_{i+1/2} \equiv \sum_{\lambda^p < 0} \mathcal{Z}^p_{i+1/2}$$

Then an update formula looks like

$$Q_i^{n+1} = Q_i^n - \frac{\Delta t}{\Delta x}\left(\mathcal{A}^+ \Delta Q_{i-1/2} + \mathcal{A}^- \Delta Q_{i+1/2}\right)$$

$$+ \quad \text{(second order correction terms)}$$

Bathymetry in GeoClaw is handled in the Riemann solver. Only friction coefficients and the Coriolis terms are explicitly treated in the source term.

Higher dimensions

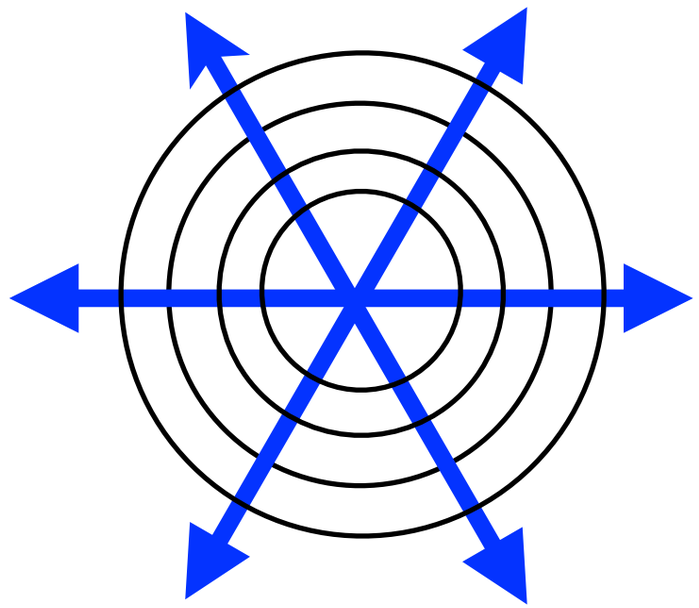
Our aim is to extend the one dimensional wave propagation to higher dimensional logically Cartesian meshes.

Why Cartesian?

- Solution is not dependent on the quality of the mesh
- Algorithms are easier to construct on smooth logically Cartesian meshes and the results are more accurate than on unstructured, non-smooth meshes
- Layout of the Cartesian data maps directly to the computer memory layout, improving runtime performance

Hyperbolicity in two space dimensions

$$q_t + Aq_x + Bq_y = 0$$



$$\xi = \mathbf{n} \cdot \mathbf{x} = n^x x + n^y y$$

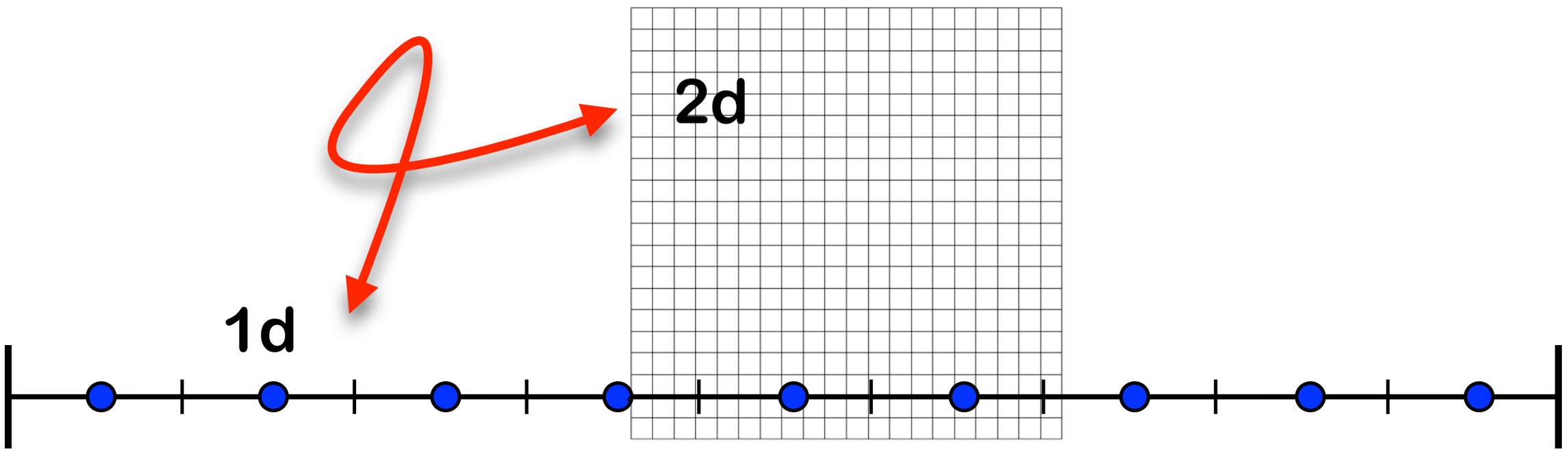
We expect wave-like behavior in any direction $\mathbf{n}$.

The matrix $A \cdot \mathbf{n} = n^x A + n^y B$ must be diagonalizable with real eigenvalues.

Riemann problems in 2d?

- Many hyperbolic problems of interest (Euler equations, shallow water wave equations) are isotropic, or rotationally invariant. The solution to the Riemann problem has the same mathematical structure in any direction.

We can re-use our one-dimensional Riemann solvers in higher dimensions.



Basic strategy

1) Solve 1d Riemann problems in the x-direction (an "x-sweep")
2) Solve 2d Riemann problems in the y-direction (a "y-sweep")
3) Update the solution, either after each sweep, or after both sweeps are done.

Question :

- How do we couple the results of x-sweeps and y-sweeps? Do we store solutions and update all at once? Or do we perform y-sweeps on updated x-sweep update?

Donor cell upwind method

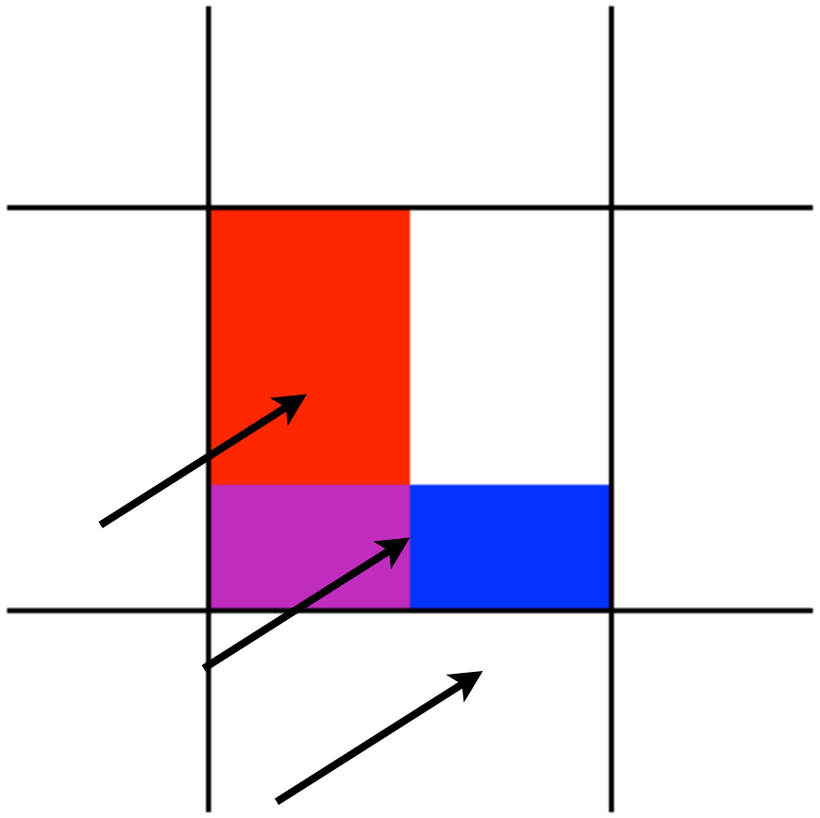
$$q_t + \mathbf{u} \cdot \nabla q = q_t + u\, q_x(x,y) + v\, q_y(x,y) = 0$$

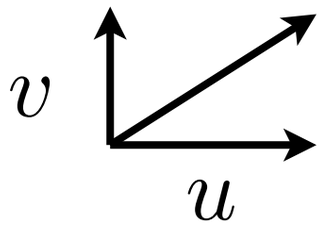
Donor cell upwind method

$$Q_{ij}^{n+1} = Q_{ij}^n - \frac{\Delta t}{\Delta x}\left[u^+(Q_{ij}^n - Q_{i-1,j}^n) + u^-(Q_{i+1,j}^n - Q_{ij}^n)\right]$$
$$- \frac{\Delta t}{\Delta y}\left[v^+(Q_{ij}^n - Q_{i,j-1}^n) + v^-(Q_{i,j+1} - Q_{ij})\right]$$
$$+ \quad (\text{higher order correction terms})$$

- Easy to implement and uses only normal Riemann solves
- Updates are done only after both “sweeps” have been performed.

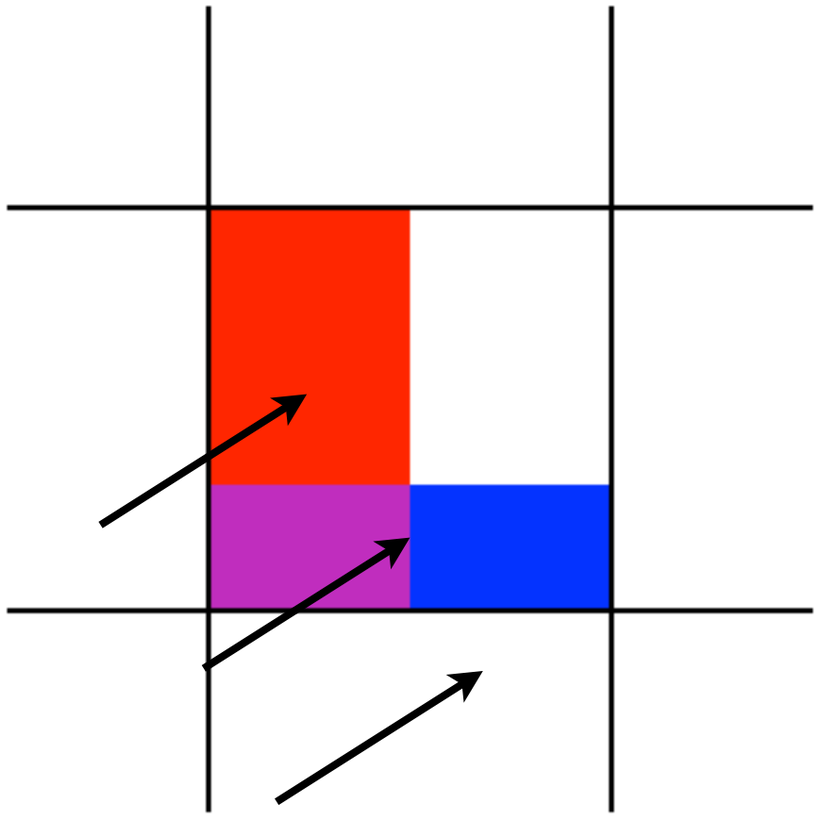
Donor cell upwind method (DCU)

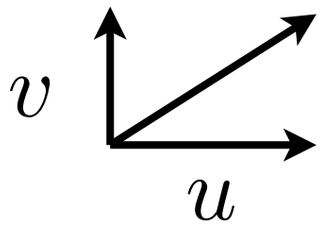




$$Q_{ij}^{n+1} = Q_{ij}^{n} - \frac{\Delta t}{\Delta x}\left[u^{+}(Q_{ij}^{n} - Q_{i-1,j}^{n}) + u^{-}(Q_{i+1,j}^{n} - Q_{ij}^{n})\right]$$
$$- \frac{\Delta t}{\Delta y}\left[v^{+}(Q_{ij}^{n} - Q_{i,j-1}^{n}) + v^{-}(Q_{i,j+1} - Q_{ij})\right]$$

Donor cell upwind method (DCU)





Stability limit :

$$\left|\frac{u\Delta t}{\Delta x}\right| + \left|\frac{v\Delta t}{\Delta y}\right| \leq 1$$

Donor cell upwind

Even with second order correction terms included, the donor cell upwind method still misses the cross derivative terms

$$q_{xy}(x,t)$$

needed to get full second order accuracy in two space dimensions.

Result : Donor cell method is only first order accurate.

