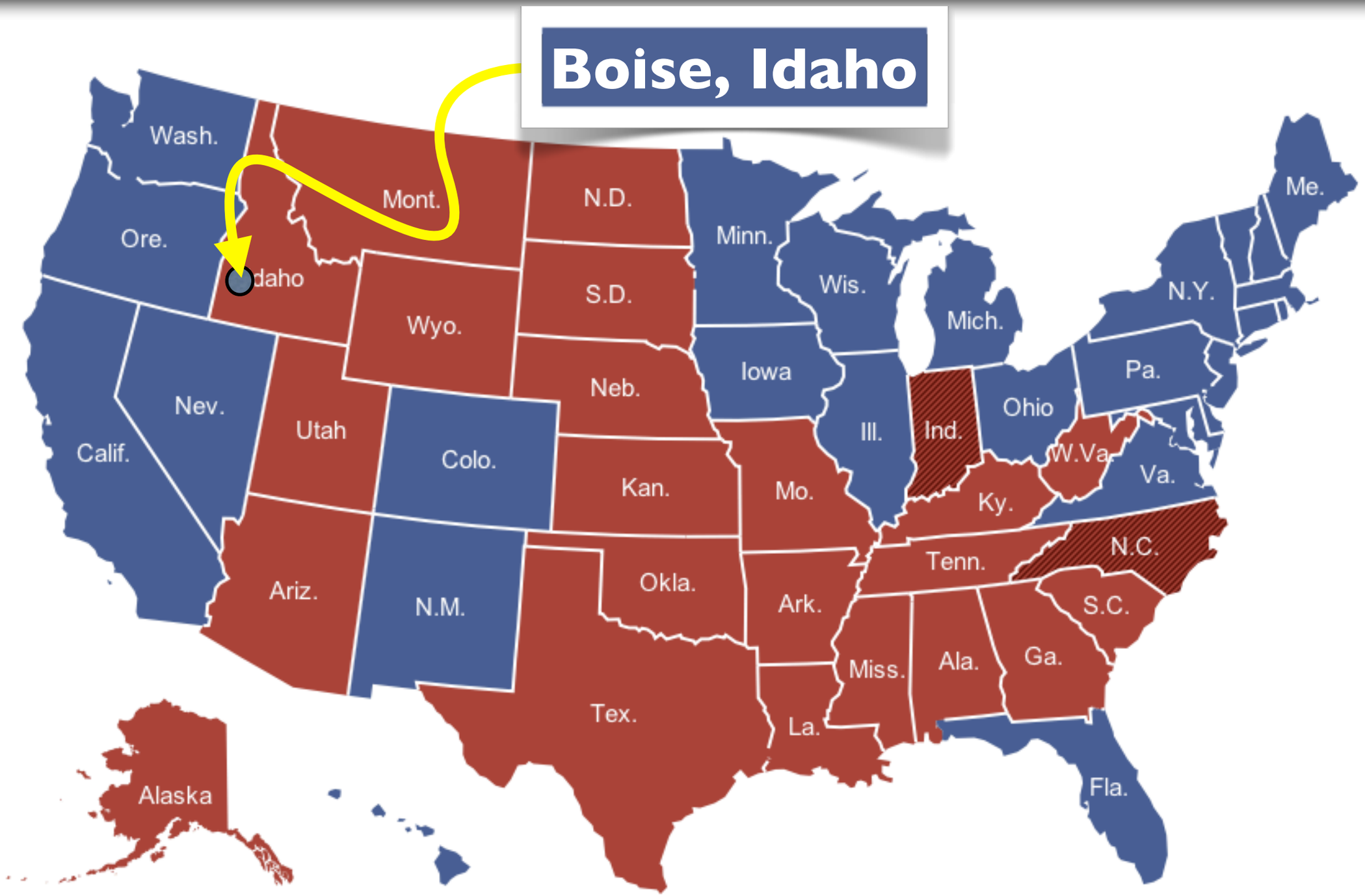
A locally adaptive Cartesian, finite-volume framework for solving PDEs on surfaces

Donna Calhoun (Boise State University)

with Carsten Burstedde (University of Bonn, Germany), Marsha Berger (NYU), Randall J. LeVeque (Univ. of Washington), Christiane Helzel (Ruhr-University Bochum)

Mathematical Institute
University of Oxford
Nov 29, 2012

Where is Boise?



Boise : “boisé = “wooded”

Outline

- Finite volume schemes on logically Cartesian grids surface meshes
- Adaptive mesh refinement (AMR)
- Existing numerical software for AMR
- One example : ForestClaw (on going project)
- Demonstrations

Finite volume schemes

Assume a conservation law of the form

$$q_t + f(q)_x = 0$$

Define cell averages over the interval $C_i = [x_{i-1/2}, x_{i+1/2}]$

$$Q_i^n = \frac{1}{\Delta x} \int_{C_i} q(x, t_n)\, dx$$

How does the average evolve?

$$\begin{aligned} \frac{d}{dt} \int_{C_i} q(x,t)\, dx &= -\int_{C_i} \frac{d}{dx} f(q(x,t))\, dx \\ &= f(q(x_{i-1/2}, t)) - f(q(x_{i+1/2}, t)) \end{aligned}$$

Finite volume schemes

Evolution of the cell average value :

$$\frac{d}{dt}\int_{C_i} q(x,t)\,dx = f(q(x_{i-1/2},t)) - f(q(x_{i+1/2},t))$$

discretize in time

$$\int_{C_i} q(x,t_{n+1})\,dx = \int_{C_i} q(x,t_n)\,dx + \int_{t_n}^{t_{n+1}} \left[f(q(x_{i-1/2},t)) - f(q(x_{i+1/2},t))\right]\,dt$$

$$\boxed{\approx \Delta x\, Q_i^{n+1}}$$

Finite volume schemes

Using numerical approximations to the average values of the edge flux over the time interval, we obtain the update formula (in flux form) :

$$Q_i^{n+1} = Q_i^n - \frac{\Delta t}{\Delta x}\left[F_{i+1/2}^n - F_{i-1/2}^n\right]$$

Numerical fluxes

Written as

$$\frac{Q_i^{n+1} - Q_i^n}{\Delta t} + \frac{F_{i+1/2} - F_{i-1/2}}{\Delta x} = 0$$

this form resembles the conservation law :

$$q_t + f(q)_x = 0$$

What about 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

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

Hyperbolic problems

$$q_t + f(q)_x = 0, \qquad q \in \mathcal{R}^m$$

A problem is *hyperbolic* if the flux Jacobian has real eigenvalues and a complete set of eigenvectors.

$$f'(q) = R\Lambda R^{-1}$$

$$R = [r^1, r^2, \ldots r^m] \qquad \Lambda = \mathrm{diag}(\lambda^1, \lambda^2, \ldots \lambda^m)$$

eigenvectors eigenvalues

depend on q

For many physical systems (shallow water wave equations, gas dynamics, acoustics, elasticity, ...) , this eigen-decomposition is known analytically and is the basis for many numerical methods.

Numerical fluxes for a hyperbolic PDE

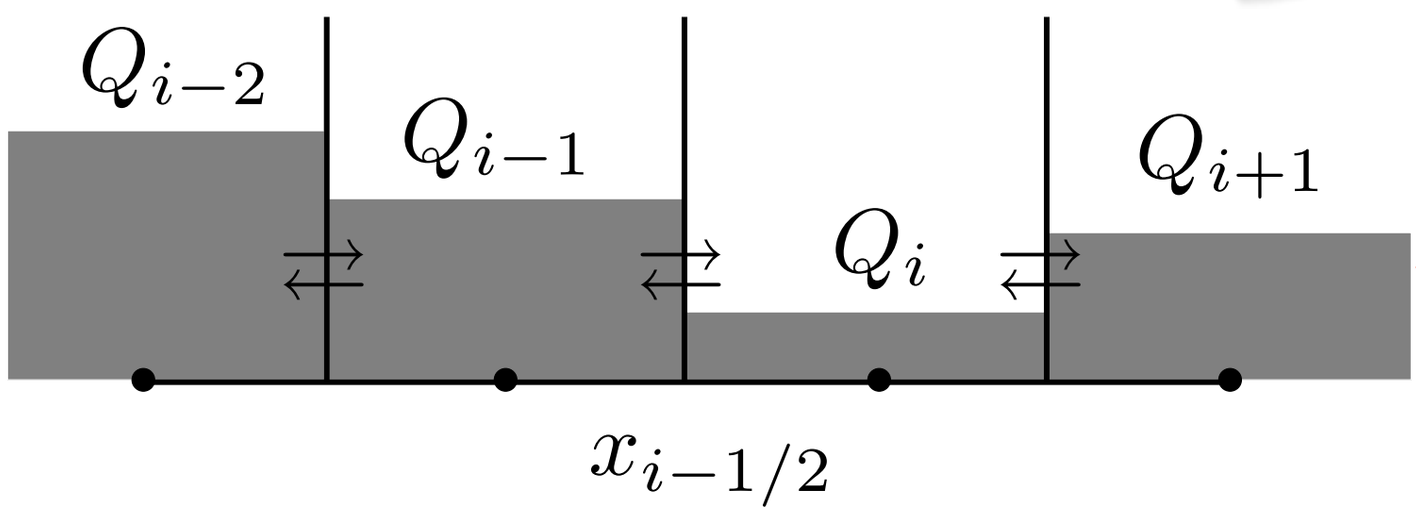
We assume an explicit time stepping scheme

$$F^n_{i-1/2} \approx \int_{t_n}^{t_{n+1}} f(q(x_{i-1/2}, t))\, dt$$

so that we try to find formulas for the flux of the form

$$F^n_{i-1/2} = \mathcal{F}(Q^n_i, Q^n_{i-1})$$

Piecewise constant reconstruction of a solution average in each cell



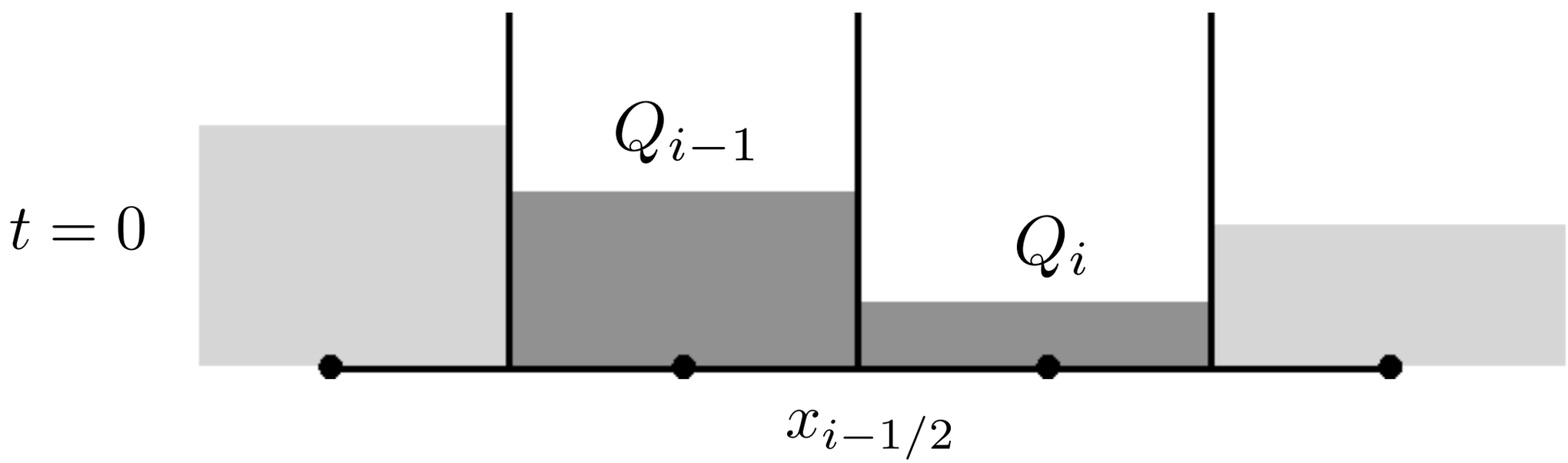
Riemann problem

At each cell interface, solve the hyperbolic problem with special initial data, i.e.

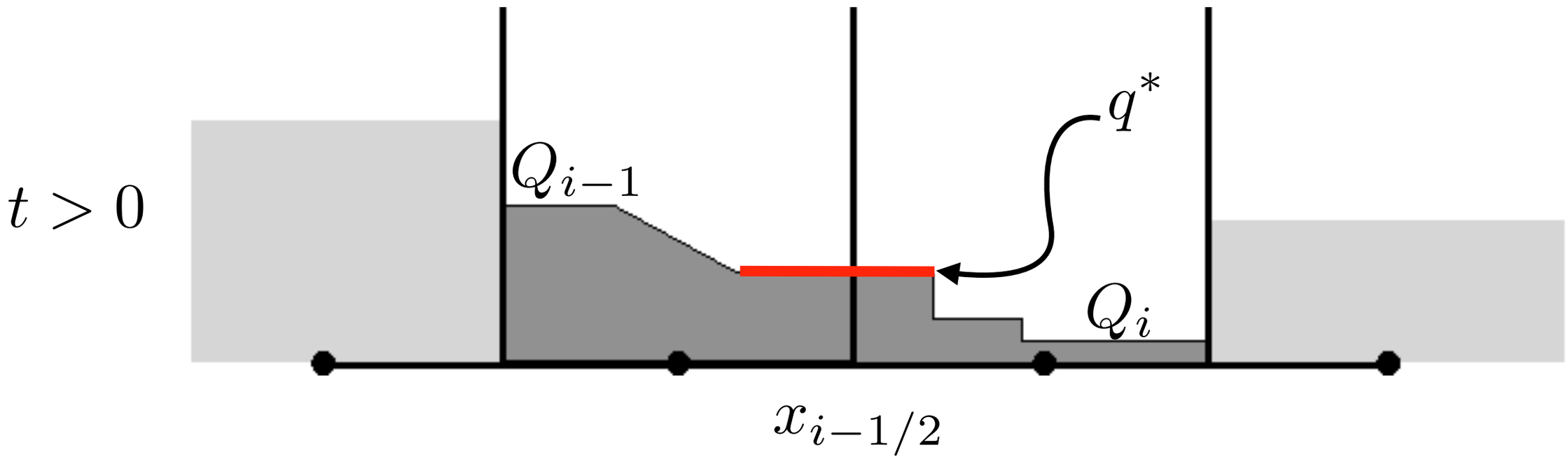
$$q_t + f(q)_x = 0$$

subject to

$$q(x,0) = \begin{cases} Q_{i-1} & x < x_{i-1/2} \\ Q_i & x > x_{x-1/2} \end{cases}$$



Riemann problem



Numerical flux at cell interface is then approximated by

$$F_{i-1/2} = f(q^*)$$

This is the classical Godunov approach for solving hyperbolic conservation laws.

