Challenges in coupling codes in large scale PDE solvers

Donna Calhoun (Boise State University)

Collaborators : C. Burstedde (Univ. of Bonn); J. Snively (Embry-Riddle; S. Aiton (BSU); Clawpack Developers and many others

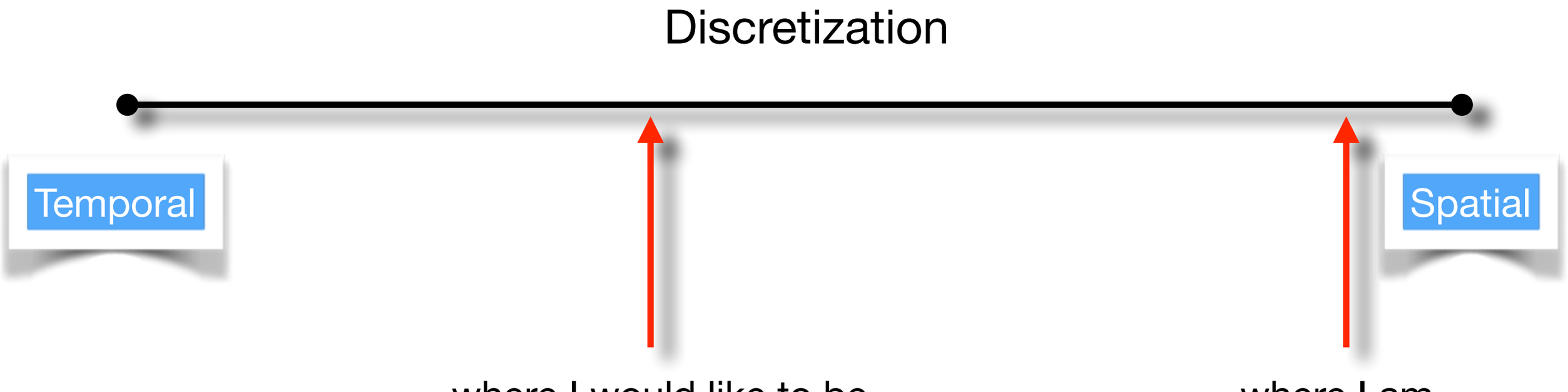
Holistic Design of Time-Dependent PDE Discretizations

January 10-14, 2022

ICERM

Providence,RI

The temporal-spatial scale



What are the gains we can hope from sophisticated time stepping

- Local time stepping can provide enormous performance gains,
- A basic understanding of coupling complex codes will lead to more robust simulations

ForestClaw Project

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

Features of **ForestClaw** include :

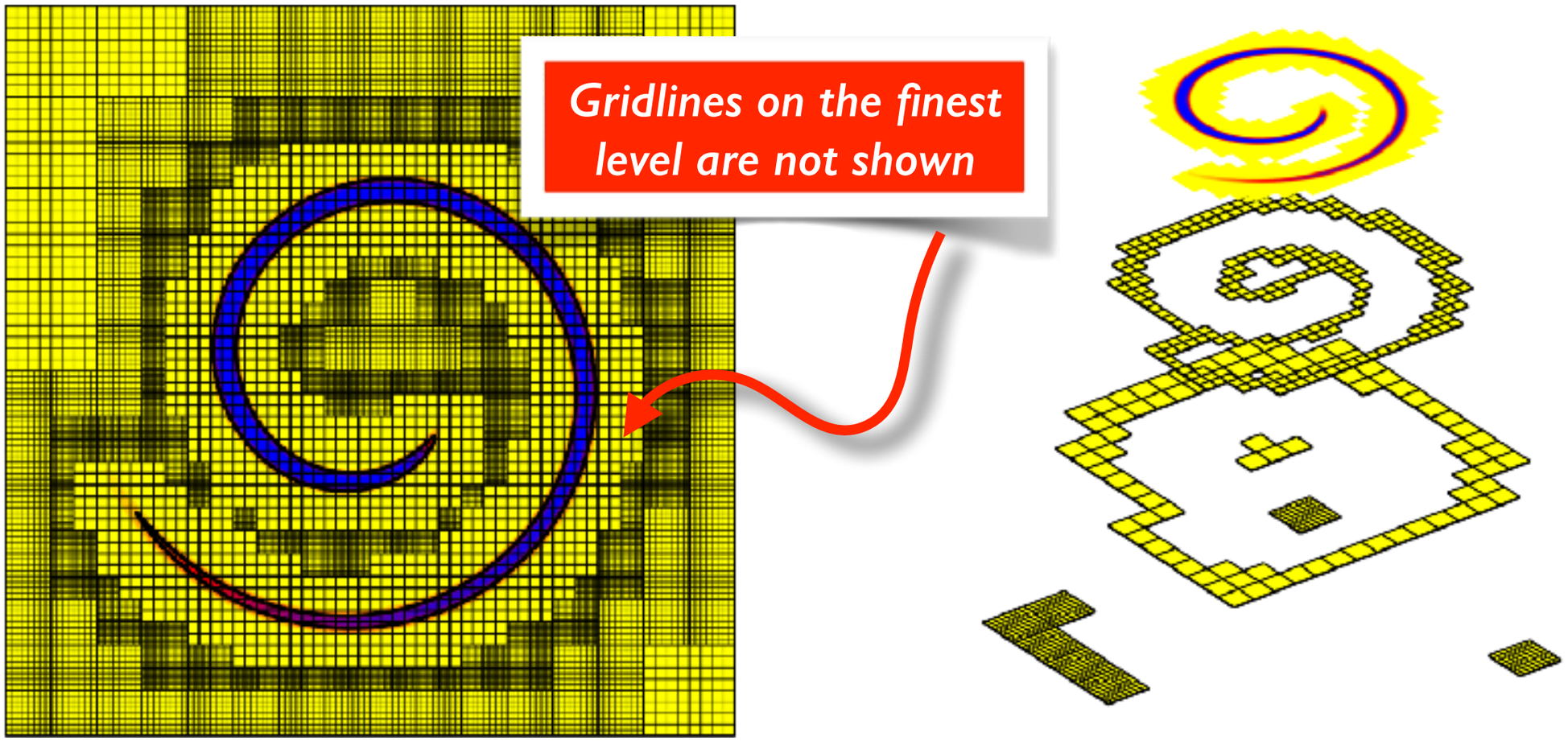
- **Block-based** AMR - Each leaf of the quadtree contains a fixed-size grid,
- Uses the **highly scalable p4est** dynamic grid management library (C. Burstedde, Univ. of Bonn, Germany)
- Has **mapped, multi-block** capabilities, (cubed-sphere, for example) to allow for flexibility in physical domains,
- **Extensible** with custom solvers
- Optional **adaptive** time stepping strategy,
- Uses essentially the same **algorithmic components** as Berger-Oliger-Collela patch-based AMR used by AMReX, Chombo, AMRClaw, and others

ForestClaw development supported by the National Science Foundation

www.forestclaw.org **www.github.com/ForestClaw**

Adaptive Mesh Refinement (AMR)

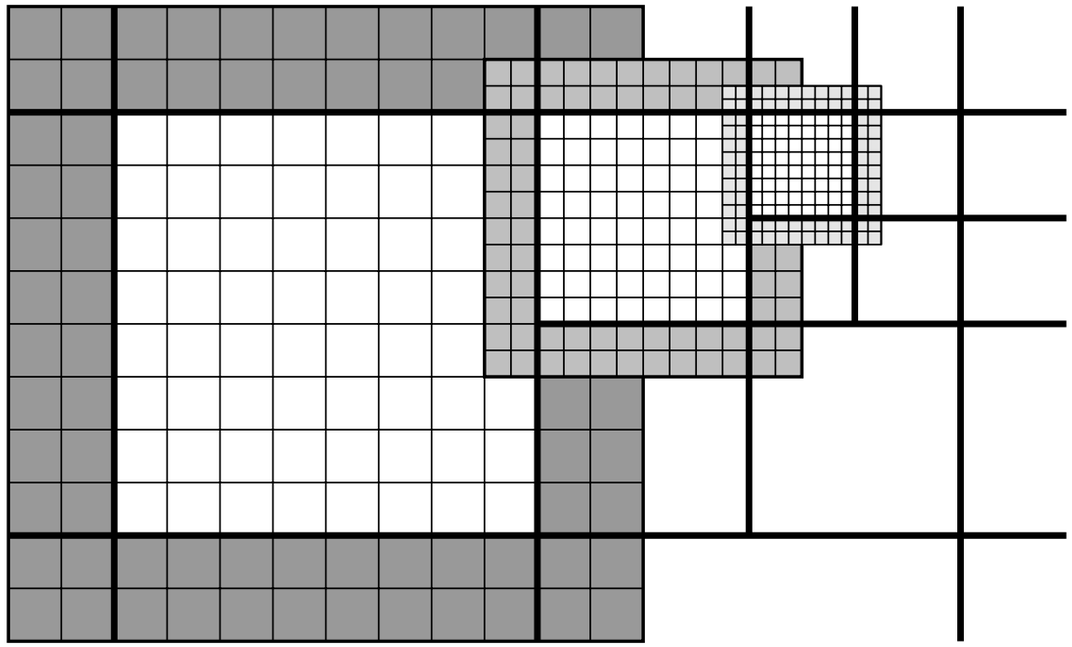
Block-based AMR (regular sized, non-overlapping blocks in a quadtree/octree)



ForestClaw

Other codes using similar idea : FLASH (U. Chicago), Racoon II (U. Bochum), Nirvana (Potsdam), "Building Cubes" (Tohoku); ENZO (enzo-project.org), ...

ForestClaw : a PDE layer



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.

ForestClaw philosophy

- Enable users to **port existing Cartesian grid codes** to highly scalable, parallel adaptive environment.
- Starting point : **Users are experts in their application** and solvers, and have put much thought and work into developing their codes
- To the greatest extent possible, users should be able to **leverage any existing code** they have already developed. Encourage re-use of legacy Cartesian codes.
- If the programming paradigm is clear enough, users can reason about their interaction with the code and can be involved in technical details of getting their application running.
- Most users are not experts in computer science, nor do they want to be. So **language constructs need to be reasonably simple**, i.e. limit use of C++. Emphasize procedures over objects. Don't try to invent DSLs that are meaningless to everyone but the developer.
- Encourage mixed programming, i.e. Fortran+C.
- Allow for easy customization of most details of adaptive process, including tagging criteria.

Programming paradigms in ForestClaw

Paradigms

- Iterators
- Callbacks
- Virtual tables
- Encapsulated *extension libraries* for defining how patches get updated, and how data within a patch is stored.

Extension libraries

- A solver library can update a solution on a single grid, or, in the case of an elliptic solver, return a solution on the mesh hierarchy. Solver libraries are typically wrappers for legacy code.
- Solvers work together with *patch libraries.*
- Configuration parameters for solvers and patch types (cell-centered, node centered, etc) are contained within the library,
- Composibility : Libraries are design not to clash with each other, so multiple versions of the same library can be compiled together for selection at run-time.