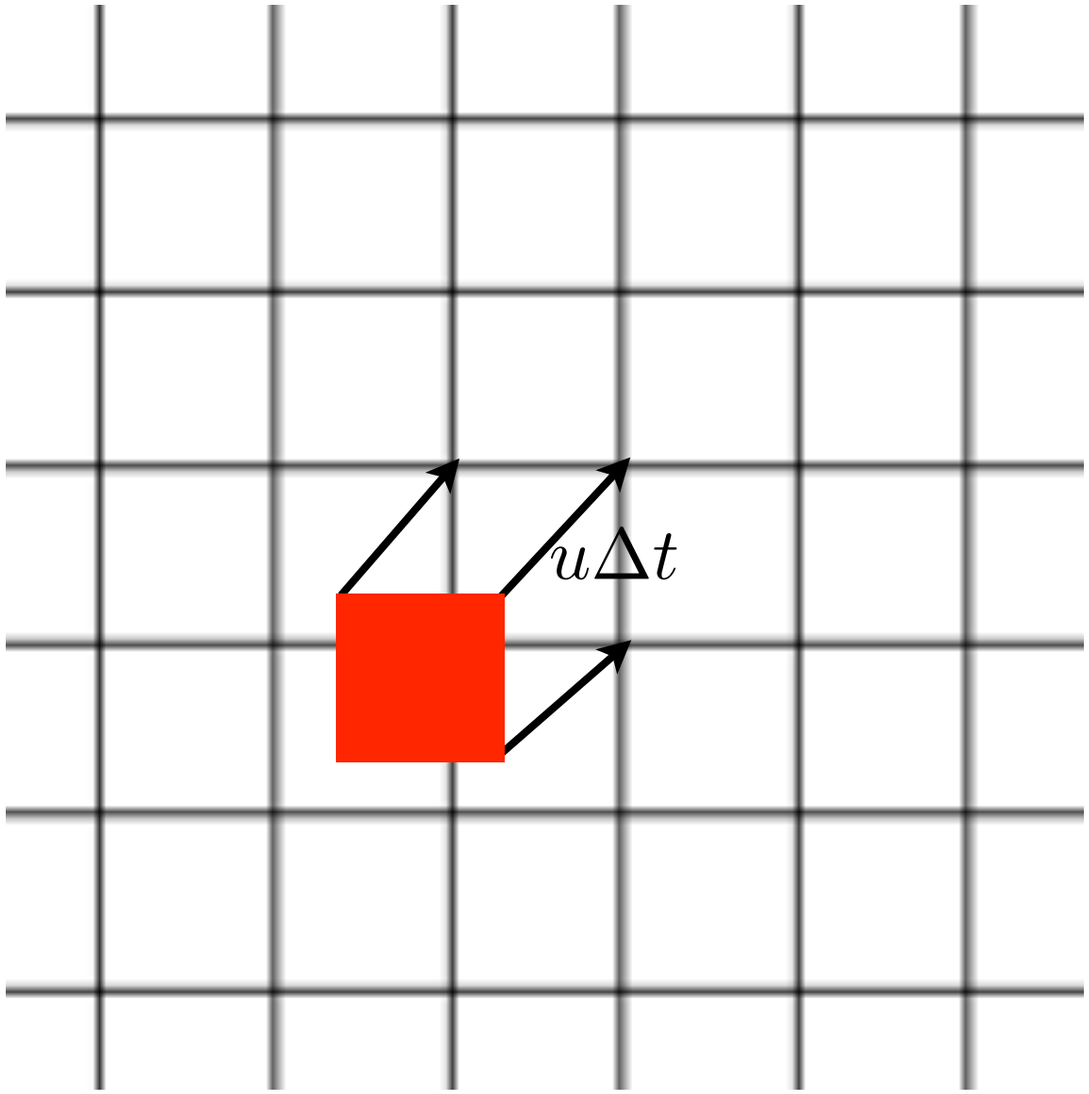
Dimensional splitting

Update the solution after each sweep :

$$
\begin{aligned}
Q^{*}_{ij} &= Q^{n}_{ij} - \frac{u\Delta t}{\Delta x}\left(Q^{n}_{ij} - Q^{n}_{i-1,j}\right) + \dots \\
Q^{n+1}_{ij} &= Q^{*}_{ij} - \frac{v\Delta t}{\Delta y}\left(Q^{*}_{ij} - Q^{*}_{i,j-1}\right) + \dots
\end{aligned}
$$

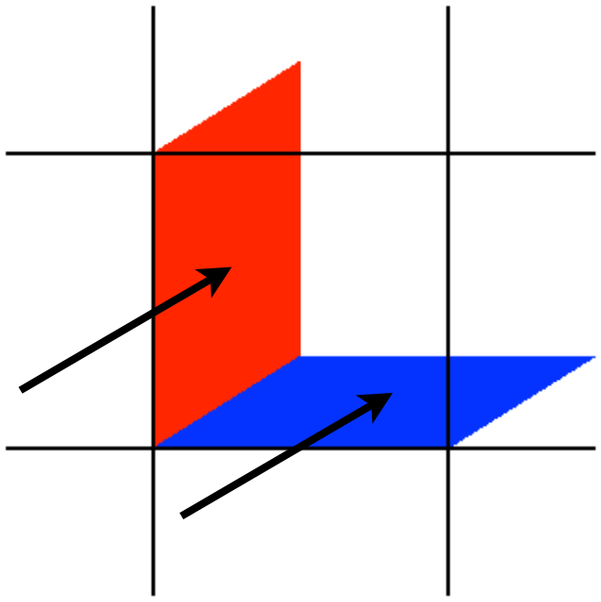
- If second order corrections are included, the dimensional split algorithm is second order,
- No need to explicitly include cross derivatives, since they are treated automatically by the two stage process,
- Uses only 1d normal Riemann solves
- However, dimensionally split algorithms are not always practical

Corner transport upwind method

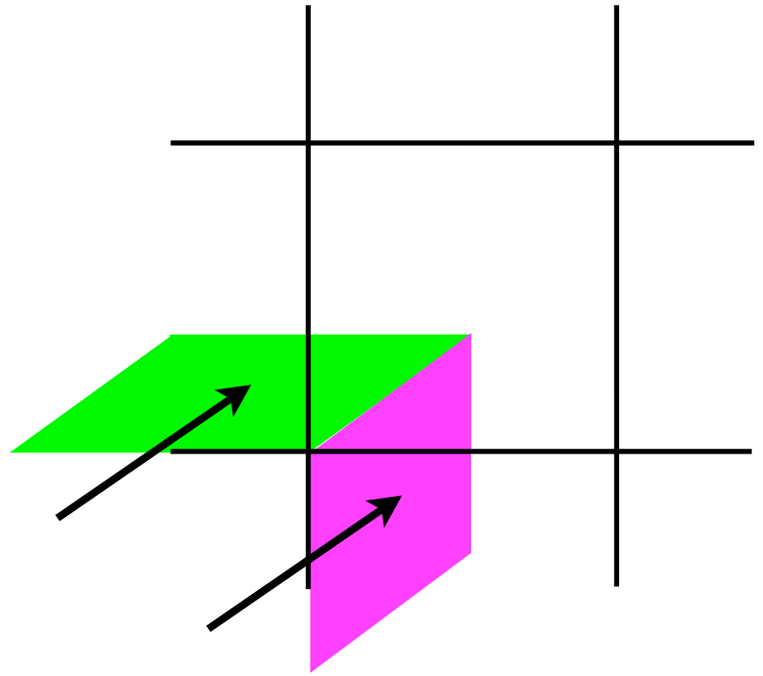


Trace back to get the contents of the cell at the current time.

Corner transport upwind (CTU) method



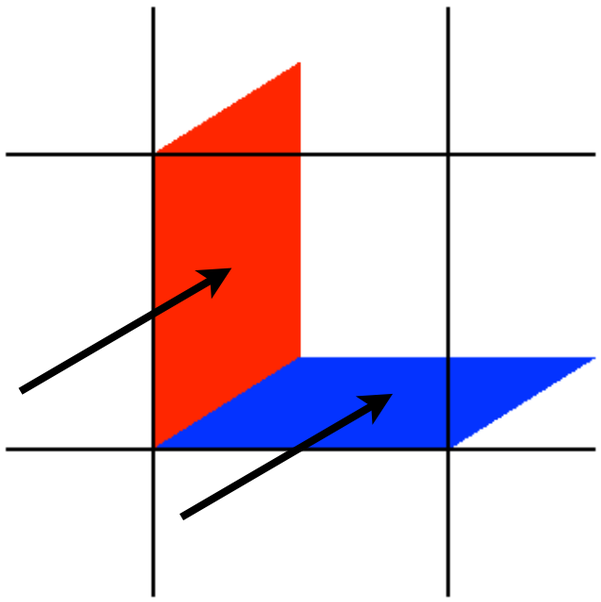
$$u > 0, \quad v > 0$$



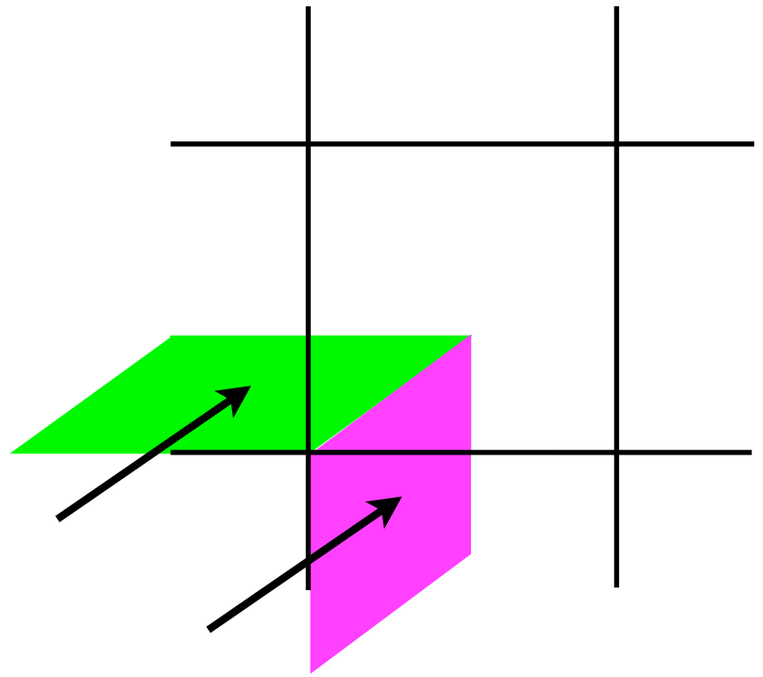
Wave propagation algorithm

$$Q_{ij}^{n+1} = Q_{ij}^n - \frac{u\Delta t}{\Delta x}\left(Q_{ij}^n - Q_{i-1,j}^n\right) - \frac{v\Delta t}{\Delta y}\left(Q_{ij}^n - Q_{i,j-1}^n\right)$$
$$+\frac{1}{2}(\Delta t)^2\left\{\frac{u}{\Delta x}\left[\frac{v}{\Delta y}(Q_{ij}^n - Q_{i,j-1}^n) - \frac{v}{\Delta y}(Q_{i-1,j}^n - Q_{i-1,j-1})\right]\right.$$
$$\left.+\frac{v}{\Delta y}\left[\frac{u}{\Delta x}(Q_{ij}^n - Q_{i-1,j}^n) - \frac{u}{\Delta x}(Q_{i,j-1}^n - Q_{i-1,j-1})\right]\right\}$$

CTU method



$$u > 0, \quad v > 0$$



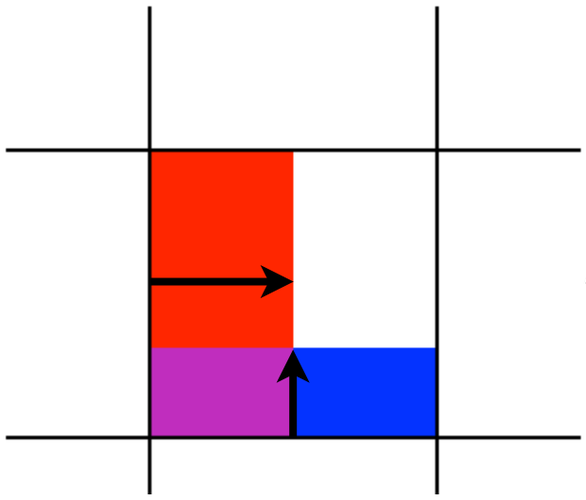
$$\widetilde{\mathcal{F}}_{i-1/2,j} = -\frac{1}{2}\frac{\Delta t}{\Delta y} uv\,(Q_{i-1,j} - Q_{i-1,j-1})$$

$$\widetilde{\mathcal{F}}_{i+1/2,j} = -\frac{1}{2}\frac{\Delta t}{\Delta y} uv\,(Q_{ij} - Q_{i,j-1})$$

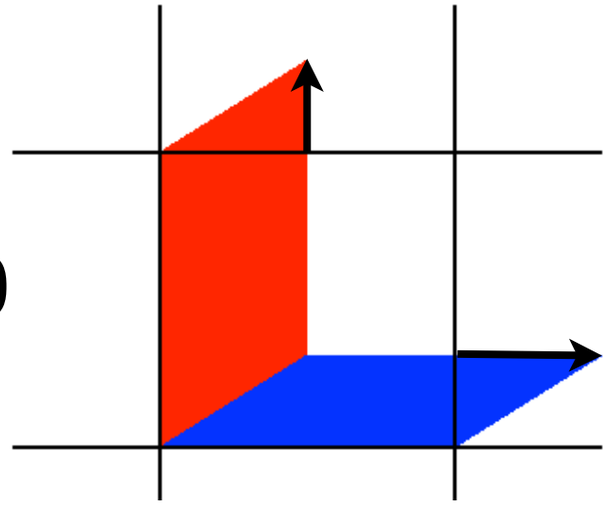
$$\widetilde{\mathcal{G}}_{i,j-1/2} = -\frac{1}{2}\frac{\Delta t}{\Delta x} uv\,(Q_{i,j-1} - Q_{i-1,j-1})$$

$$\widetilde{\mathcal{G}}_{i,j+1/2} = -\frac{1}{2}\frac{\Delta t}{\Delta x} uv\,(Q_{ij} - Q_{i-1,j})$$

Wave propagation algorithm



$u > 0, \quad v > 0$



Normal waves *Transverse waves*

Wave propagation = CTU + second order correction terms

$$Q_{ij}^{n+1} = Q_{ij}^{n} - \frac{\Delta t}{\Delta x}\left[u^{+}\mathcal{W}_{i-1/2,j} + u^{-}\mathcal{W}_{i+1/2,j}\right] - \frac{\Delta t}{\Delta y}\left[v^{+}\mathcal{W}_{i,j-1/2} + v^{-}\mathcal{W}_{i+1/2,j}\right]$$
$$- \frac{\Delta t}{\Delta x}\left[\widetilde{\mathcal{F}}_{i+1/2,j} - \widetilde{\mathcal{F}}_{i-1/2,j}\right] - \frac{\Delta t}{\Delta y}\left[\widetilde{\mathcal{G}}_{i,j+1/2} - \widetilde{\mathcal{G}}_{i,g-1/2}\right]$$

Second order correction terms and transverse propagation

Wave propagation in 2d

- With high resolution correction terms, the CTU method is second order accurate,

- Has better stability properties than the two dimensional version of Lax-Wendroff, since cross derivative terms are taken in the upwind direction rather than averaged.