- Resolves shocks and rarefactions and is conservative
- First order accurate

Parabolic and elliptic problems

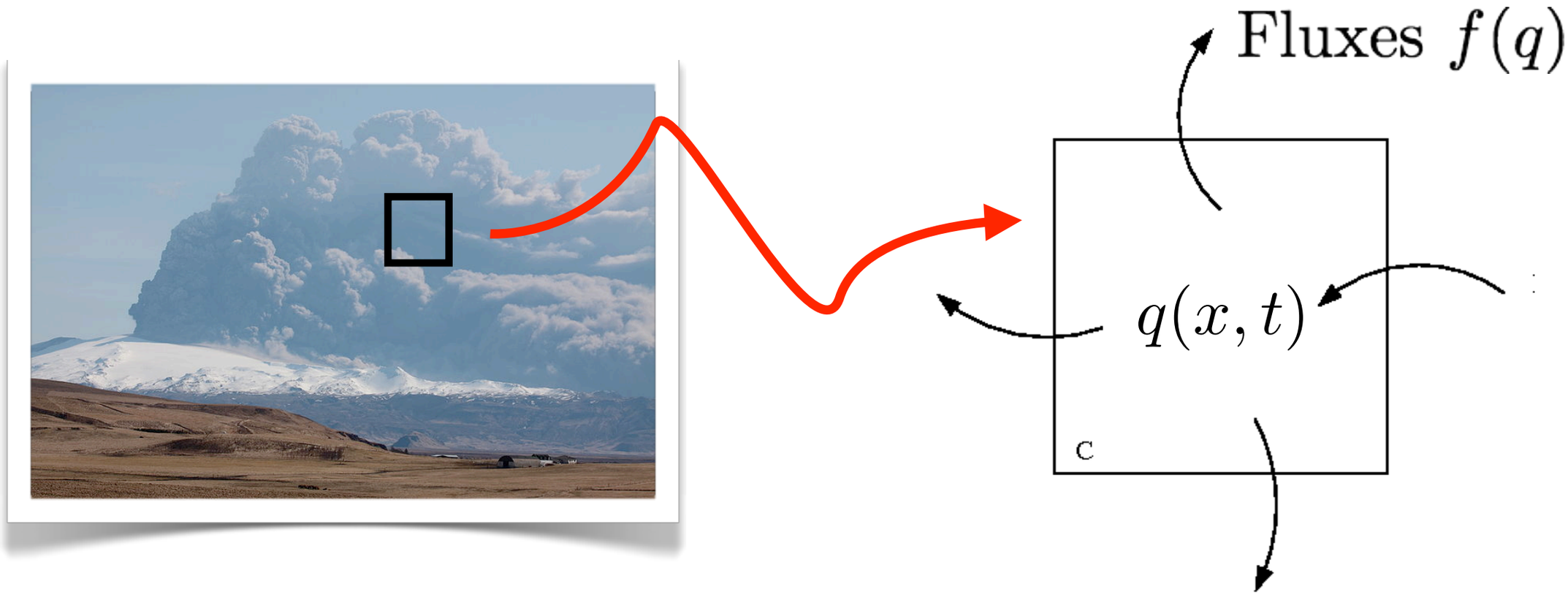
$$q_t + (\beta(x) q_x)_x = 0$$

$$(\beta(x) q_x)_x = g(x)$$

$$\beta(x) q_x \approx F_{i-1/2} = \beta_{i-1/2} \frac{Q_{i+1} - Q_i}{\Delta x}$$

- *Elliptic problems -> solve linear systems*
- *Parabolic systems -> solve implicitly to avoid time step restrictions imposed by standard explicit solvers*

Extensions to higher dimensions



A model of transport of $q(\mathbf{x}, t)$ in a *control volume* C :

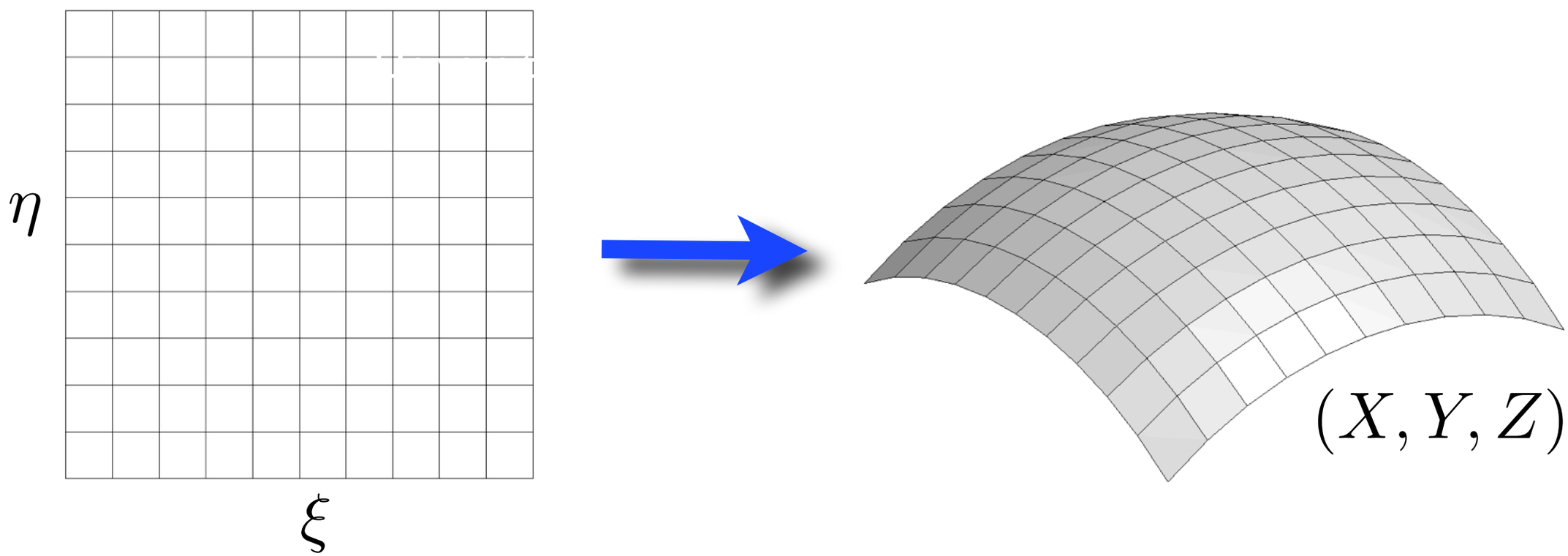
$$\frac{d}{dt}\int_C q\, dA \quad = \quad -\int_C \nabla \cdot \mathbf{f}(q)\, dA = -\int_{\partial C} \mathbf{f}(q) \cdot \mathbf{n}\, dL$$

divergence theorem

Finite volume schemes on curvilinear grids

We have extended the schemes described above to curvilinear mapped grids

$$\mathbf{T}(\xi,\eta) = (X(\xi,\eta), Y(\xi,\eta), Z(\xi,\eta))^T$$



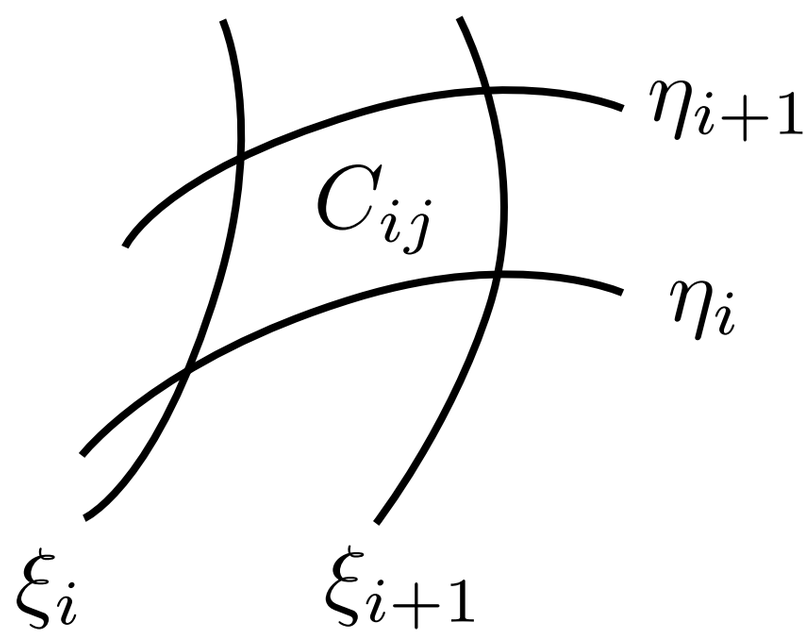
Computational space

Physical space

Finite volume scheme on mapped grids

$$\frac{d}{dt}\int_C q(\mathbf{x},t)\,dA = -\int_C \nabla\cdot\mathbf{f}(\mathbf{q})\,dA = -\int_{\partial C}\mathbf{f}(\mathbf{q})\cdot\mathbf{n}\,ds$$

holds generally over any 2d region where $q(\mathbf{x},t)$ is smooth.
On a transformed grid, our region is a mapped grid cell :



Assume grid remains logically Cartesian

Finite volume scheme on mapped grids

Defining the average as :

$$Q_{ij}^n \approx \frac{1}{A_{ij}} \int_{C_{ij}} q(\mathbf{x}, t)\, dA$$

A finite volume scheme for an equation in conservative form on the mapped grid can be written as :

Quadrilateral mesh cells

$$Q_{ij}^{n+1} = Q_{ij}^n - \frac{\Delta t}{A_{ij}} \sum_{k=1}^{4} \ell_k \bar{F}_k$$

where ℓ_k is the length of a cell edge.

Metric terms

Assume a smooth mapping of the form

$$T(\xi,\eta) = (X(\xi,\eta), Y(\xi,\eta), Z(\xi,\eta))^T$$

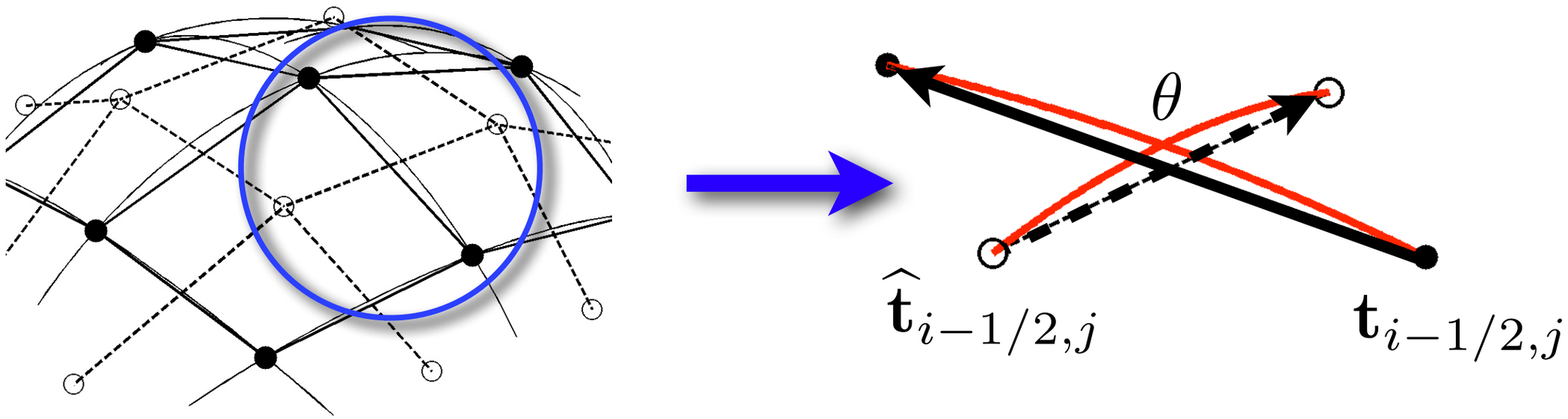
Then

$$\mathbf{t}_{(1)} \equiv T_\xi, \qquad \mathbf{t}_{(2)} \equiv T_\eta$$

$$\mathbf{a} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \equiv \begin{pmatrix} \mathbf{t}_{(1)} \cdot \mathbf{t}_{(1)} & \mathbf{t}_{(1)} \cdot \mathbf{t}_{(2)} \\ \mathbf{t}_{(2)} \cdot \mathbf{t}_{(1)} & \mathbf{t}_{(2)} \cdot \mathbf{t}_{(2)} \end{pmatrix}$$

$$a = \det(\mathbf{a})$$

Discrete metric terms



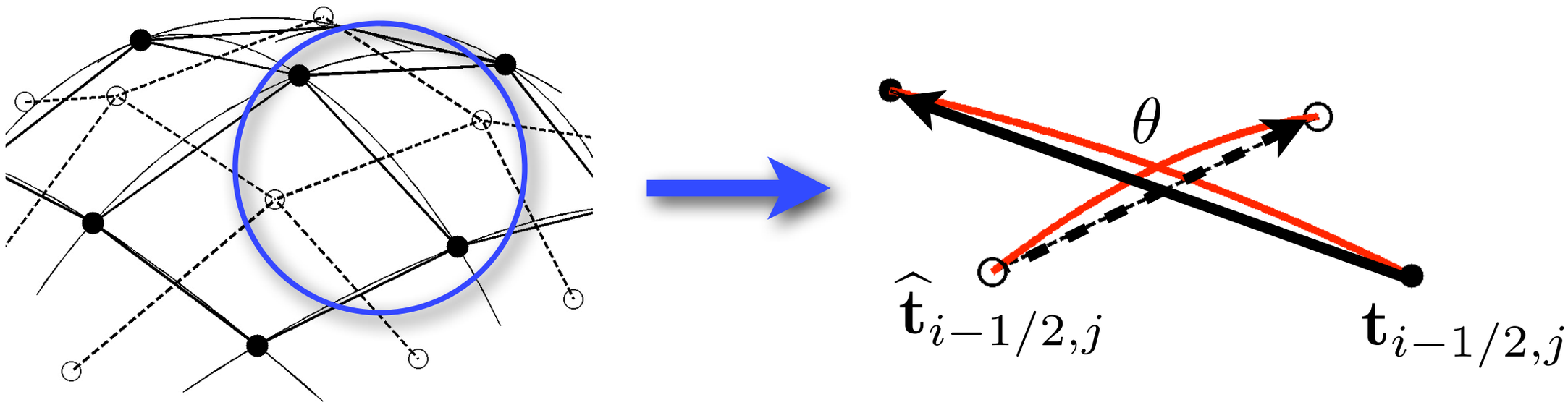
$$a_{11} = \mathbf{T}_\xi \cdot \mathbf{T}_\xi \approx \mathbf{t} \cdot \mathbf{t} = \|\mathbf{t}\|^2$$

$$a_{12} = a_{21} = \mathbf{T}_\xi \cdot \mathbf{T}_\eta \approx \mathbf{t} \cdot \widehat{\mathbf{t}} = \|\mathbf{t}\| \|\widehat{\mathbf{t}}\| \cos(\theta)$$

$$a_{22} = \mathbf{T}_\eta \cdot \mathbf{T}_\eta \approx \widehat{\mathbf{t}} \cdot \widehat{\mathbf{t}} = \|\widehat{\mathbf{t}}\|^2$$

$$\sqrt{a} = \| \mathbf{T}_\xi \times \mathbf{T}_\eta \| \approx \| \mathbf{t} \times \widehat{\mathbf{t}} \| = \|\mathbf{t}\| \|\widehat{\mathbf{t}}\| \sin(\theta)$$

Normals and lengths



Lengths

$$\ell_{i-1/2,j} = \int_{\eta_{j-1/2}}^{\eta_{j+1/2}} \| \mathbf{T}_\eta \| \; d\eta \approx \| \mathbf{t}_{i-1/2,j} \|$$

Edge normals tangent to the surface

$$\mathbf{n}_{i-1/2,j} \approx \frac{\mathbf{t}^{(1)}}{\| \mathbf{t}^{(1)} \|} \approx \frac{1}{\| \mathbf{t} \|} \left(\frac{\| \mathbf{t} \|}{\| \widehat{\mathbf{t}} \|} \csc(\theta) \, \widehat{\mathbf{t}} - \cot(\theta) \mathbf{t} \right)$$

Parabolic and elliptic problems

Need to approximate the Laplace Beltrami operator :

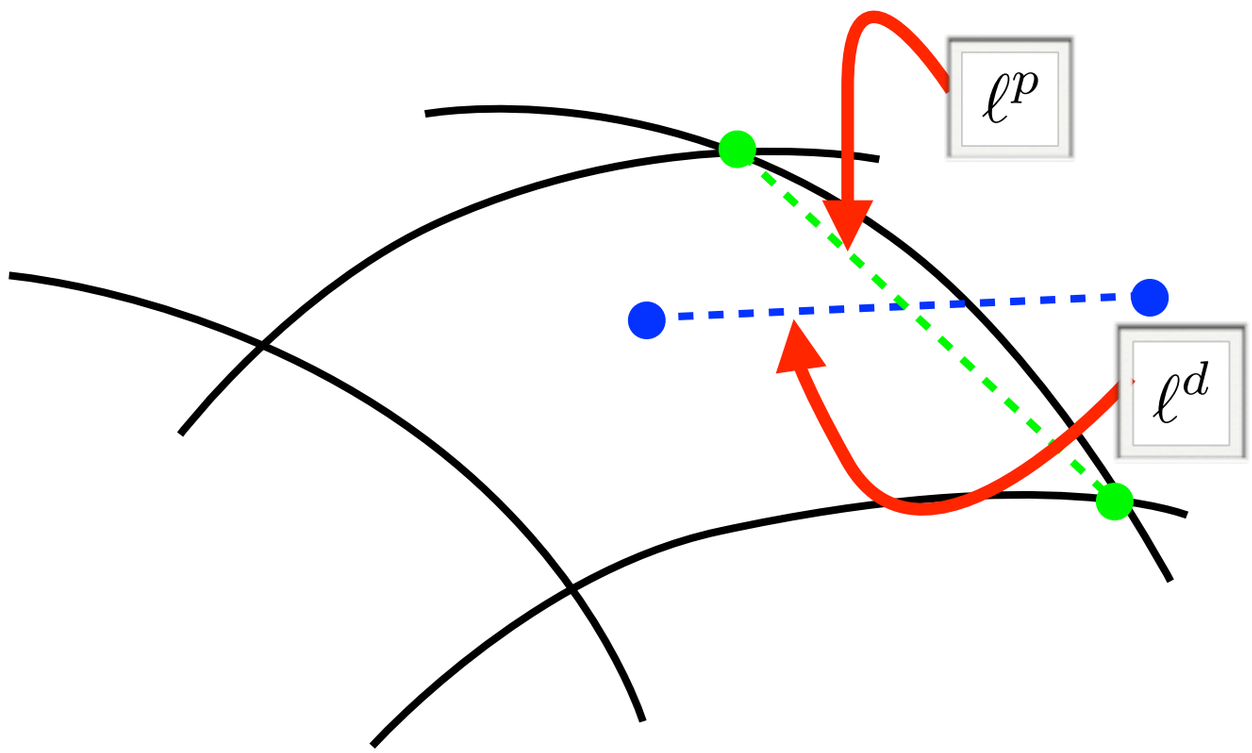
$$\nabla^2 q = \frac{1}{\sqrt{a}} \left\{ \frac{\partial}{\partial \xi} \sqrt{a} \left(a^{11} \frac{\partial q}{\partial \xi} + a^{21} \frac{\partial q}{\partial \eta} \right) + \frac{\partial}{\partial \eta} \sqrt{a} \left(a^{21} \frac{\partial q}{\partial \xi} + a^{22} \frac{\partial q}{\partial \eta} \right) \right\}$$

$$\nabla^2 q = \frac{1}{\sqrt{a}} \left\{ \frac{\partial}{\partial \xi} \frac{1}{\sqrt{a}} \left(a_{22} \frac{\partial q}{\partial \xi} - a_{21} \frac{\partial q}{\partial \eta} \right) - \frac{\partial}{\partial \eta} \frac{1}{\sqrt{a}} \left(a_{21} \frac{\partial q}{\partial \xi} + a_{11} \frac{\partial q}{\partial \eta} \right) \right\}$$

"Fluxes", scaled by the lengths of the finite volume edges.