Solver libraries : time stepping

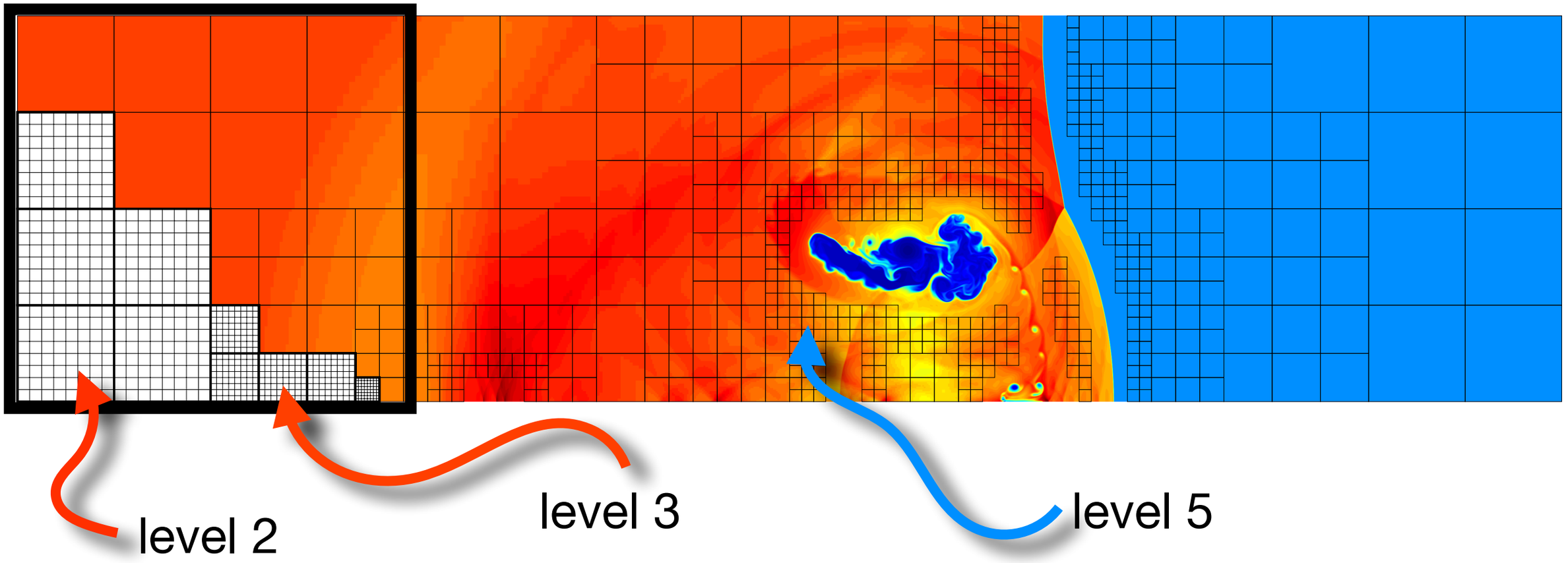
We have an existing Cartesian grid solver

- Let's assume it is an explicit time stepping solver.
- Furthermore, we have a time stepping loop that looks something like this :

```
Choose a time step dt,
for k = 1, M
   Take a single time step
   Output results
   Compute some diagnostics
```

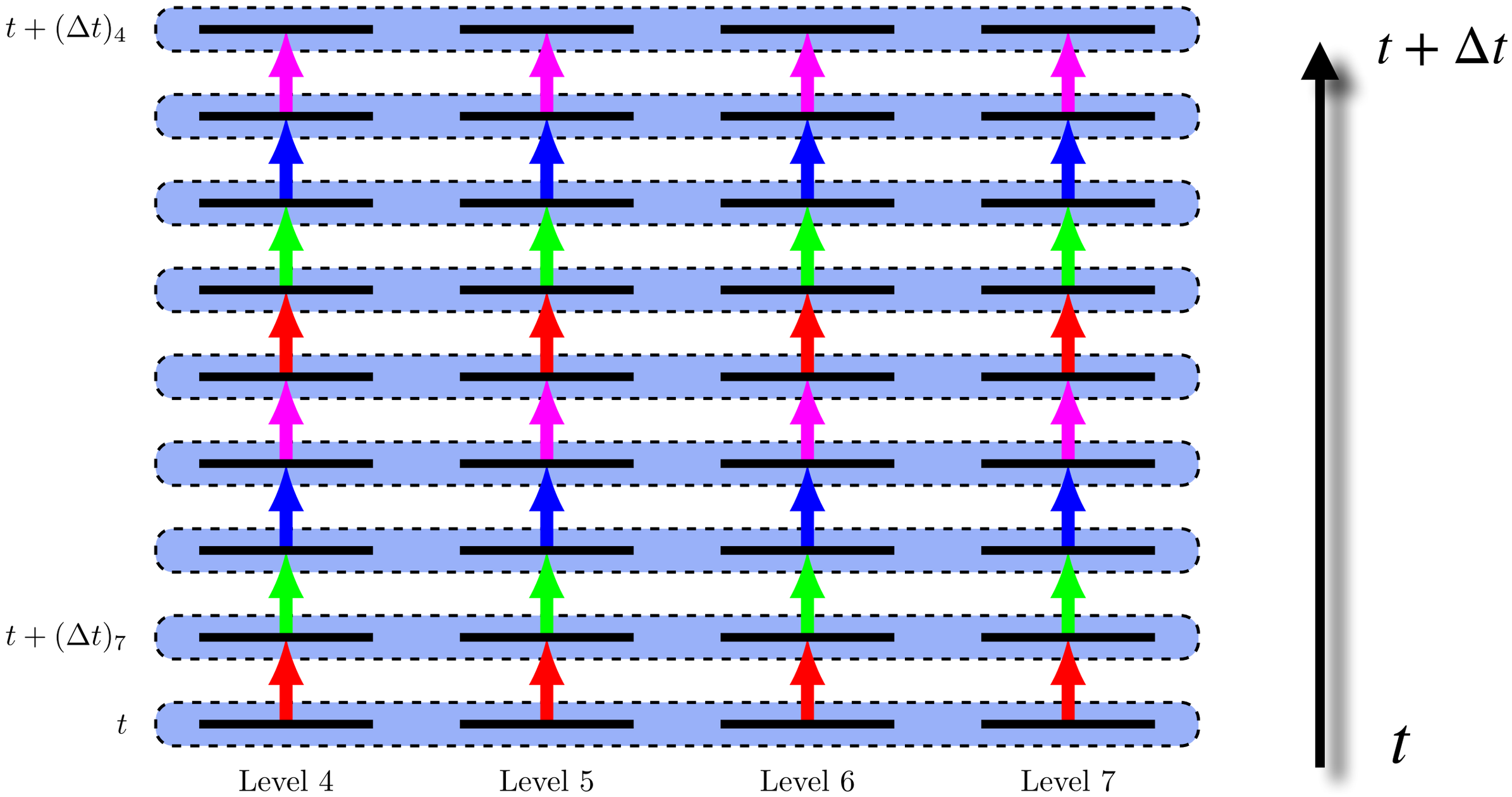
- The time step may depend on a CFL constraint, or some other constraint needed for stability.
- What does this loop look like on an AMR hierarchy?
- Focus on the single time step

Solver libraries : time stepping



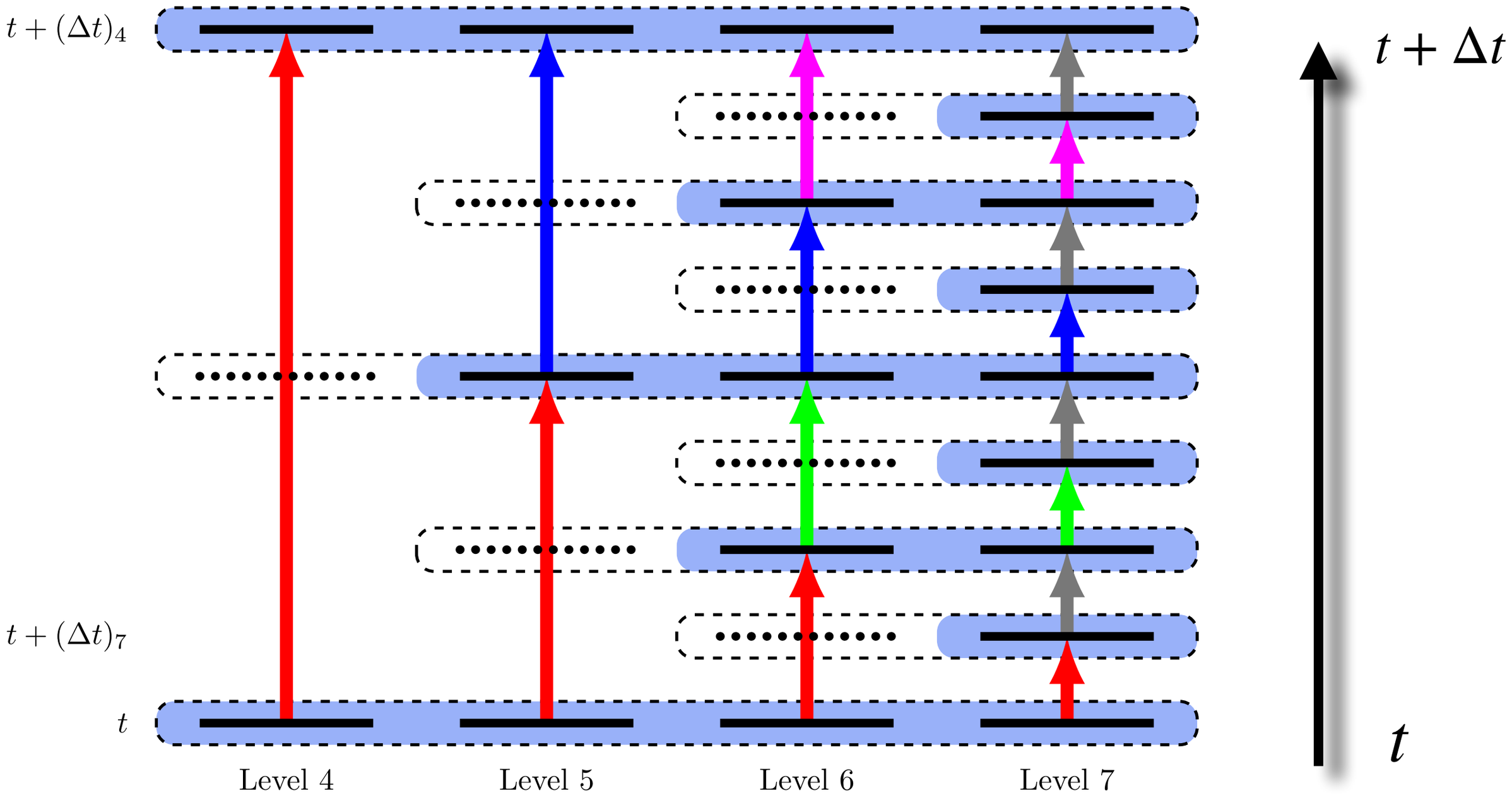
- For hyperbolic problems, the time step is often limited by cell size.
- Global time stepping : One time step Δt for all grids.
- Local time stepping : Time step size depends on cell size
- Benefits of local time stepping depend on the problem

Global time stepping



- Arrows of the same color indicate recursive calls
- Blue boxes indicate parallel ghost cell exchanges

Local time stepping



- Arrows of the same color indicate recursive calls
- Blue boxes indicate parallel ghost cell exchanges

Multirate algorithm

Require: Grids at all levels at time t must have valid ghost cells values.
  for $k = 1$ to $2^{\ell_{max}-\ell_{min}}$ **do**
    ADVANCE_SOLUTION$(\ell_{max}, (\Delta t)_{\ell_{max}})$ **Advance solution on finest level**
    if multirate **then** **Multirate**
      if $k < 2^{\ell_{max}-\ell_{min}}$ **then**
        Find largest integer $p \geq 0$ such that 2^p divides k.
        $\ell_{time} = \ell_{max} - p - 1$ **Intermediate synchronization**
        UPDATE_GHOST$(\ell_{time} + 1)$
      end if
    else **Global time stepping**
      UPDATE_GHOST(ℓ_{min})
    end if
  end for
  UPDATE_GHOST(ℓ_{min}).