- Stability constraint given by

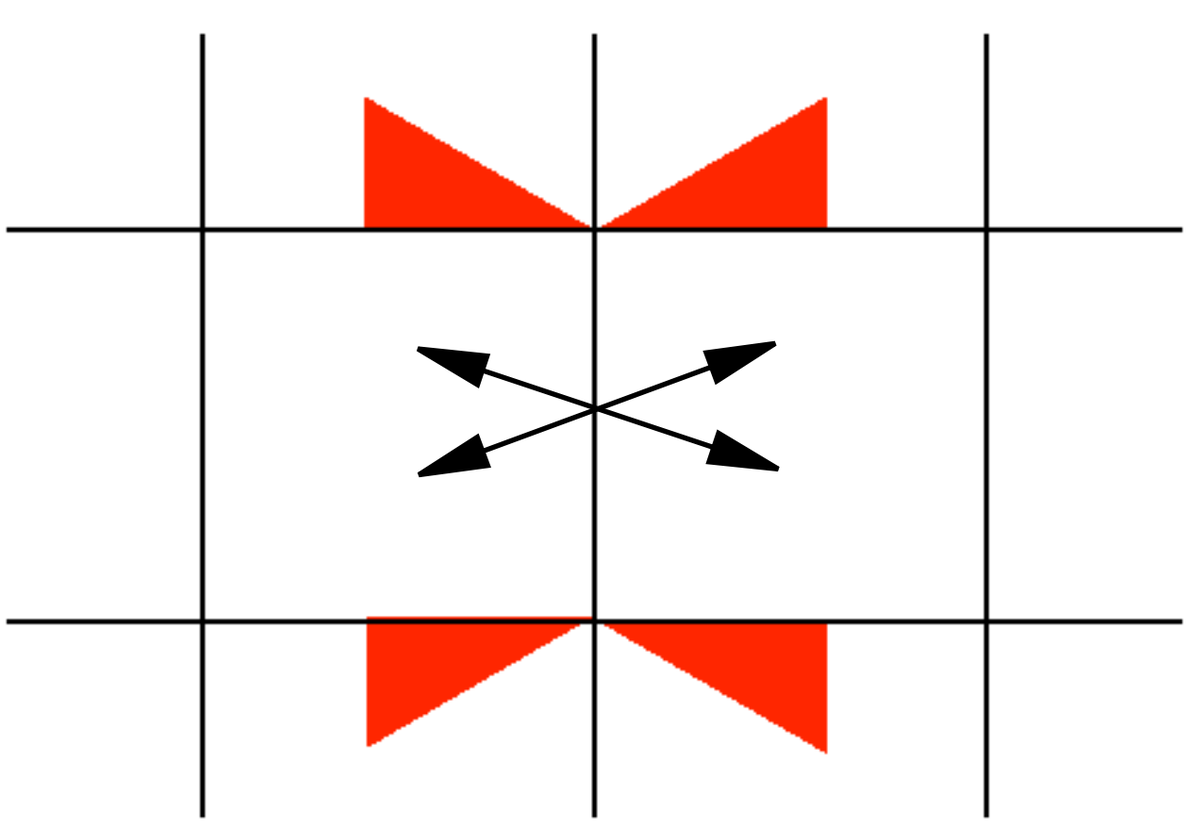
$$\max\left(\frac{|u|\Delta t}{\Delta x}, \frac{|v|\Delta t}{\Delta y}\right) \leq 1$$

- Extends naturally to variable coefficient problems

Transverse propagation

Two dimensional wave propagation algorithm requires two Riemann solvers

- Normal Riemann solver : `rpn2.f`
- Transverse Riemann solver : `rpt2.f`



Transverse Riemann solver

In Clawpack, two parameters control the manner in which the second dimension is handled :

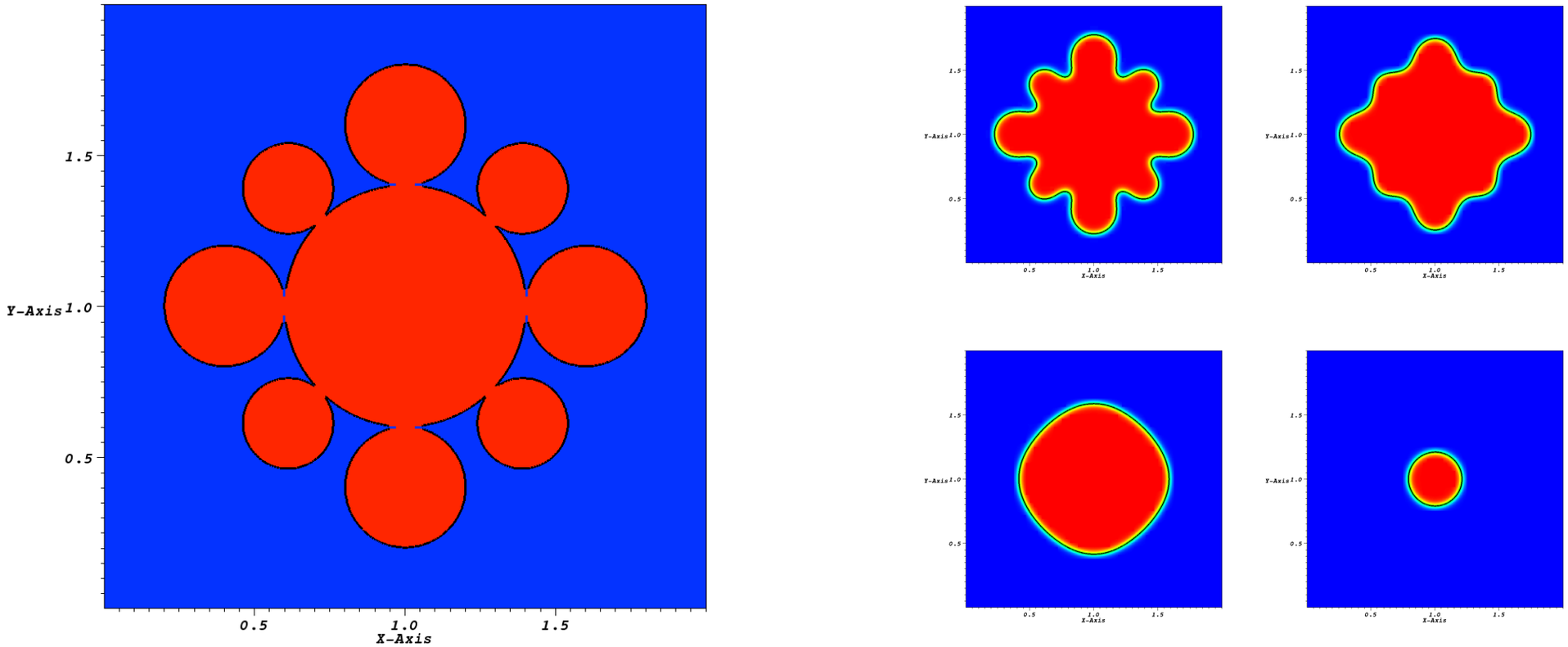
In **`setrun.py`** :

```
# ------------------
# Method to be used:
# ------------------

# Method (1,0)  : Donor Cell Upwind (CFL <= 0.5)
# Method (1,1)  : Corner Transport Upwind  (CFL <= 1.0)
# Method (2,2)  : Full Wave propagation (CFL <= 1.0)
# Method (2,-2) : Dimensionally split algorithm

# Order of accuracy:  1 => Godunov,  2 => Lax-Wendroff plus limiters
clawdata.order = 2

# Transverse order for 2d or 3d (not used in 1d):
clawdata.order_trans = 2
```

Try running at CFL close to 1.0, with `clawdata.order_trans=0`

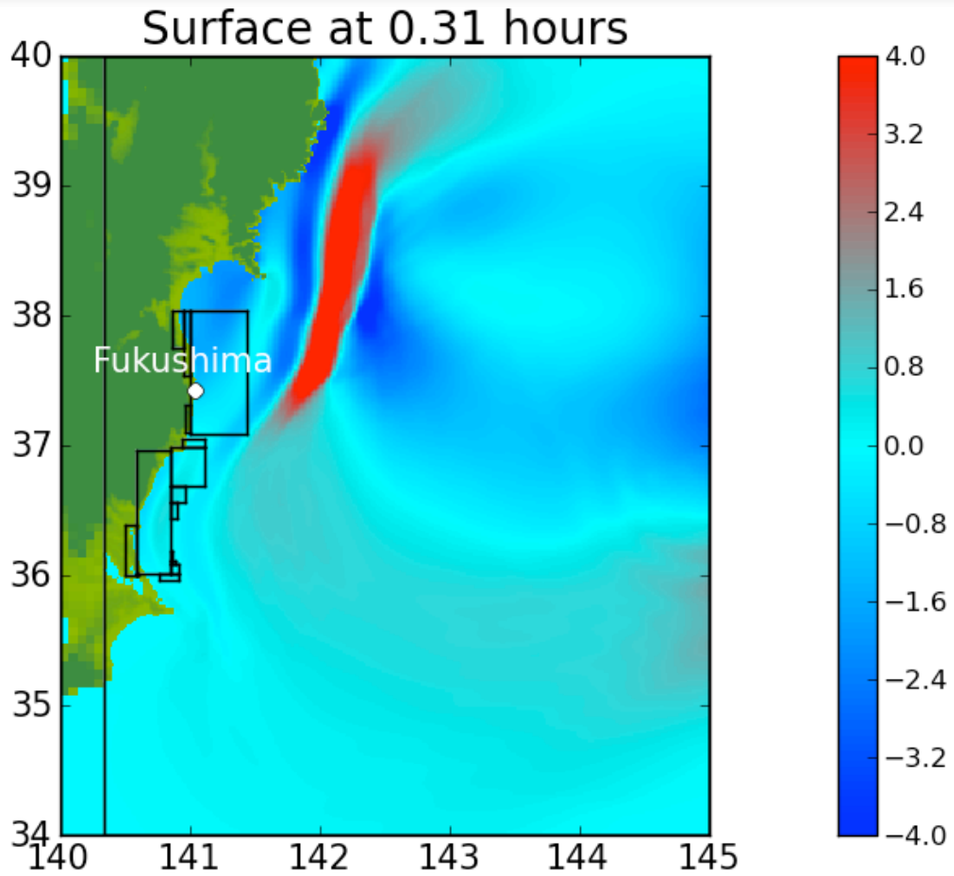
Multi-resolution schemes

When solving PDEs using mesh based methods, it is generally recognized that many problems could benefit enormously from a multi-resolution grid.

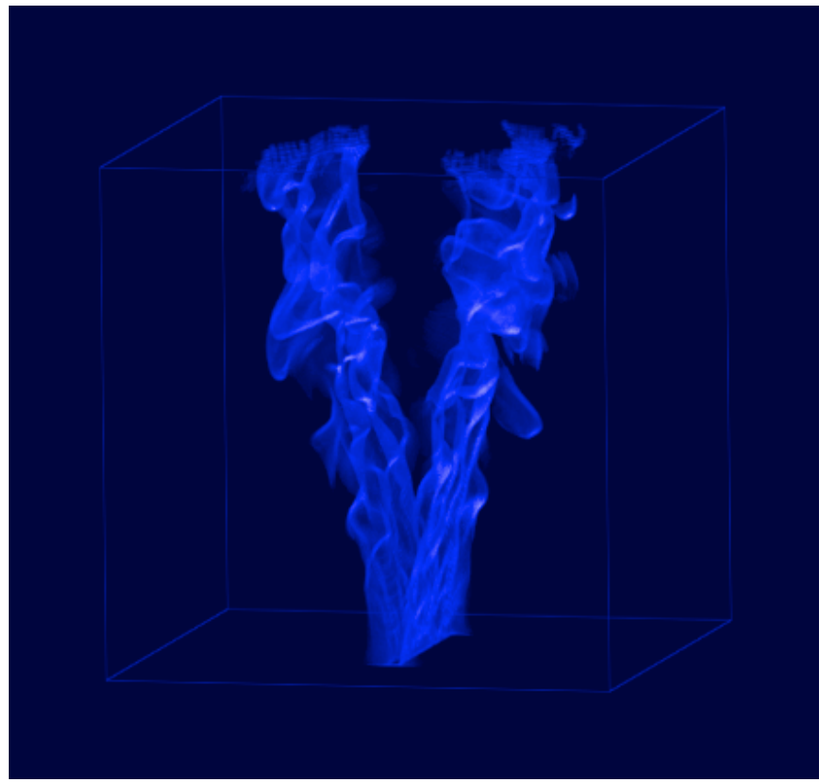


Allen Cahn equation - Flow by mean curvature

Adaptive mesh refinement (AMR)



Tsunami modeling (R. LeVeque, D. George, M. Berger)

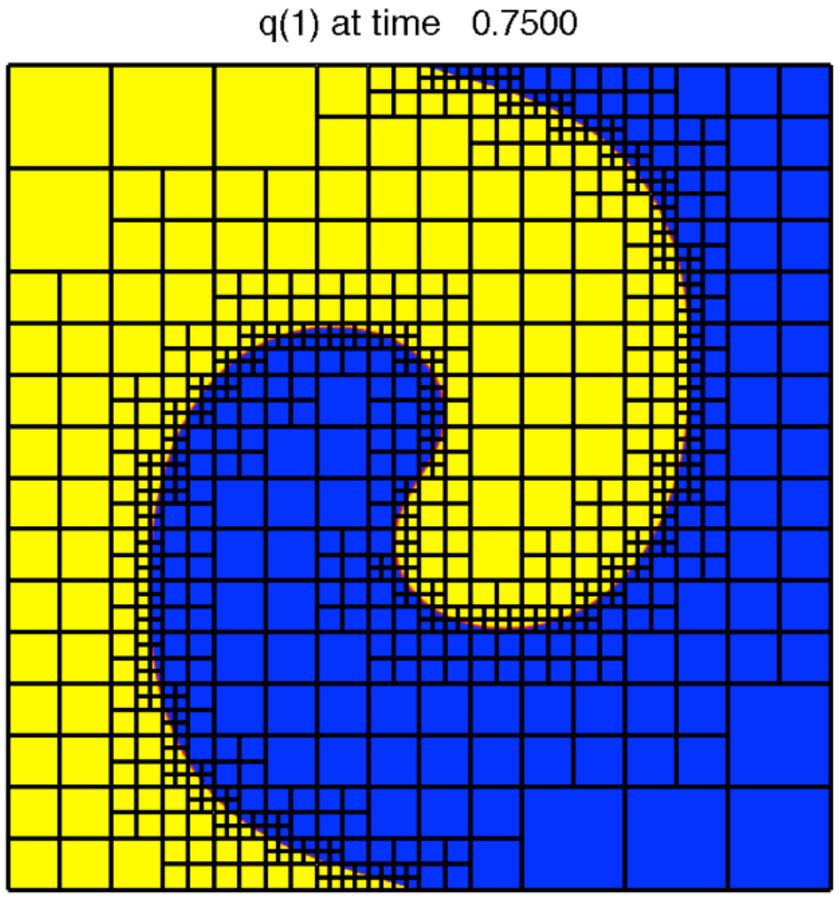


Rod stabilized V-flame (J. B. Bell, Lawrence Berkeley Lab)

- Tracer transport in the atmosphere
- Astrophysics
- Shock capturing for aerodynamic applications
- Regional weather forecasting, hurricanes