Parabolic and elliptic problems

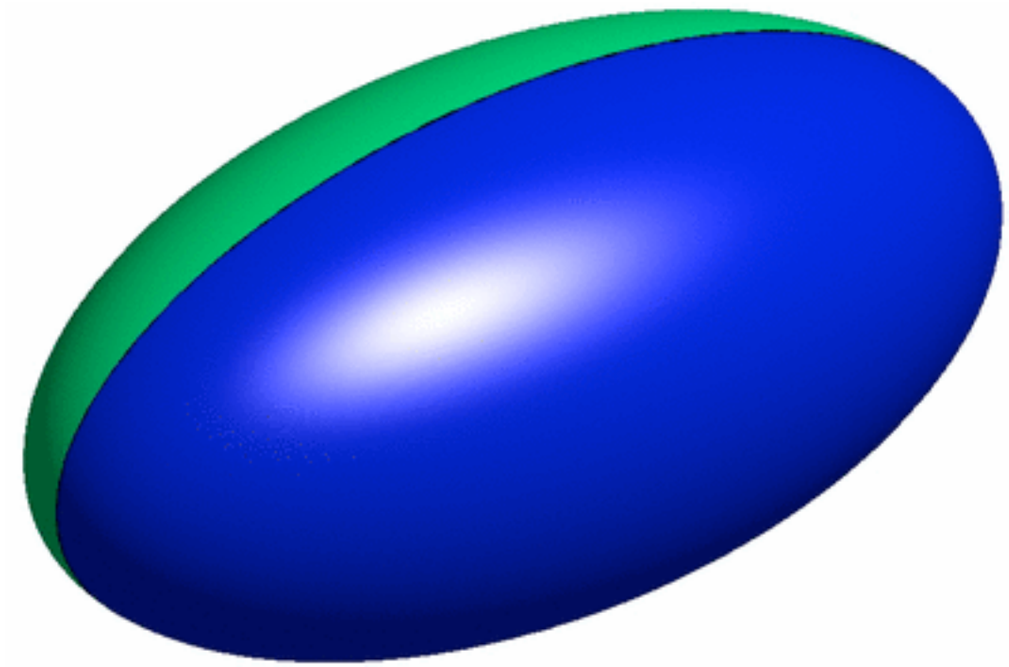
$$\nabla^2 q \approx L(q) \equiv \frac{1}{\text{Area}} \sum_{k=1}^{4} \frac{|\ell_k^p|}{|\ell_k^d|} \csc(\theta_k) \Delta_k q - \cot(\theta_k)\, \Delta_k \widehat{q}$$



Construct coefficients and solve resulting system

DC, C. Helzel, SISC 2008

Spiral waves



$$u_t = \nabla^2 u + \varepsilon^{-1} u(1-u)(u + a^{-1}(v+b))$$
$$v_t = u - v$$

Barkley model

Hyperbolic problems

Shallow water wave equations and the Euler equations both satisfy a rotational invariance property :

Given fluxes $\mathbf{f}(q) = (f_1(q), f_2(q))$ and a rotation matrix

$$R = \begin{pmatrix} 1 & 0 & 0 \\ 0 & n_\xi^1 & n_\xi^2 \\ 0 & \tau_\xi^1 & \tau_\xi^2 \end{pmatrix}$$

edge normals

we have

$$R^{-1}\mathbf{f}(Rq) = n_\xi^1 \, f_1(q) + n_\xi^2 \, f_2(q)$$

Hyperbolic problems

$$R^{-1}\mathbf{f}(Rq) = n_\xi^1 \; f_1(q) + n_\xi^2 \; f_2(q)$$

- This allows us to use the same Riemann solver in any direction.

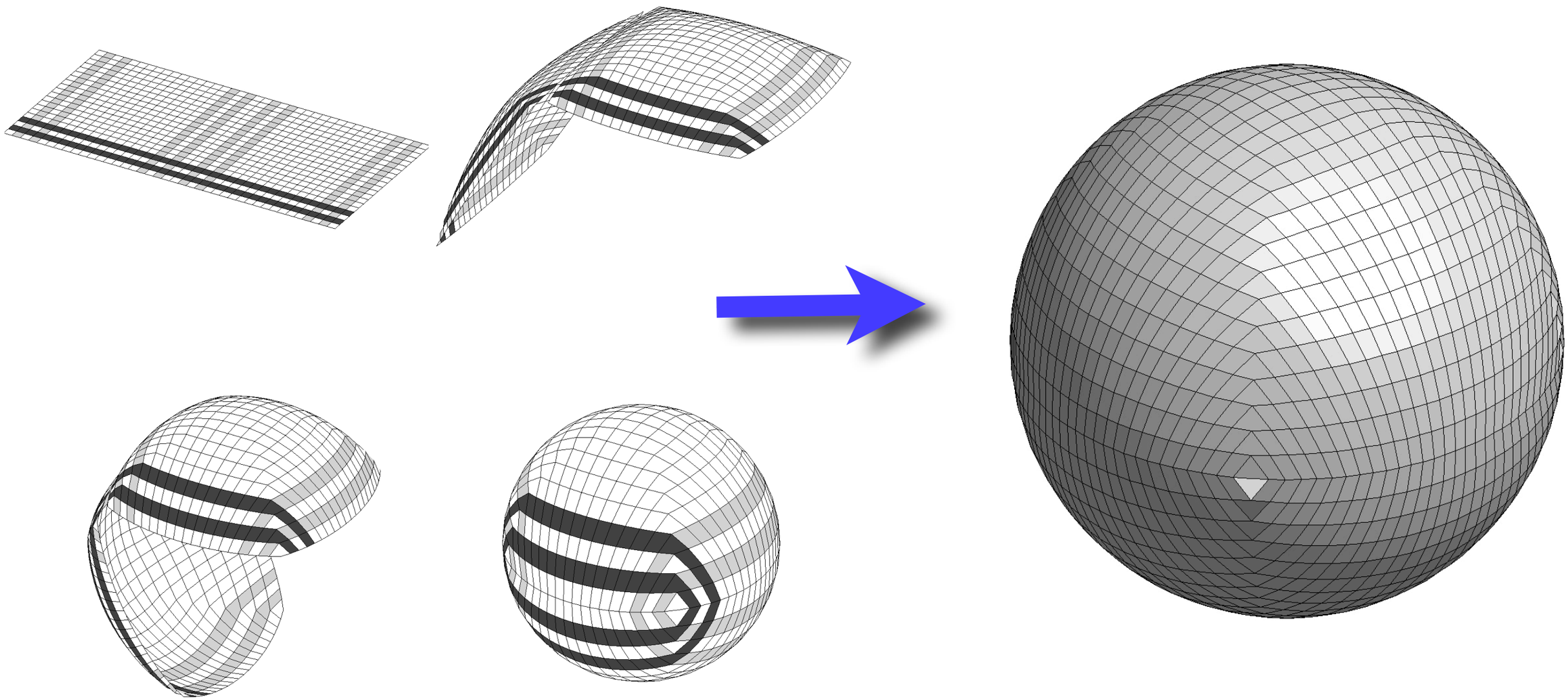
(1) Rotate data in left and right states to align with the normal and tangential directions at cell edge

(2) Use these rotated velocities to solve Riemann problem, either exactly or approximately,

(3) Scale the resulting speeds using edge scale factors

(4) Rotate resulting waves back to their Cartesian components

Logically Cartesian sphere grid



D. Calhoun, C. Helzel, and R. LeVeque, SIAM Review, 50 (2008)

Shallow water on the sphere

$$\begin{aligned} \phi_t + \nabla \cdot \mathbf{u}\phi &= 0 \\ (\mathbf{u}\phi)_t + \mathbb{P}\nabla \cdot \widetilde{\mathbf{F}}(q) &= -\frac{2\Omega \mathbf{z}\phi}{R^2}(\mathbf{x} \times \mathbf{u}) - \phi \mathbb{P}\nabla B \end{aligned}$$

- $\phi(\mathbf{x}, t)$ is the ocean depth
- $\mathbf{u} = (u, v, w)$ Cartesian components of velocity
- $B(\xi, \eta)$ is the ocean bathymetry
- $\mathbb{P}$ is a projection operator that projects momentum components onto the surface of the sphere
- $q = [\phi, \phi u, \phi v, \phi w]$ is the vector of conserved quantities.
- Ω rotation due to Coriolis forces

Shallow water on the sphere

$$\widetilde{\mathbf{F}}(q) = \begin{pmatrix} \phi u^2 + \frac{1}{2} g \phi^2 & \phi u v & \phi u w \\ \phi u v & \phi v^2 + \frac{1}{2} g \phi^2 & \phi v w \\ \phi u w & \phi v w & \phi w^2 + \frac{1}{2} g \phi^2 \end{pmatrix}$$

- Rotational invariance of the SWE allows us to formulate the equations in three dimensions, even though they are only solved in two dimensions
- Momentum is projected onto the sphere via a projection operator
- Cartesian components of the momentum are updated

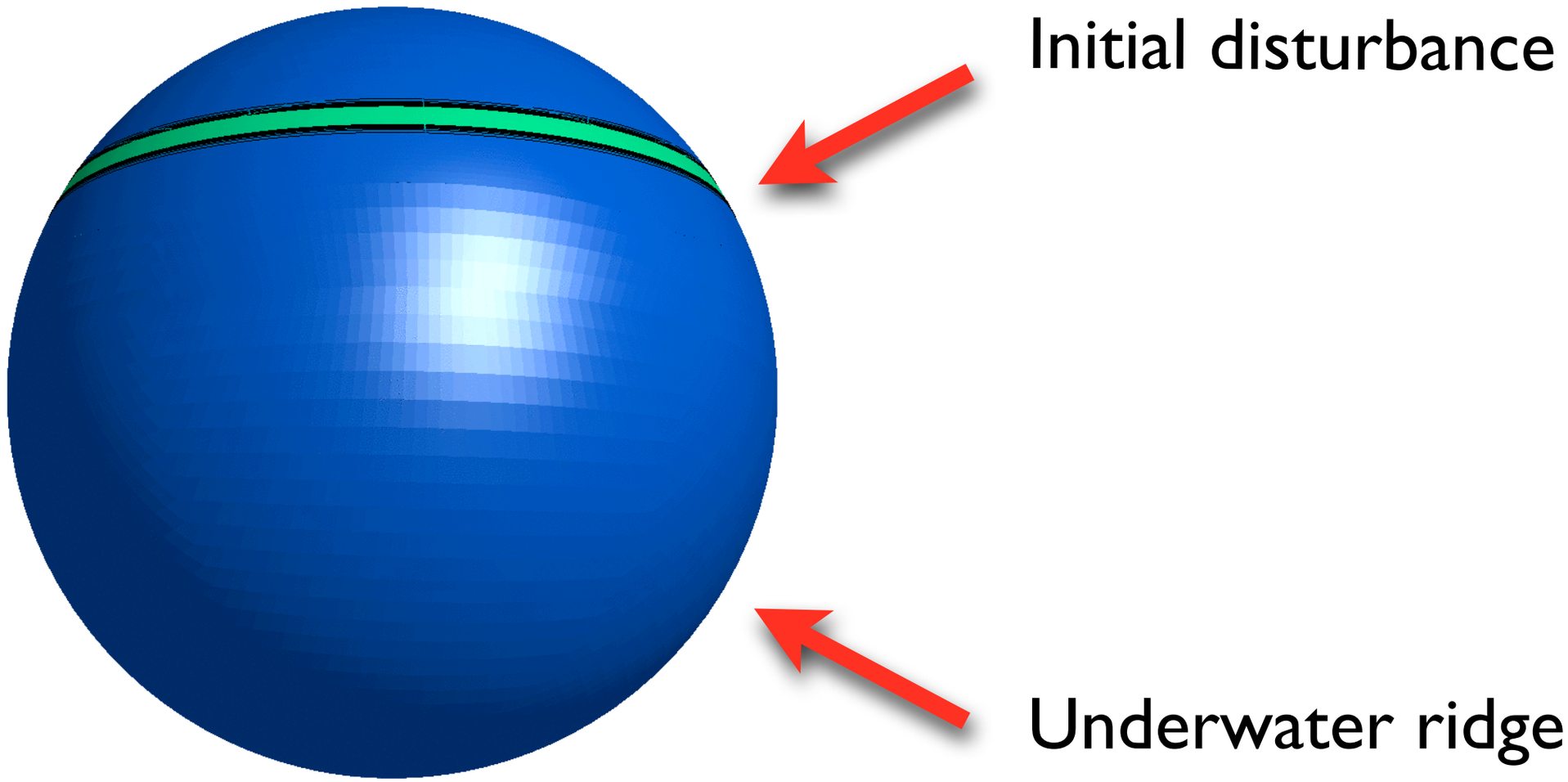
Shallow water test case

Initial disturbance

Underwater ridge

- *M. Berger, D. Calhoun, C. Helzel, R. J. LeVeque, Phil. Trans. Proc. R. Soc. A, (367) 2009*
- *GeoCLAW, R. J. LeVeque, www.clawpack.org (code available)*

Shallow water test case



- *M. Berger, D. Calhoun, C. Helzel, R. J. LeVeque, Phil. Trans. Proc. R. Soc. A, (367) 2009*
- *GeoCLAW, R. J. LeVeque, www.clawpack.org (code available)*

Logically Cartesian meshes

Why use Cartesian meshes instead of unstructured meshes?