procedure ADVANCE_SOLUTION(level $= \ell$, dt_stable $= \Delta t$)
  for all grids g on level ℓ **do**
    Update solution $Q^{n+1} = Q^n + \Delta t\, F(Q^n, t_n)$.
  end for
  if $\ell > \ell_{min}$ **then**
    if multirate **then**
      if levels ℓ and $\ell - 1$ are time synchronized **then**
        ADVANCE_SOLUTION$(\ell - 1, 2\Delta t)$
        TIME_INTERPOLATE$(\ell - 1, t + 2\Delta t)$
      end if
    else
      ADVANCE_SOLUTION$(\ell - 1, \Delta t)$
    end if
  end if
end procedure

Recursive advance, followed by a time interpolation

Iterators and callback functions

Iterators are used to visit each grid in the tree.

```
double fclaw2d_update_single_step(fclaw2d_global_t *glob, int level,
                                  double t, double dt) {
    /* Store time step, t in struct ss_data */
    fclaw2d_global_iterate_level(glob, level, cb_single_step, &ss_data);
}
```

Additional customizable callback functions can be customized to supply

- Initialization routines
- Averaging and interpolating to fill ghost cells and update mesh
- Tagging grids for coarsening or refinement,
- Diagnostic accumulation (error, conservation)
- Applying physical boundary conditions
- Customizing output

Callback function for each patch

Many options that control when to return from time stepping routines

See details available in p4est Summer School Tutorial

ForestClaw

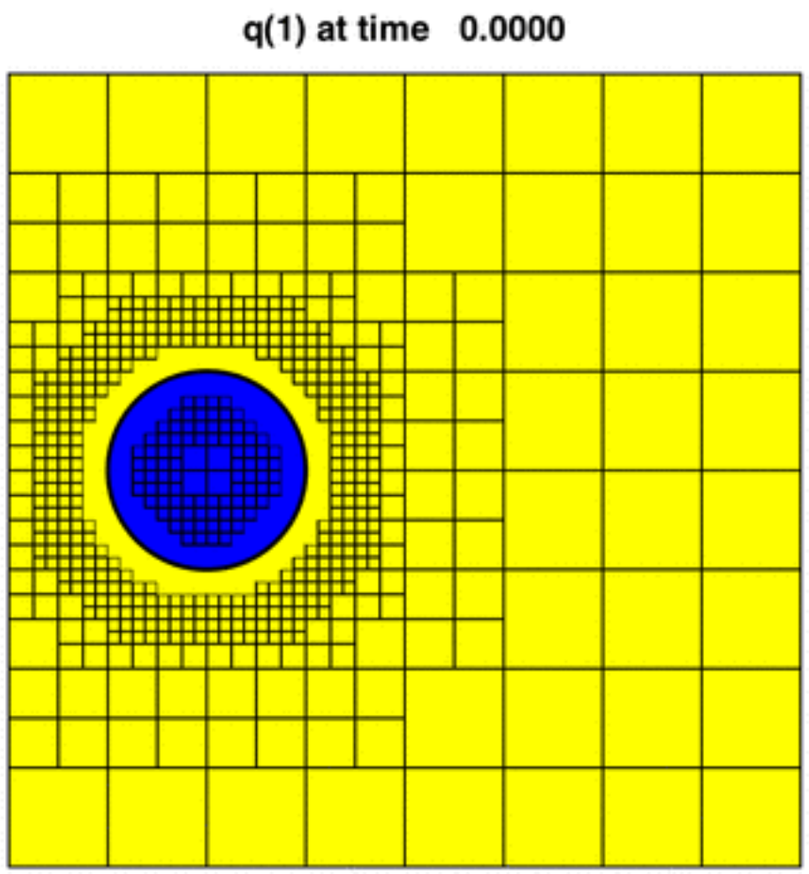
- Single step, explicit “ADER” type time stepping. Formal second order is achieved by dimensionally split or unsplit Lax-Wendroff-like spatial discretizations.
  - For dynamically evolving meshes, multi-step methods are impractical.
  - Multi-stage methods are potentially more tractable, especially if a global time step is used.
- Godunov operator splitting used to separate a hyperbolic step from a source term step (e.g. diffusion, reaction, geometric source terms, bathymetry)
- Accuracy is typically CFL dependent. A CFL closer to 1 will generally lead to more highly resolved solution.
- On AMR grids, conservative corrections are needed to maintain conservation
- Local time stepping (aka subcycling, or **multirate** time stepping) included from the start.
  - Main goal is time savings; secondary goal is increased accuracy.
  - Depending on choice of AMR meshing strategy, multirate + dynamically evolving meshes creates challenges.

Three case studies

Three case studies in multirate/coupling of codes

1. Tracer transport : How does performance gains using multirate compare to other areas of where we can save time?
2. Multi-rate shallow water wave equations (choice of Δt governed by fluid depth (less obvious how to manage a multi-rate scheme)
3. Complex coupling of codes using different meshes, elliptic terms, stiff diffusive terms (coupling with a global time step is already challenging)

Case study 1 : Tracer Transport

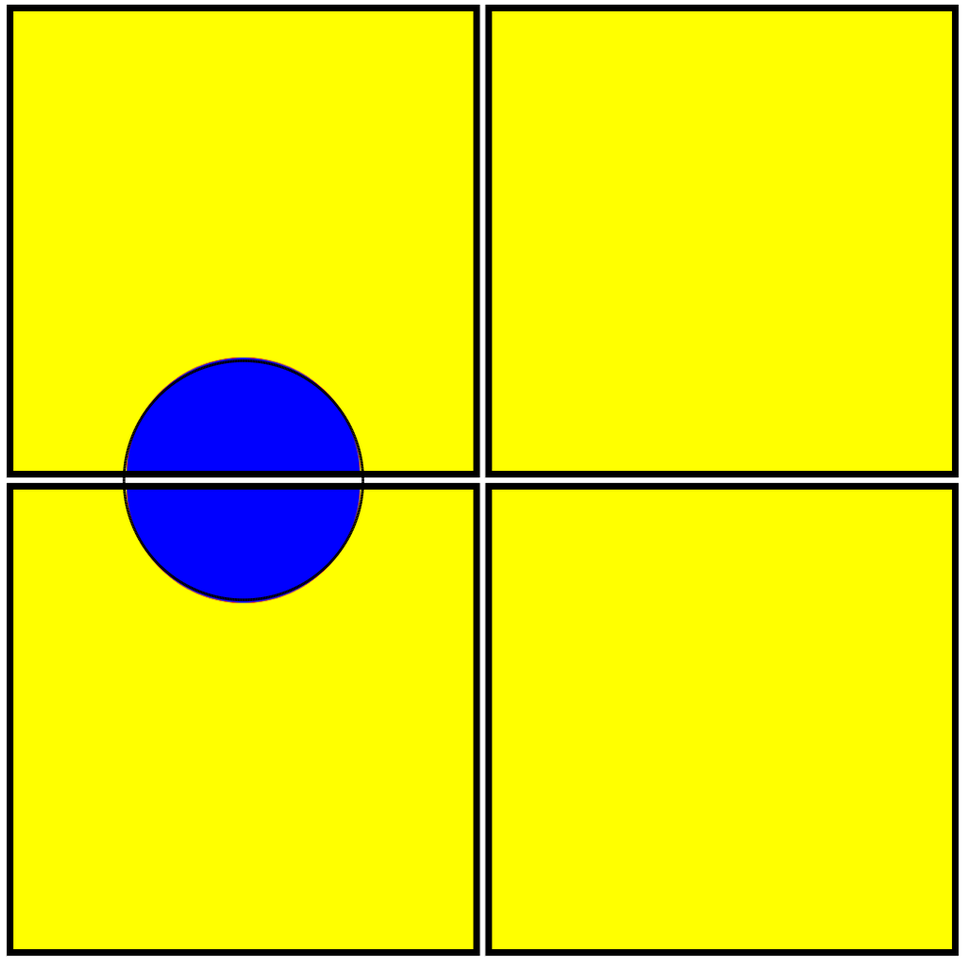


Tracer transport in smooth flow field can form thin filaments

Underlying mesh management done using p4est (C. Burstedde, Univ. of Bonn)

Subdivide the domain

6997.1s (1.9hr)



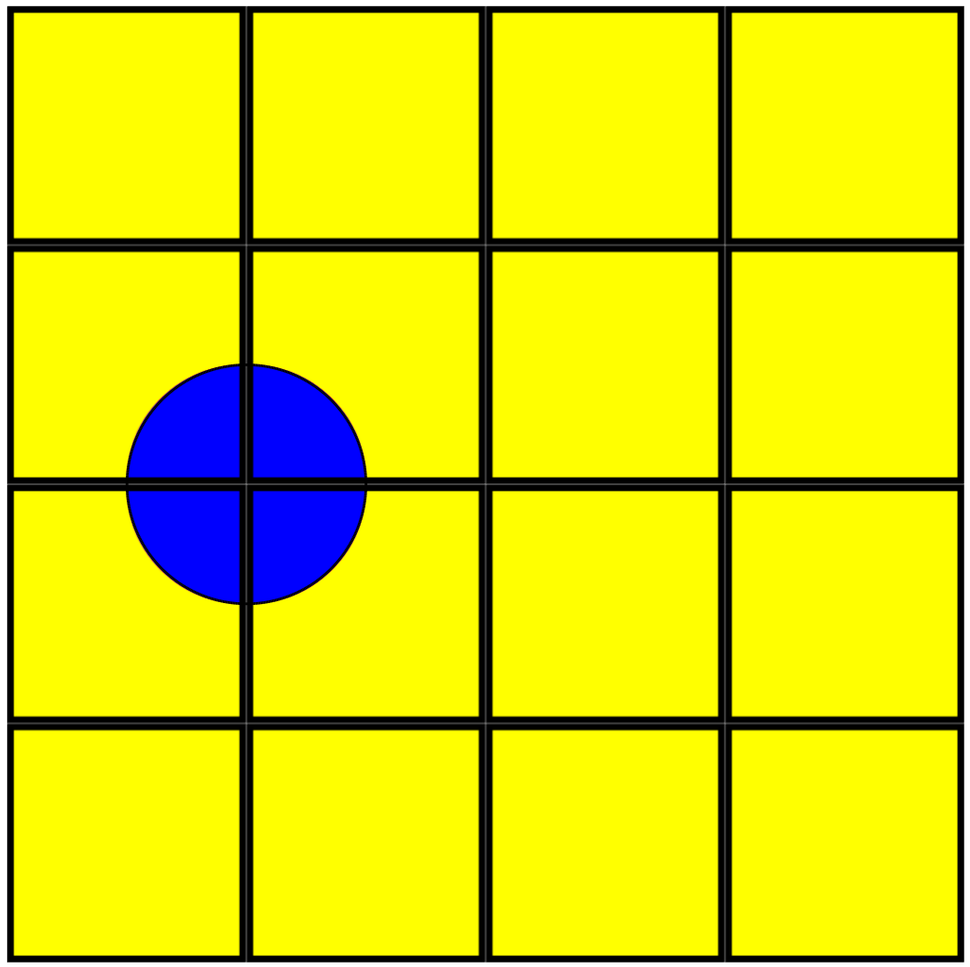
;

2×2 array of 512×512 grids

This will improve cache performance enormously

Subdivide the domain

4131.7s (69min)