Adaptive mesh refinement

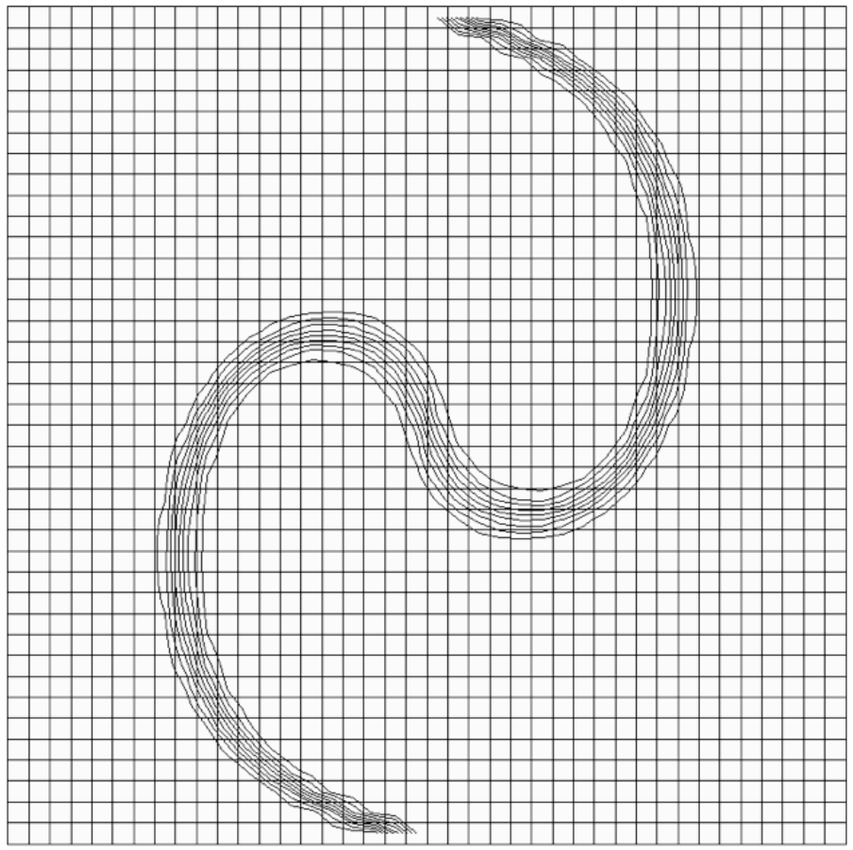
- Block-structured AMR (Berger, Oliger, Colella, ...)
- Tree-based adaptivity (Popinet, Tessyier, ...)
- Finite-element adaptivity includes both h-refinement (increase mesh resolution) and p-refinement (increase order of accuracy), with or without hanging nodes.

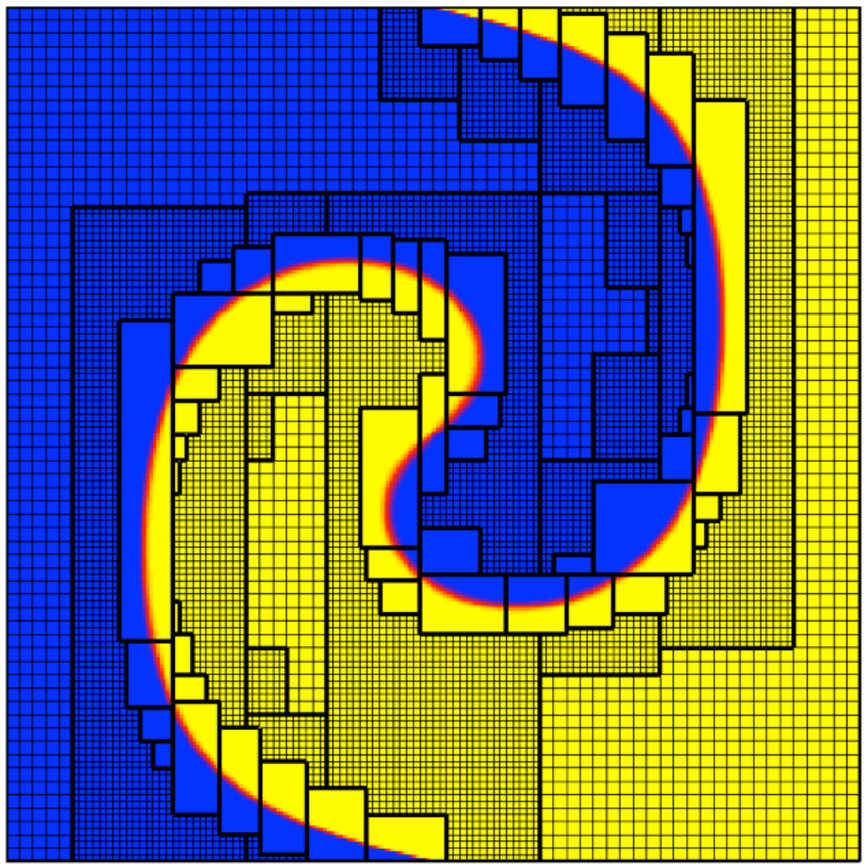


Tree-based adaptivity :

- *Gerris* (S. Popinet, NIWA, NZ),
- *Ramses* (R. Tessyier)
- and many other codes (including several in astrophysics)

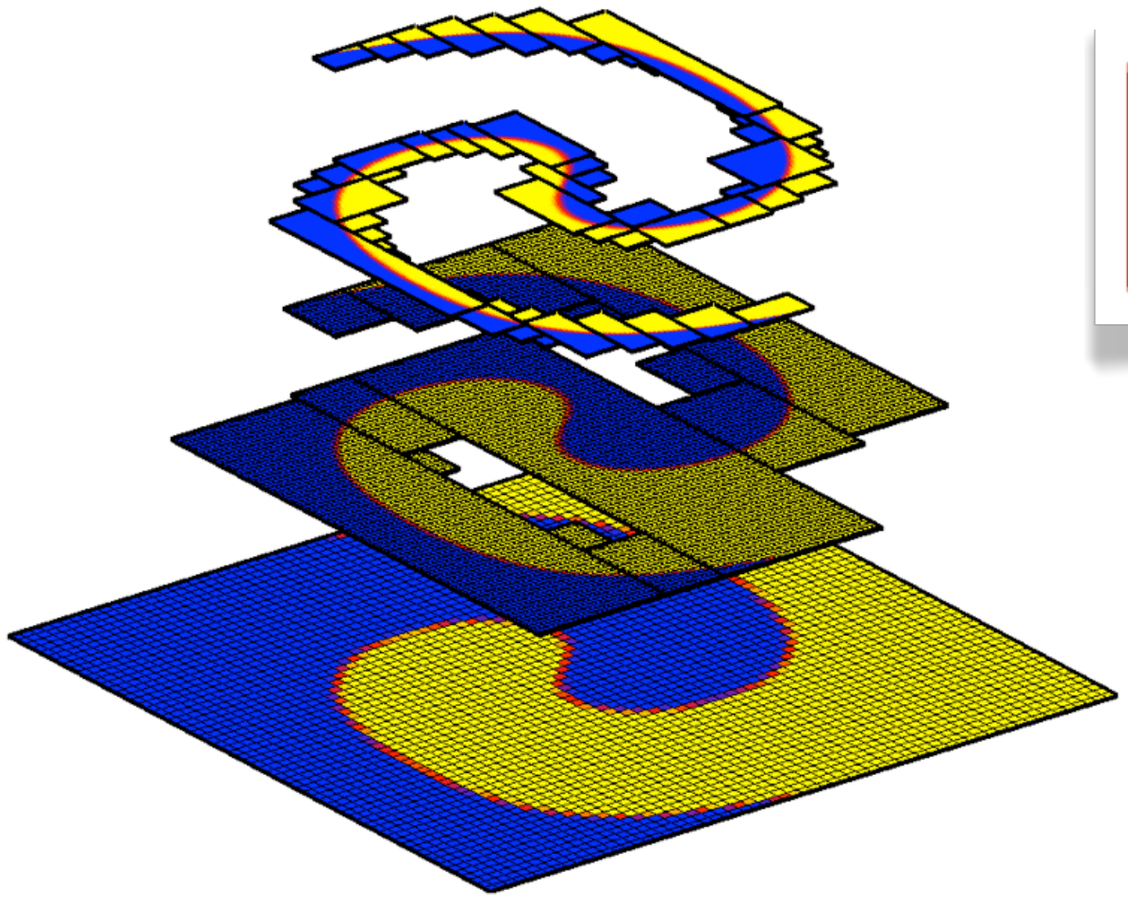
Block structured AMR





- Originally designed to improve shock capturing methods
- Gained widespread use in many application areas
- Colella, Bell, LeVeque, Almgren, Deiterding, and many others have developed methods and solvers

Block structured AMR



GeoClaw grid data is stored in the this layered fashion as well

- Data is stored in hierarchy of logically Cartesian grids,
- Multi-rate time stepping based on mesh size,
- Grids are dynamically (or "adaptively") refined and de-refined to track the solution features of interest.
- Communication between grids is done via a layer of ghost cells.

AMR requirements

- Fine grid boundaries are aligned with coarse grid coordinate lines
- Finer meshes are properly nested into coarser ones
- Assume that we use the same discretization scheme at each level
- Ghost cell values are obtained from coarse grid or neighboring fine grids, if available
- Do not allow grids which overlap multiple levels of refinement
- Averaging fine grid solution to coarse grid; interpolate coarse grid solution to the fine grid.

AMR numerical requirements

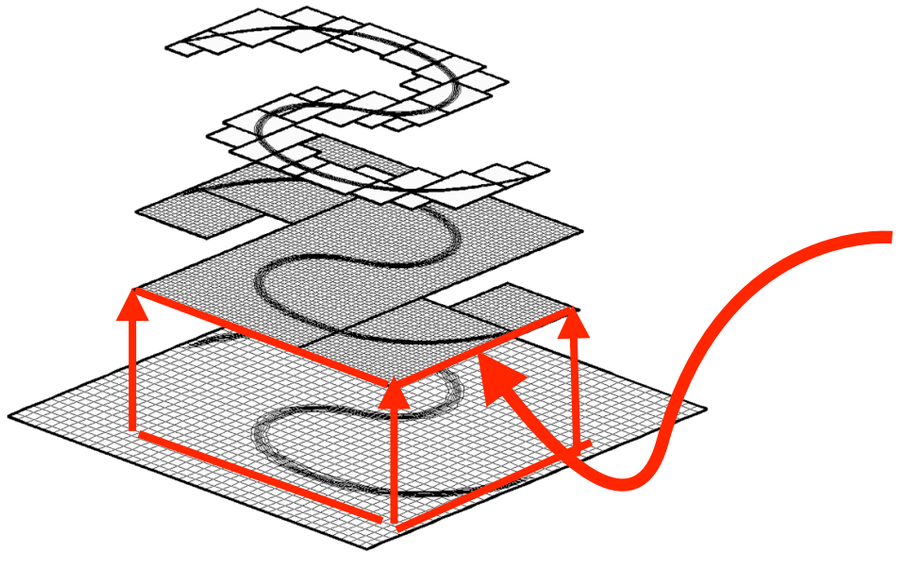
- Single grid Cartesian layout should be used whenever possible
- Avoid use of complicated stencils at coarse/fine grid interfaces
- Numerical solution on grid hierarchy should have the same order of accuracy as the single grid algorithm.
- Conservation should be maintained if PDE is in conservative form
- Overhead in managing multiple grid levels should not impact performance significantly

Multirate time stepping algorithm

A single time step advance, assuming a refinement factor of R.

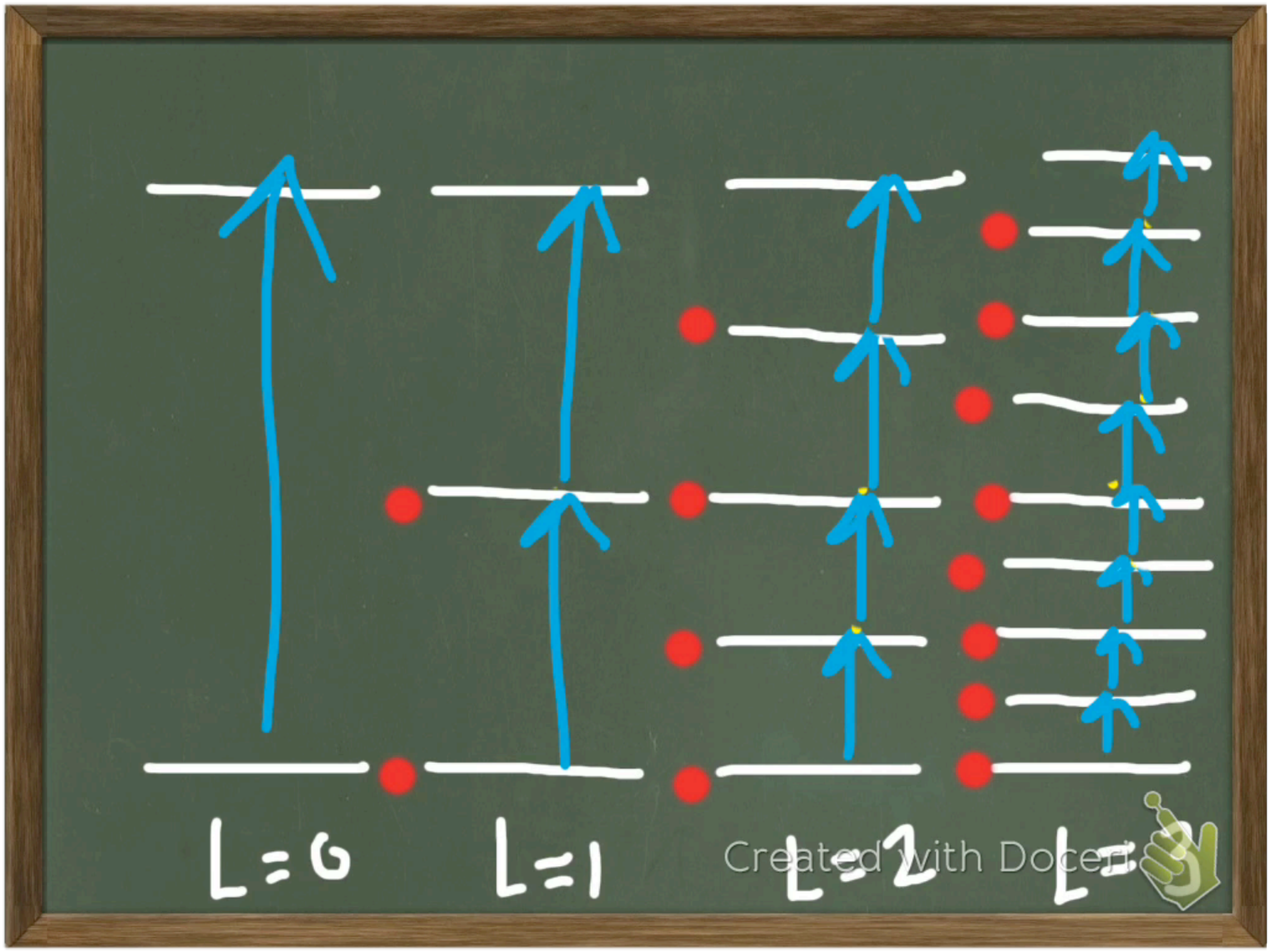
1. Advance at the coarsest level by time step Δt
2. Interpolate coarse grid solution to fine grid ghost cells
3. Advance fine grid R time steps, by a time step $\Delta t/R$
4. Average solution from fine grids to coarse grid,
5. Adjust coarse grid solution to assure flux continuity at the coarse/fine boundaries,
6. Tag cells for refinement and regrid

Grids at the same level exchange ghost cell values directly



Fine grid boundary conditions interpolated in space and time from coarse grid

AMR multi-rate scheme



Patch-based AMR scaling with sub-cycling

$$T_{single}(R,L) = R^{(d+1)L}$$

$$T_{amr}(R,L,\alpha) = 1 + R(\alpha R)^d + R^2((\alpha R)^d)^2 + \ldots + R^L((\alpha R)^d)^L$$

$$S_{amr}(R,L,\alpha) = \frac{T_{single}(R,L)}{T_{amr}(R,L,\alpha)} = \frac{1}{p}\left(\frac{1}{1+\varepsilon+\varepsilon^2+\ldots+\varepsilon^L}\right) < \frac{1}{p}$$

$$\varepsilon = \frac{1}{R^{(d+1)}p^{1/L}}$$

Want this small!

