Block-structured adaptive mesh refinement for finite volume methods on Cartesian grids

Donna Calhoun (Boise State University)

Carsten Burstedde, Univ. of Bonn, Germany

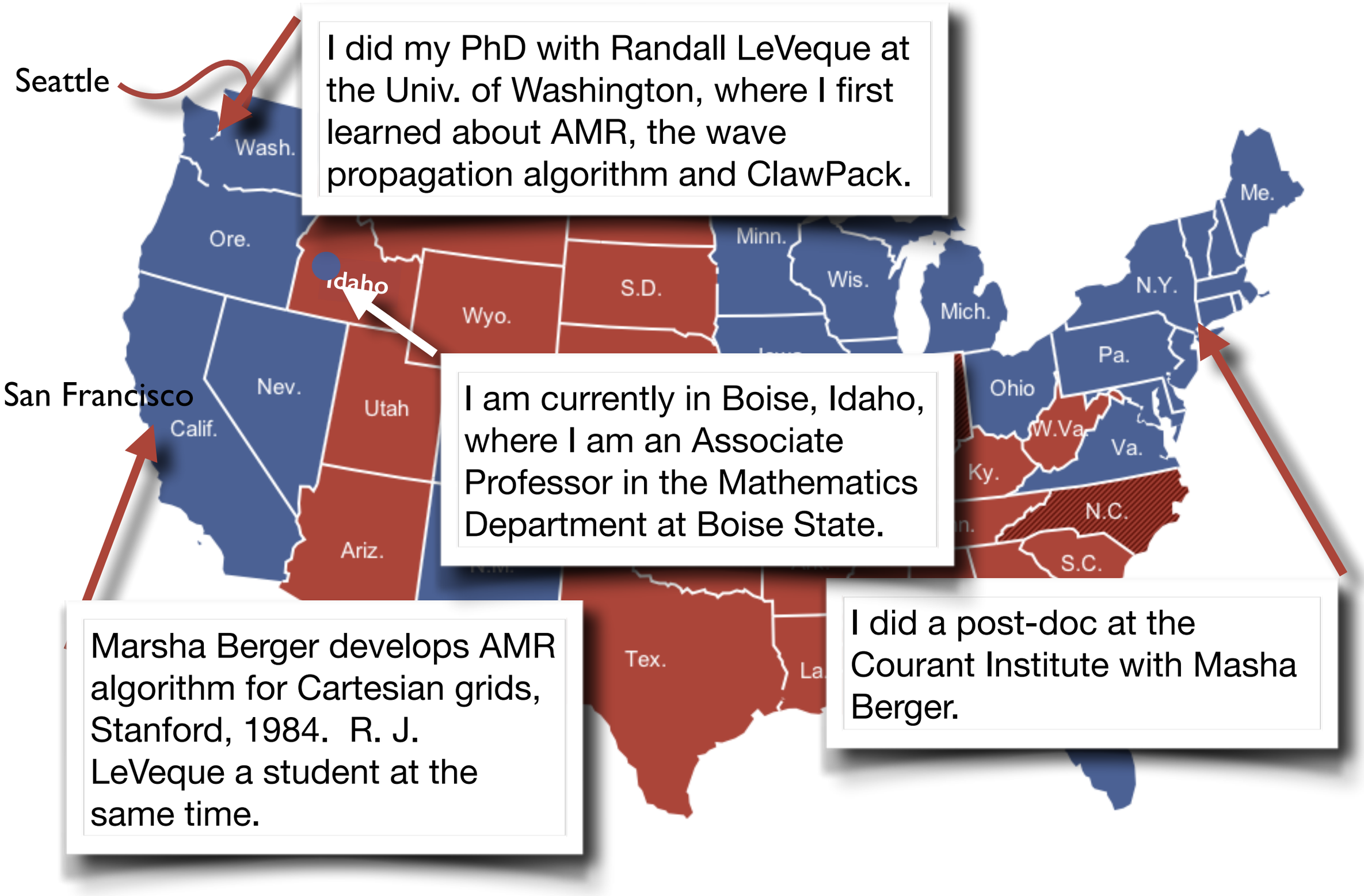
Other collaborators : M. Shih (NYU); S. Aiton (BSU); X. Qin (Univ. of Washington); R. J. LeVeque (Univ. of Washington); K. Mandli (Columbia University) and many others.

p4est Summer School

July 20 - 25, 2020

Bonn, Germany (Virtual)

My (brief) AMR story



I did my PhD with Randall LeVeque at the Univ. of Washington, where I first learned about AMR, the wave propagation algorithm and ClawPack.

I am currently in Boise, Idaho, where I am an Associate Professor in the Mathematics Department at Boise State.

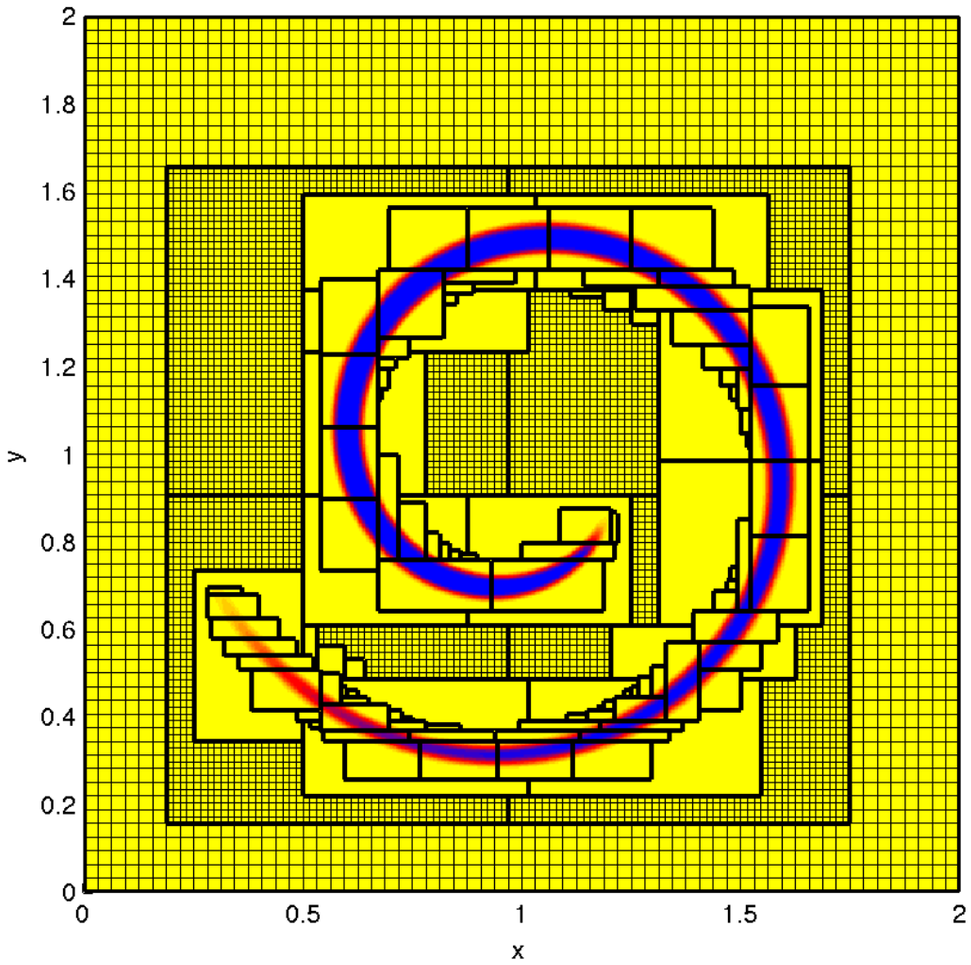
Marsha Berger develops AMR algorithm for Cartesian grids, Stanford, 1984. R. J. LeVeque a student at the same time.

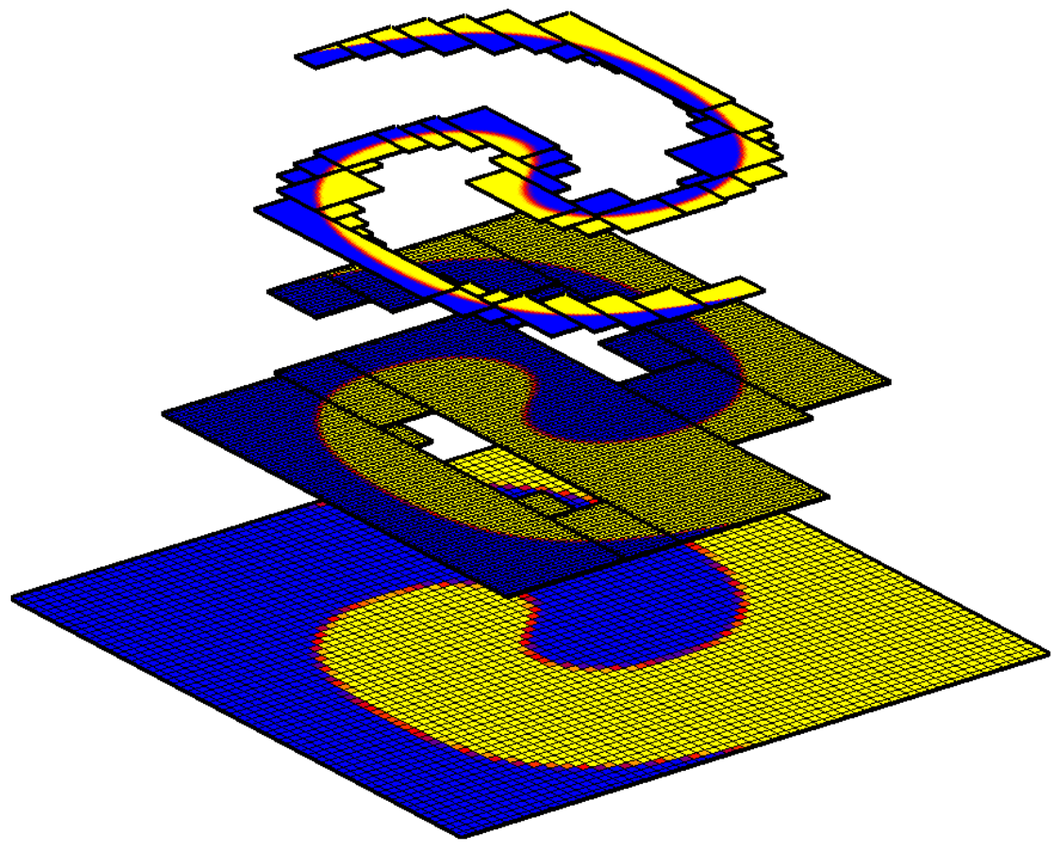
I did a post-doc at the Courant Institute with Masha Berger.

Adaptive Mesh Refinement (AMR)

Overlapping patch-based AMR (Structured AMR or SAMR)

Original approach (Berger, 1984)





Codes : Chombo (LBL), AMRClaw and GeoClaw (UW, NYU) , AMReX (LBL), SAMRAI (LLNL), AMROC (Univ. of South Hampton); Uintah (Univ. of Utah) and many others

See my website for extensive list of patch-based AMR codes

Original “Adaptive Mesh Refinement”

M. Berger, 1984 thesis

- Idea was to leverage **existing solvers** for Cartesian, finite volume meshes in a **multi-level hierarchy** of overlapping Cartesian grid patches,
- The multi-level mesh **dynamically evolved** with solution features of interest,
- **Adaptive time stepping** included in earliest AMR algorithms,
- Early numerical methods are **explicit finite volume solvers** for hyperbolic conservation laws : Piecewise-Parabolic Method (PPM, P. Colella), wave-propagation algorithms (Clawpack, R. J. LeVeque), MUSCL schemes (van Leer),
- Original applications included **shock hydrodynamics** and **weather.**

Patch-based AMR

Features :

- Finer patches overlap coarser patches, and the solution exists at each refinement level
- Buffer cells built into each grid prevent the solution features of interest from running off the finest levels
- Efficient Berger-Rigoustos algorithm allows for dynamic regridding, done every 2-3 time steps
- Maintaining conservation (“hanging node problem”) at the coarse/fine boundary was an early technical challenge.
- Averaging and interpolation (using limiters) used to communicate the solution between levels; ghost cells used to communicate between grids at coarse/fine boundaries.
- Adaptive time stepping maintains constant CFL across levels

Advantages :

- Patches can be of arbitrary size (may lead to more flexible meshing)
- Refinement factors can be 2,4,6, 12, 30, ... (not just limited to 2)