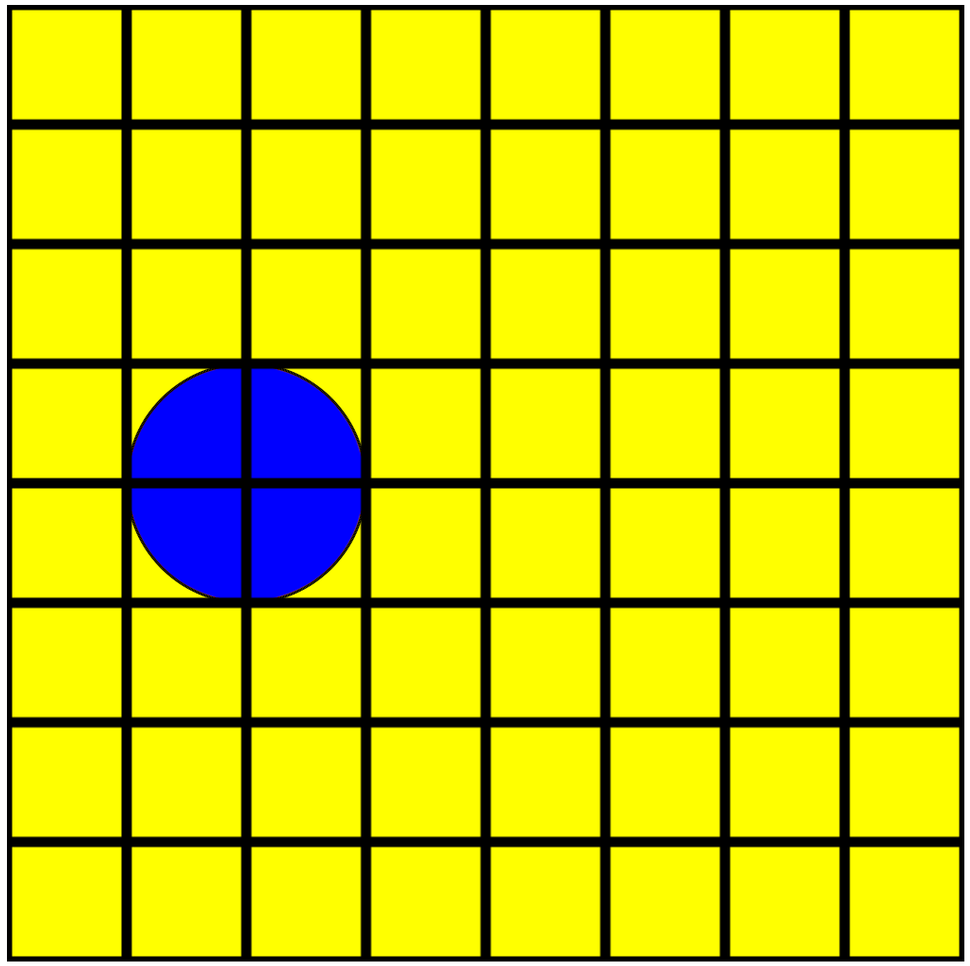
;

4×4 array of 256×256 grids

This will improve cache performance enormously

Subdivide the domain

3813.7s (64min)



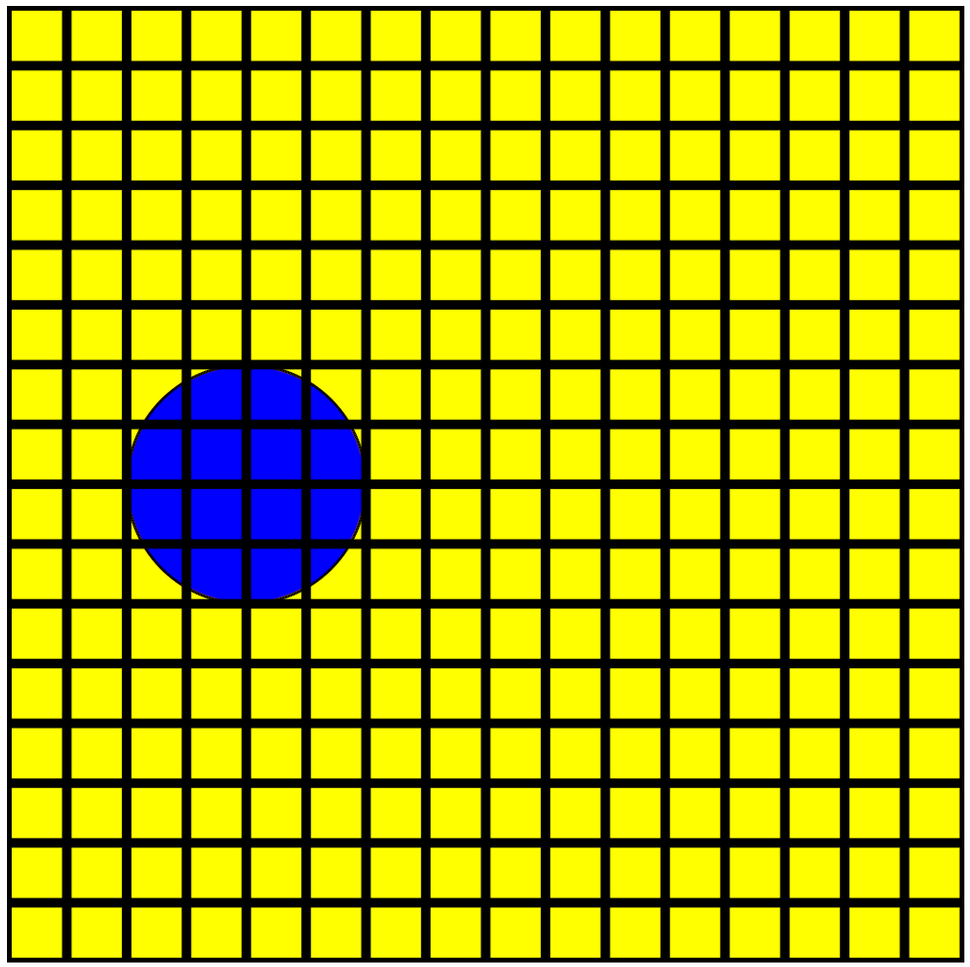
;

8×8 array of 128×128 grids

This will improve cache performance enormously

Subdivide the domain

3865.3s (64min)



;

16×16 array of 64×64 grids

This will improve cache performance enormously

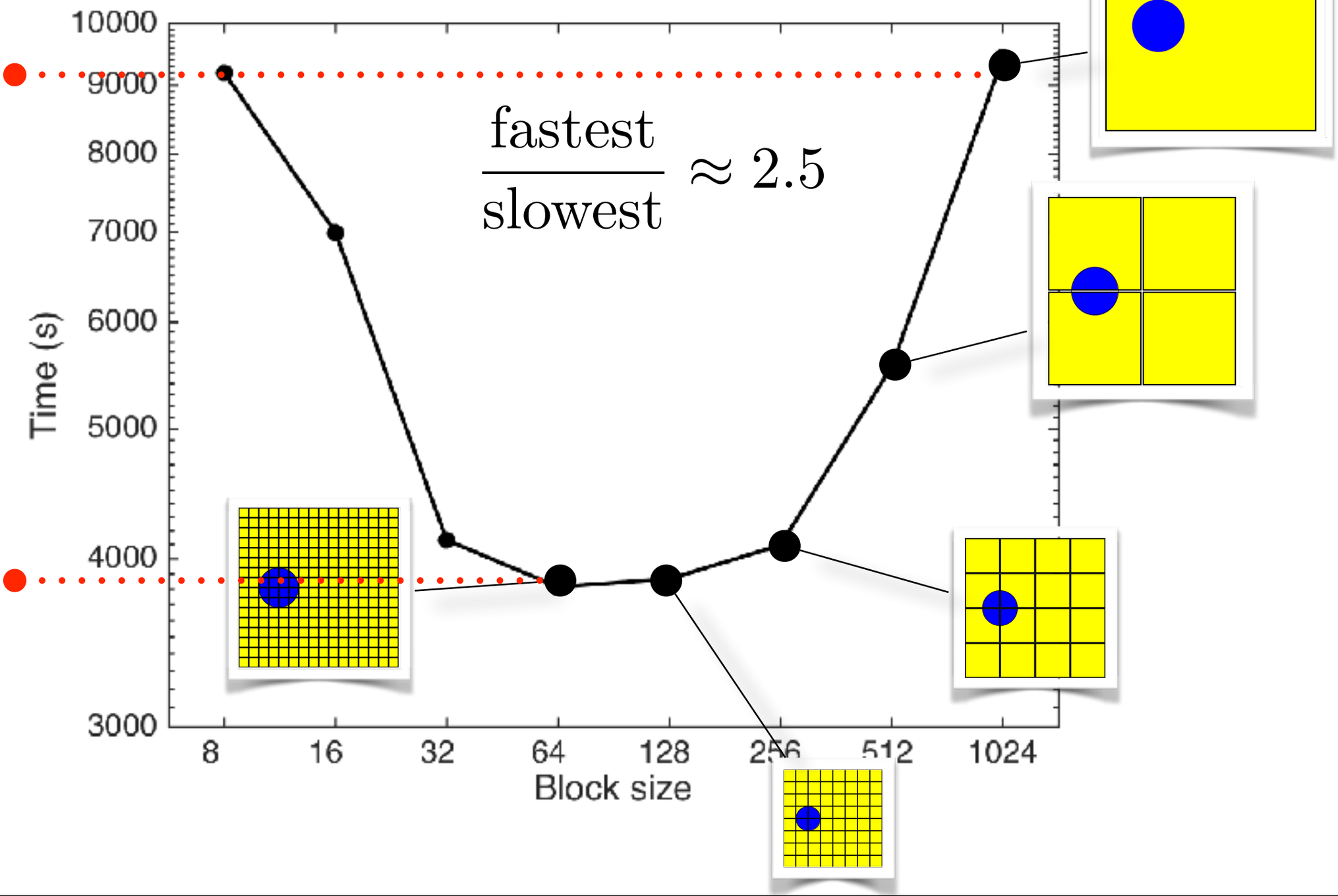
Timing vs block size

9194.5s (1)

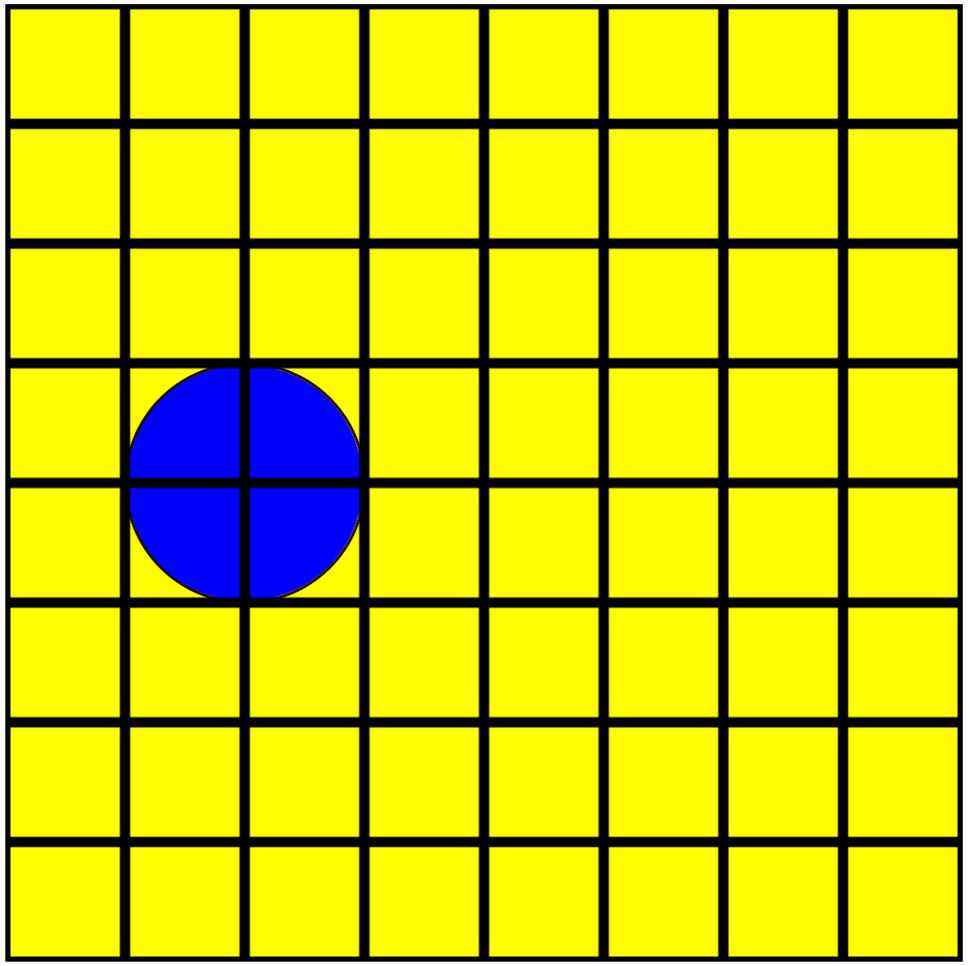
(2.5 hrs)