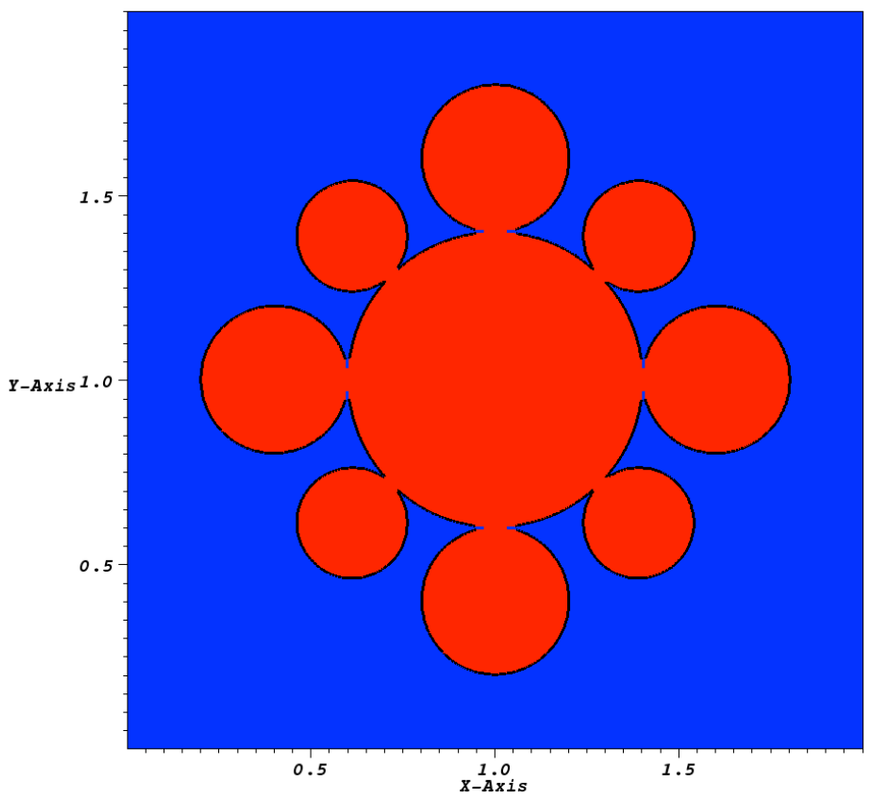
- Mesh generation is easier, if not trivial
- Solution is not dependent on the quality of the mesh
- Algorithms are easier to construct on smooth logically Cartesian meshes and the results are more accurate than on unstructured, non-smooth meshes
- Layout of the Cartesian data maps directly to the computer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

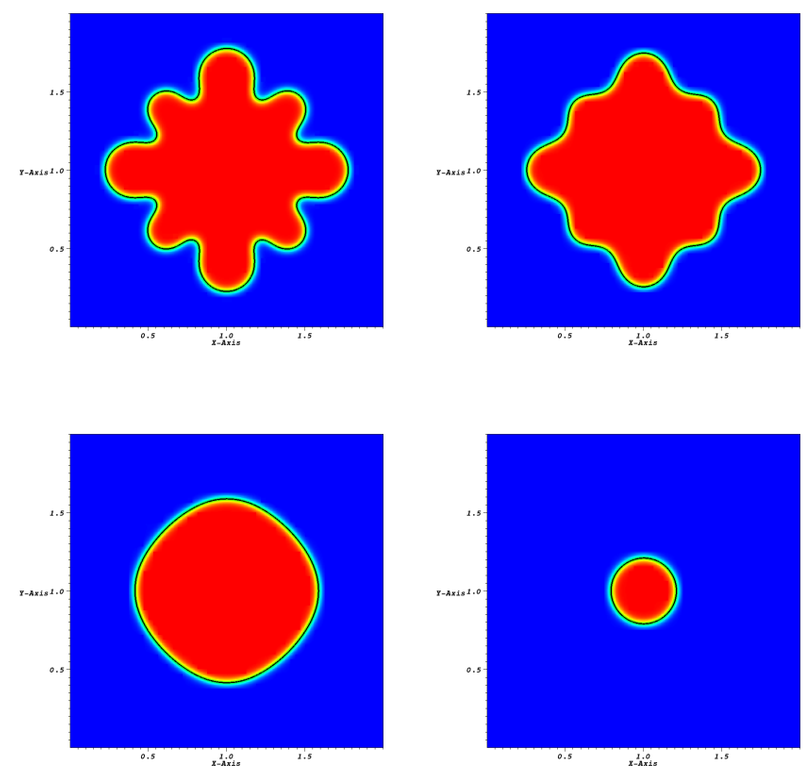
But ...

Use of computational resources is not efficient.

Adaptive Mesh Refinement

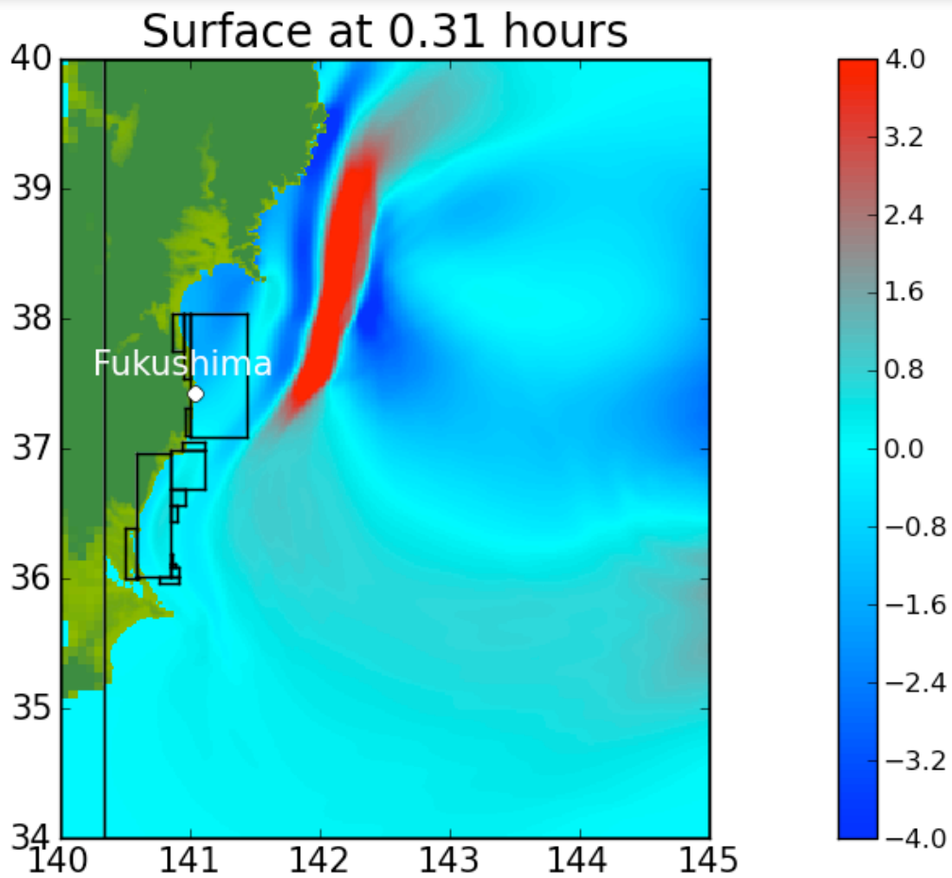
When solving PDEs using mesh based methods, it is generally recognized that many problems could benefit enormously from a multi-resolution grid.



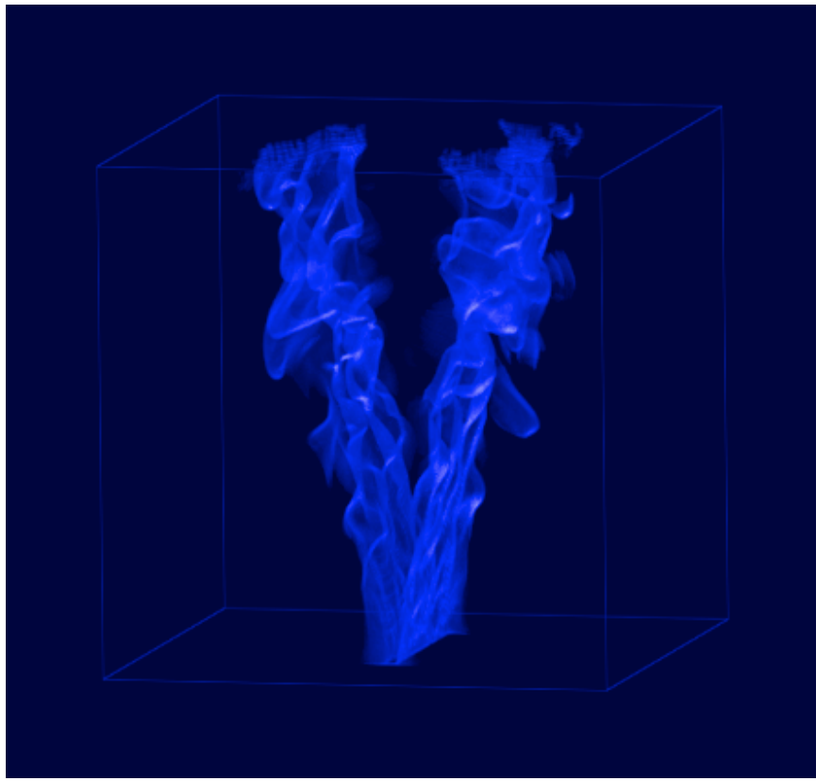


Allen Cahn equation - Flow by mean curvature

Applications for AMR



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

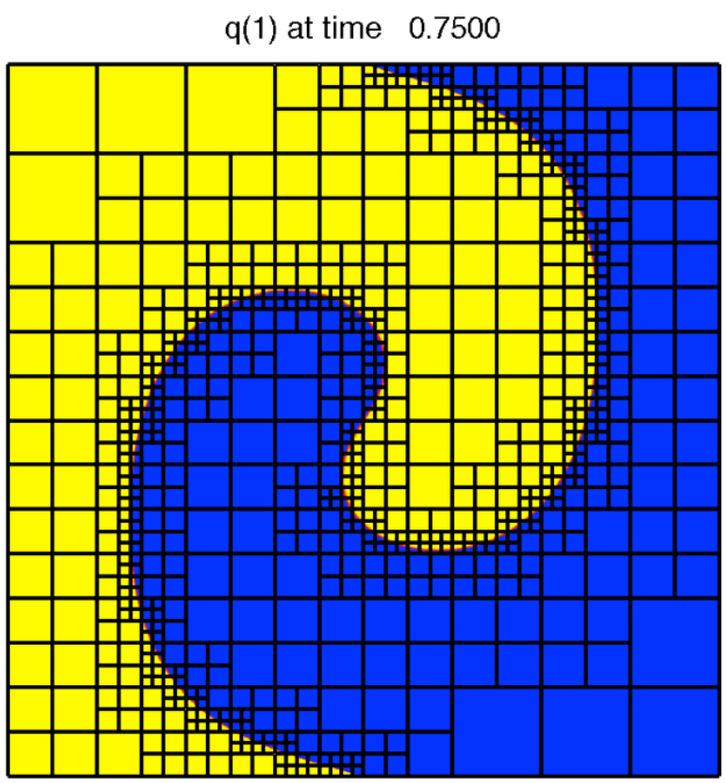


Rod stabilized V-flame (J. B. Bell, Lawrence Berkeley Lab)

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

Many flavors of adaptivity

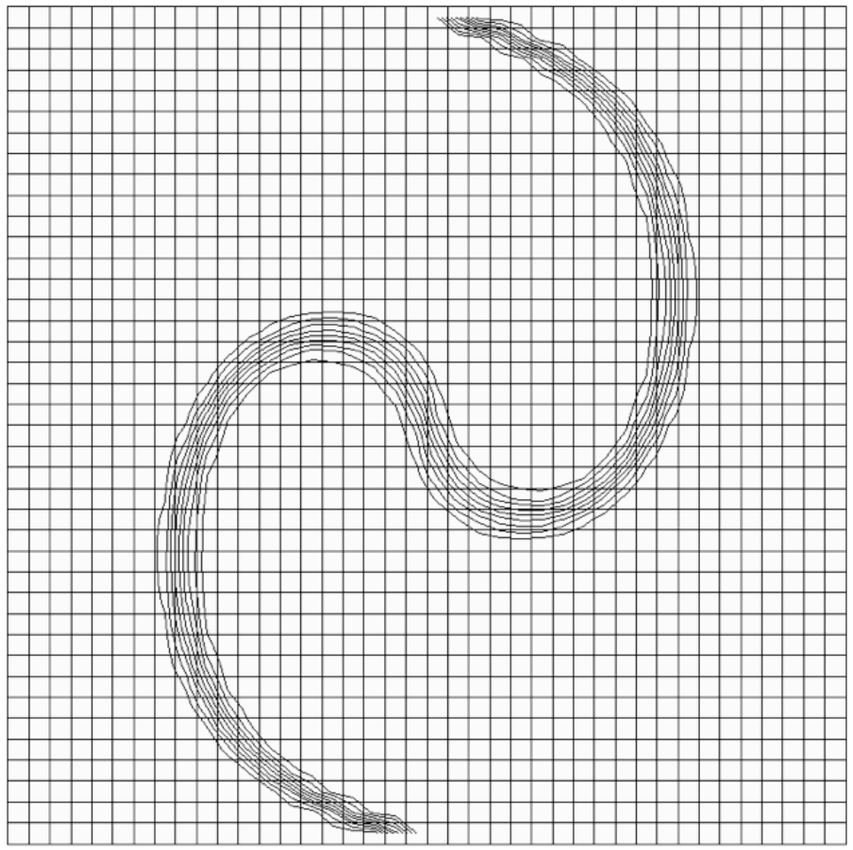
- Block-structured AMR (Berger, Oliger, Colella, ...)
- Tree-based adaptivity (Popinet, Tessyier, ...)
- Finite-element adaptivity includes both h-refinement (increase mesh resolution) and p-refinement (increase order of accuracy)

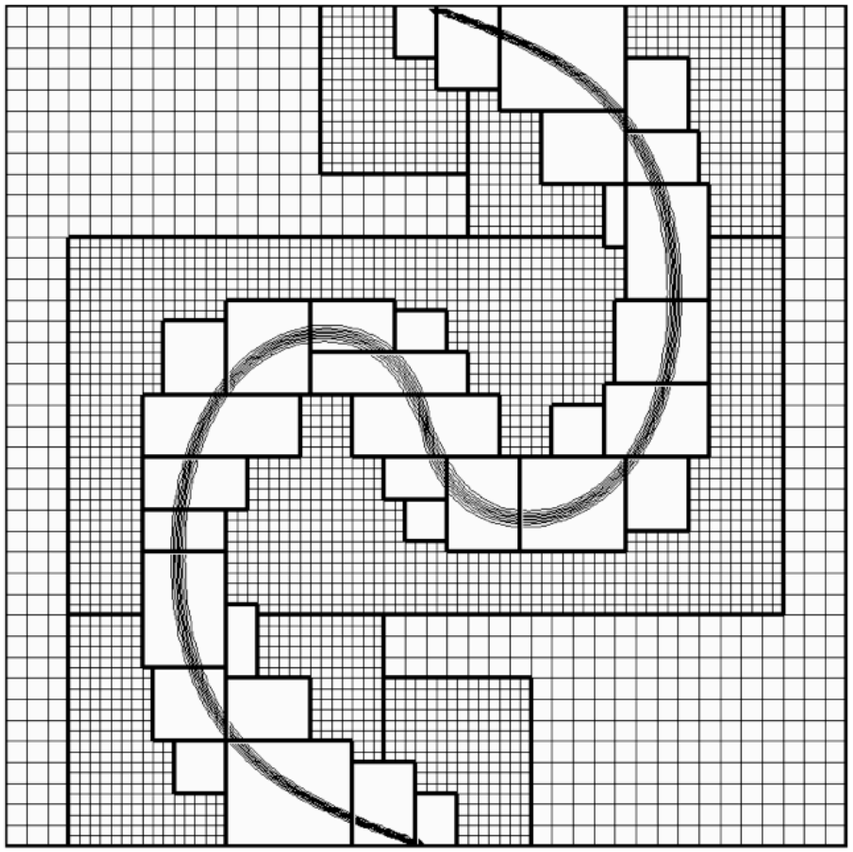


Tree-based adaptivity :