R – Refinement factor (2,4,8,...)
d – Dimension $(2,3)$
L – Number of refined levels (1,2,3...)
α – Fraction of grid refined at each step $(0 < \alpha < 1)$
p – Fraction of domain at finest level $(0 < p = \alpha^{dL} < 1)$

Conservation?

Integrating over entire domain, we have

$$\frac{d}{dt}\int_{x_a}^{x_b} q(x,t)\,dx = -\int_{x_a}^{x_b} (f(q))_x\,dx = f(q(x_a,t)) - f(q(x_b,t))$$

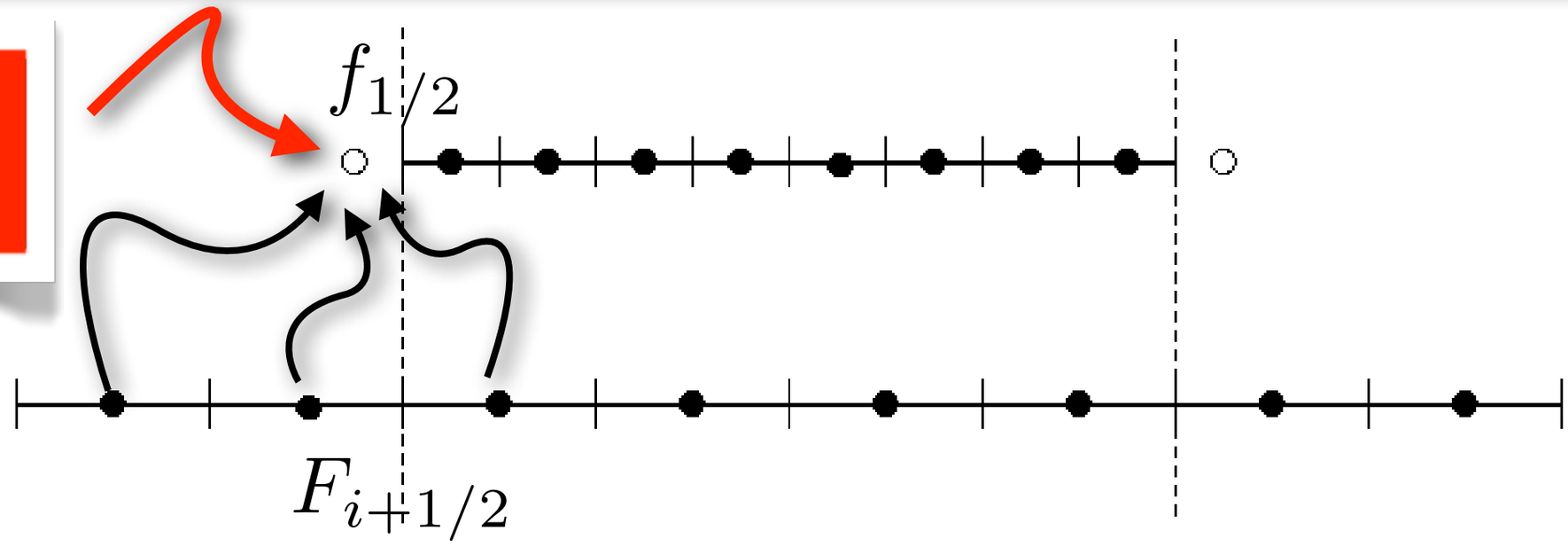
Discrete case

$$\begin{aligned}\sum_{i=1}^{M} Q_i^{n+1} &= \sum_{i=1}^{M} Q_i^n - \frac{\Delta t}{\Delta x}\sum_{i=1}^{M}\left(F_{i+1/2} - F_{i-1/2}\right)\\ &= \sum_{i=1}^{M} Q_i^n - \frac{\Delta t}{\Delta x}\left(F_{M+1/2} - F_{1/2}\right)\end{aligned}$$

Quantities are conserved up to fluxes at domain boundaries.

Conservation at coarse/fine boundaries

interpolated ghost cell value



On the coarse grid, the update is

$$Q_i^{n+1} = Q_i^n - \frac{\Delta t}{\Delta x}(F_{i+1/2}^n - F_{i-1/2}^n)$$

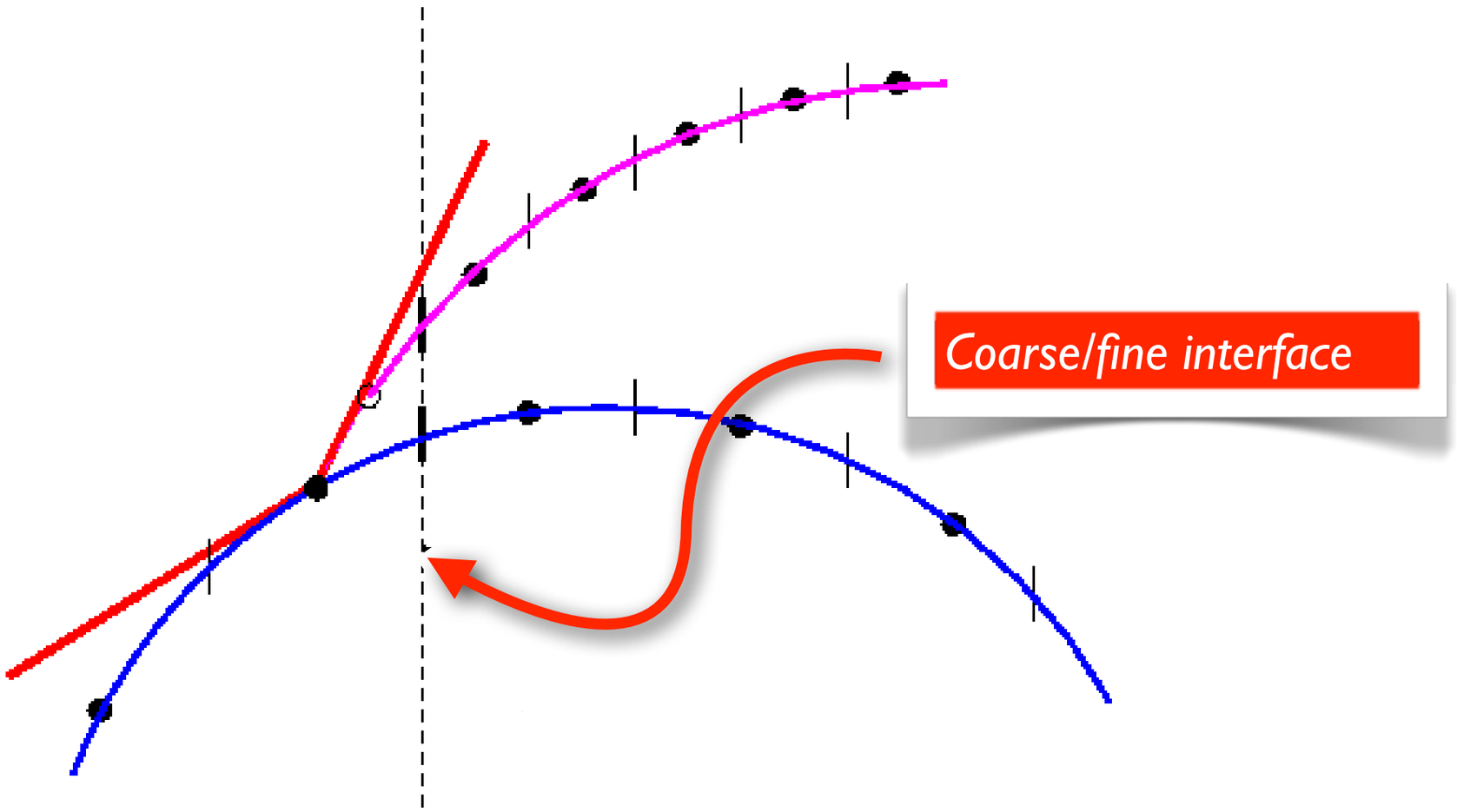
On the fine grid, the update is

$$Q_{i,f}^{n+1} = Q_{i,f}^n - \frac{\Delta t}{\Delta x}(f_{3/2}^n - f_{1/2}^n)$$

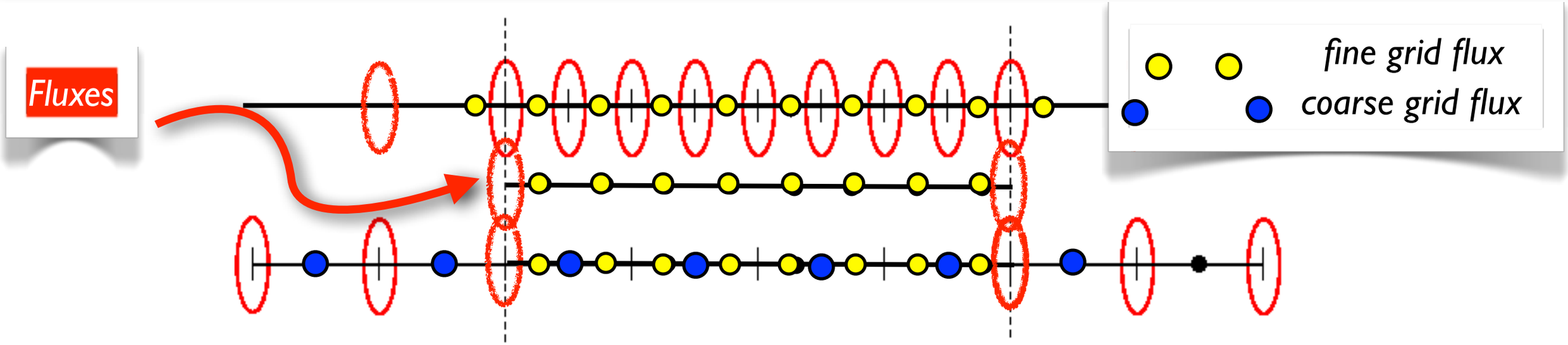
Fluxes do not match at the coarse/fine interface

Conservation

- Not differentiable at the coarse/fine interface
- Not conservative, since two different fluxes are used at the coarse/fine interface



Conservative fix-up



For conservation, we want

$$Q^{n+1} = Q^n - \frac{\Delta t}{\Delta x}\left(f^n_{1/2} - F^n_{i-1/2}\right)$$

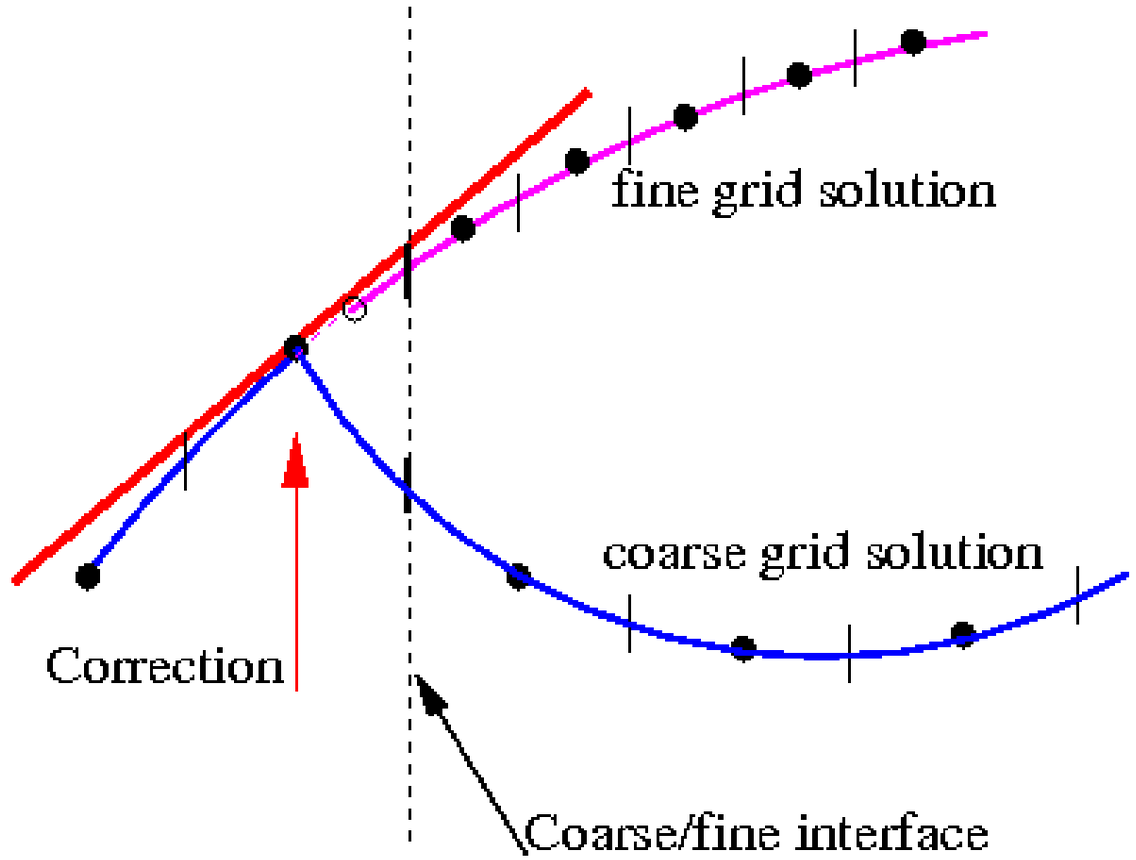
Add correction explicitly

correction term

$$\begin{aligned} Q^{n+1} &= Q^n - \frac{\Delta t}{\Delta x}\left(F^n_{i+1/2} - F^n_{i-1/2}\right) - \frac{\Delta t}{\Delta x}\left(f^n_{1/2} - F^n_{i+1/2}\right) \\ &= Q^n - \frac{\Delta t}{\Delta x}\left(F^n_{i+1/2} - F^n_{i-1/2}\right) - \Delta t \delta^n_i \end{aligned}$$

Corrected solution

- Composite solution is smooth at coarse-fine interface (although coarse grid solution is not smooth at the coasre/fine interface).
- Final solution is conservative, since a single flux is used at the coarse fine interface.