Patch-based AMR

Hyperbolic conservation laws with explicit, adaptive time stepping

- 1985 : M. Berger and J. Oliger, "Adaptive mesh refinement for **hyperbolic** partial differential equations" (from M. Berger's thesis at Stanford)
- 1989 : M. Berger and P. Colella, "Local adaptive mesh refinement for **shock hydrodynamics**" (NYU + Berkeley effort)
- 1987 : W. Skamarock, "Adaptive Grid Refinement for Numerical Weather Prediction" (another Stanford thesis; uses M. Berger's code)
- 1991 : M. Berger and I. Rigoustos, "An algorithm for point clustering and grid generation" - NYU
- 1991 : A. Almgren - "A fast adaptive vortex method using local corrections" (thesis; Berkeley)
- 1998 : M. Berger and R. J. LeVeque publish "Adaptive mesh refinement using wave-propagation algorithms for **hyperbolic** systems" - (AMRClaw; University of WA + NYU)

1985 ⟷ 1998

* incomplete and lab biased

Patch-based AMR

Elliptic solvers, Navier-Stokes and incompressible Euler equations

- 1996 : D. Martin and K. Cartwright write technical report "Solving Poisson's Equation using Adaptive Mesh Refinement (LBL)
- 1997 : L. Howell and J. Bell, "An Adaptive Mesh Projection Method for Viscous Incompressible Flow" (LBL)
- 1998 : A. Almgren, J. Bell, P. Colella, et al "A Conservative Adaptive Projection Method for the Variable Density Incompressible Navier-Stokes Equations" (LBL)
- 2000 : D. Martin and P. Colella, "A Cell-Centered Adaptive Projection Method for the Incompressible Euler Equations" (LBL)
- 2000 : M. Day and J. Bell, "Numerical Simulation of Laminar Reacting Flows with Complex Chemistry" (LBL)
- 2000 : J. Huang and L. Greengard, "A Fast Direct Solver for Elliptic Partial Differential Equations on Adaptively Refined Meshes" (NYU)
- 2008 : D. Martin, P. Colella, D. Graves, "A cell-centered adaptive projection method for the incompressible Navier-Stokes equations in three dimensions"

1996 ⟷ 2008

Patch-based AMR

Parallel scaling and performance

- 1999 : C. Rendelman, V. Beckner, et al, "Parallelization of Structured, Hierarchical Adaptive Mesh Refinement Algorithms" (p-Boxlib)
- 2001 : A. M. Wessink, R. D. Hornung, et al, "Large scale parallel structured AMR calculations using the SAMRAI framework" (LLNL)
- 2006 : M. Welcome, C. Rendleman, et al, "Performance Characteristics of an Adaptive Mesh Refinement Calculation on Scalar and Vector Platforms" (LBL)
- 2007 : P. Colella, J. Bell, N. Keen et al, "Performance and Scaling of Locally-Structured Grid Methods for Partial Differential Equations" (LBL)
- 2007 : T. Wen, J. Su, P. Colella, et al, "An adaptive mesh refinement benchmark for modern parallel programming languages" (LBL)
- 2010 : J. Luitjens and M. Berzins, "Improving the performance of Uintah: A large-scale adaptive meshing computational framework" (Univ. of Utah)
- Present : Exascale Computing Project (DOE ECP).

1999 ⟵ present

Patch-based AMR

Excellent survey of widely used codes currently available (Chombo, Cactus, Boxlib (now AMReX), Uintah, FLASH)

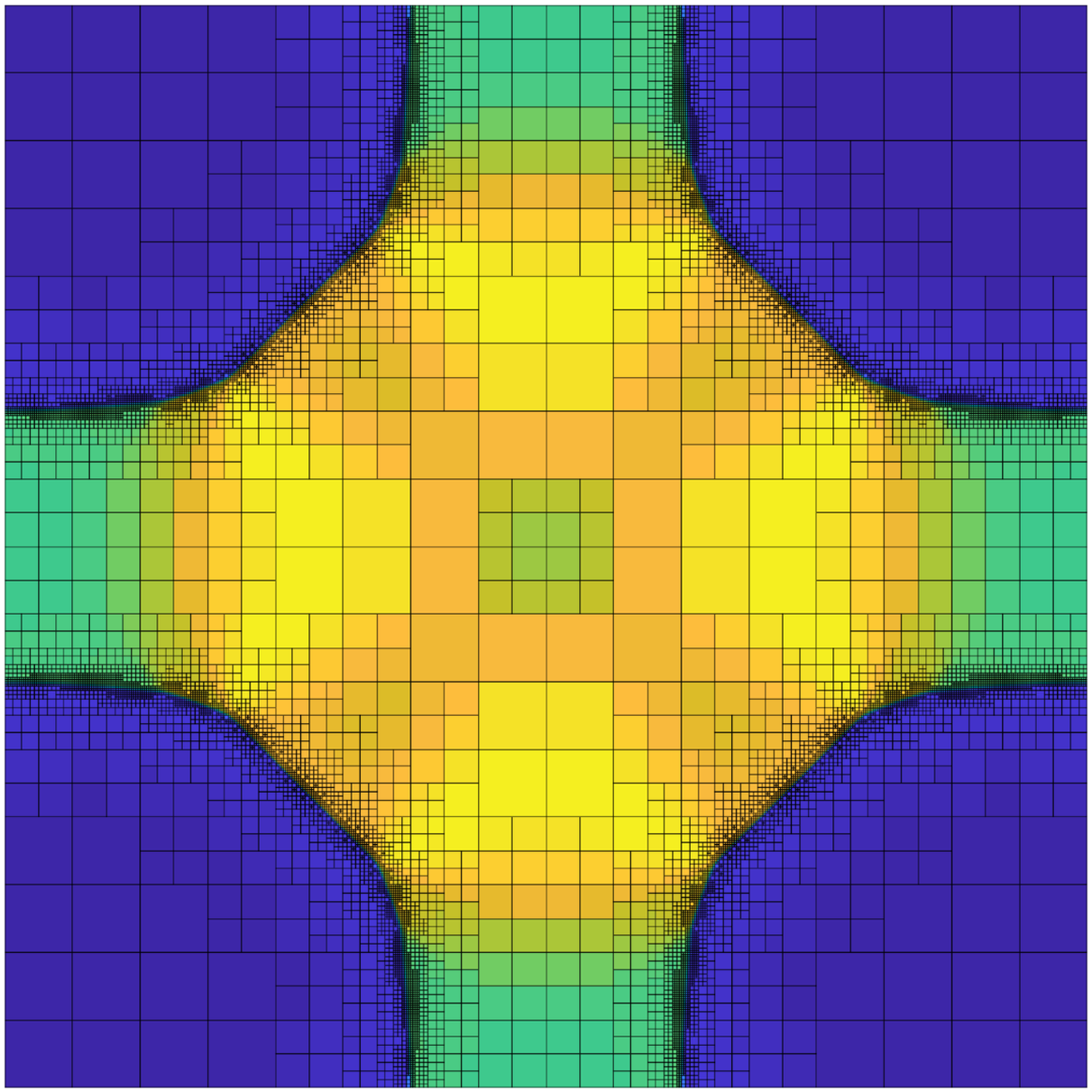
A. Dubey, A. Almgren, J. B. Bell, M. Berzins, S. Brandt, G. Bryan, P. Colella, D. Graves, M. Lijewski, F. Laeffler, B. O'Shea, E. Schnetter, B. V. Straalen, and K. Weide, "A survey of high level frameworks in block-structured adaptive mesh refinement packages", *Journal of Parallel and Distributed Computing*, (2014).

AMReX code (used in DOE Exascale Computing Project (ECP))

W. Zhang, A. Almgren, V. Beckner, J. Bell, J. Blaschke, C. Chan, M. Day, B. Friesen, K. Gott, D. Graves, M. P. Katz, A. Myers, T. Nguyen, A. Nonaka, M. Rosso, S. Williams, and M. Zingale, AMReX: a framework for block-structured adaptive mesh refinement, J. Open Source Software, 4 (2019). https://ccse.lbl.gov/AMReX

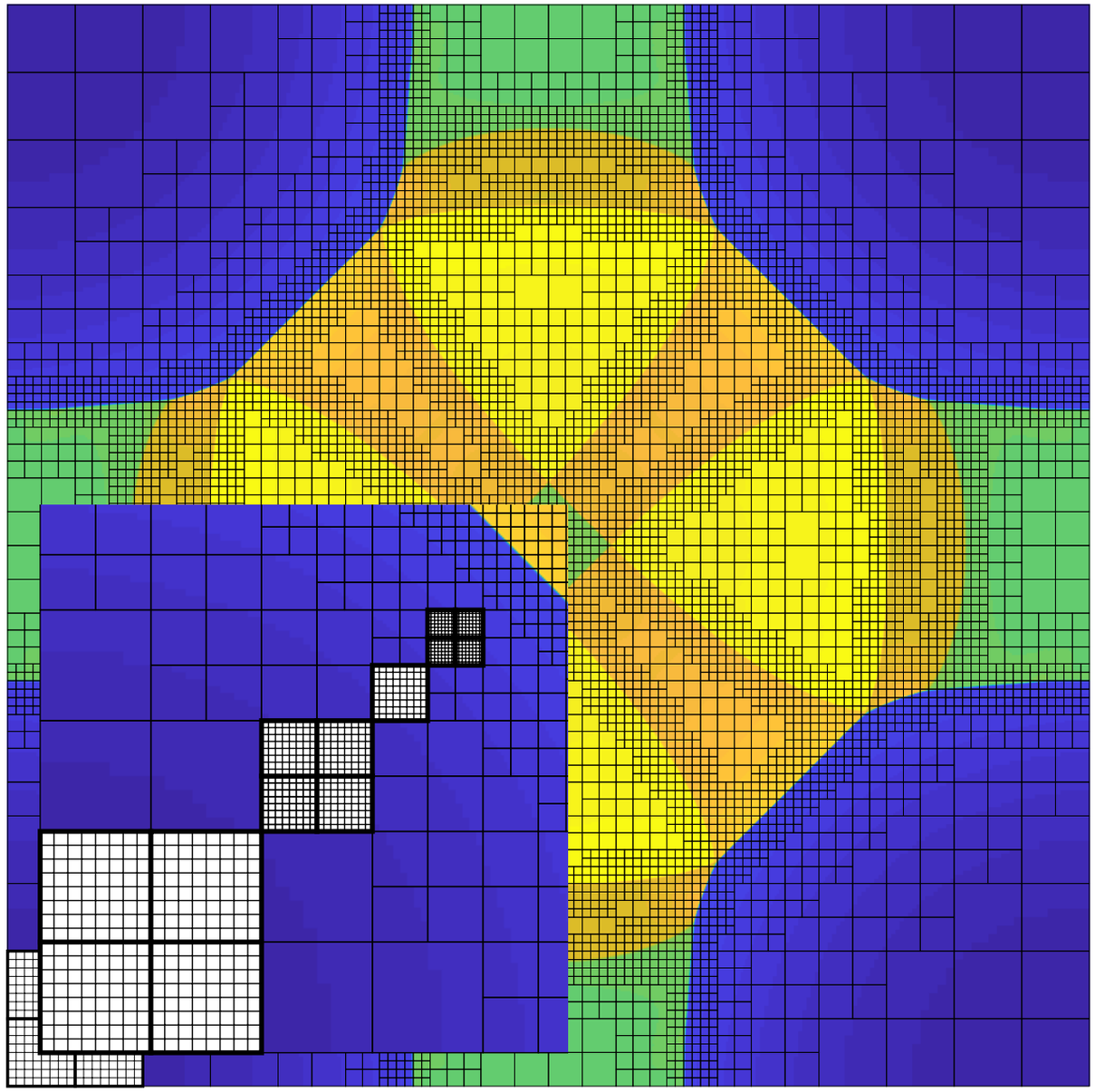
Quadtree/Octree based refinement

Cell-based refinement



Basilisk (S. Popinet) : One degree of freedom per leaf

Block-based refinement



ForestClaw (D. Calhoun) : Fixed size grid per leaf

Quadtree/Octree based refinement

"Block-based" and "cell-based" AMR

- 2000 : P. MacNiece, K. Olson et al, "**PARAMESH**: A parallel adaptive mesh refinement community toolkit" (FLASH code based on PARAMESH)
- 2002 : R. Tessyier, "Cosmology Hydrodynamics with adaptive mesh refinement. A new high resolution code called **RAMSES**" (Lausanne, Switzerland)
- 2004 : U. Ziegler, "An ADI-based adaptive mesh Poisson solver for the MHD code **NIRVANA**" (Potsdam, Germany)
- 2005 : J. Dreher and R. Grauer, "**Racoon**: A parallel mesh-adaptive framework for hyperbolic conservation laws" (Bochum, Germany)
- 2011 : C. Burstedde, L. Wilcox, O. Ghattas, "**p4est**: Scalable Algorithms for Parallel Adaptive Mesh Refinement on Forests of Octrees" (Univ. Texas)
- 2011 : K. Komatsu, T. Soga et al "Parallel processing of the **Building-Cube Method** on a GPU platform" (Tohoku, Japan)
- 2016 : S. Popinet. "**Basilisk**: simple abstractions for octree-adaptive scheme". SIAM conference on Parallel Processing for Scientific Computing, April 12-15 2016, Paris, 2016.