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

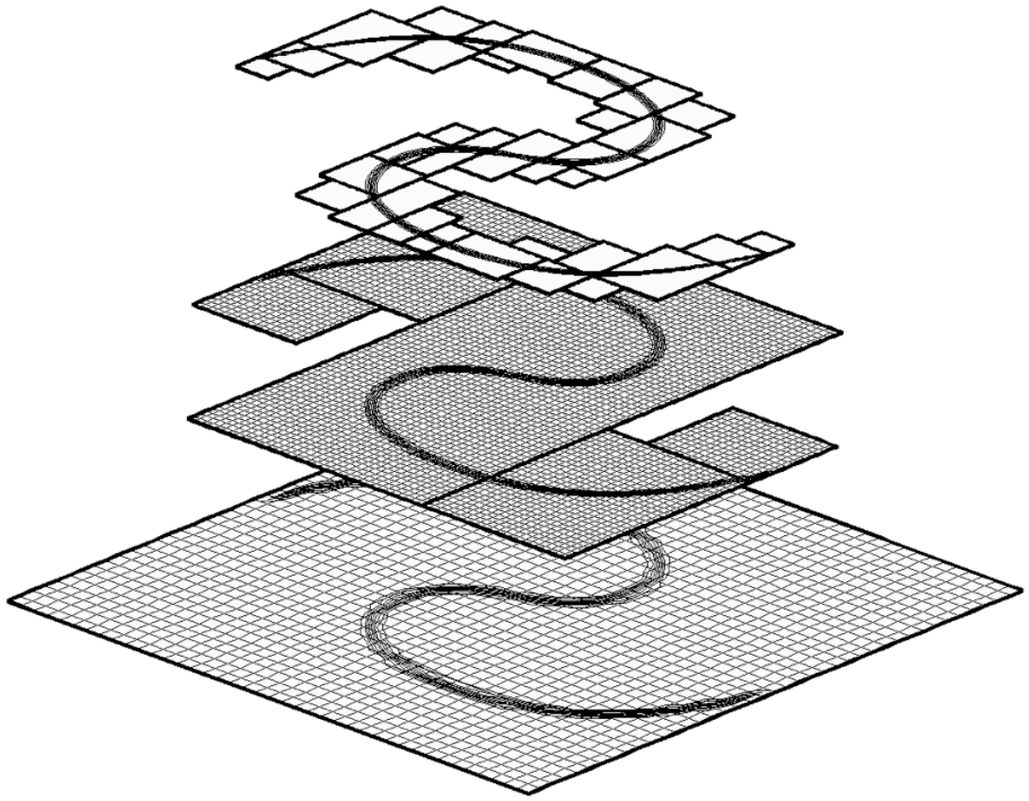
Block structured AMR (Berger, Oliger 1984)





- 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 (ala Berger and Oliger)



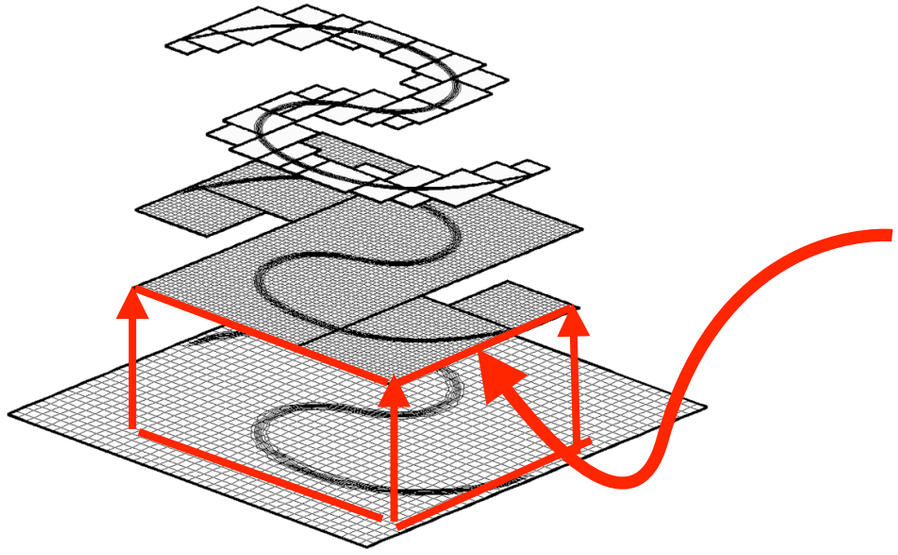
- Data is stored in nested, layered hierarchy of logically Cartesian grids,
- Multi-rate time stepping based on mesh size,
- Grids are dynamically refined and de-refined to adapt to the solution features of interest.
- Communication between grids is done via ghost cells.

Solution procedure

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

Software for AMR

Great idea! How can I add adaptivity to my code?

- It is much harder than it looks (and you don't really just "add" adaptivity)
- And there are several codes already available that you can feed your single grid solver to.

Even better! What's available?

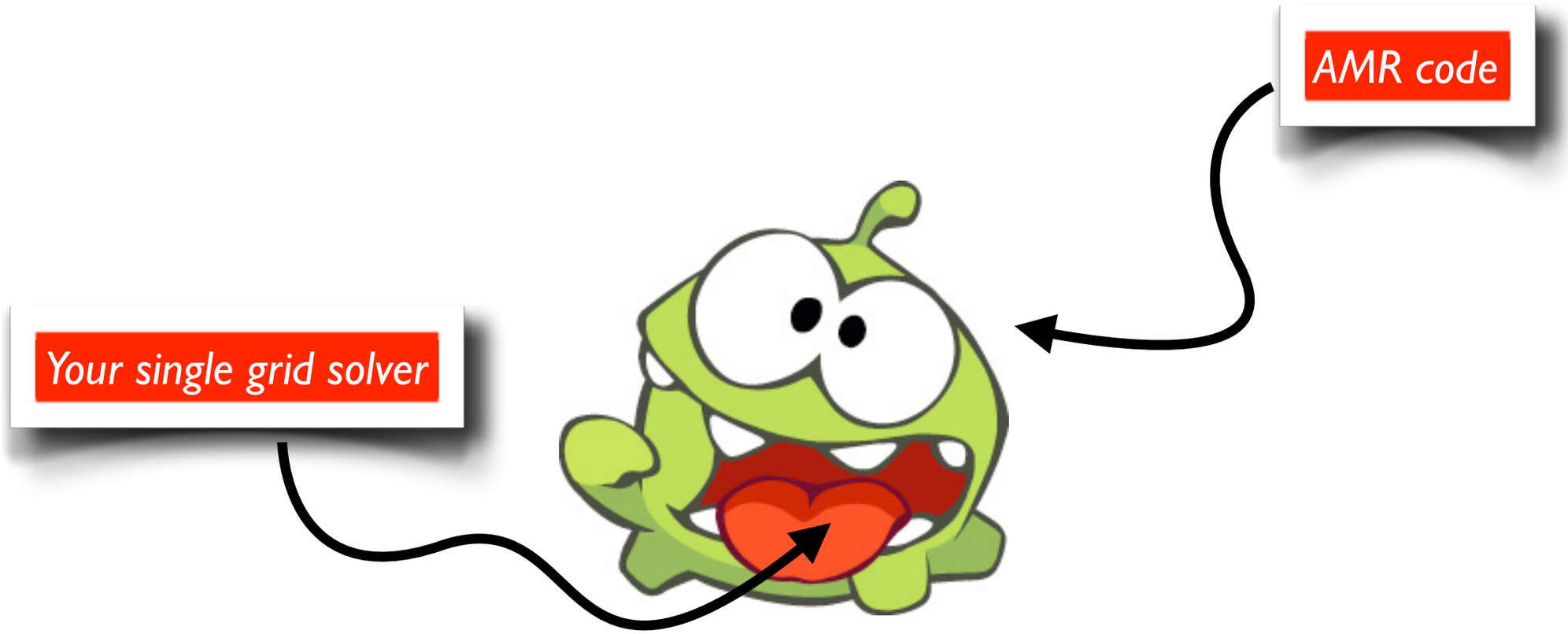
Block structured AMR codes

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

See my website for a list of several more application specific codes

Using block structured AMR codes

Using SAMRAI, Chombo, AMRClaw, BoxLib, Paramesh, ...



** Idea for code name : OmNum*

Block structured AMR codes

The Dream

```
AMR.run(max_time, max_steps);
```

Your single grid solver is called from here.

Block structured AMR codes

The Reality

```
Tuple< RefCountedPtr<AMRLevelOpFactory<
LevelData<FArrayBox> > >, SpaceDim> velTGAOpFactoryPtrs;

for (int idir = 0; idir < SpaceDim; idir++)
{velTGAOpFactoryPtrs[idir] =
    RefCountedPtr<AMRLevelOpFactory<LevelData
    <FArrayBox> > >
    ((AMRLevelOpFactory<LevelData<FArrayBox> >*)
    (new AMRPoissonOpFactory())); //.....
```

Your single grid solver

you

Retro...

```
node(ndjhi,mptrnx) = node(ndjhi,mptr)
node(ndjhi,mptr)   = node(ndjlo,mptr) + nyl - 1
node(ndjlo,mptrnx) = node(ndjhi,mptr) + 1
node(ndihi,mptrnx) = node(ndihi,mptr)
node(ndilo,mptrnx) = node(ndilo,mptr)

rnode(cornxlo,mptrnx)   = cxlo
rnode(cornylo,mptrnx)   = cymid
rnode(cornyhi,mptrnx)   = cyhi
rnode(cornxhi,mptrnx)   = cxhi
node(nestlevel,mptrnx)  = node(nestlevel,mptr)
rnode(timemult,mptrnx)  = rnode(timemult,mptr)
go to 10
```

Block structured AMR codes

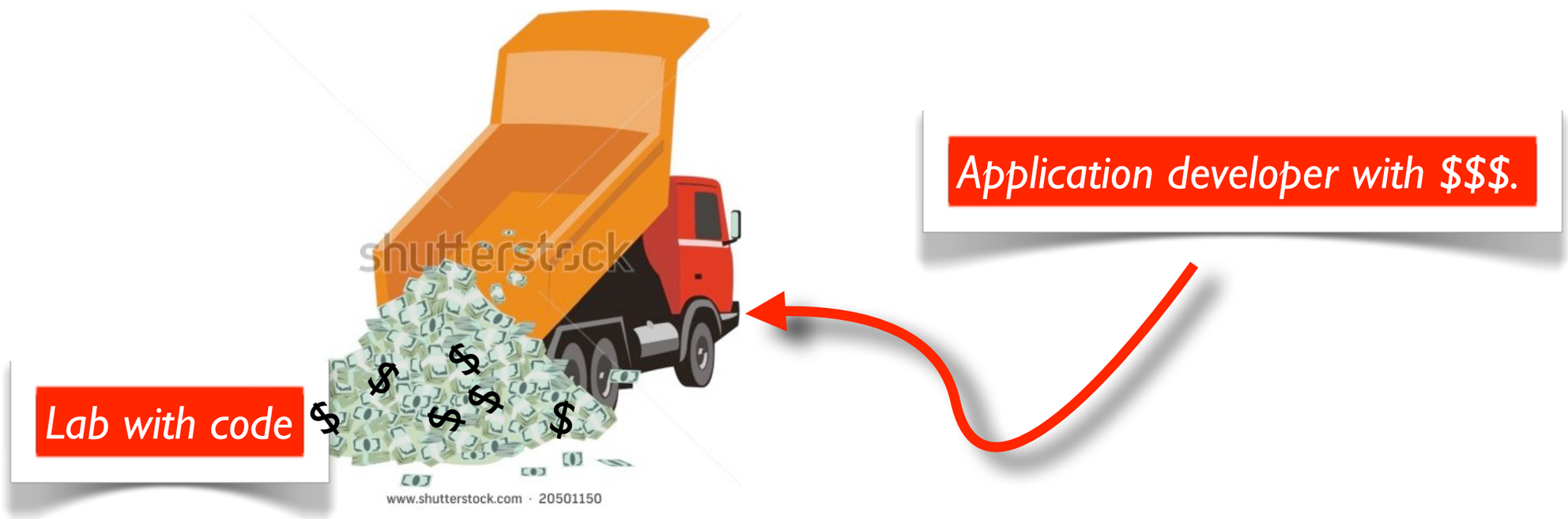
"PARAMESH is a package of Fortran 90 subroutines designed to provide an application developer with an easy route to extend an existing serial code which uses logically Cartesian structured mesh into a parallel code with adaptive mesh refinement"

SAMRAI - "Object oriented C++ library developed to provide algorithmic and software support to large scale multiphysics problems relevant to the US Department of energy (DOE)"

Boxlib - "These libraries provide the software infrastructure for the computational activities (combustion, astrophysics, porous media flow) at the Center for Computational Sciences and Engineering (CCSE) at Lawrence Berkeley Labs"

Block-structured AMR codes

Most of these codes were designed mainly to support large scale applications developers and in-house scientific research.



But what if you have ideas about ...

- Multi-stage, multi-step, IMEX, SSP, parallel-in-time, and other time stepping schemes in an adaptive setting,
- Accuracy of multi-rate schemes for PDEs with mixed elliptic/parabolic/hyperbolic terms.
- Elliptic and parabolic solvers (iterative? direct? Explicit? Fast multipole?)
- Parallelism in the AMR setting?
- Error estimation
- Higher order accuracy
- Complex physics

Should you write yet-another-AMR code?

AMR for the computational mathematician

What I would like an AMR package to do :

- Support for grid management that is separate from the numerics, that is intuitive, with easily manageable data structures,
- Support for multi-rate time stepping with flexibility to include new time stepping schemes (MOL solvers, for example),
- Natural code for iterating over arrays (*in Fortran?*),
- Flat data structures - little reliance on templates, and exotic object oriented data structures
- Parallelism should happen automatically.
- Simple build system

A hybrid approach to AMR

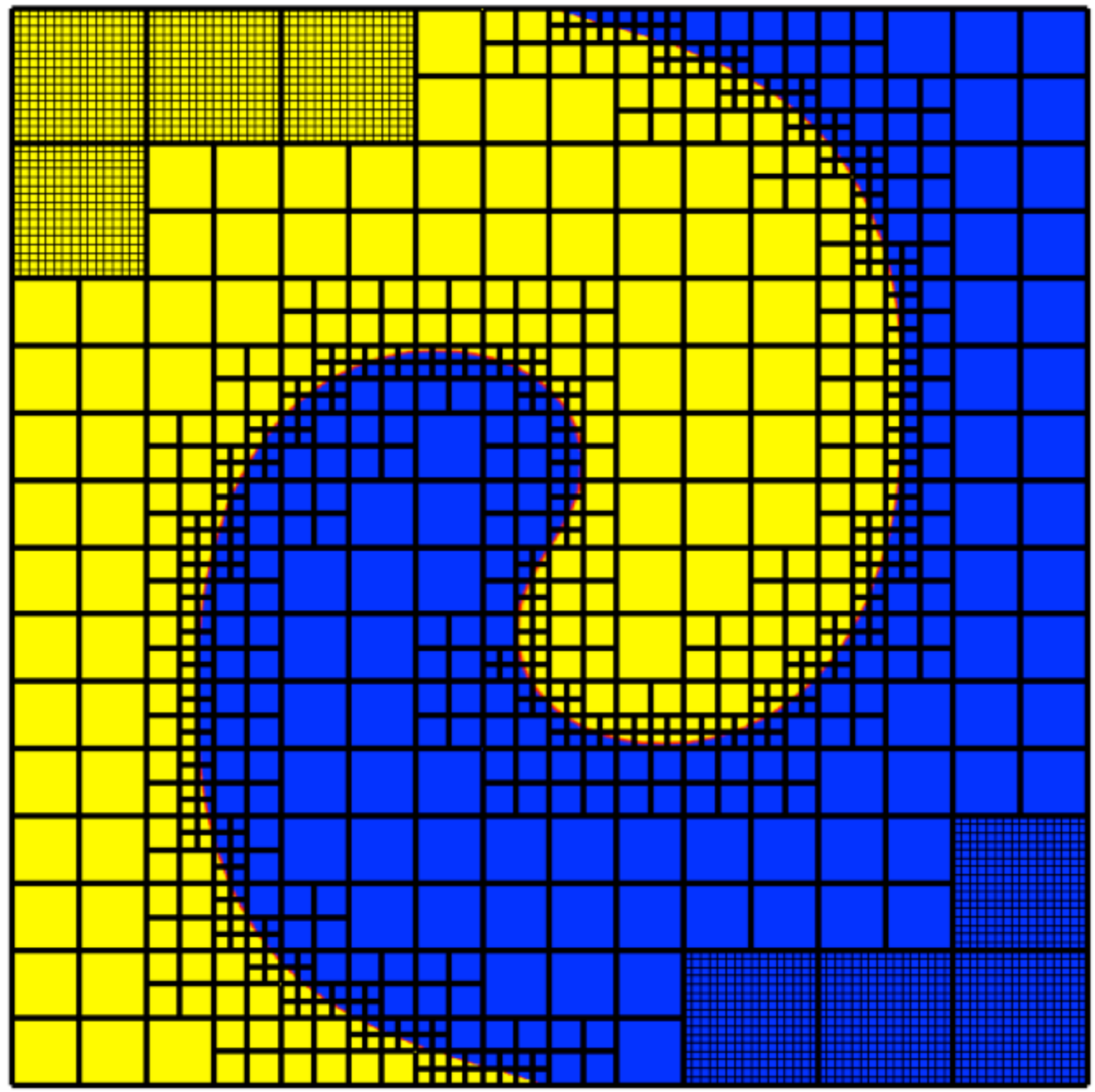
Use an existing parallel quad/octree library to do the grid management