AMR in GeoClaw

```
# max number of refinement levels:
mxnest = 3

clawdata.mxnest = -mxnest   # negative ==> anisotropic refinement in x,y,t

# List of refinement ratios at each level (length at least mxnest-1)
clawdata.inratx = [2,6]
clawdata.inraty = [2,6]

clawdata.inratt = [2,6]

clawdata.tol = -1.0     # negative ==> don't use Richardson estimator

clawdata.kcheck = 3     # how often to regrid (every kcheck steps)

clawdata.ibuff  = 2     # width of buffer zone around flagged points
```

GeoClaw bases refinement criteria on deviations in sea surface height. See the file `'flag2refine_geo.f'` in the GeoClaw source directory.

Adaptive mesh software

- General purpose (freely available) block-structured codes
  - **AMRClaw** (University of Washington/NYU. Basis for GeoClaw)
  - PARAMESH (NASA/Goddard)
  - SAMRAI (Lawrence Livermore National Lab)
  - BoxLib (Lawrence Berkeley Lab)
  - Chombo (Lawrence Berkeley Lab)
- All are large frameworks, with many developers
- Mostly C++ and Fortran libraries (no GUIs) that started life as research codes.

Block structured AMR on quad and octrees

Use the highly scalable, parallel quad/octree library **p4est** (C. Burstedde) to do the grid management

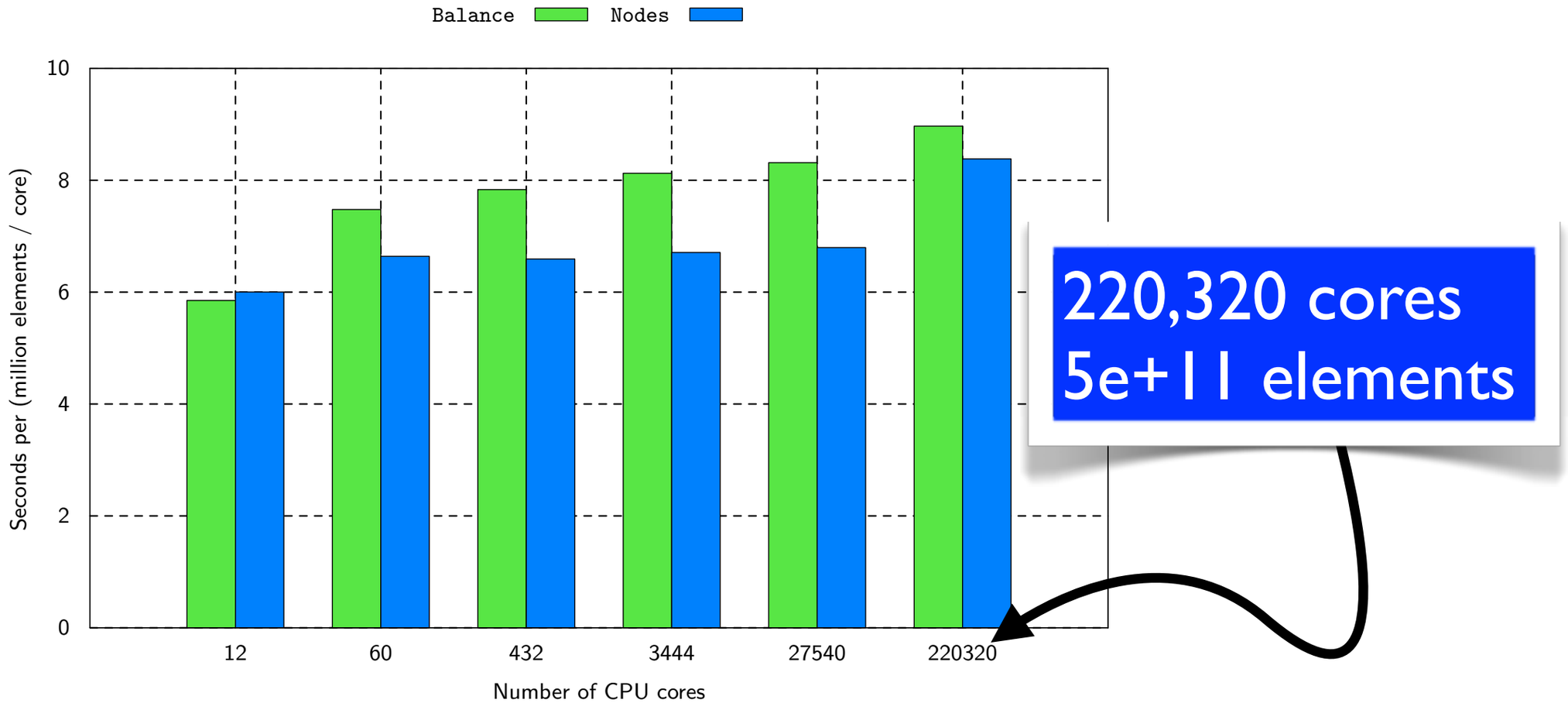
- Store fixed sized non-overlapping grids as leaves in a tree
- Refinement patches and parallel "units" are the same

Advantages

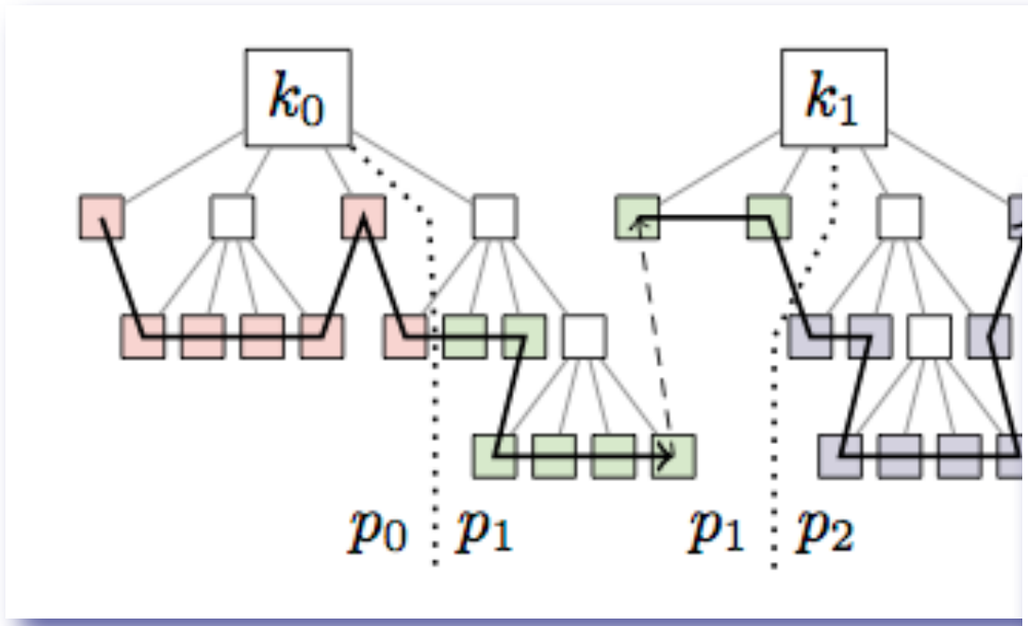
- Tree-based grid layout makes communication between grids much simpler, especially in 3d.
- Construction of refined patches is trivial,
- Clear separation between the grid management (including neighbor communication) and the numerics

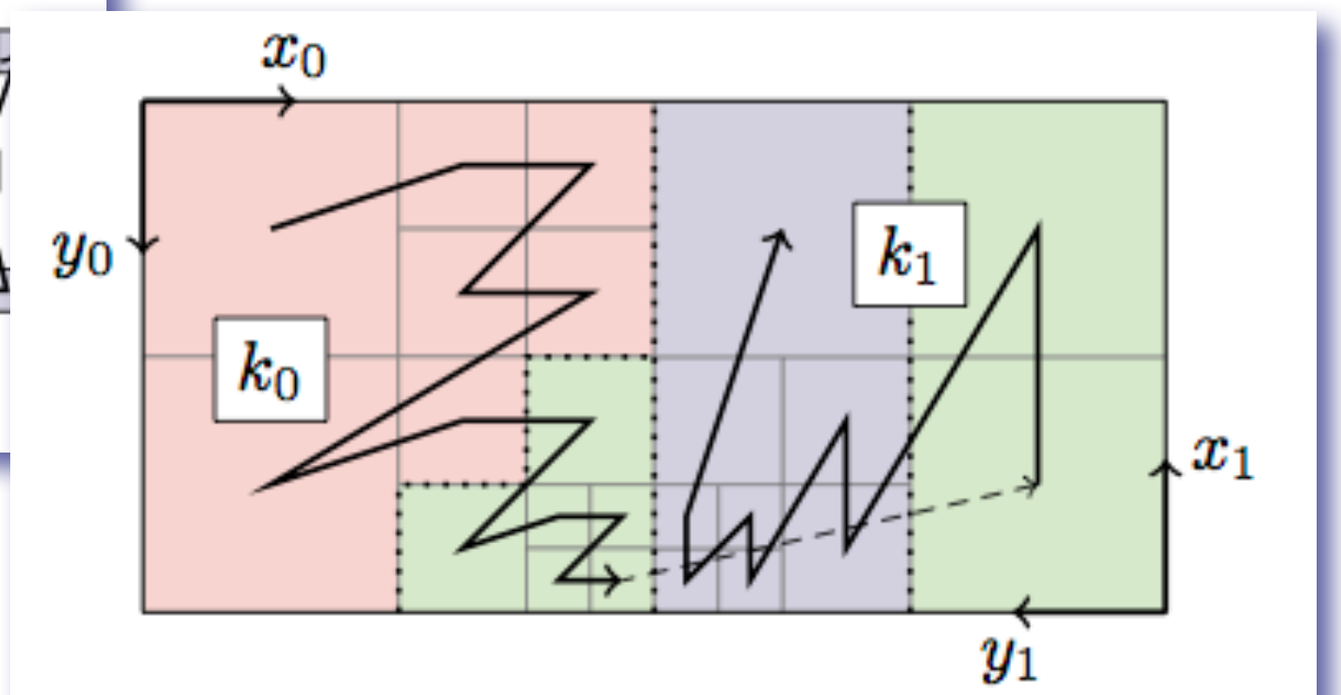
p4est

- Developed by Carsten Burstedde (Univ. of Bonn), with Wilcox, Ghattas and others
- Parallel, multiblock code for managing a forest of adaptive quad- or octrees.
- Highly scalable on realistic applications of interest



p4est

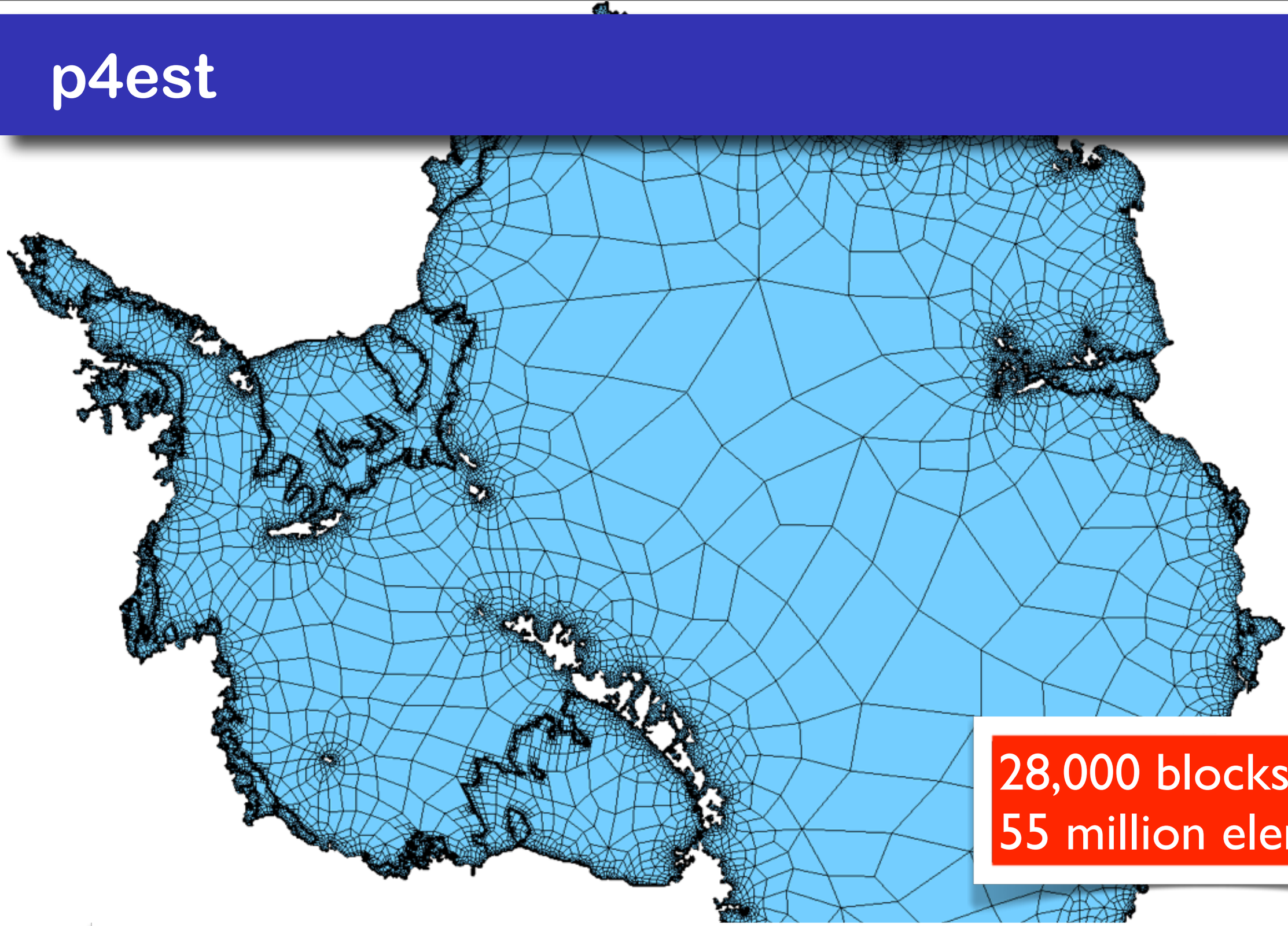




High scalability is achieved while preserving data locality by using space-filling curves.