2000 ⟷ present

Quadtree/Octree based refinement

Advantages of tree-based meshing for AMR :

- Quadtree and octree layouts simplify development of numerical methods
- Space filling curves for load balancing make parallelization much more straightforward,
- Quad or octrees partition the domain - no overlapping patches
- Leafs of the tree can be occupied by one or more degrees of freedom.
- Potential disadvantage : Refinement is limited to factor of 2
- Elliptic problems may be harder to solve.

A enormous advantage of using tree-based meshes is that libraries exist that do just the grid management and meshing (this isn't true for patch-based codes). **p4est** is particularly well suited for scientific computing.

Hybrid idea: Use patch-based algorithms with tree-based code

ForestClaw Project

A parallel, adaptive library for logically Cartesian, mapped, multi-block domains

Features of ForestClaw include :

- Uses the **highly scalable p4est** dynamic grid management library (C. Burstedde, Univ. of Bonn, Germany)
- Each leaf of the quadtree contains a fixed, uniform grid,
- Optional multi-rate time stepping strategy,
- Has **mapped, multi-block** capabilities, (cubed-sphere, for example) to allow for flexibility in physical domains,
- **Modular design** gives user flexibility in extending ForestClaw with Cartesian grid based solvers and packages.
- Uses essentially the same numerical components as patch-based AMR (e.g. Berger-Oliger-Colella)

Thanks to NSF for supporting this work

Why use quadtree/octree approach?

Advantages of the **block-based** approach using quad/octrees:

- Regular neighbor connectivity means it is easy to implement inter-grid communication
- Non-overlapping composite grid structure is intuitive. The solution exists only on one grid, not on several layers of grids.
- Quadtree/octree well suited for emerging hardware - patches can all be processed simultaneously in CUDA blocks, for example,
- Equal size patches and space-filling curve makes load-balancing straightforward, without the need for tiling patches.
- Mesh management algorithms can be decentralized
- Very limited meta-data requirements.

What is p4est?

- Highly scalable meshing library based on quadtree/octree refinement
- Manages a “forest-of-octrees” to allow for geometrically complex domains.
- Encapsulates AMR meshing details parallel load balancing, dynamic regridding, neighbor connectivity and so on
- Principle developers are Carsten Burstedde (Univ. of Bonn, Germany), Lucas Wilcox (NPS, Monterey, CA), Tobin Isaac (Georgia Tech) and several others. Originated at Univ. of Texas, Austin.
- Key component in three Gordon Bell finalists (2008, 2010, 2012) and a Gordon Bell prize winner (2015)

What does p4est provide?

- Provides **2:1 balanced quadtree/octree** mesh based on tagged quadrants/octants
- Sets up **ghost quadrants/octants** for MPI communication
- **Transfers communication buffers** (packed by the application) to remote processors
- Handles **parallel partitioning and load balancing** using a space filling curve paradigm,
- Provides many tools for **nearest neighbor lookups** (both face neighbors and corner neighbors)
- Provides **transformations** needed to implement multi-block solvers
- Companion **sc library** provides utilities for parsing and registering input options from a configuration file or command library, and memory management utilities.

What p4est does not provide

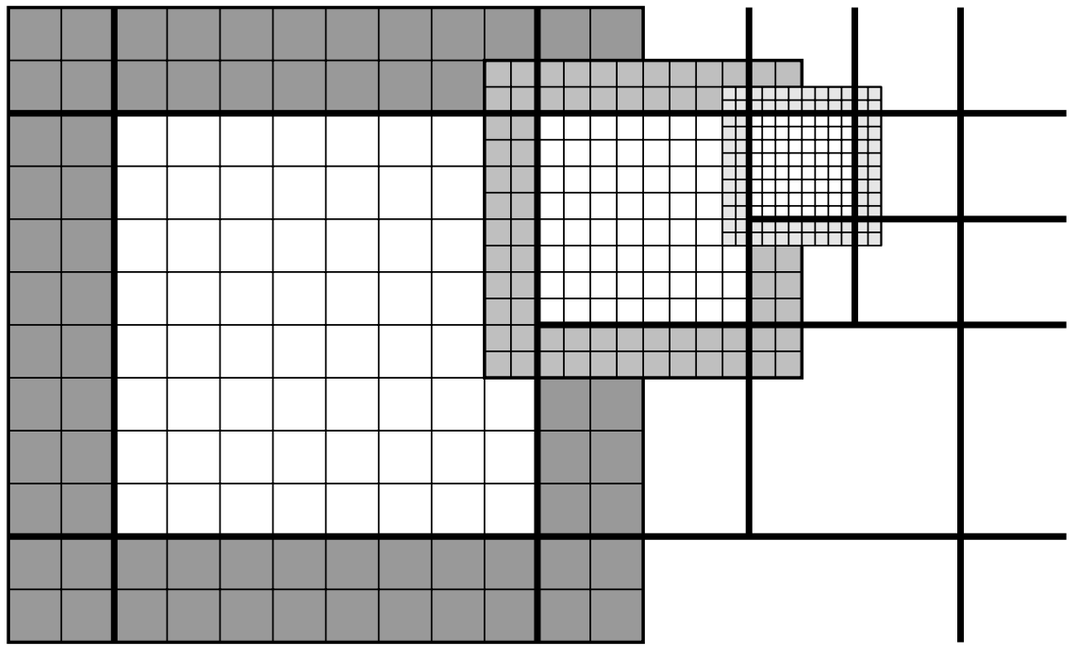
- p4est does not handle spatial discretization - user is free to fill quads/octs with any type of data (FEM/DG/FV/Cartesian grids)
- p4est does not pack and unpack data for exchange between processors
- p4est provides the user with a coarse quadrant and four refined quadrants, but it is up to the application to decide how to construct coarser or finer data from existing fine or coarse data.
- p4est does not impose any refinement criteria - the application must tag quadrants/octants for coarsening or refinement.
- p4est does not handle any time stepping - all of this must be supplied by the application

A PDE layer for applications

To use p4est to solve PDEs, a “PDE layer” is needed to handle tasks not carried out by p4est. This layer should

- Define spatial discretization,
- Manage (possibly adaptive) time stepping
- Supply refinement criteria
- Transfer solution between old and new evolving meshes
- Fill in any quadrant face and corner data at (ghost cell, halo, etc) needed so quadrants can communicate their data with each other
- Pack and unpack data for parallel exchange and load balancing
- Provide visualization
- Write output
- Post-processing diagnostics

How does ForestClaw use p4est?

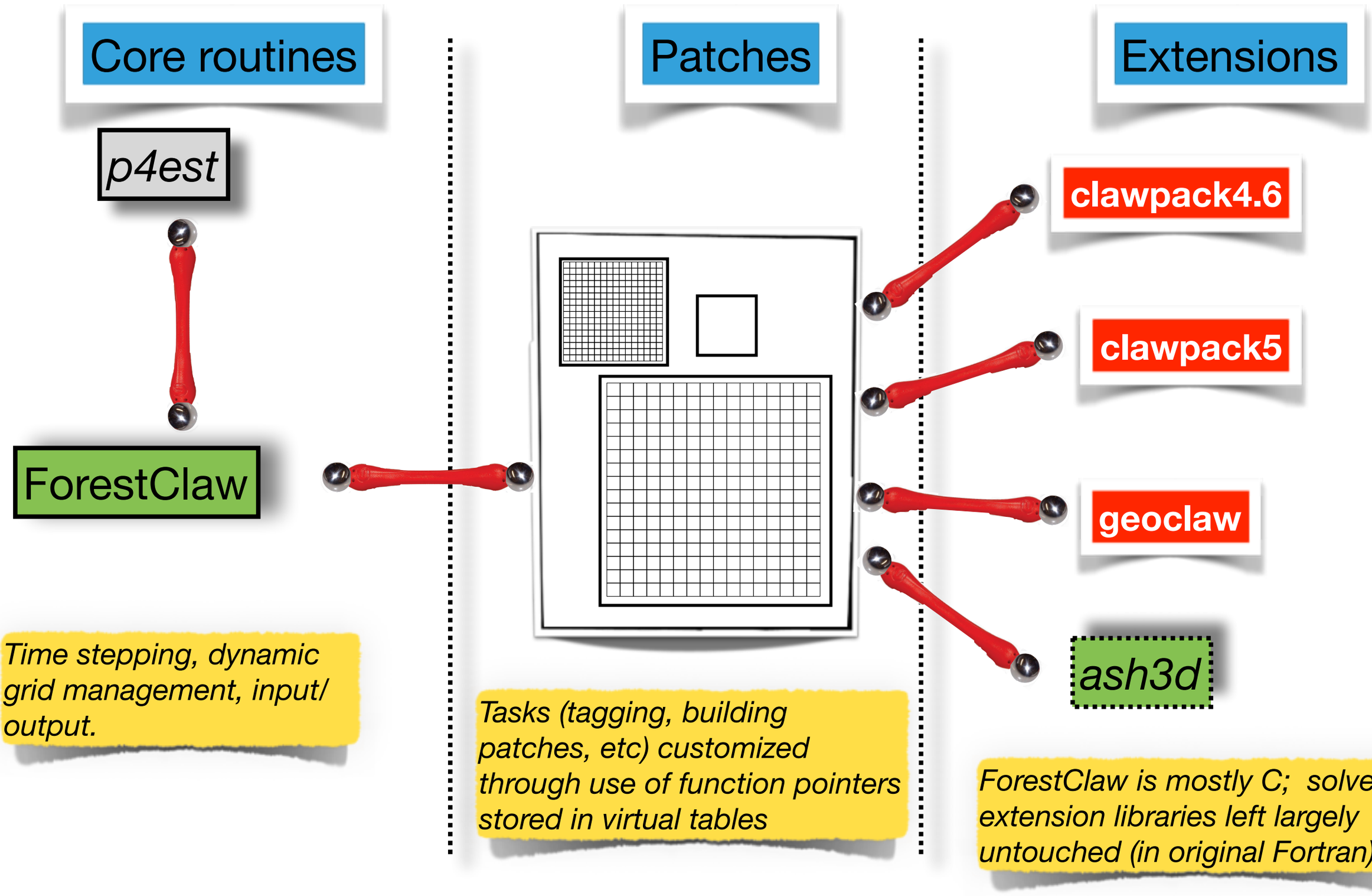


In the “clawpatch” patch (used for finite volume solvers), each p4est quadrant is occupied by a single logically Cartesian grid, stored in contiguous memory, including ghost cells.

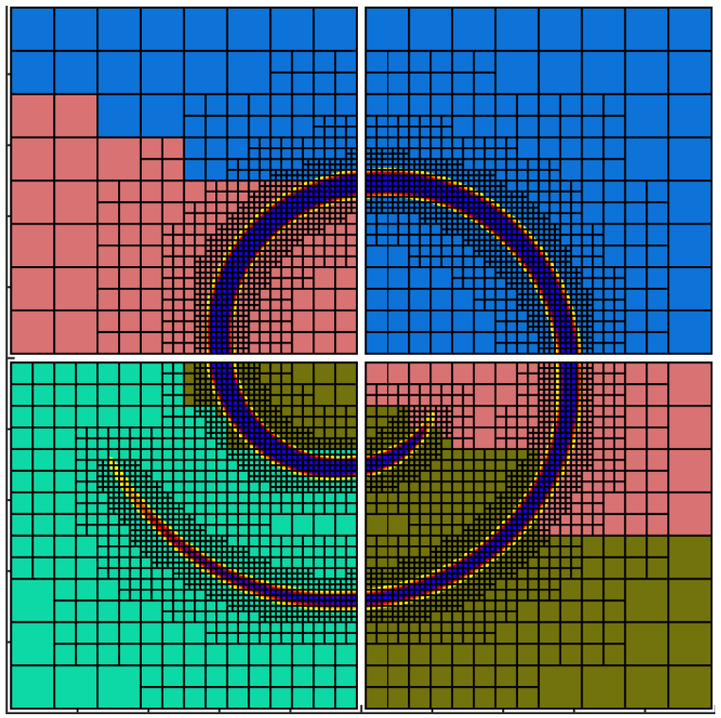
ForestClaw is a **p4est PDE layer**.