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

Advantages

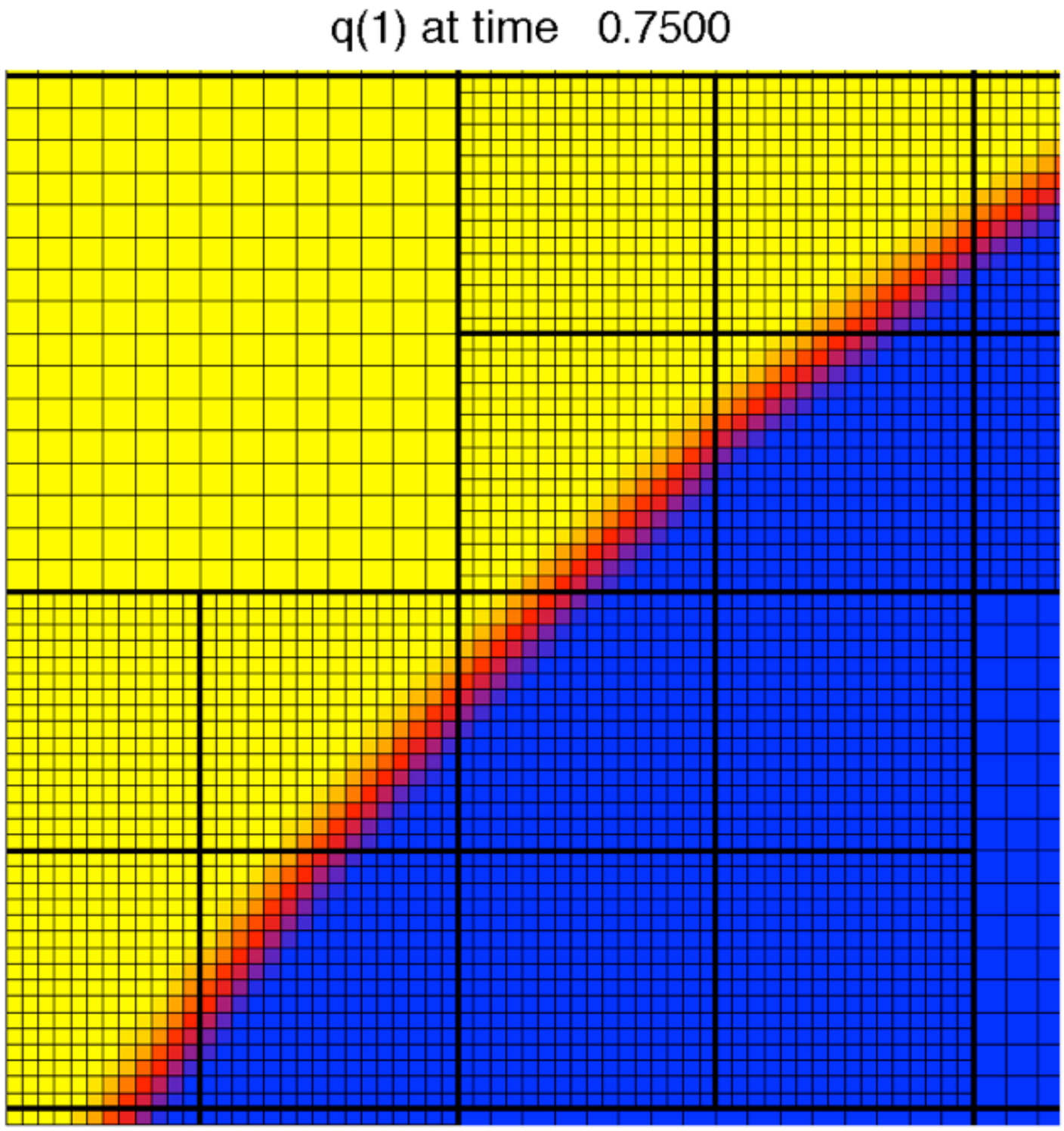
- Use of quad/octrees may be more ascetically satisfying than overlapping grids (CS people like them better...)
- Tree-based grid layout makes communication between grids much simpler,
- Construction of refined patches is trivial,
- Clear separation between the grid management (including neighbor communication) and the numerics

A hybrid approach to AMR

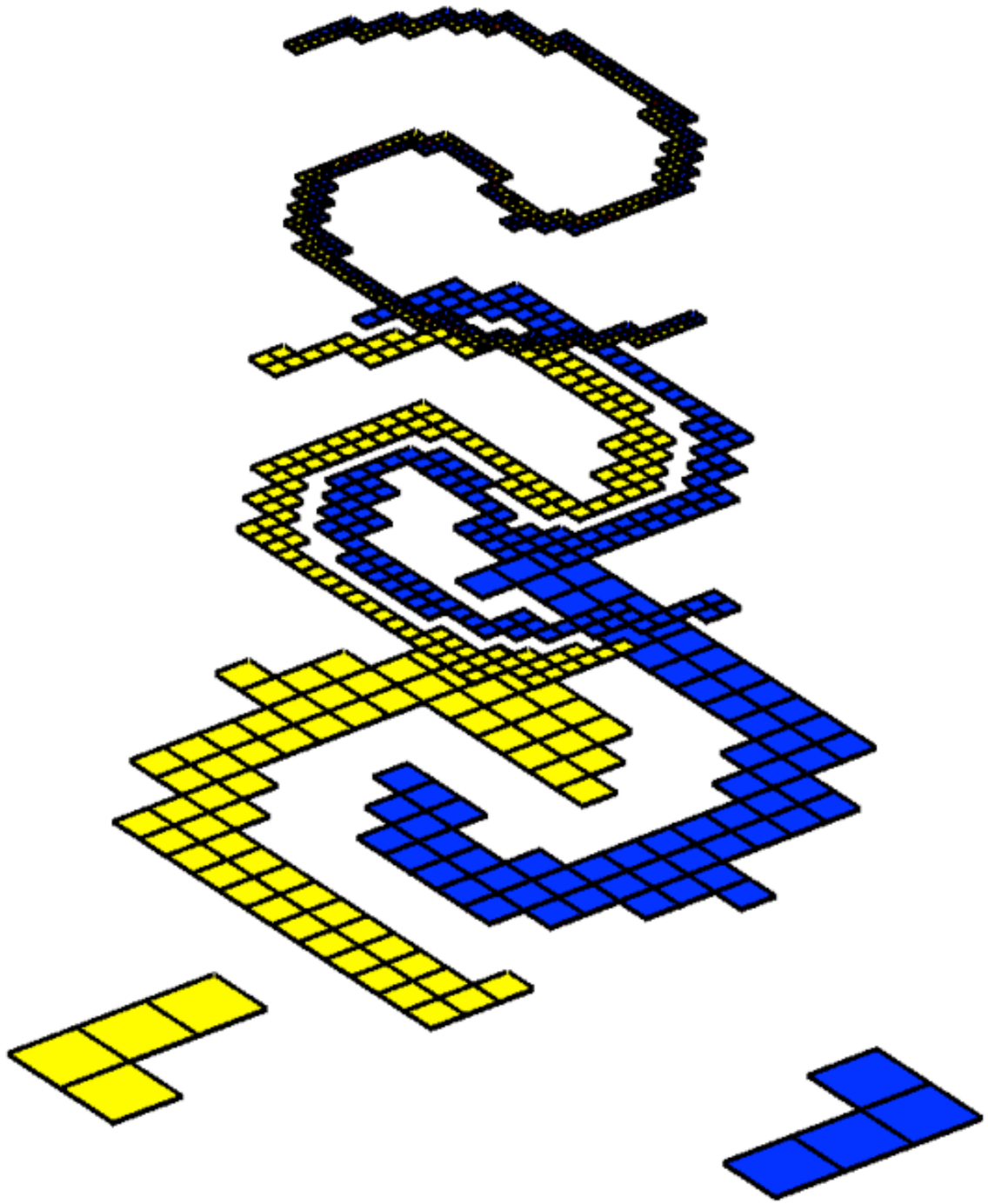


Each leaf is a fixed size grid

A hybrid approach to AMR

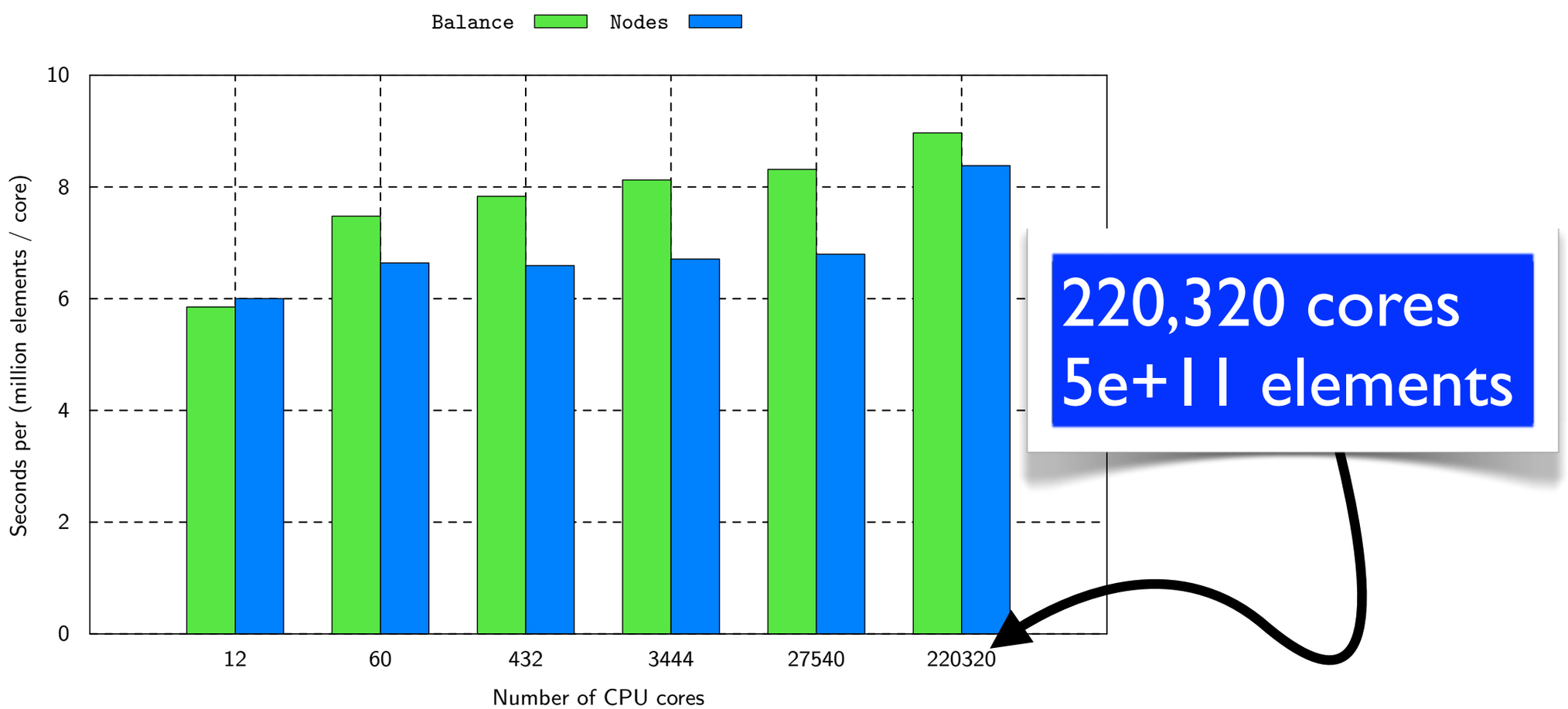


A hybrid approach to AMR

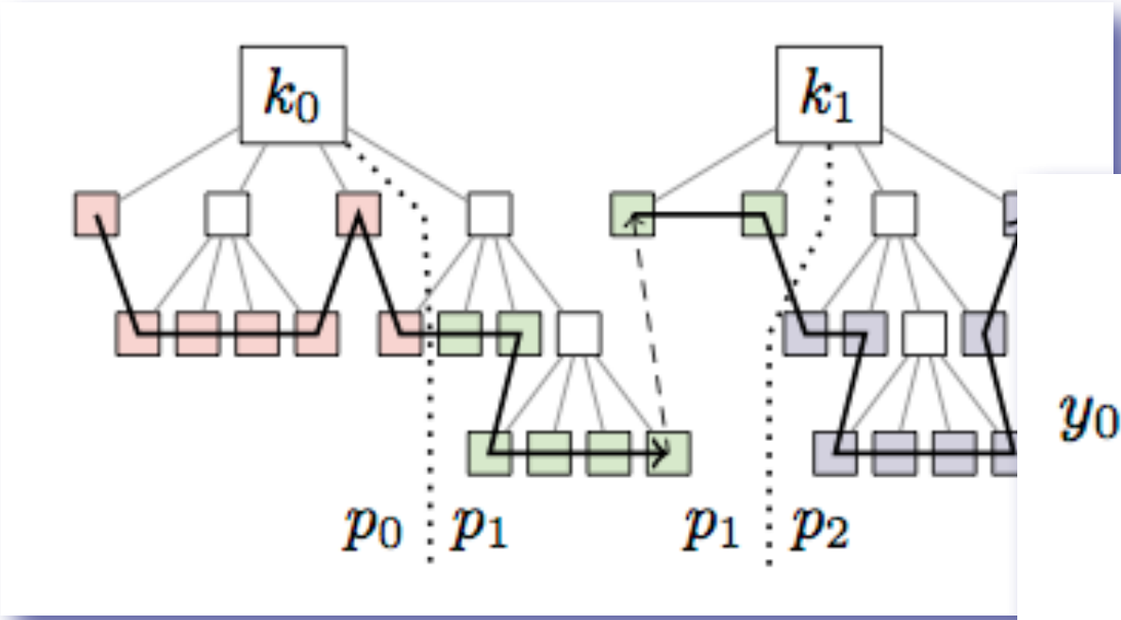


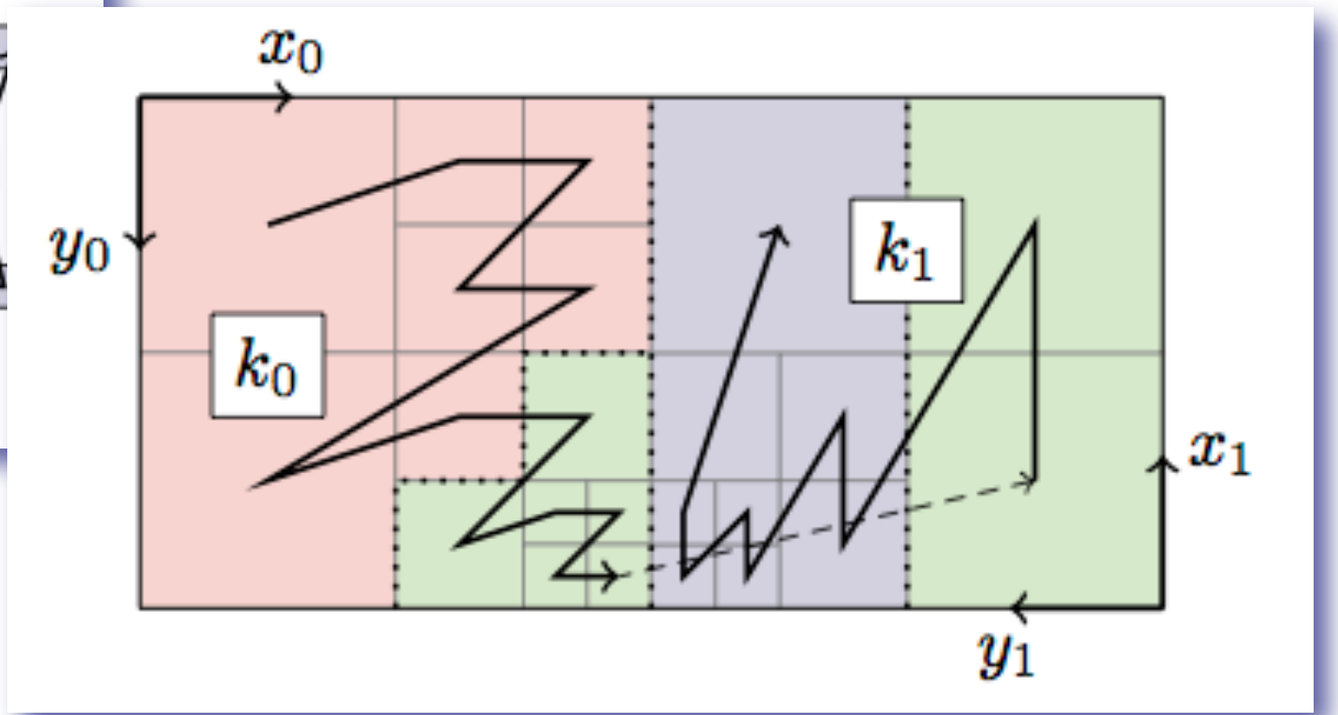
p4est (Carsten Burstedde, Univ. Bonn)