Carsten Burstedde, Lucas C. Wilcox, and Omar Ghattas, "p4est: Scalable Algorithms for Parallel Adaptive Mesh Refinement on Forests of Octrees", SISC (2011)

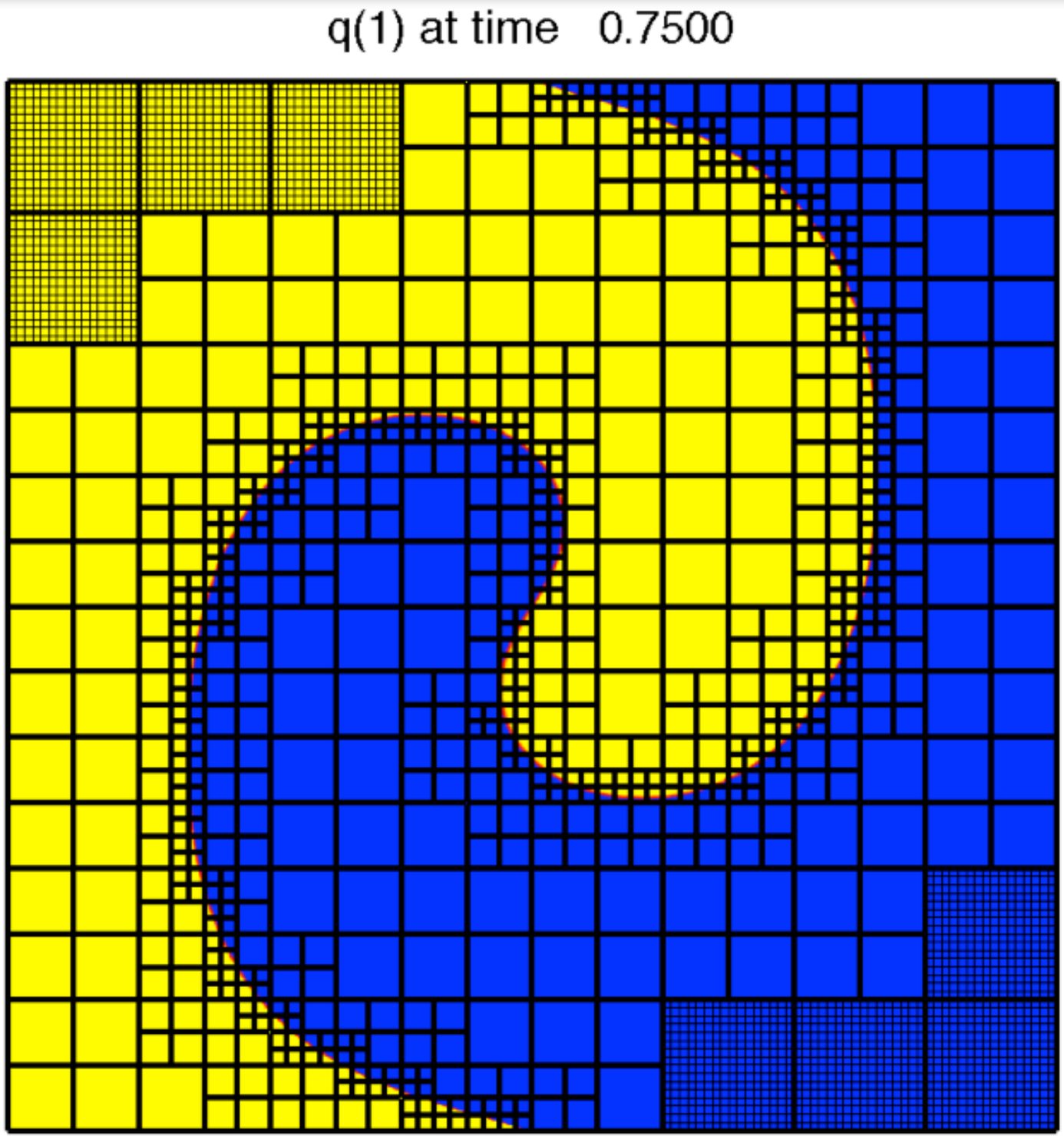
p4est



28,000 blocks
55 million elements

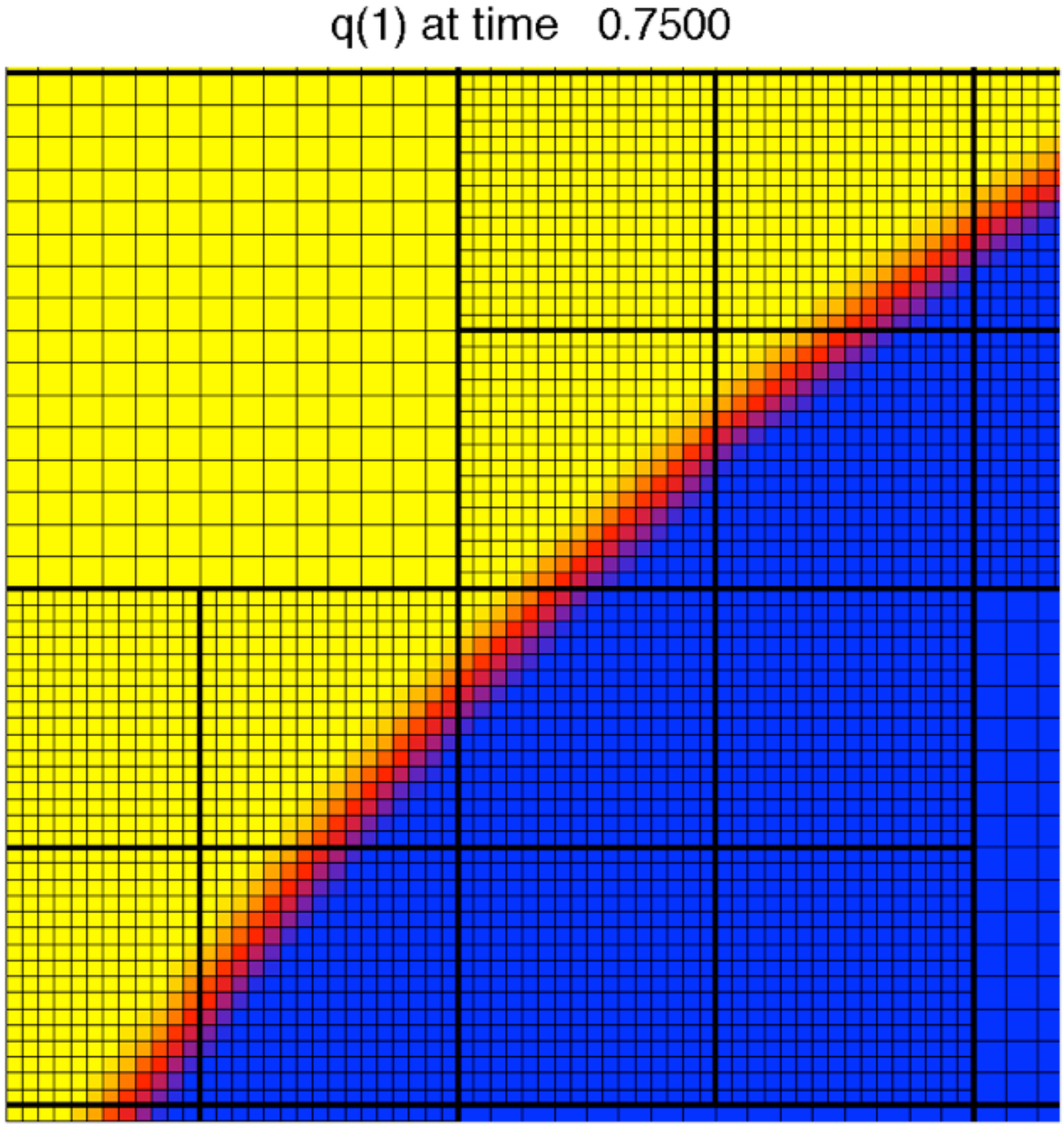
Antarctic ice sheet modeling (Tobin Isaac, C Burstedde)