- Written mostly in object-oriented C
- Core routines are agnostic as to patch data, solvers used, etc.
- Most aspects of the PDE layer, including type of patch used, solver, interpolation and averaging, ghost-filling, can be customized
- Support for legacy codes
- Several extensions include Clawpack extension, GeoClaw, Ash3d and others.
- FV solvers and meshes are available as applications.

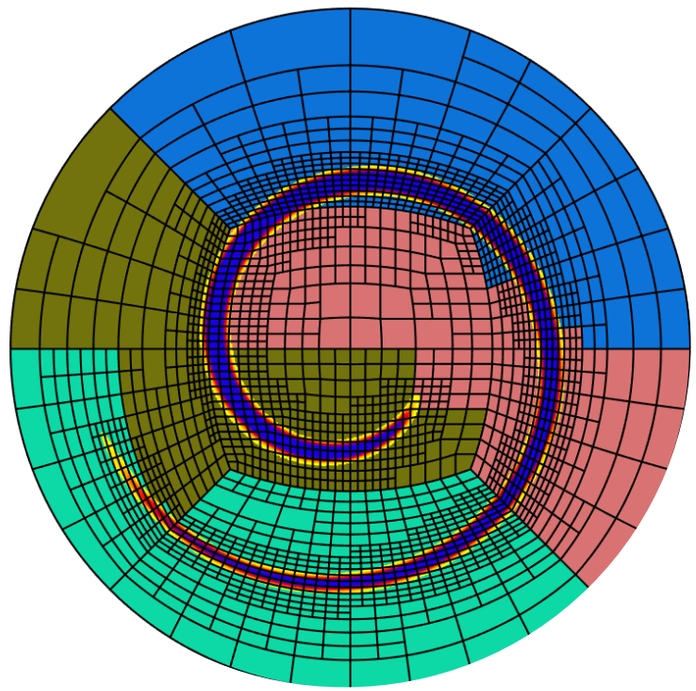
Extending ForestClaw



ForestClaw - parallel capabilities



4 block domain (4 procs)

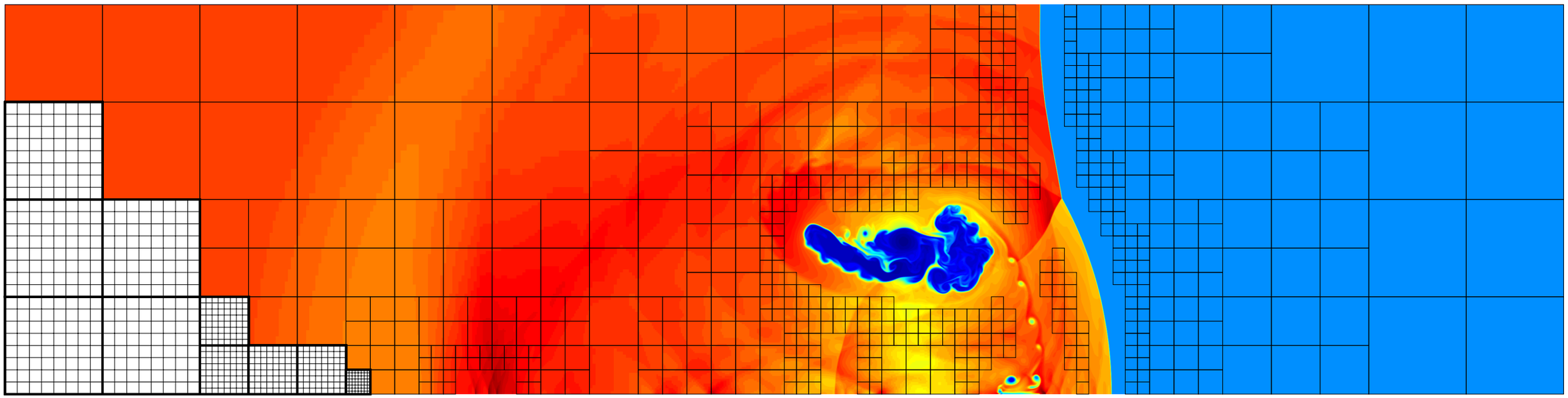


5-patch disk domain (4 procs)

Distributed (MPI) parallelism handled by p4est using space-filling-curve with Morton ordering

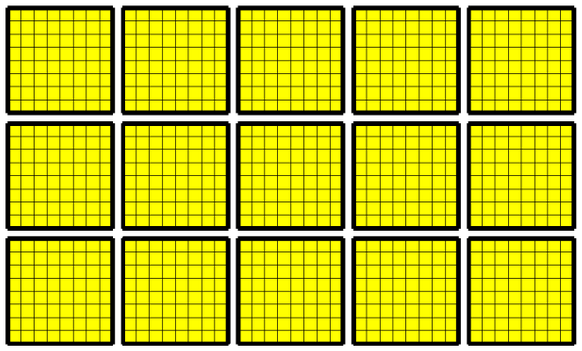
- Good scaling up to 65K cores on using 32x32 blocks on JUQUEEN (Juelich, Germany) (now de-commissioned)

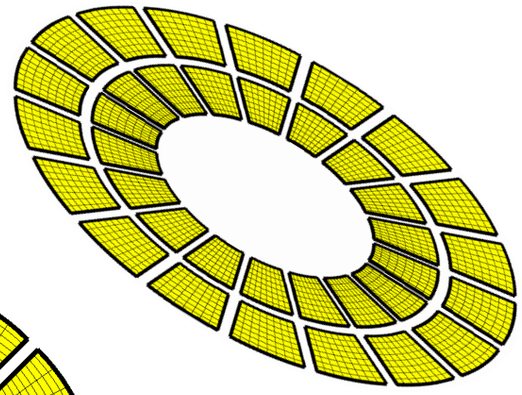
ForestClaw - multi block features

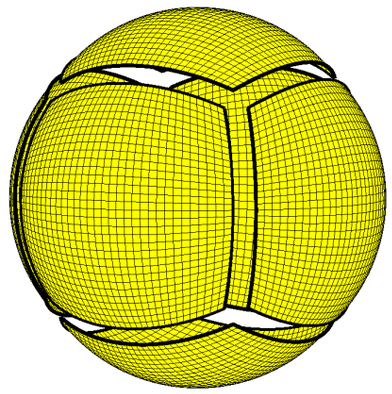


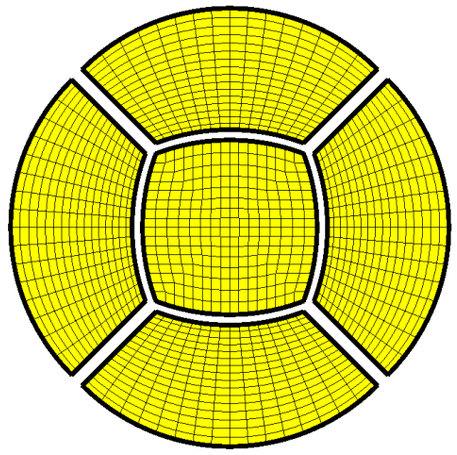
Shockbubble simulation using Clawpack (www.clawpack.org)

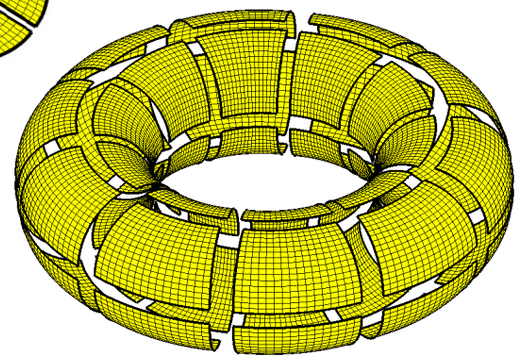
extension of ForestClaw on 4x1 multi block domain



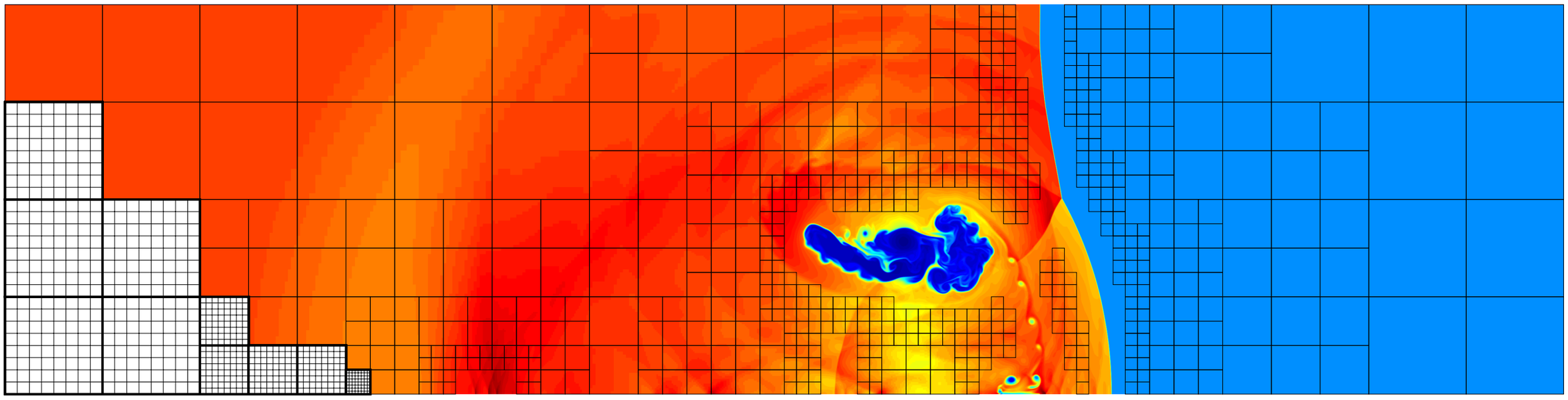






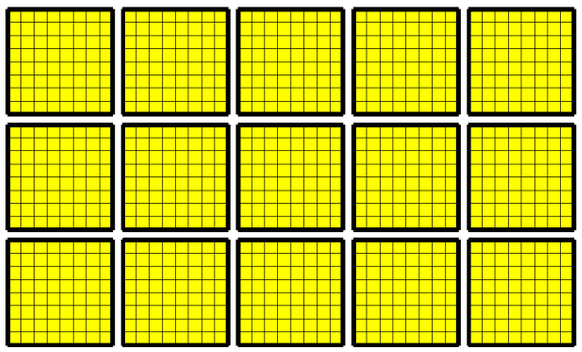


ForestClaw - multi block features

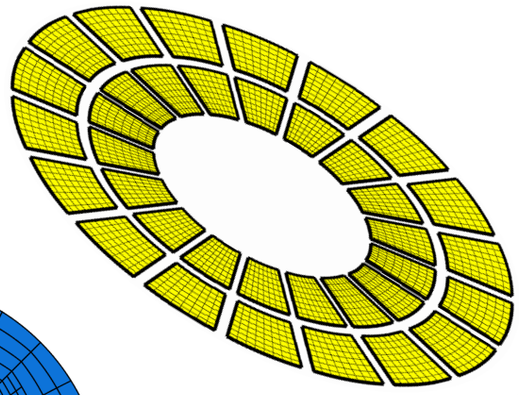


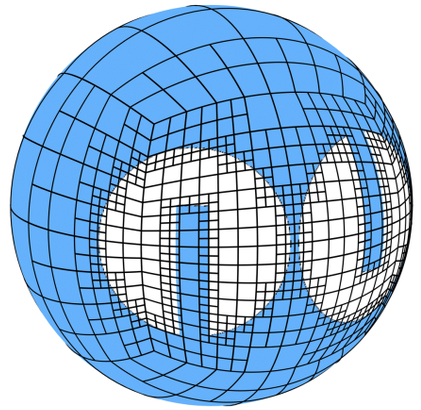
Shockbubble simulation using Clawpack (www.clawpack.org)

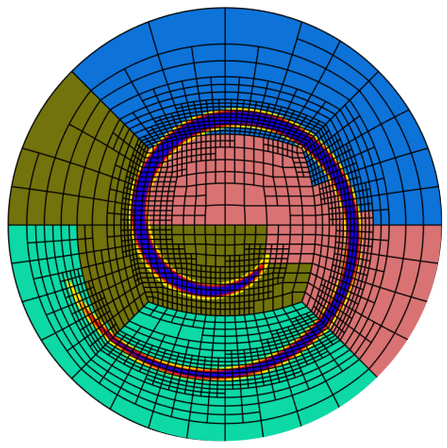
extension of ForestClaw on 4x1 multi block domain

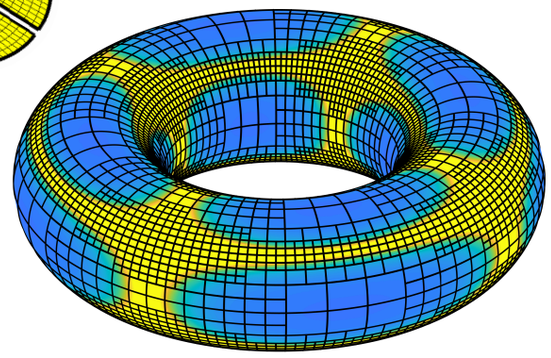


Solvers based on finite volume wave propagation algorithms in Clawpack (R. J. LeVeque)









Applications and extensions

- Most **Clawpack** examples are available (scalar advection, acoustics, Burgers, Euler equations, shallow water wave equations) (R. J. LeVeque, M. Berger, K. Mandli and many others)
- **GeoClaw** library extension for depth averaged geophysical flows (tsunamis, overland flooding, debris flows, landslides, storm surge). (www.clawpack.org) (M. Shih, D. George, R. J. LeVeque, M. Berger, and many others)
- Volcanic ash cloud modeling using USGS code **Ash3d**. (example of legacy code port.) (H. Schwaiger, DC)
- **MAGIC-Forest** (J. Snively, C. Burstedde, DC, Embry-Riddle, FL)
- **Serre-Green-Naghdi** solver (D. Chipman, S. Aiton, DC)
- **GPU solvers** for Clawpack available (5x-7x speed-up) (M. Shih, S. Aiton, X. Qin, DC)

Library extensions

ForestClaw allows for any extension library, so users can easily incorporate their own solvers into the ForestClaw PDE layer.