- 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

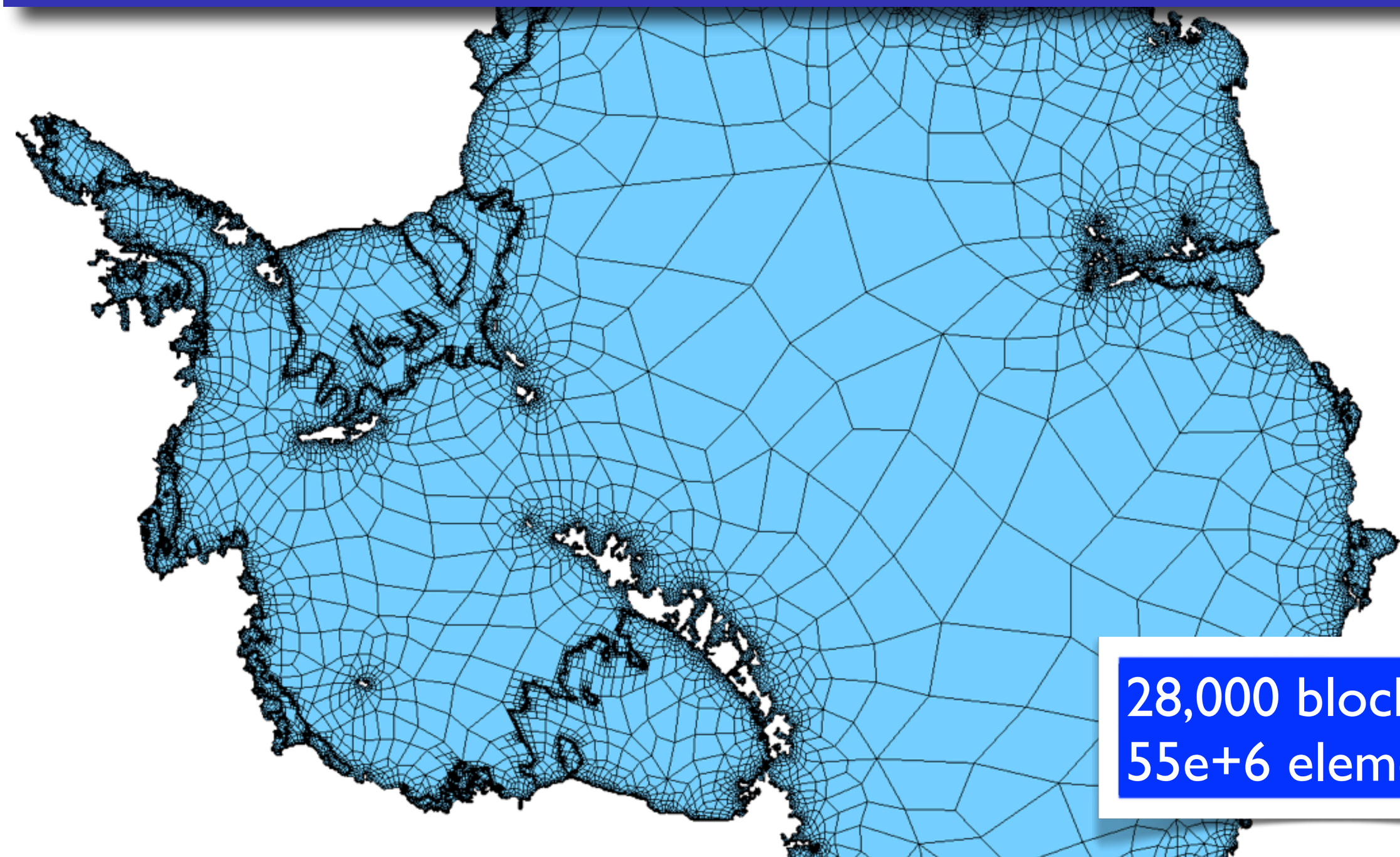




High scalability is achieved while preserving data locality by using space-filling curves.

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

Multi-block support in p4est



28,000 blocks
55e+6 elements

Antarctic ice sheet modeling (Tobin Isaac, C Burstedde)

A hybrid approach to AMR

- Use p4est to manage the parallel multiblock of quad- or octree whose leaves are non-overlapping, fixed size grids.
- p4est guides parallel transfers,
- p4est provides information about face orientations in the multiblock setting
- 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 MOL solvers
- Quad/Octree is scalable to thousands of processors

Similar approaches

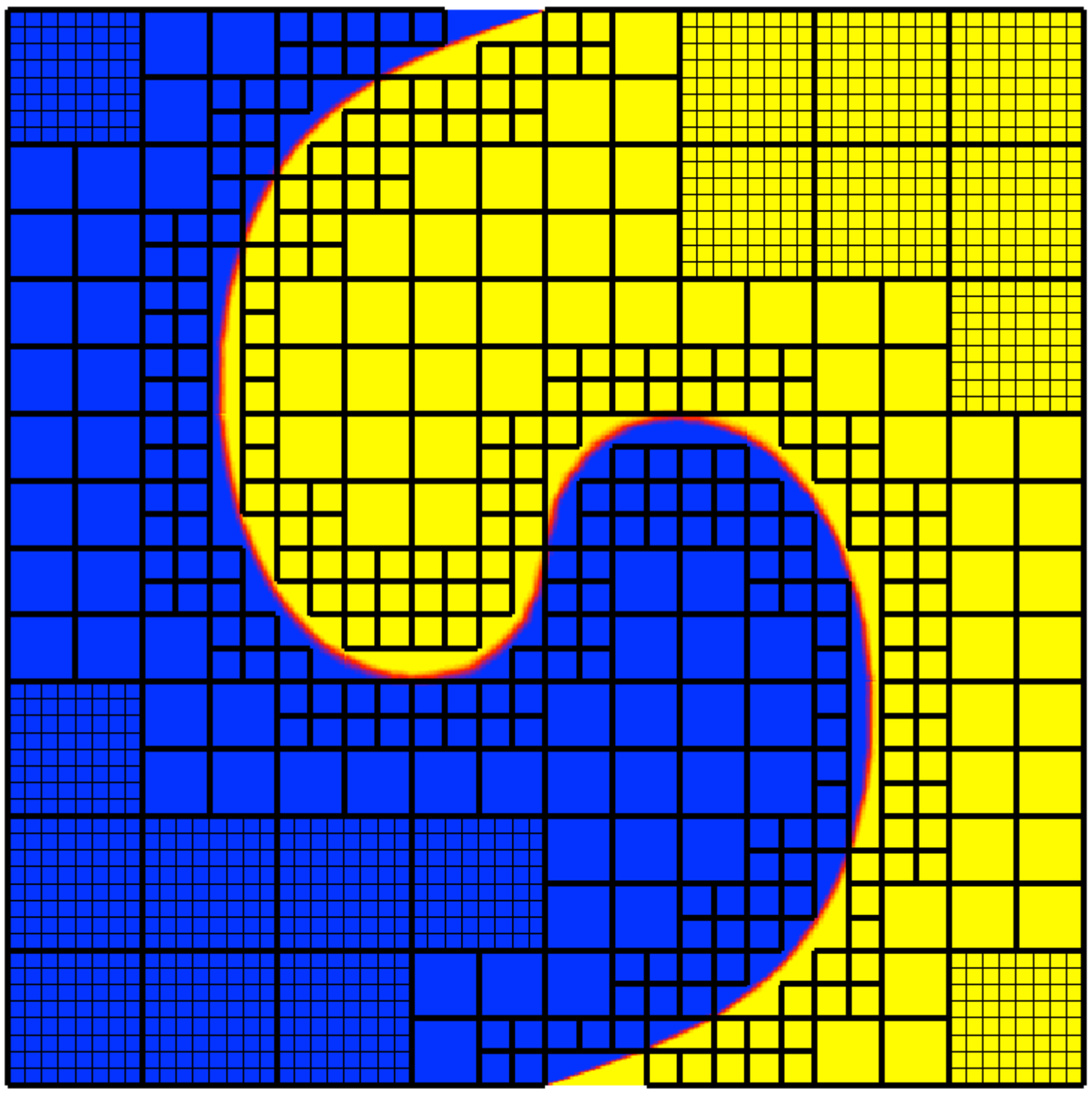
Similar approaches based on fixed size grids and/or octrees...

- "Building Cubes Method" (Sasaki, Akahito, Yamazaki, ...)
- Parallel adaptive methods for weather prediction C. Jablonowski, Oehmke, Stout and others
- NIRVANA (U. Ziegler)
- Racoon II (J. Dreher)
- Block-structured AMR codes (Chombo, Boxlib, AMRClaw, SAMRAI, AMROC, Agrif, ...) could probably be run in an octree-like mode.

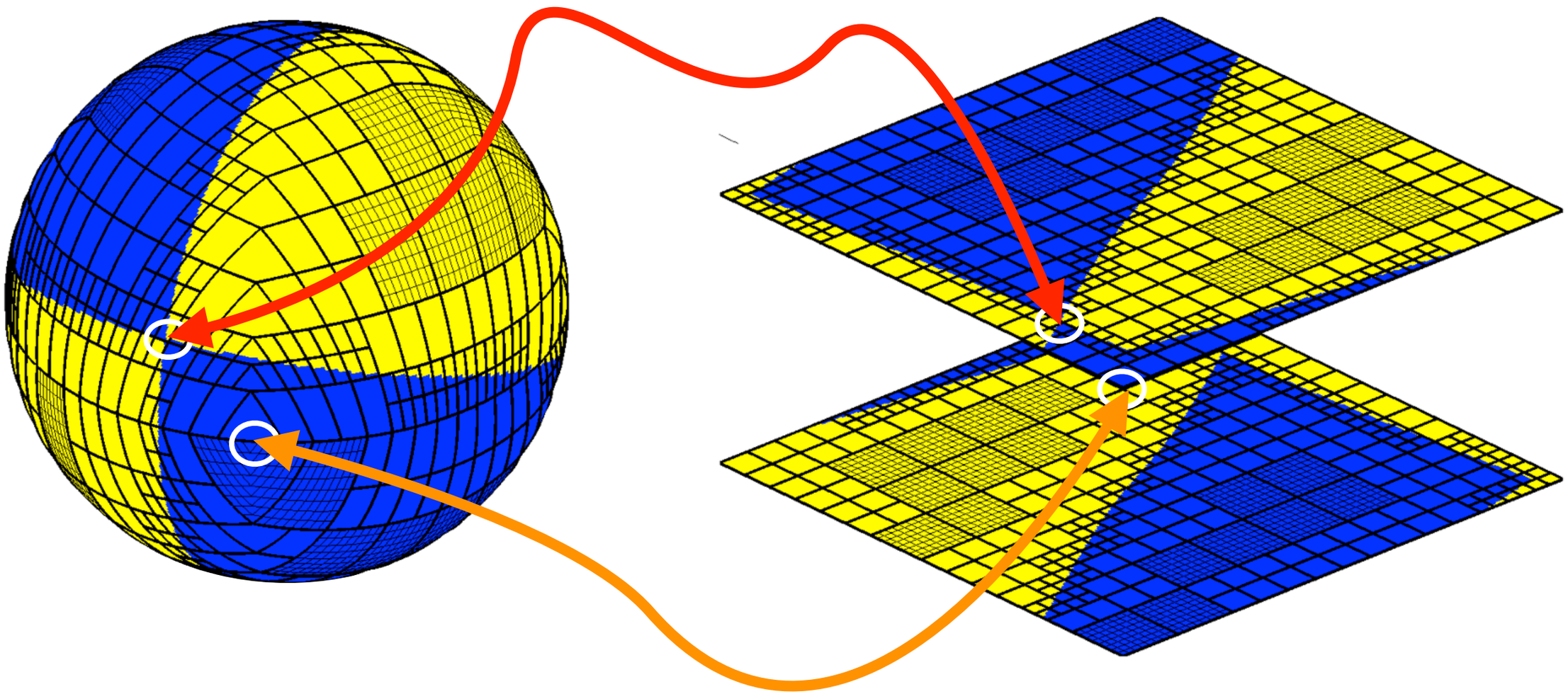
None of these codes handle general multiblock domains