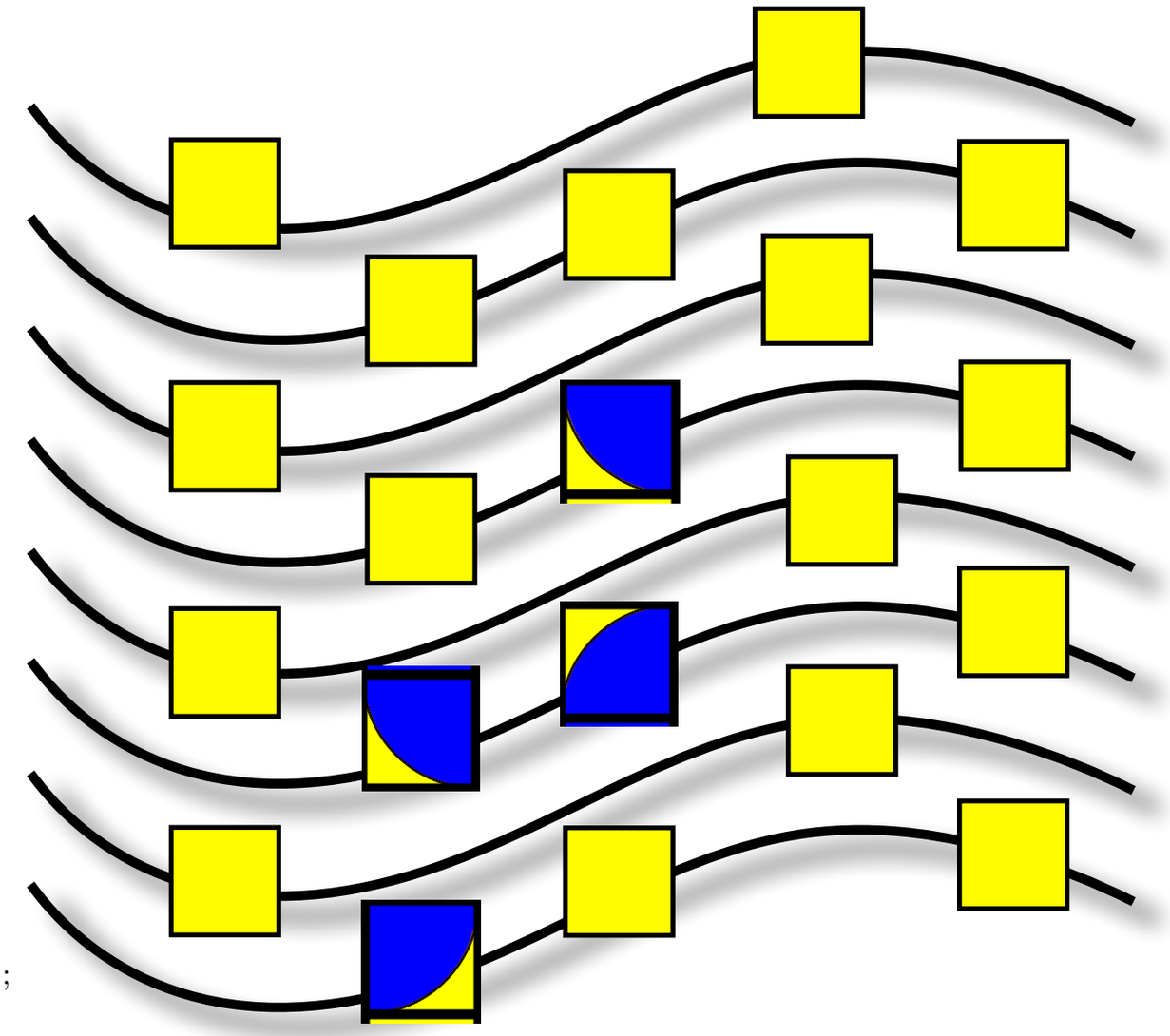
3813.7s (2.4)

(1.1 hrs)



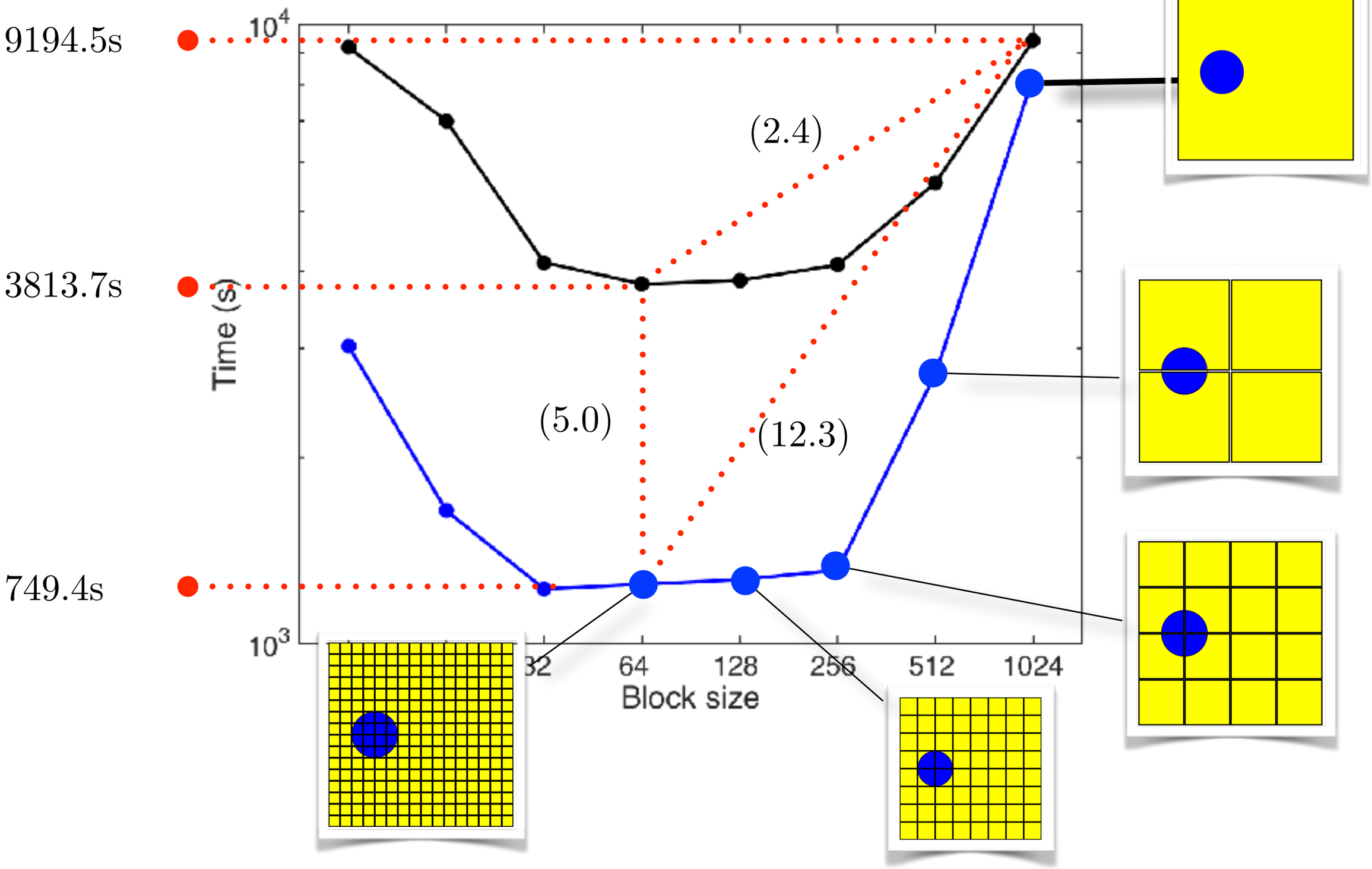
Use multiple cores





Distribute patches to hardware threads. On a four core machine, this should lead to about a factor four speedup.

Speed up using 4 cores (Intel i7)

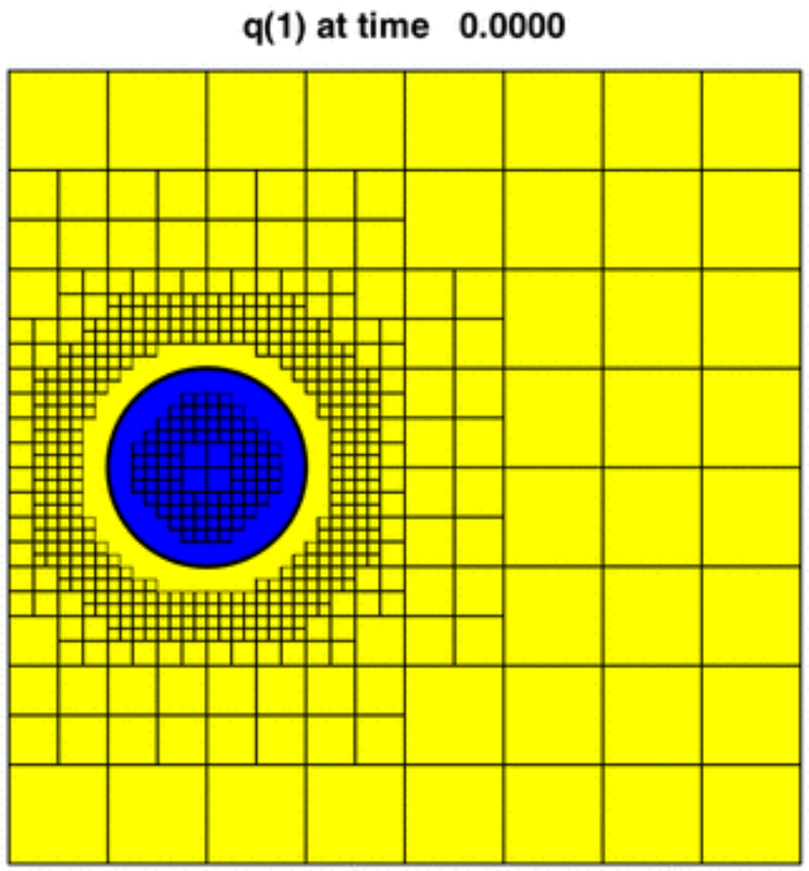


Can we improve on this?