AMR on quadtrees

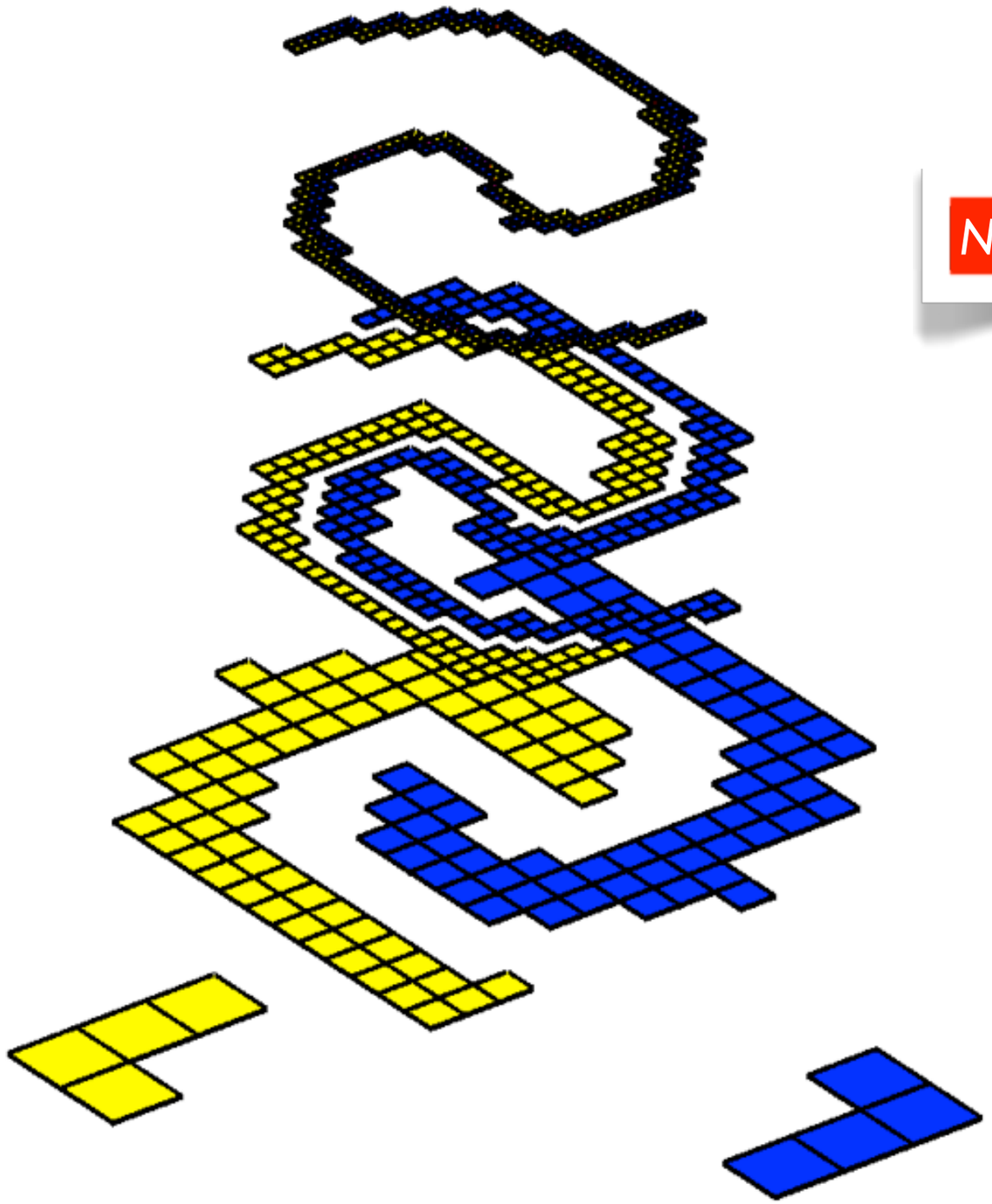


Each leaf is a fixed size grid

AMR on quadtrees

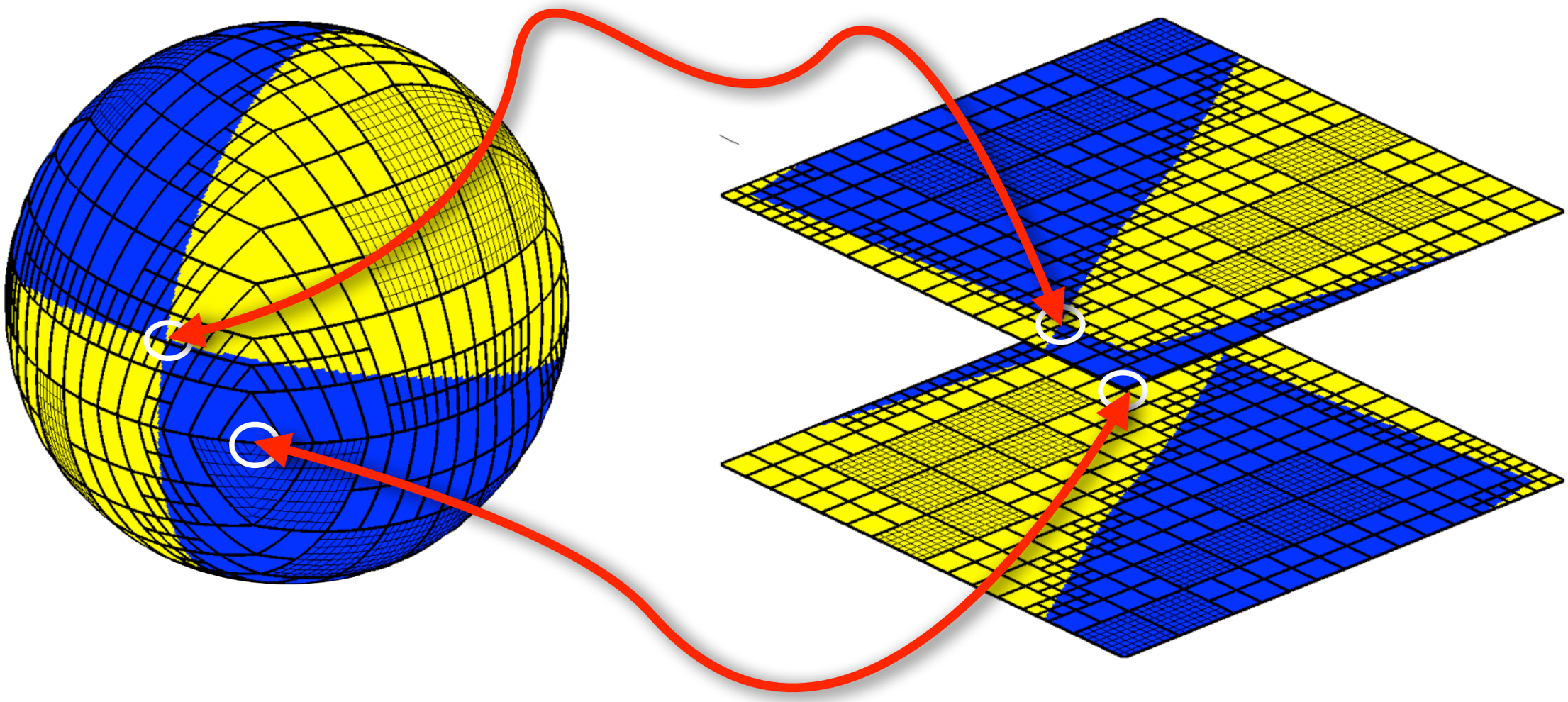


AMR on quadtrees



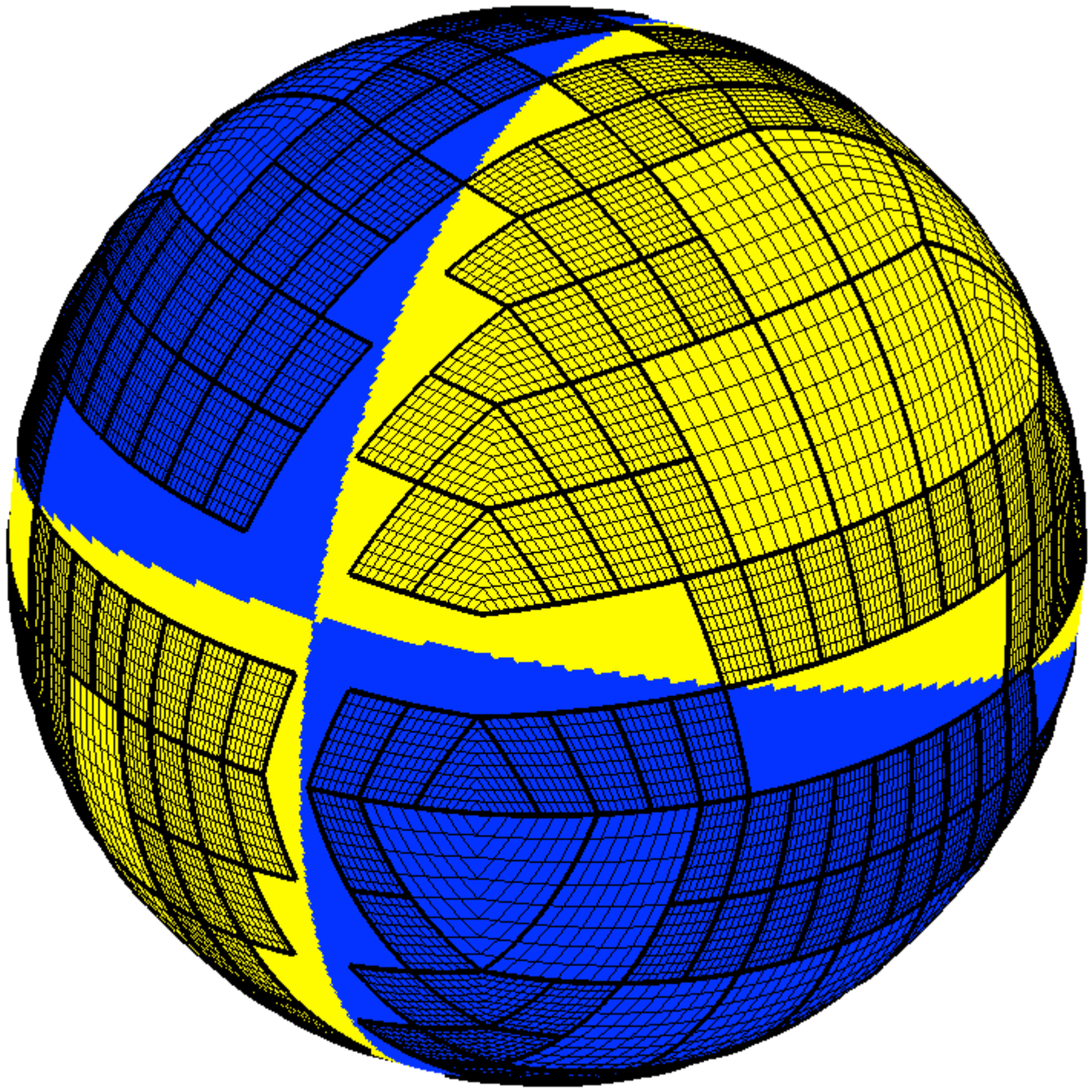
No overlapping grids

Scalar transport on the sphere



Scalar transport on the sphere

Scalar transport on the sphere

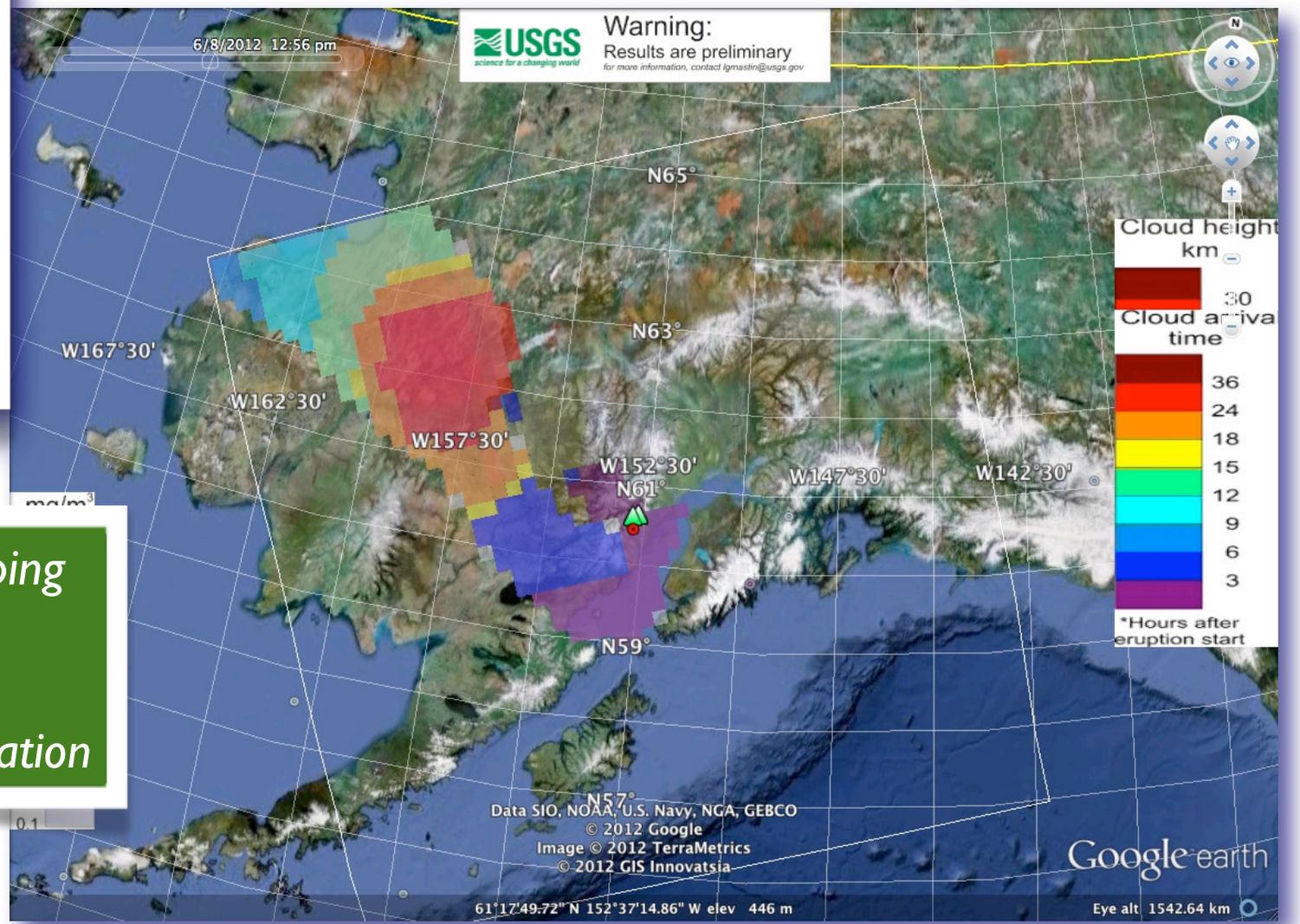


Forestclaw

- Use p4est to manage the parallel multiblock of quad- or octree whose leaves are non-overlapping, fixed size grids.
- p4est guides parallel transfers,
- Support for multiblock inherited from p4est
- Flat data structure - the space filling curve.
- Essentially same block structured algorithms can be implemented,
- Wave propagation algorithms in Clawpack, and other finite volume solvers
- Support for multirate method-of-line solvers
- Quad/Octree is scalable to thousands of processors

See www.forestclaw.org

Ash3d



- *Split horizontal, vertical time stepping*
- *Fully conservative,*
- *Eulerian, finite volume*
- *Algorithms based on wave propagation*

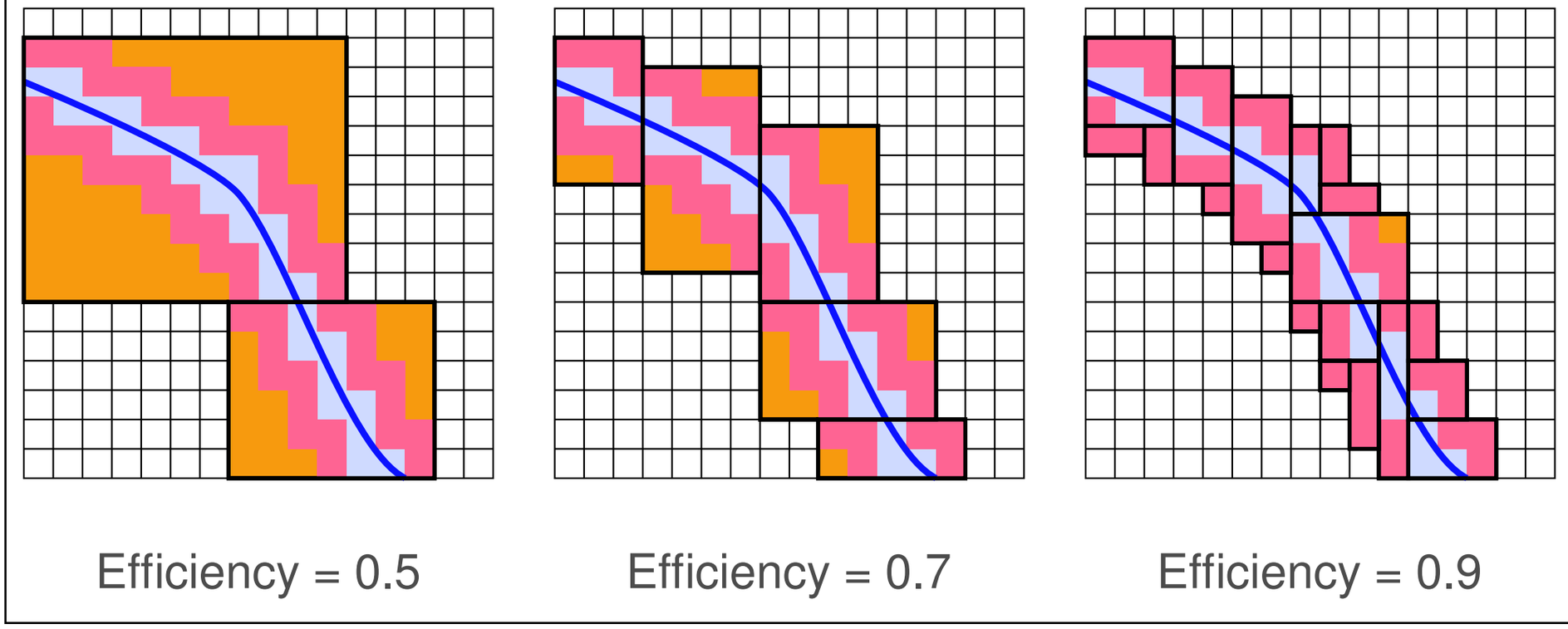
Ash3d :A finite-volume, conservative numerical model for ash transport and tephra deposition, Schwaiger, Denlinger, Mastin, JGR (2012)

Thanks!

Thanks to Lorena and all the PASI organizers for making this workshop such a success!

Clustering algorithm (Berger and Rigoustos)

- Fill data at level 0
- Estimate where refinement is needed and buffer
- Group cells into patches according to a prescribed "grid efficiency" and refine $\Rightarrow B_1, ..., B_n$ (Berger and Rigoustos, 1991)
- Repeat for next level and adjust for proper nesting



John Bell, Short course on "Block structured adaptive mesh refinement", (Cambridge, UK, 2004).