Scalar advection

Scalar advection

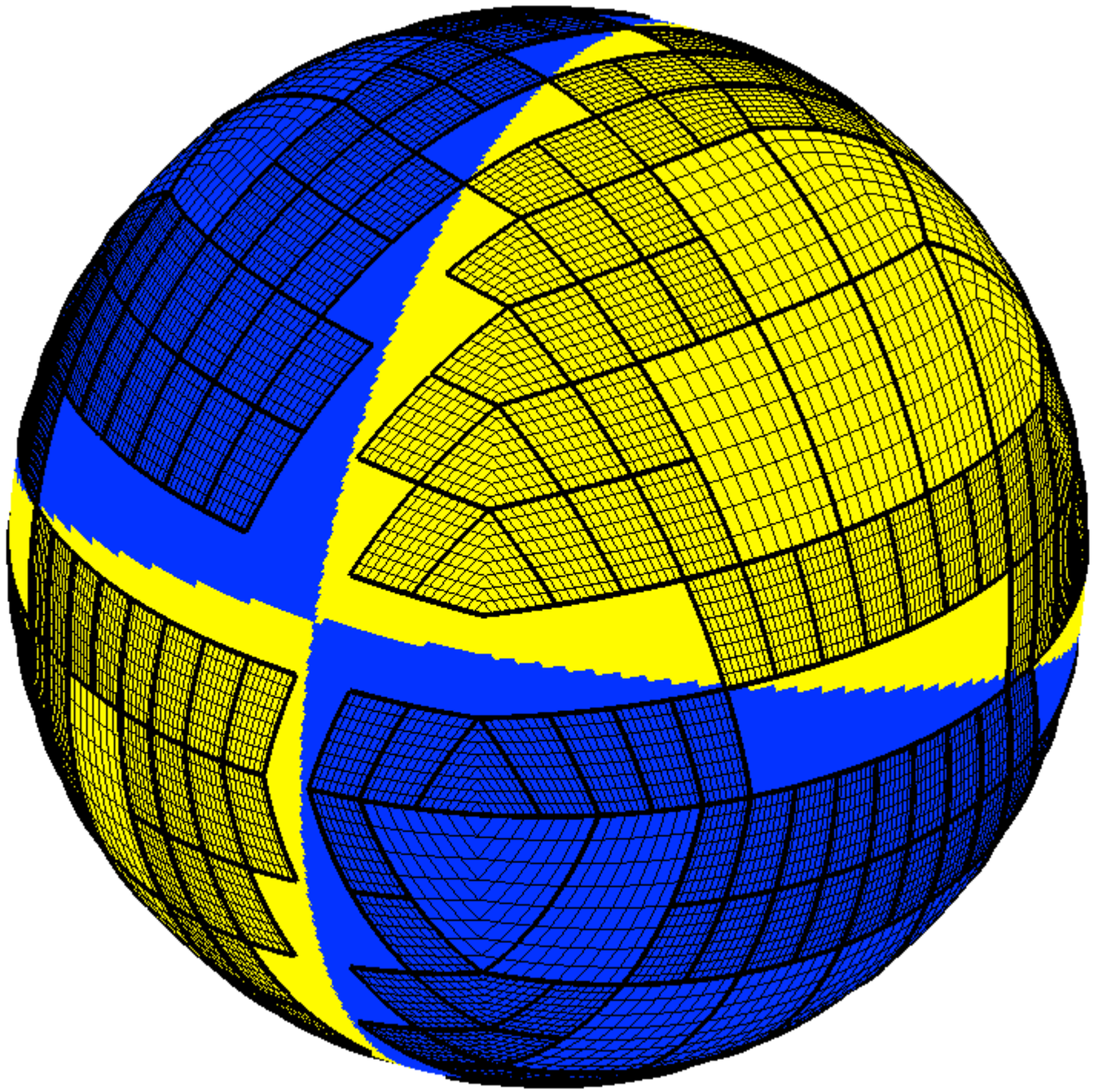


Scalar transport on the sphere

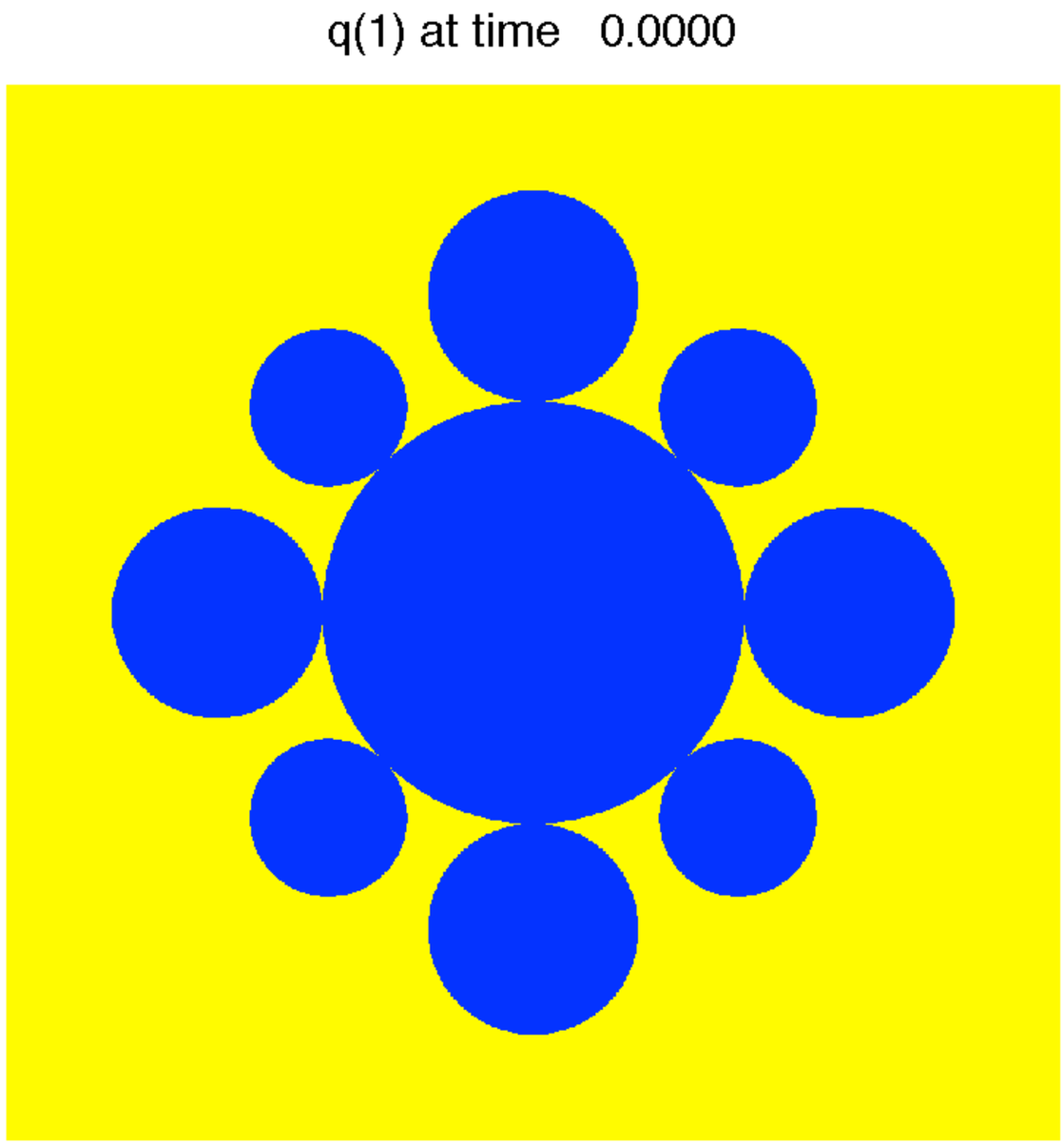


Scalar advection

Scalar advection



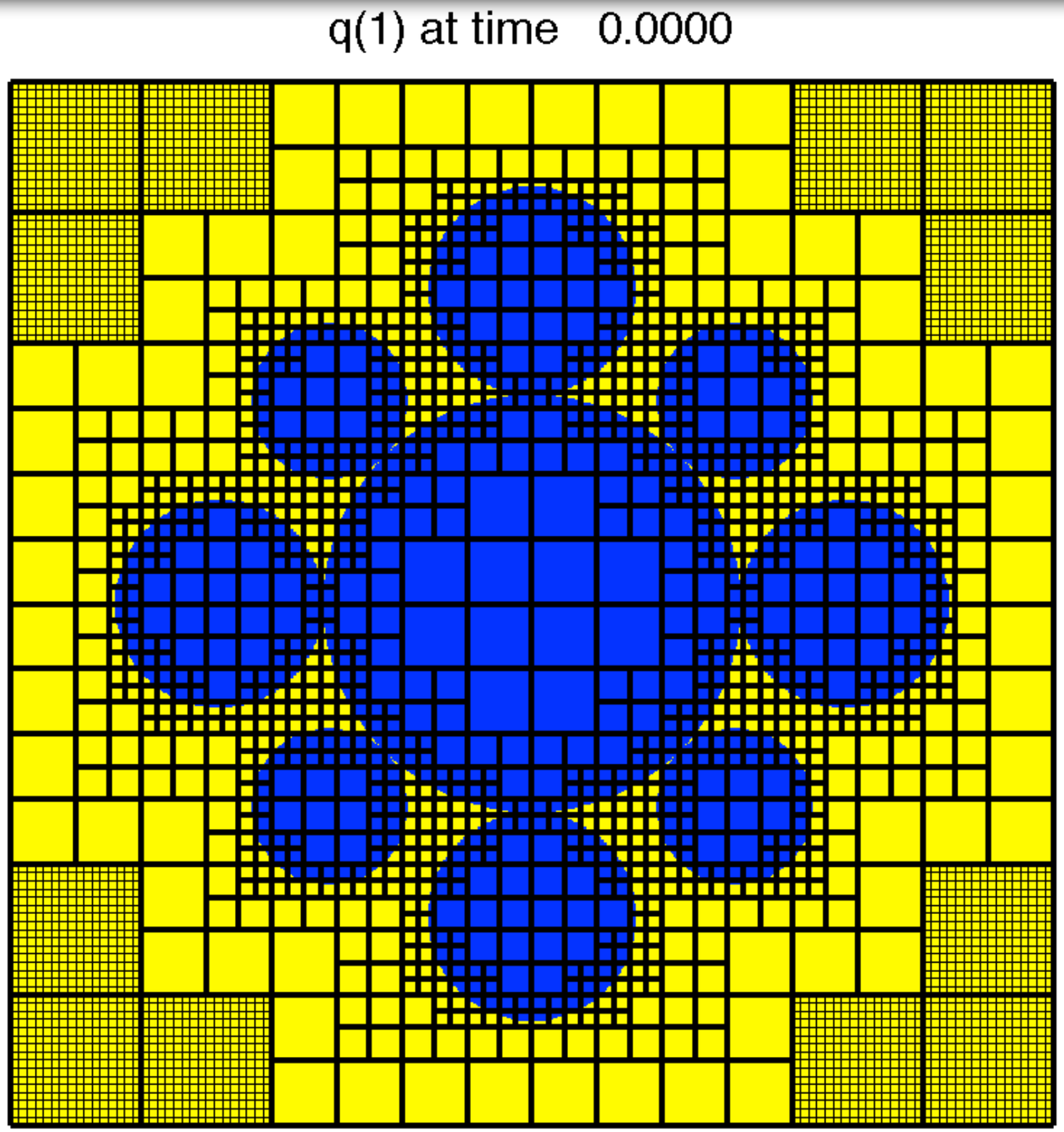
Flow by mean curvature



Time stepping done using stabilized RKC method

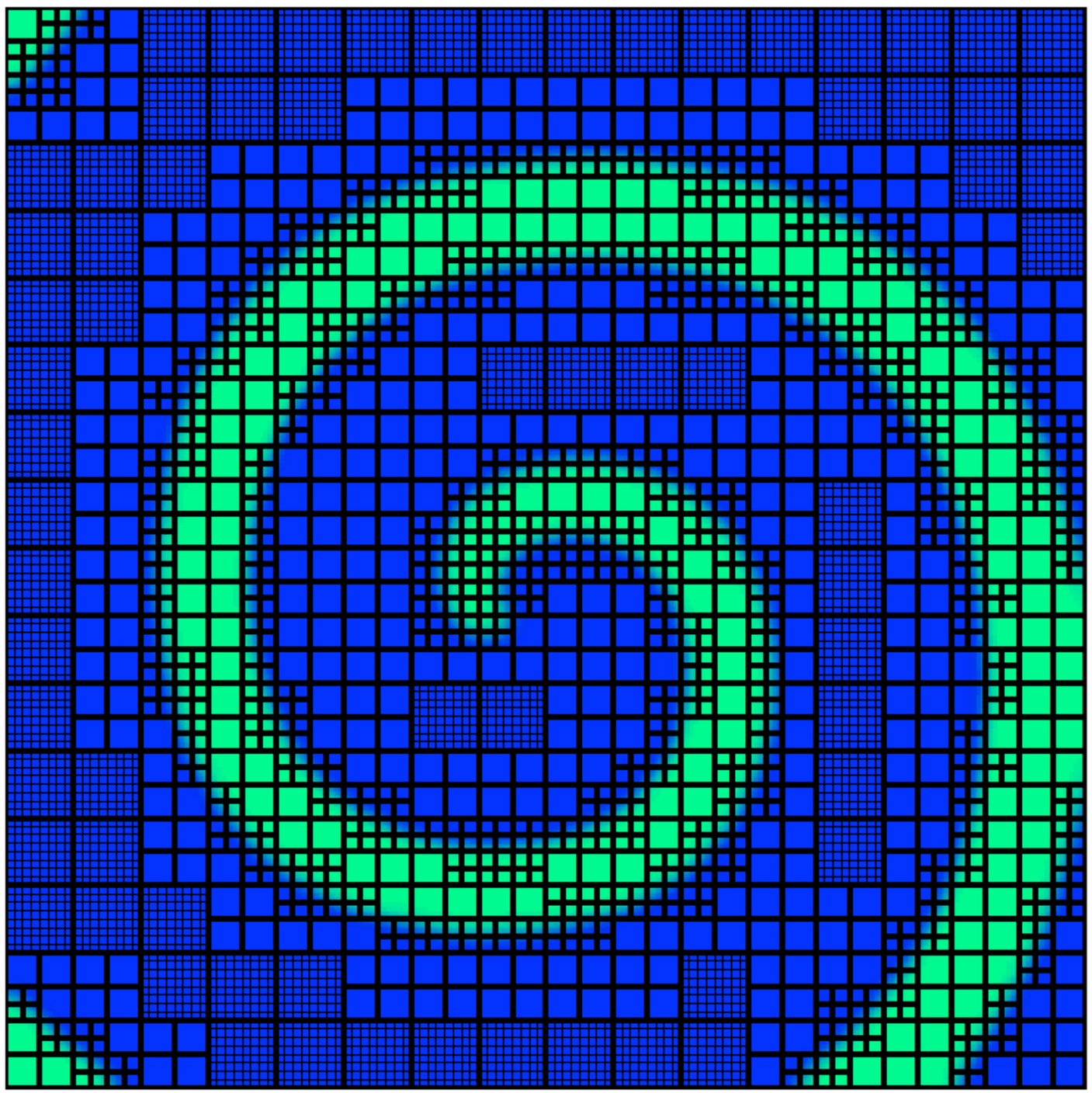
Flow by mean curvature

Flow by mean curvature



Spiral waves (Barkely model)

Spiral waves (Barkely model)



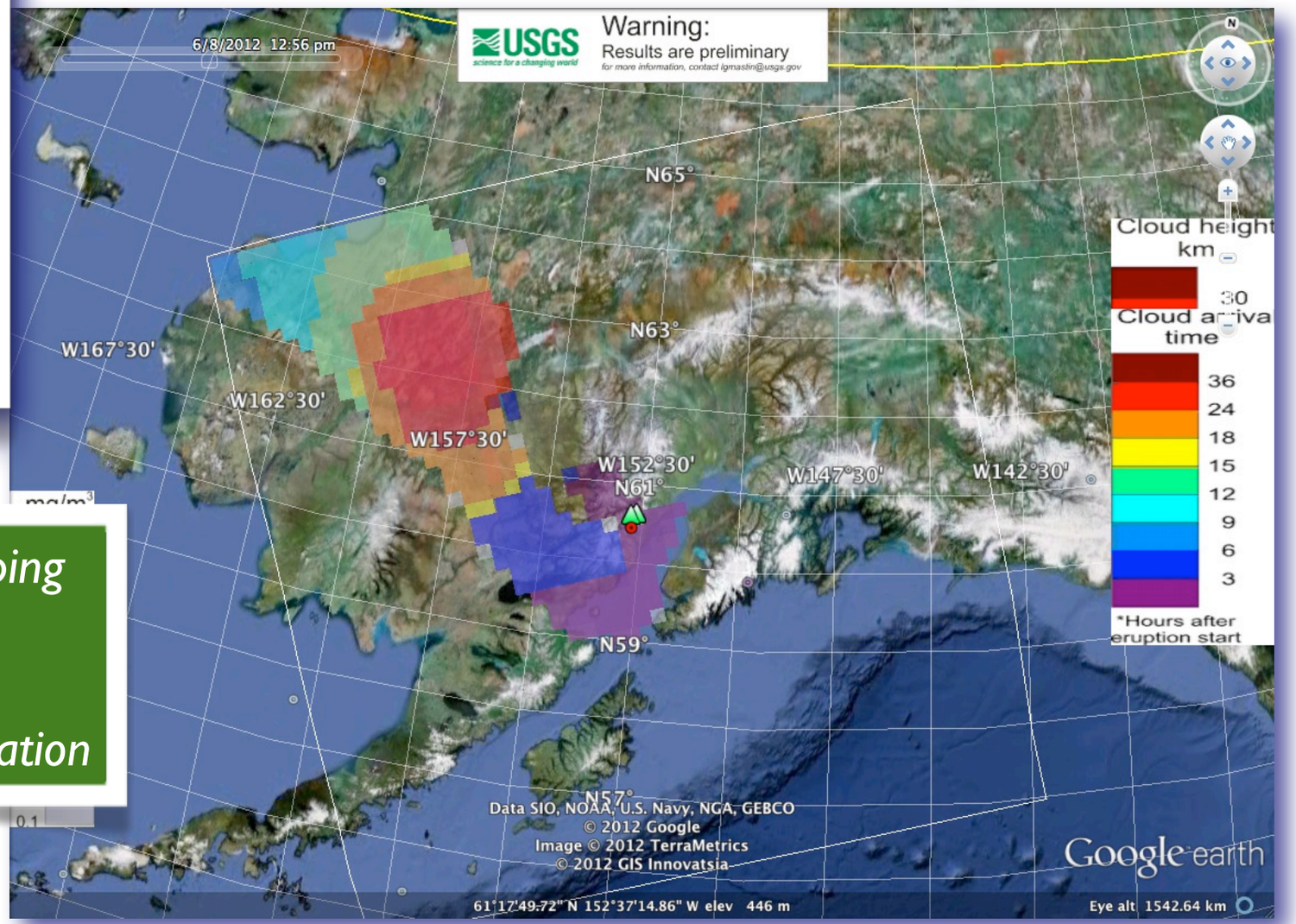
Future ...

- Multi-rate time stepping for multi-stage time stepping schemes
- Extensions to 3d
- Efficient parallel solvers for elliptic problems
- Implementation of Discrete Duality Finite Volume (DDFV) schemes for elliptic equations on mapped grids,
- Implementation of exact far-field boundary conditions for elliptic and parabolic equations,
- Assessement of performance

see http://www.forestclaw.org

Ash cloud modeling

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)