- Hardware solutions : Use more computing units (cpus/threads/cores?
  - Eventually, however, the communications costs will overwhelm any gains made in subdividing the domain further
  - Problems always outgrow the hardware that is available

- Software solutions : Reduce the “factor 8” scaling law and put resources only where they are needed.
  - This is much more difficult than just adding more computing unit
  - Should complement hardware advances

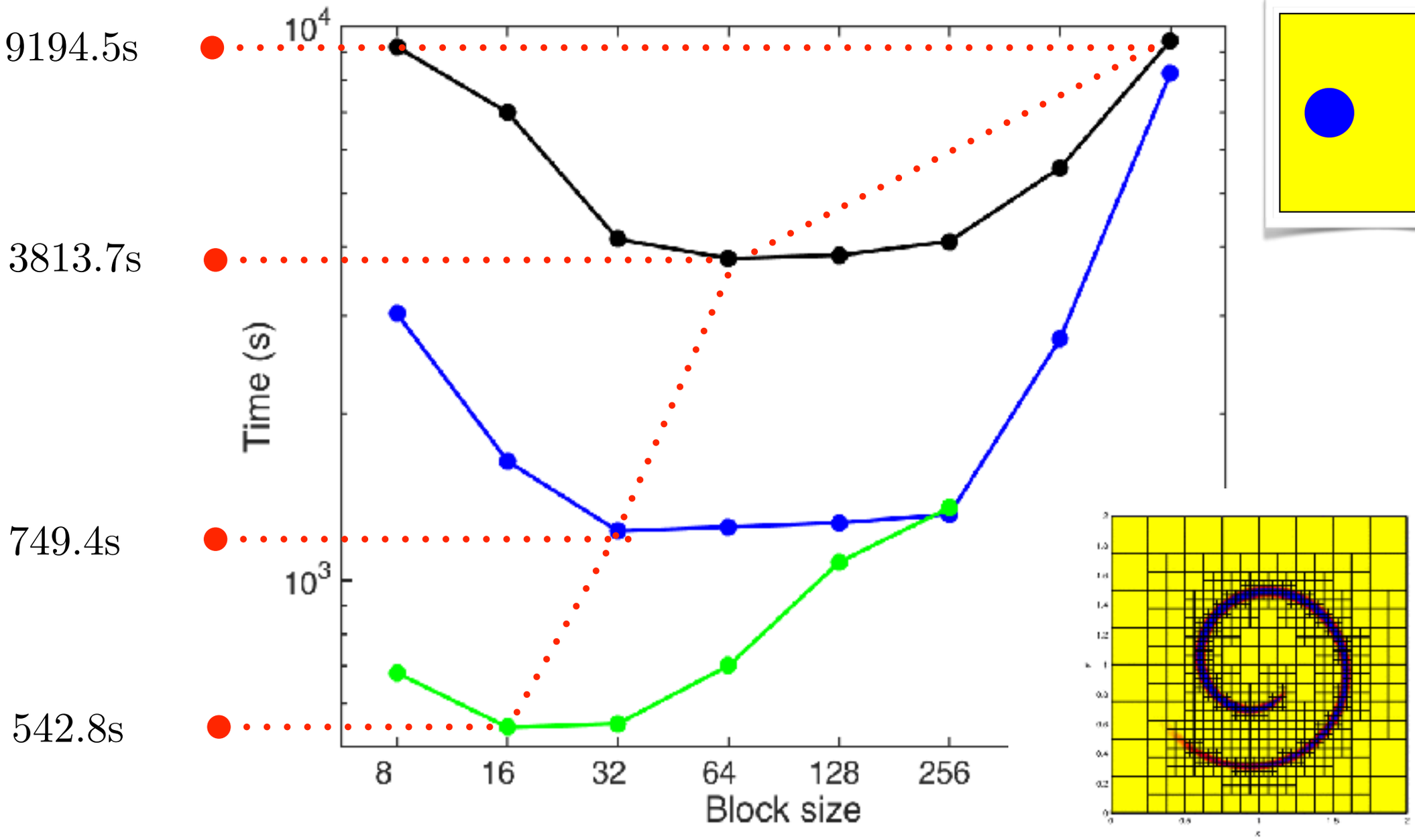
Spatial adaptivity



Tracer transport in smooth flow field can form thin filaments

Underlying mesh management done using p4est (C. Burstedde, Univ. of Bonn)

With adaptive mesh refinement



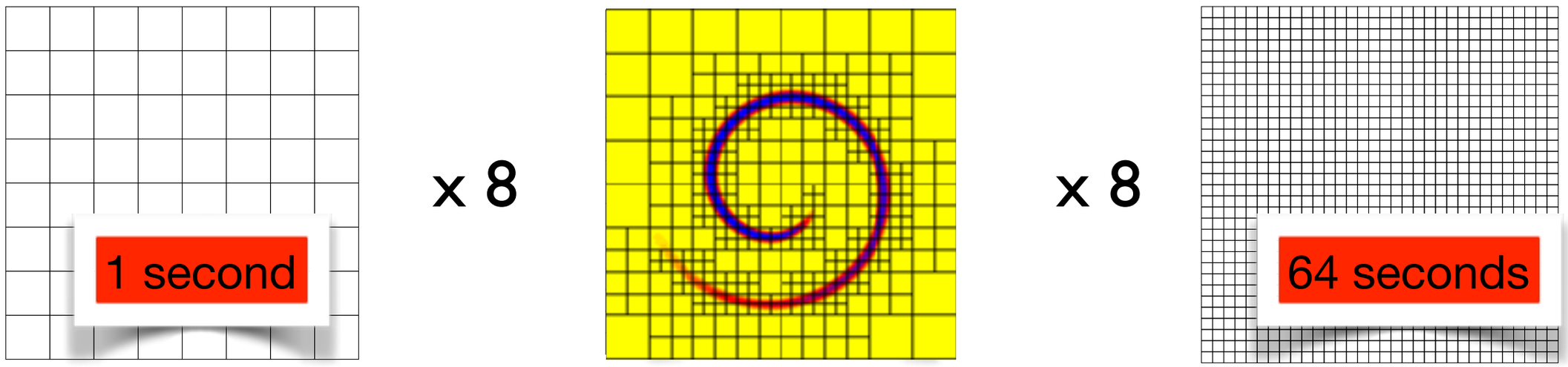
Almost a factor of 15 improvement !

One more tweak …

In two dimensions, if we double the resolution, we increase the computational cost by a factor of 8

- Number of grid cells increases by a factor of 4
- Number of time steps increases by a factor of 2

The CFL condition constrains the size time step we can take on a grid. For stability, smaller grid cells need smaller time steps.