Core ForestClaw routines (doesn't include patch libraries)

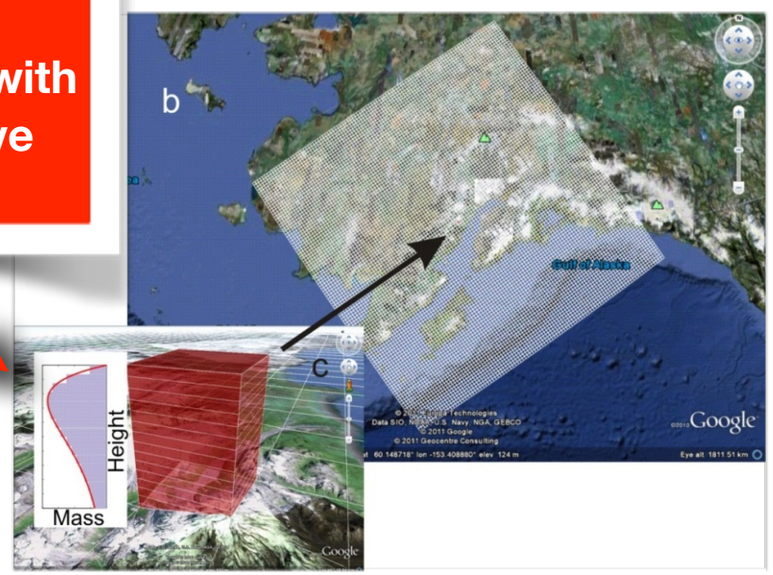
Language	files	blank	comment	code
C	40	2277	2182	9570
C/C++ Header	38	1055	1977	2574
C++	1	44	46	225
SUM:	79	3376	4205	12369

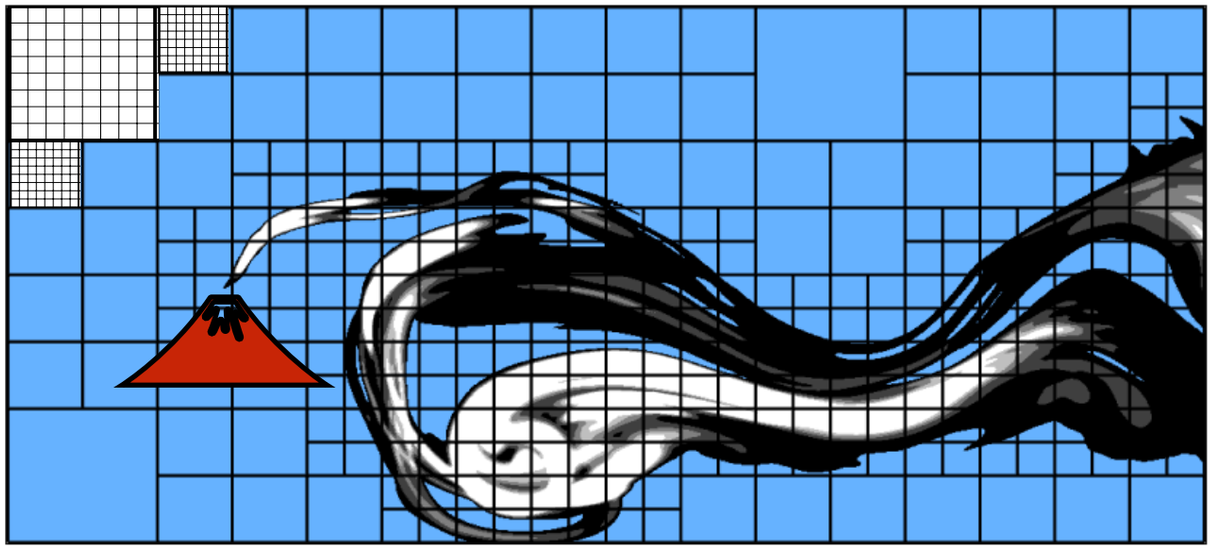
Clawpack 4.x library routines (www.clawpack.org)

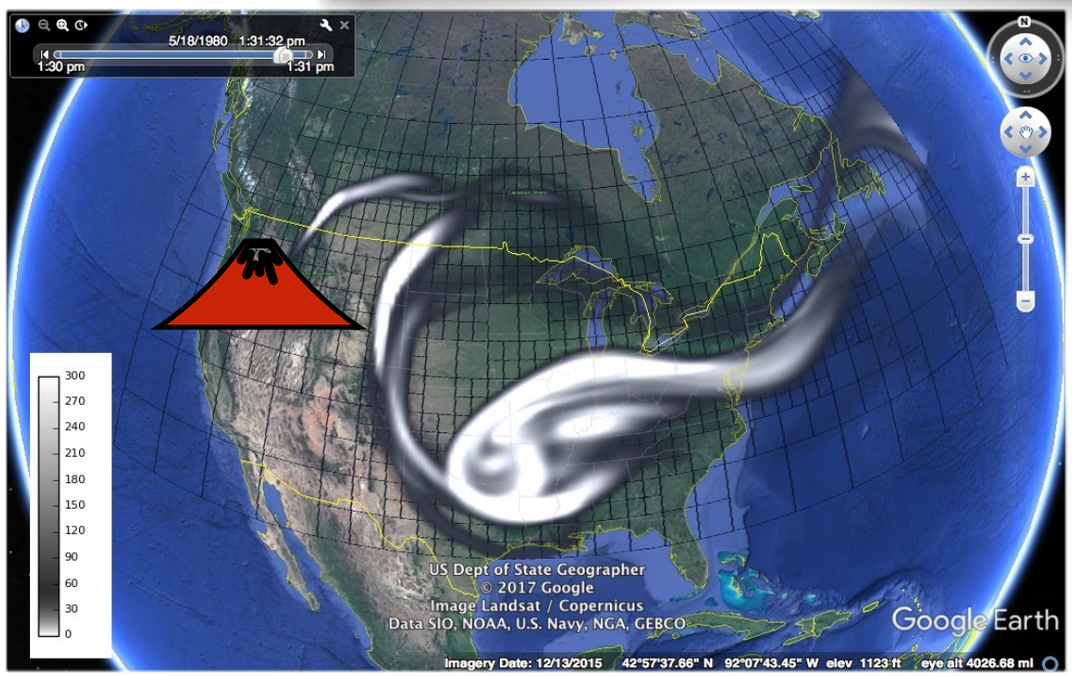
Language	files	blank	comment	code
Fortran 77	17	260	592	1472
C/C++ Header	4	144	103	425
C++	1	152	50	421
C	1	55	30	142
SUM:	23	611	775	2460

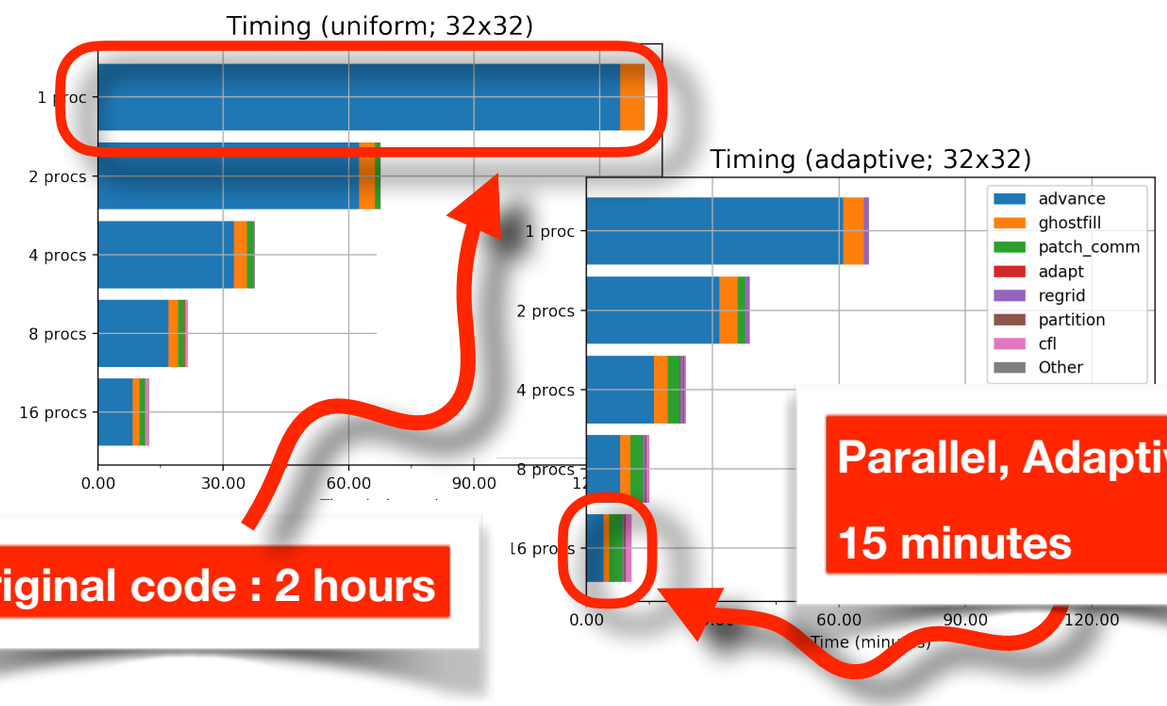
Volcanic ash transport

Extend existing volcanic ash transport model with a parallel, adaptive capabilities