Computational performance

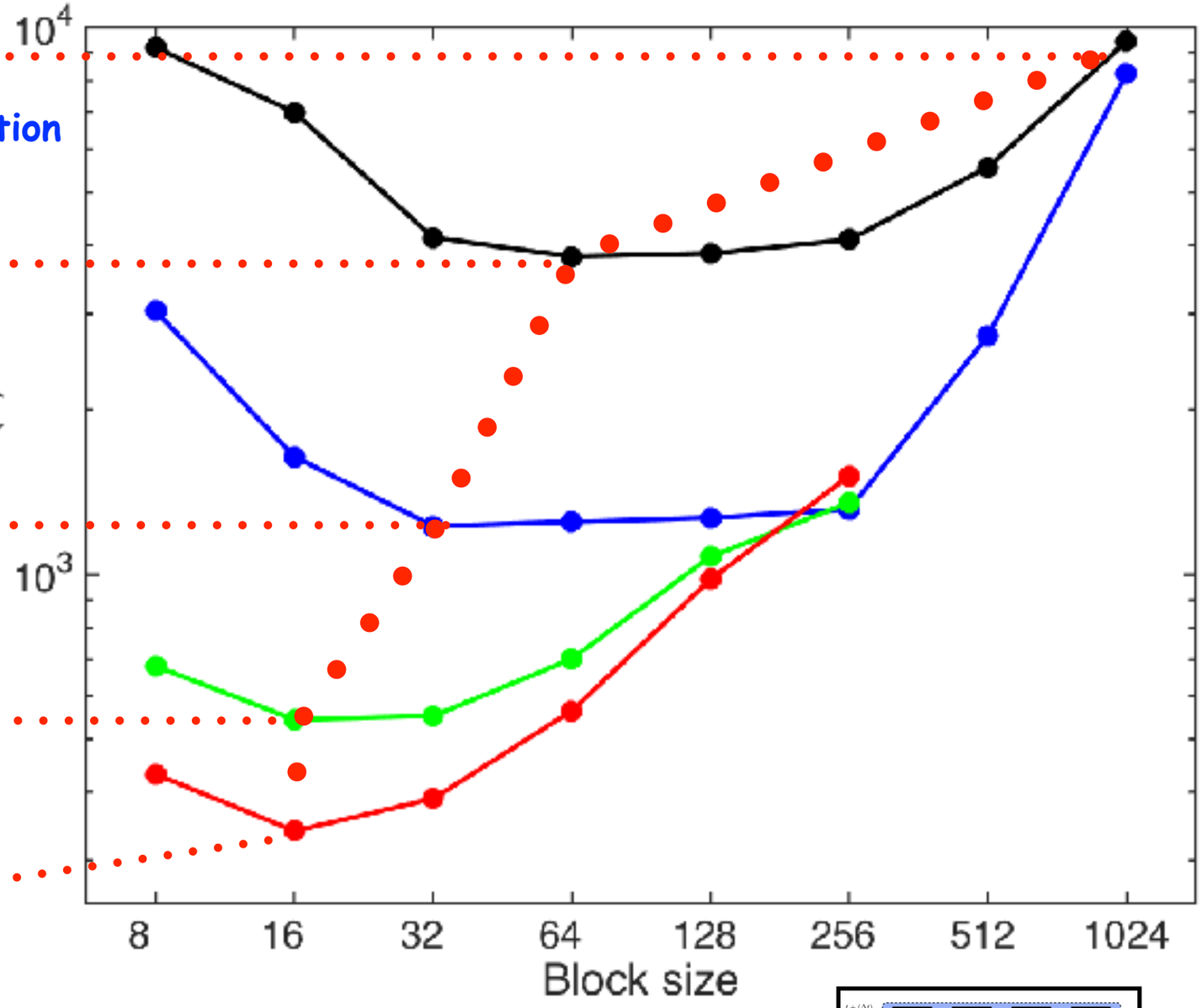
~2.5 h
Single grid calculation

~1 h
Uniformly refined

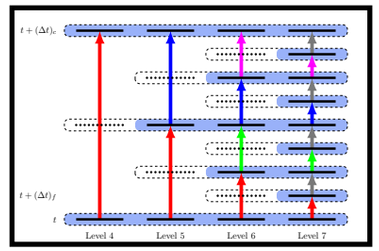
~12.5 min
Parallelism

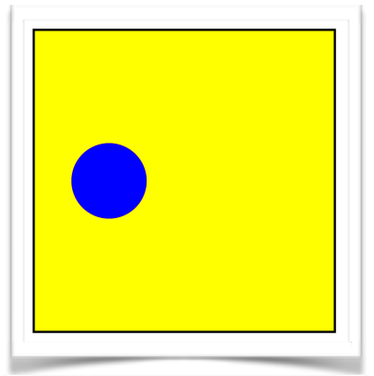
~9 min
Spatial adaptivity

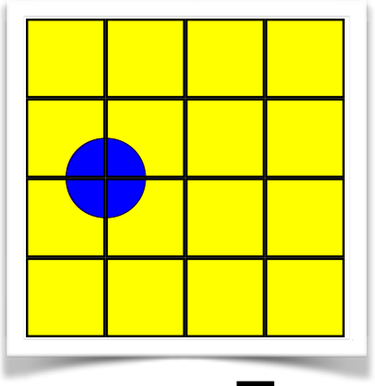
~5.5 min
Adaptive time stepping

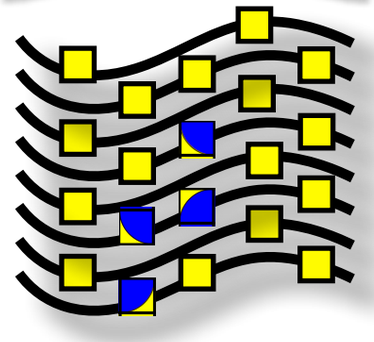


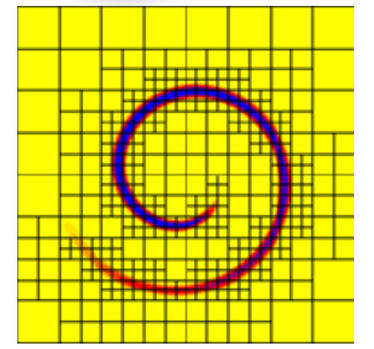
Almost 30 times improvement







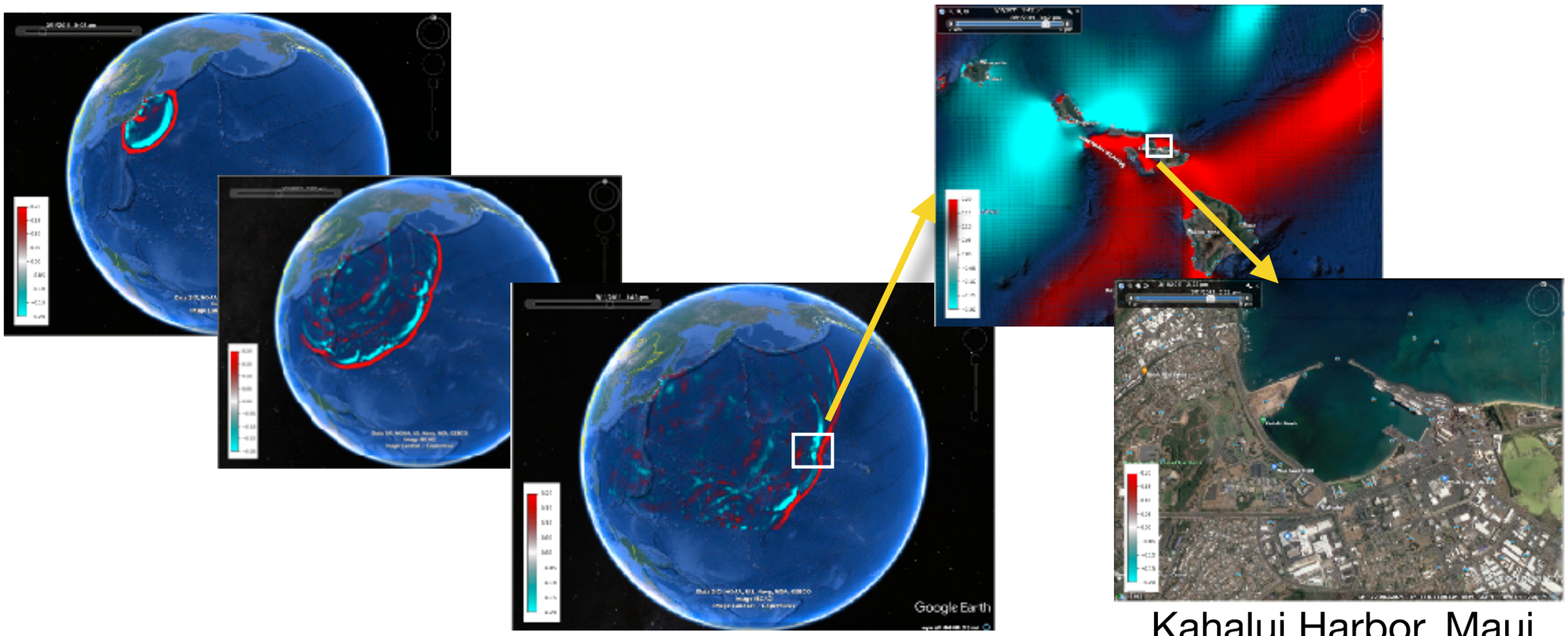




Case study 2 : SWE with bathymetry

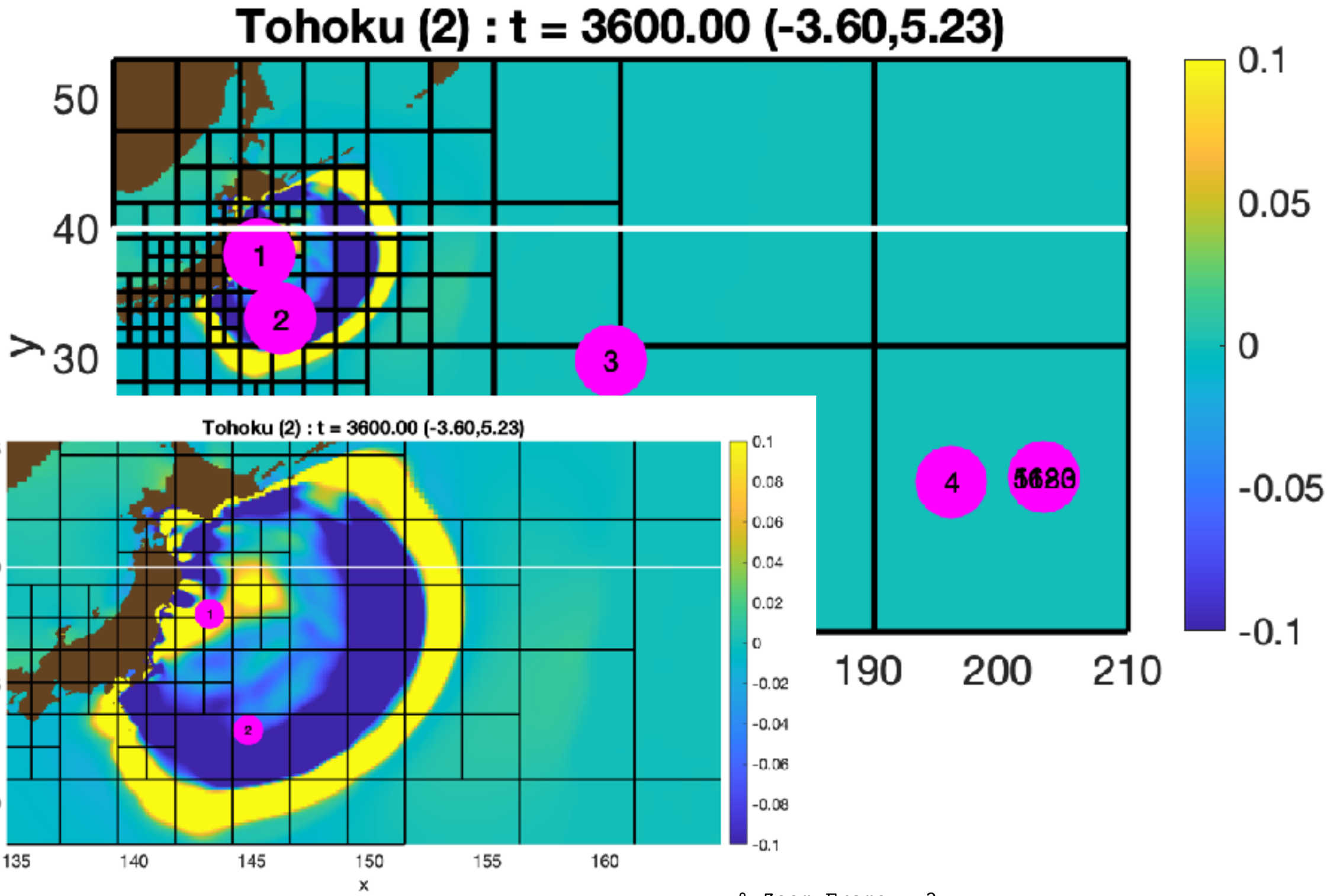
Using GeoClaw (www.geoclaw.org) extension of ForestClaw, we simulate the 2011 Tohoku tsunami.

- Length scales on the order of 6200km; ocean depth approximate 4km, so shallow water model is appropriate
- Tsunami started in Japan; goal is to model effects in Kahului Harbor in Maui