Original code : 2 hours

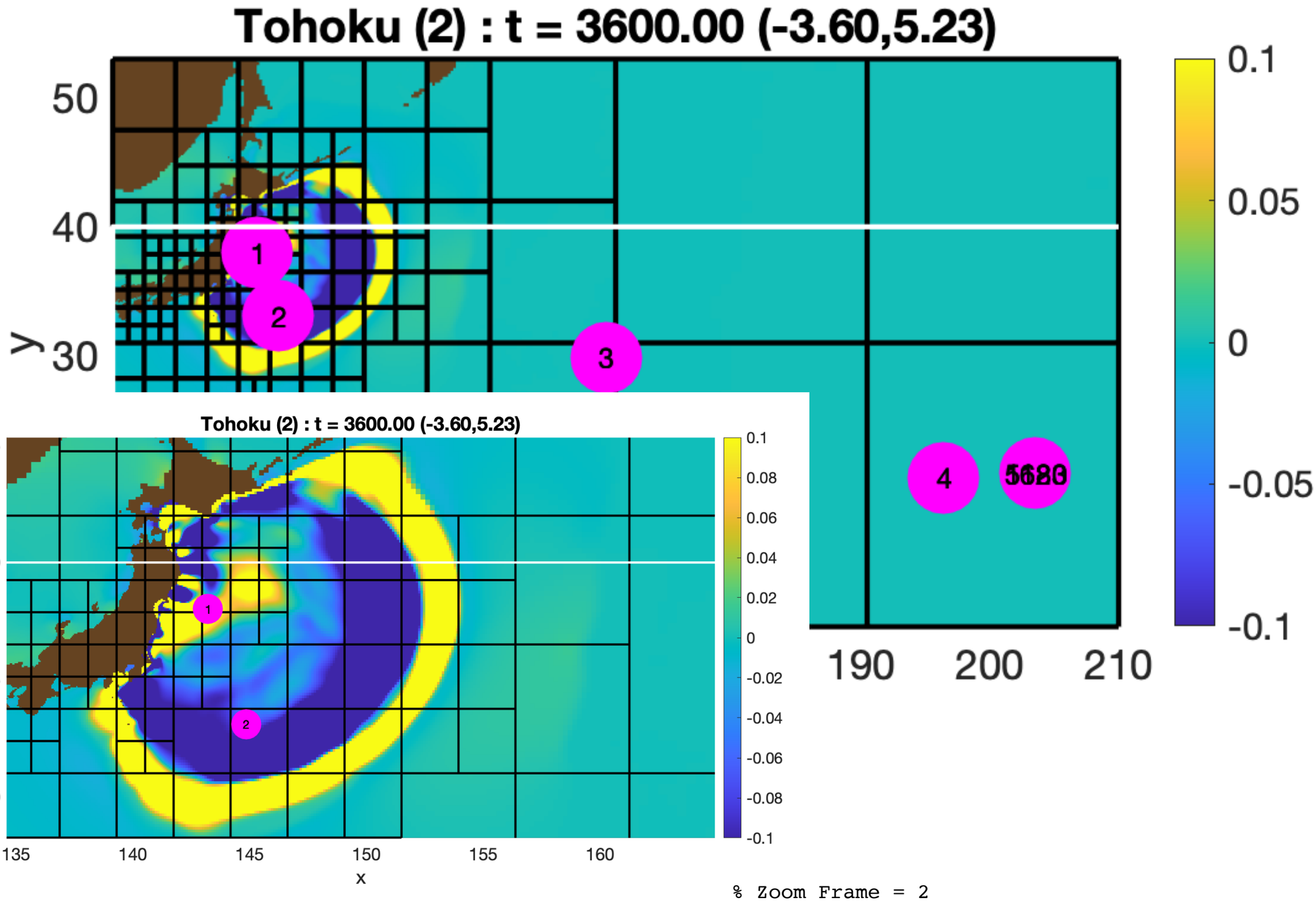
Parallel, Adaptive: 15 minutes

Volcanic ash transport using Ash3d (H. Schwaiger, USGS) extension of ForestClaw

Donna Calhoun (Boise State Univ.)

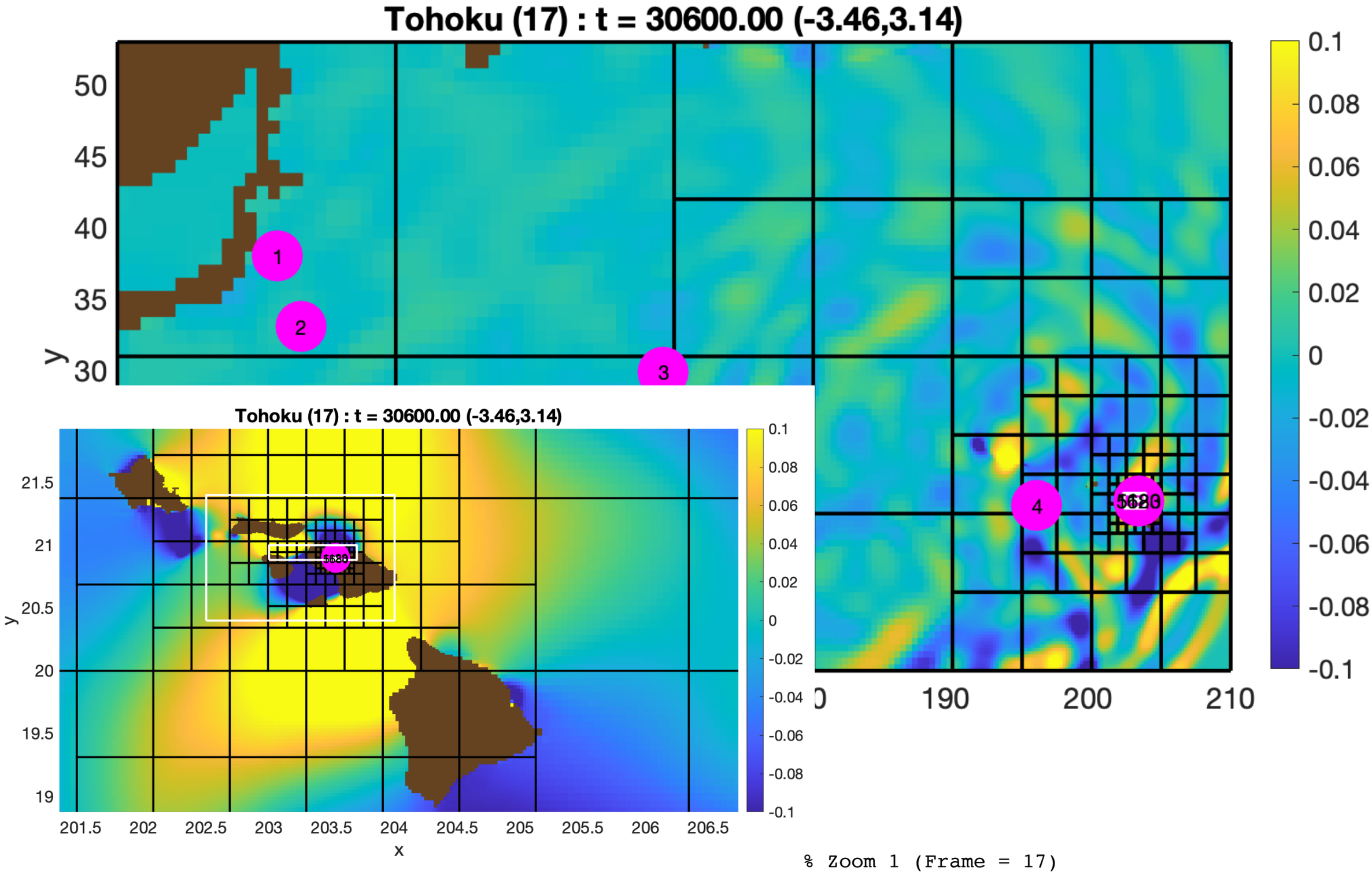
www.forestclaw.org

2011 Tohoku tsunami (GeoClaw extension)



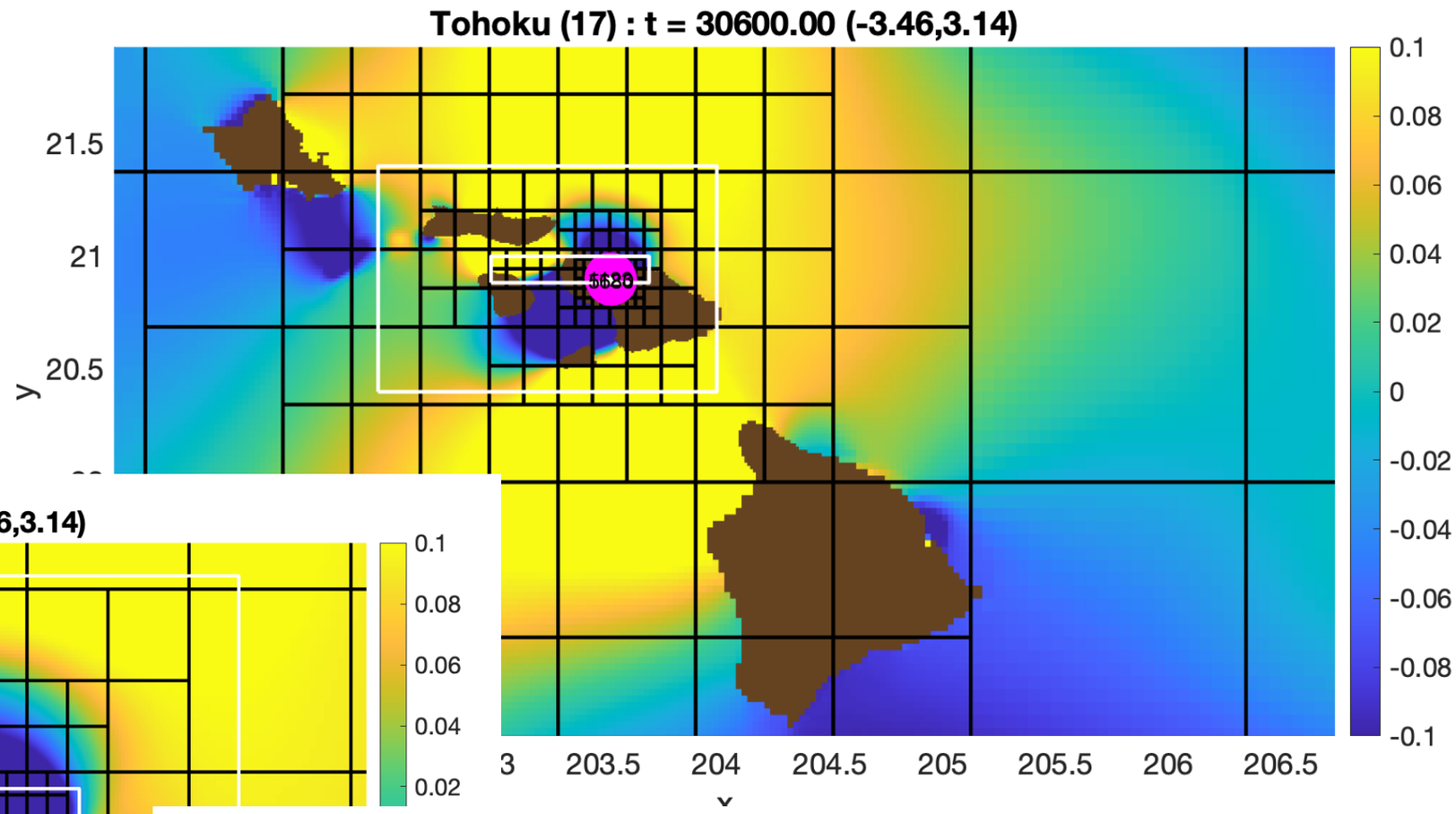

```
% Zoom Frame = 2
% axis([134.6060,  164.9044,   28.2413,   45.3327])
```

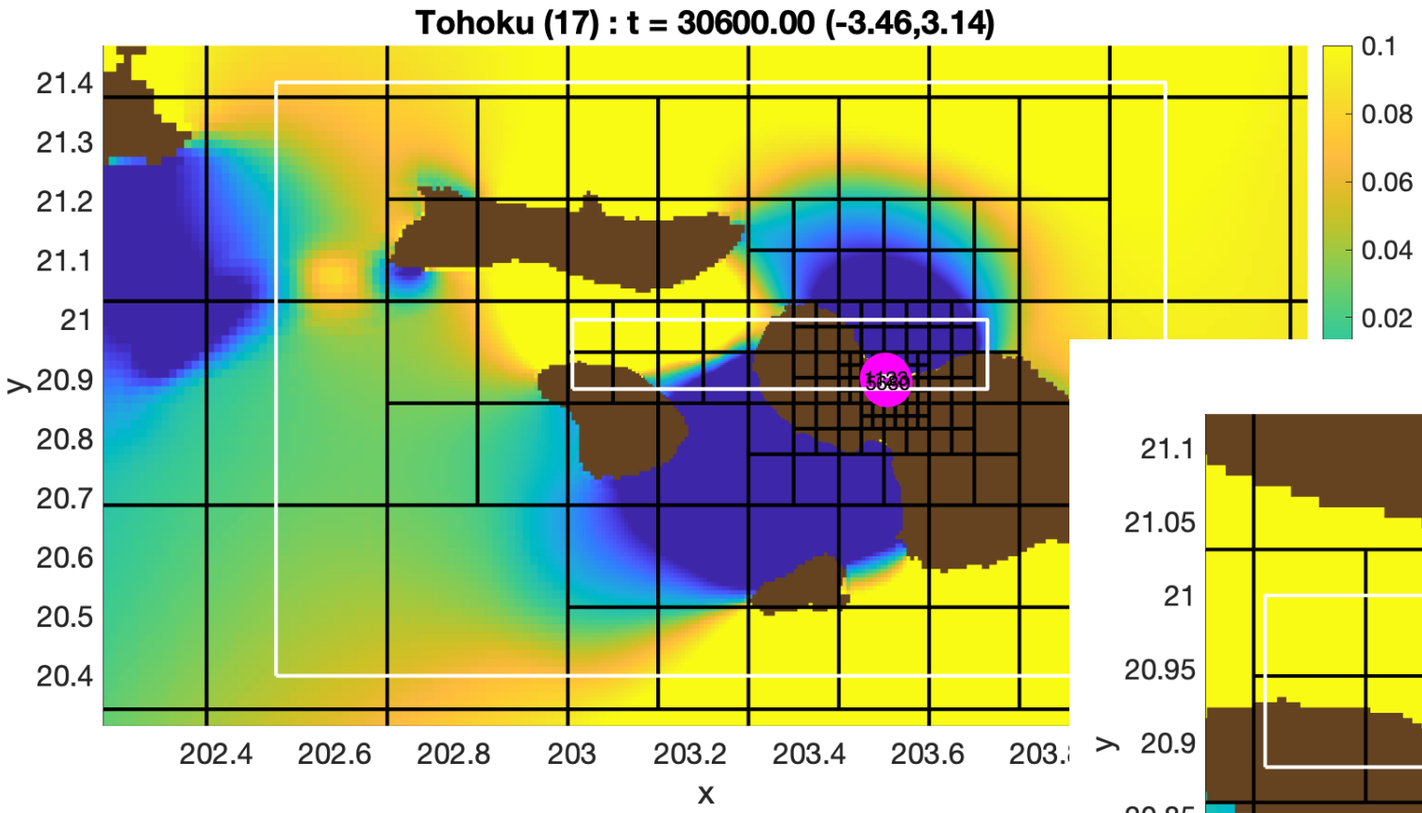
2011 Tohoku tsunami

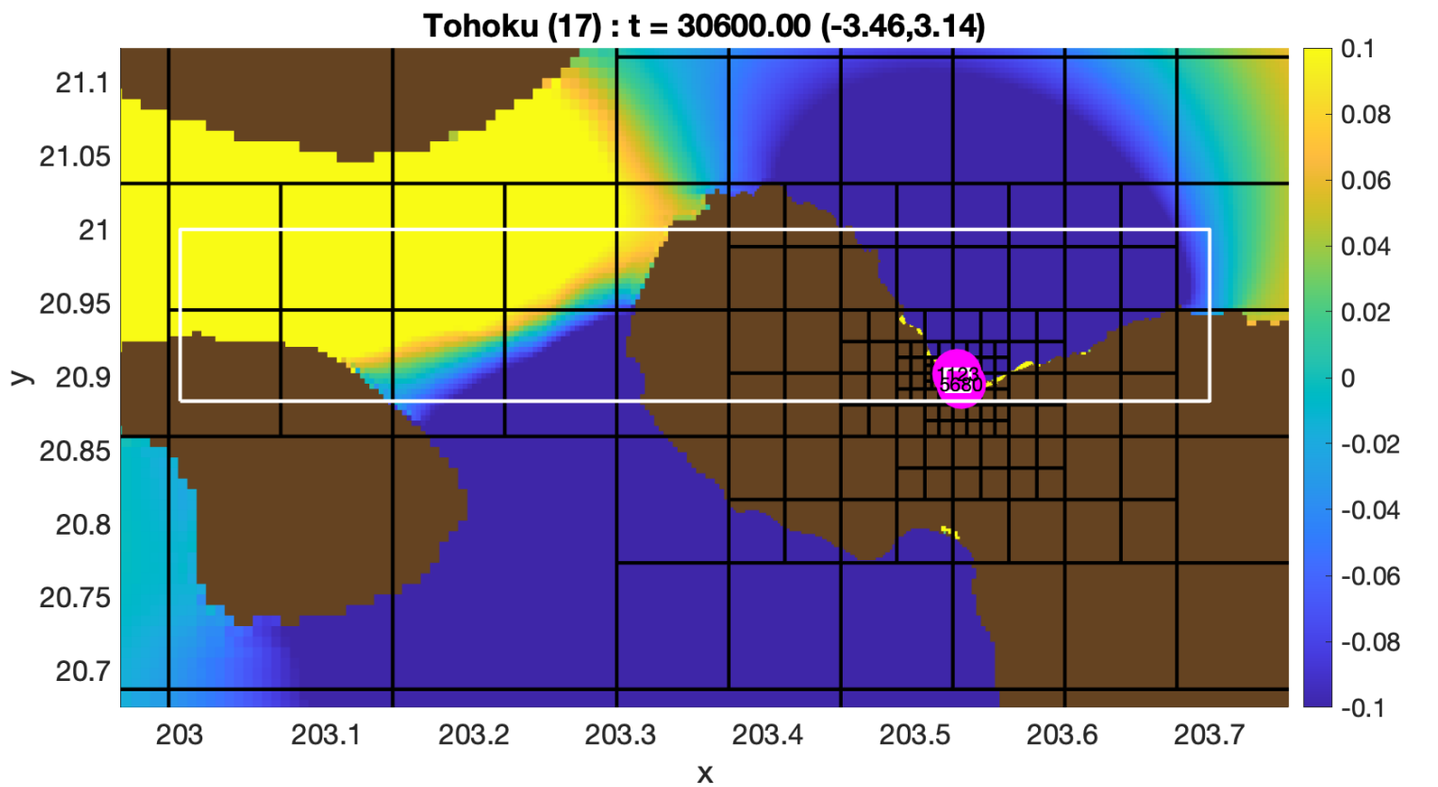



```
% Zoom 1 (Frame = 17)
% axis([201.3305,  206.7376,   18.8768,  21.9269]);
```

2011 Tohoku tsunami

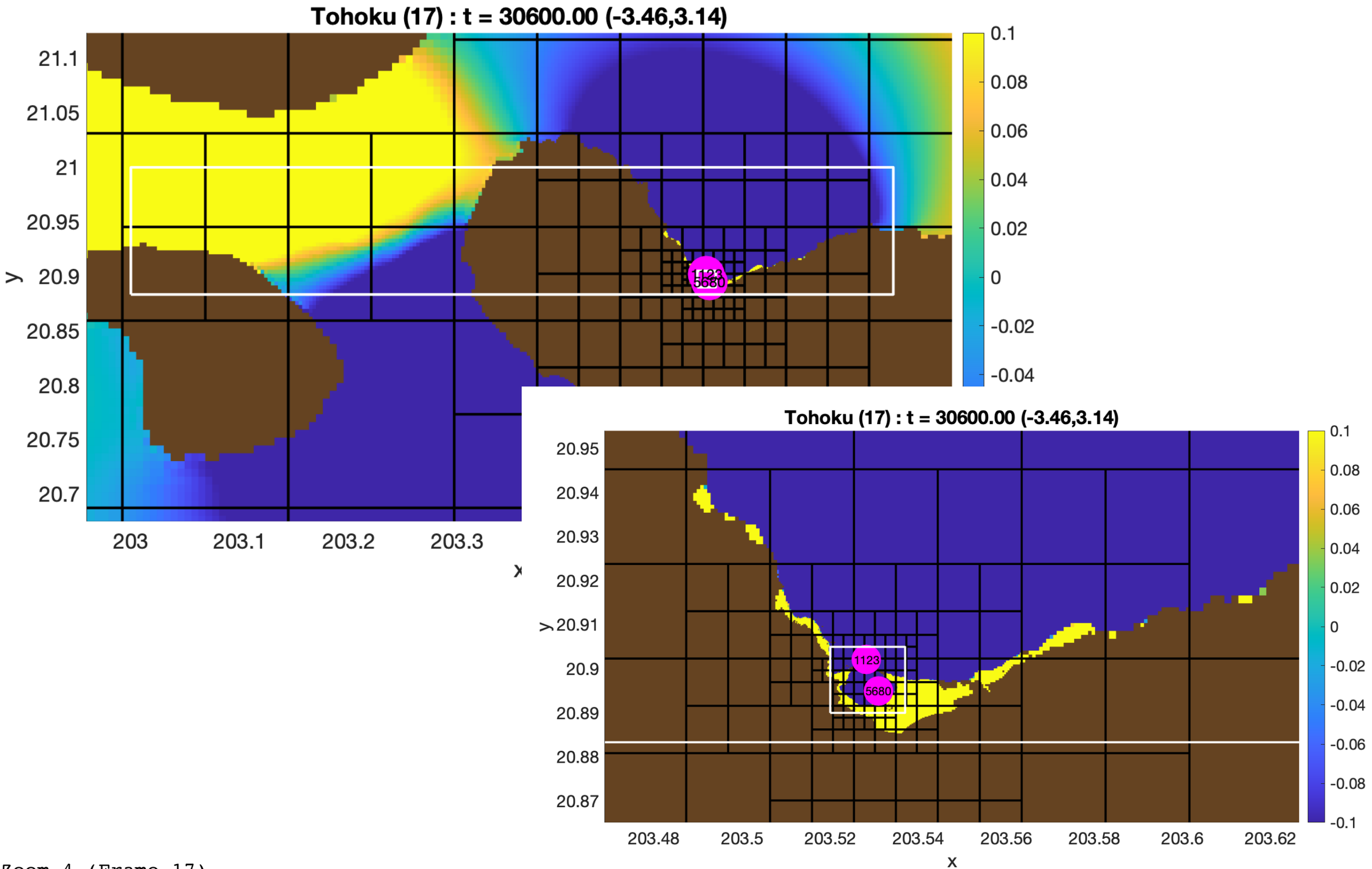






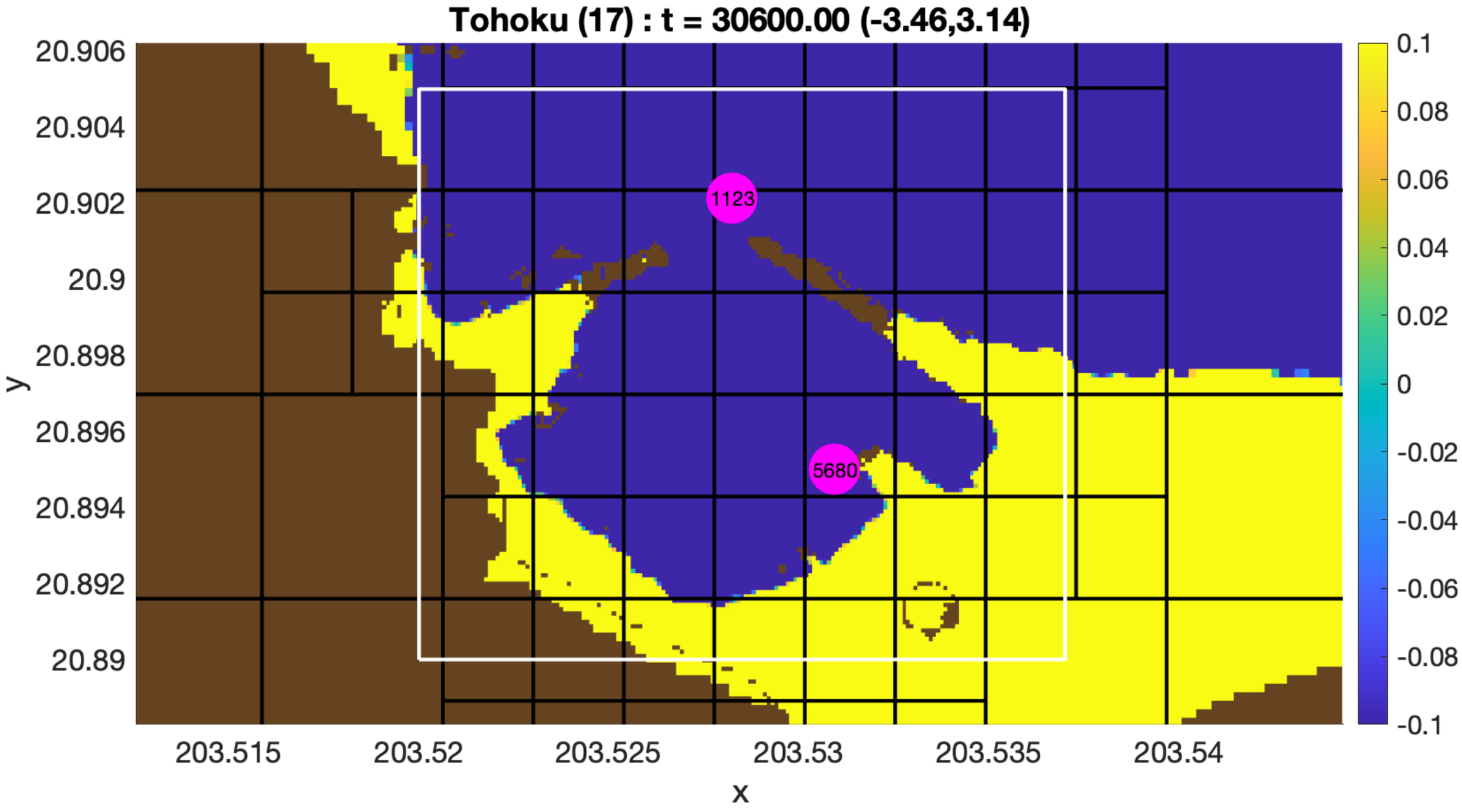

```
% Zoom 2 (Frame 17)
% axis([202.2093, 204.2401, 20.3161, 21.4617])

% Zoom 3 (Frame 17)
% axis([202.9598, 203.7538, 20.6753, 21.1232]);
```

2011 Tohoku tsunami



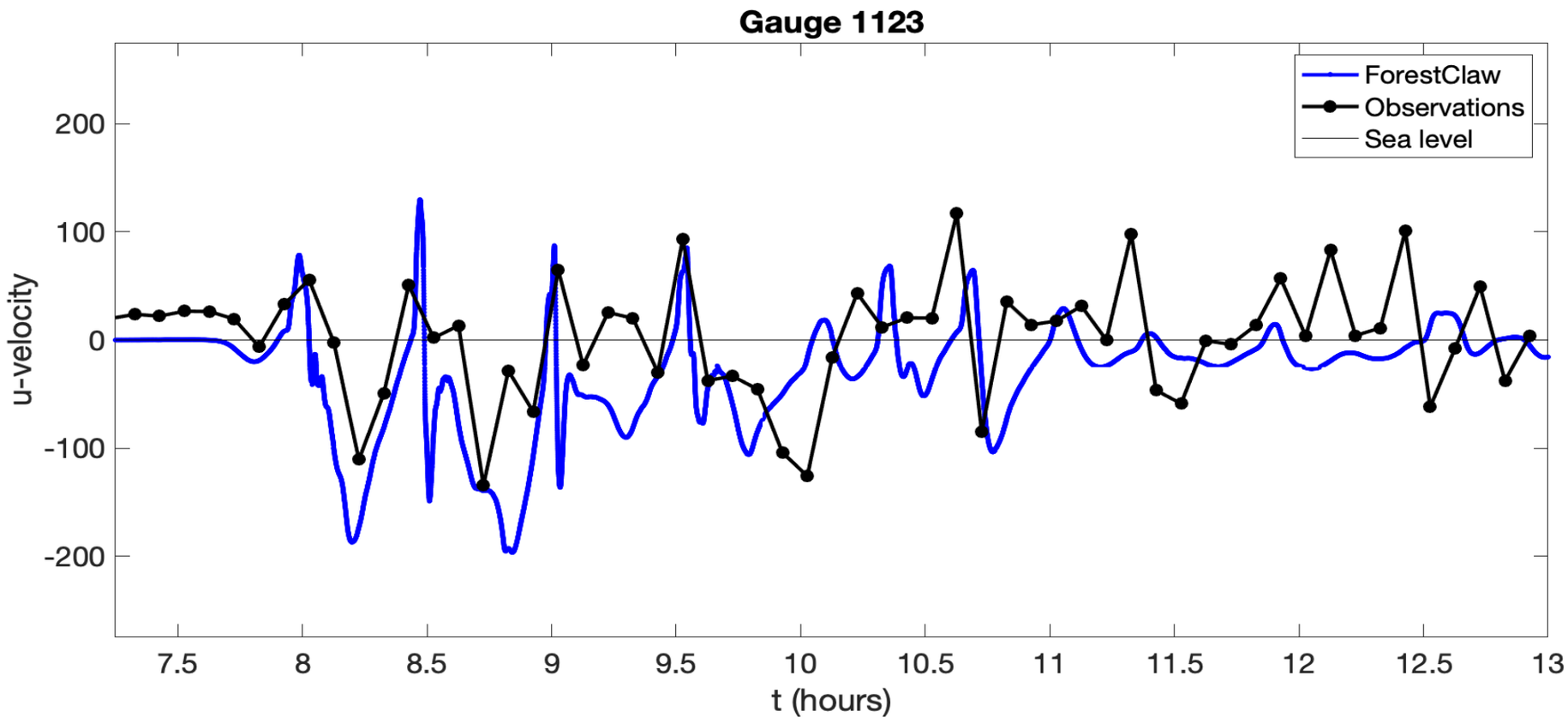

```
% Zoom 4 (Frame 17)
% axis([203.4689,  203.6265,   20.8651,   20.9540]);
```

2011 Tohoku tsunami



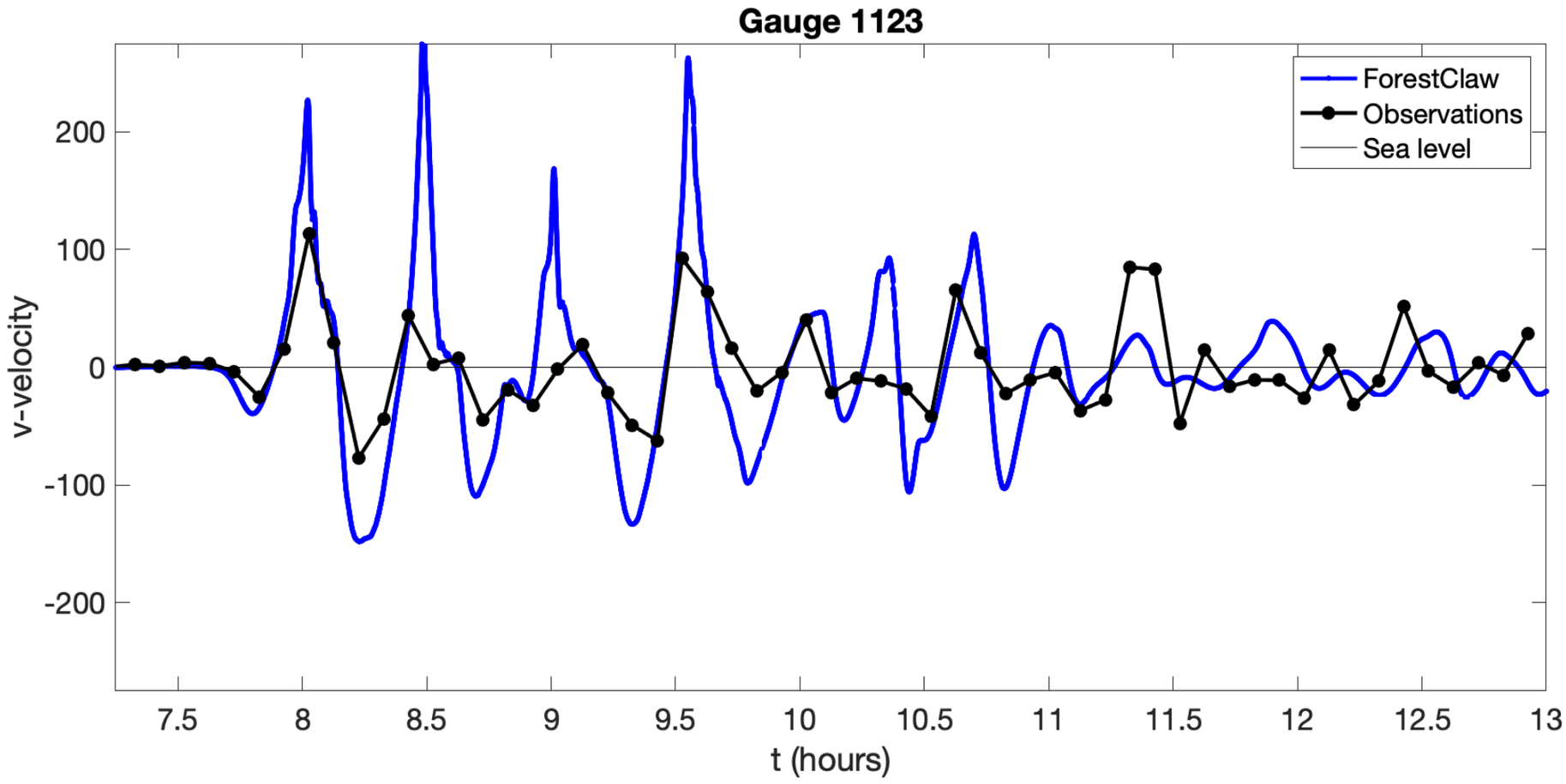

```
% Zoom 5 (Frame 18)
% axis([203.5126,  203.5443,   20.8883,   20.9062]);
```

u-velocity (ForestClaw)



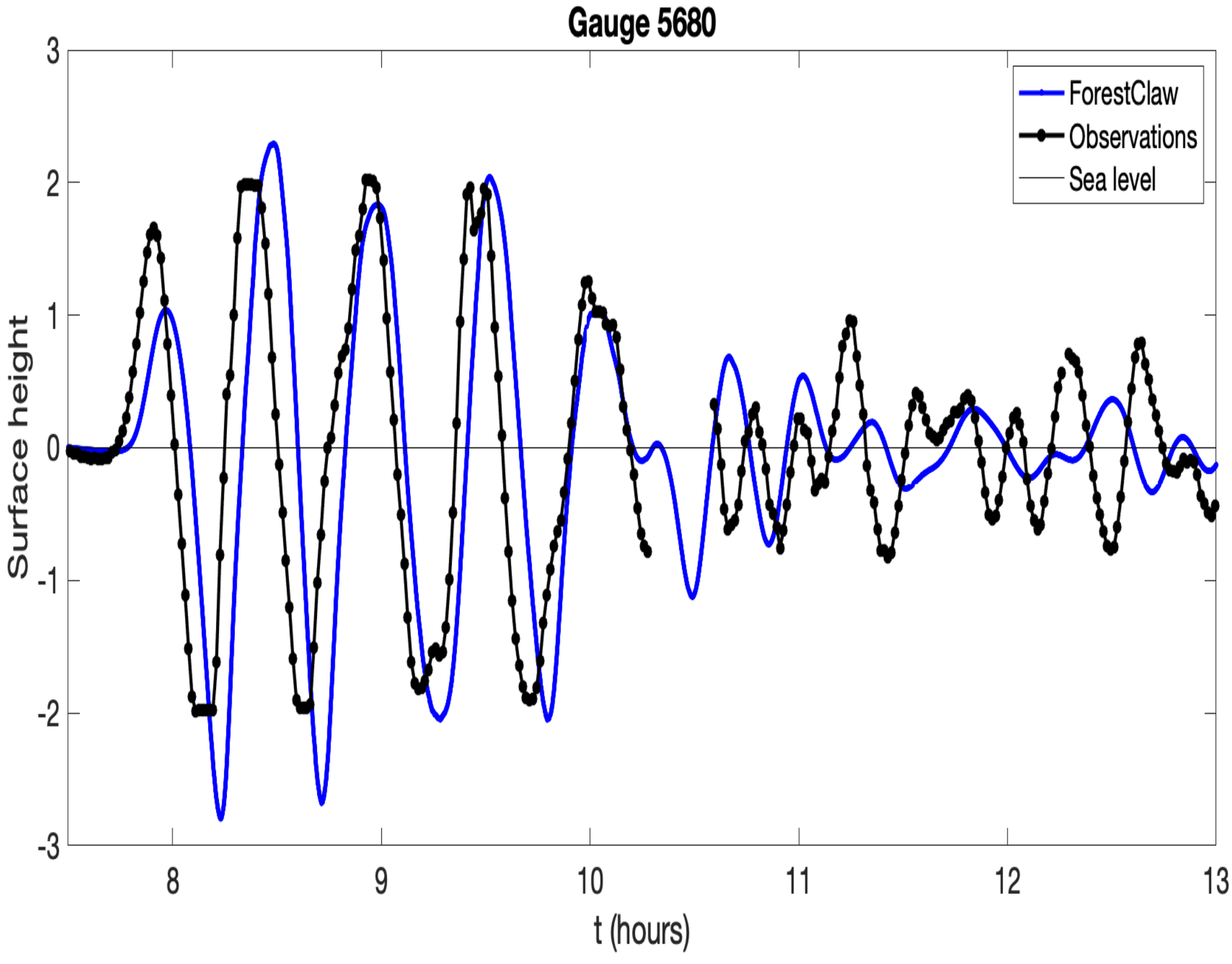
time shift=5min

v-velocity (ForestClaw)



time shift=5min

Surface height (ForestClaw)



Using ForestClaw

- Github site : www.github.com/ForestClaw
- See the wiki for installation instructions and examples
- Most users find it fairly easy to get basic ForestClaw examples running.

Topics that could be discussed further :

- Wave propagation algorithm (Clawpack)
- Building your own library extension?
- Using mapped, multi-block features?
- Parallel features and GPU extension?
- GeoClaw applications?
- Refinement? Adaptive time stepping?