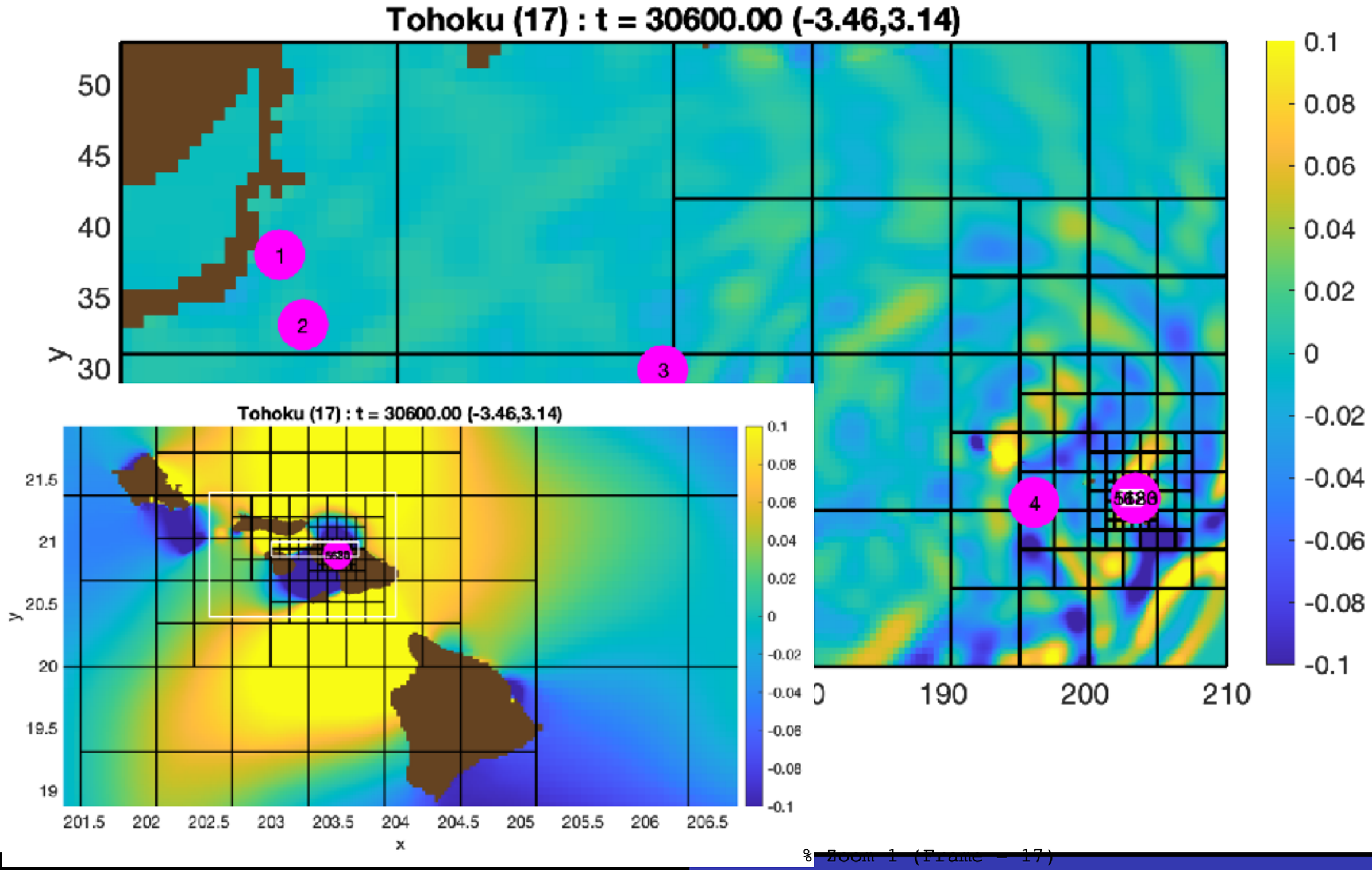
Kahalui Harbor, Maui

Tohoku 2011

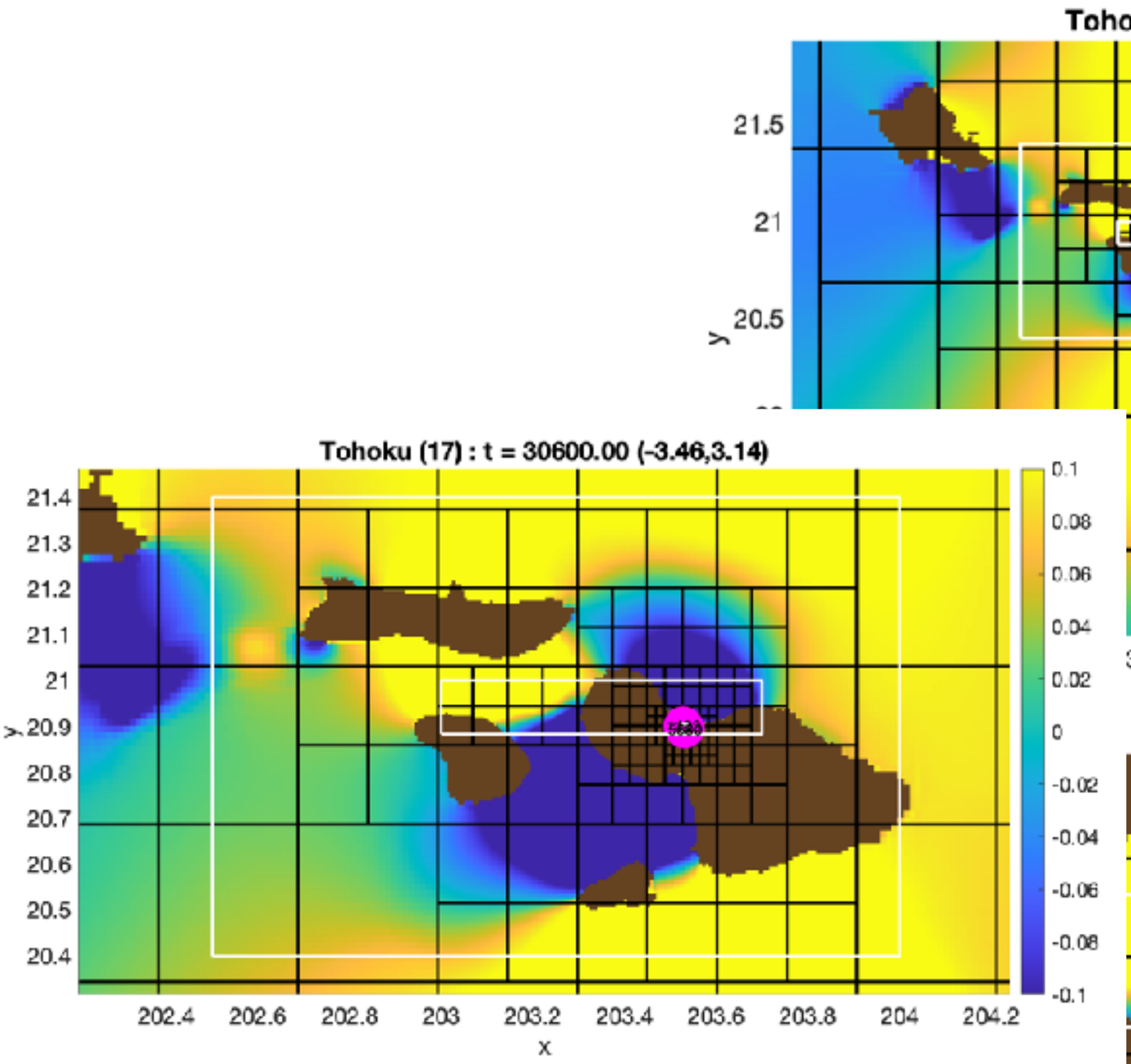


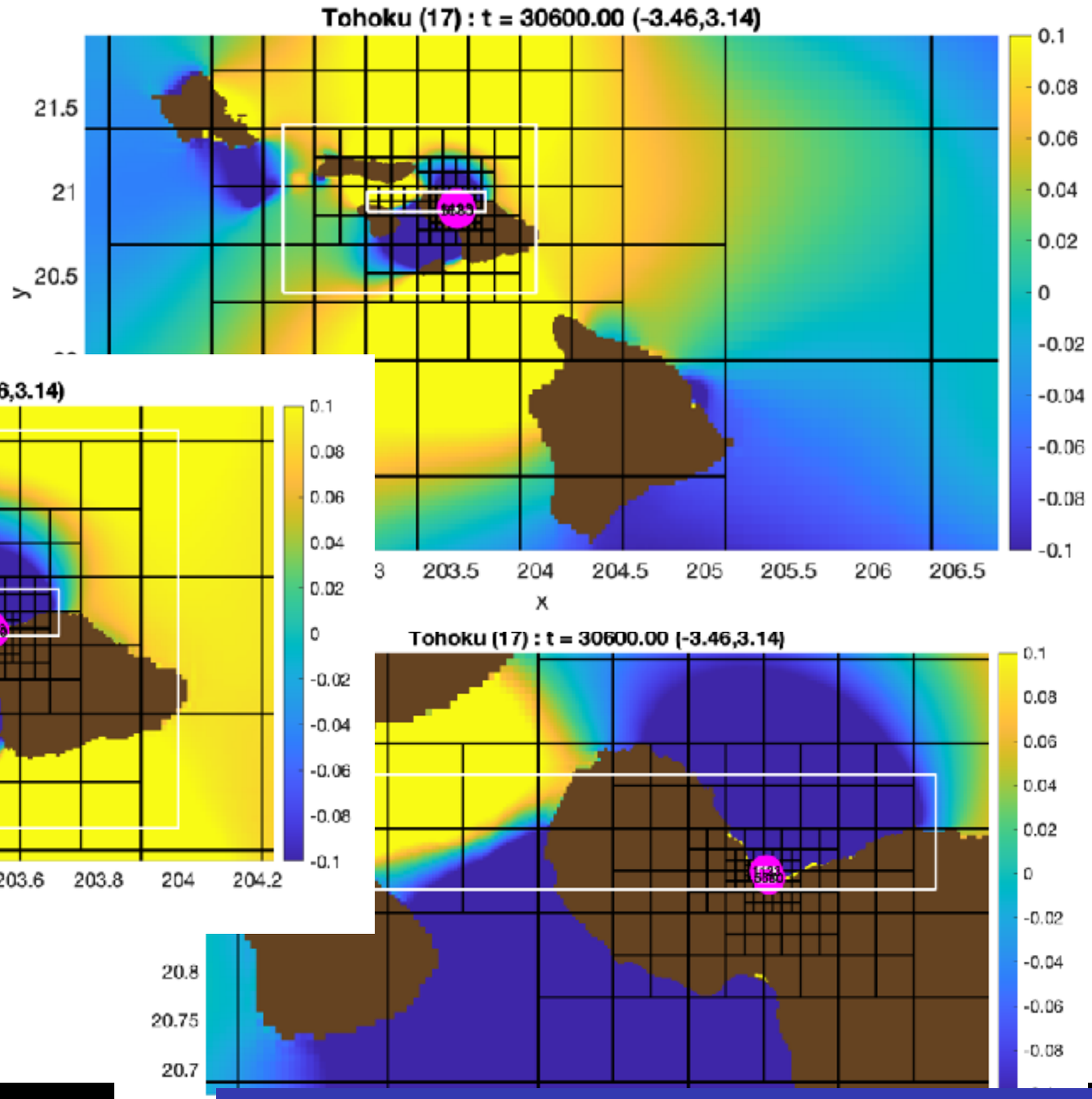

```
% Zoom Frame = 2
% axis([134.6060,  164.9044,  28.2413,  45.3327])
```

Tohoku 2011

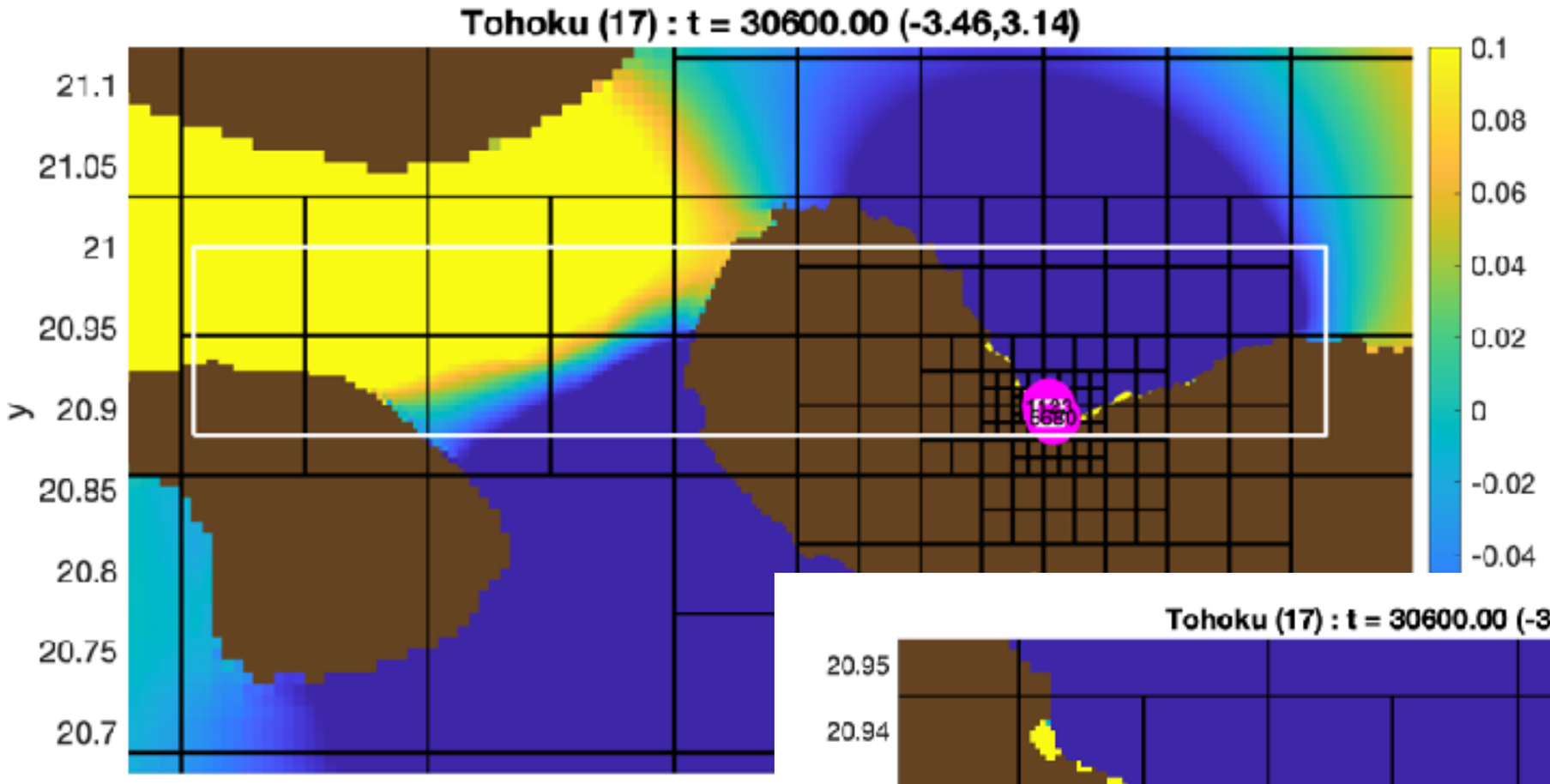

Tohoku (17) : t = 30600.00 (-3.46,3.14)
50
45
40
35
30
y
1
2
3
4
190
200
210
0.1
0.08
0.06
0.04
0.02
0
-0.02
-0.04
-0.06
-0.08
-0.1
Tohoku (17) : t = 30600.00 (-3.46,3.14)
21.5
21
20.5
20
19.5
19
y
201.5
202
202.5
203
203.5
204
204.5
205
205.5
206
206.5
x
% Zoom 1 (Frame = 17)
% axis([201.3305, 206.7376, 18.8768, 21.9269]);

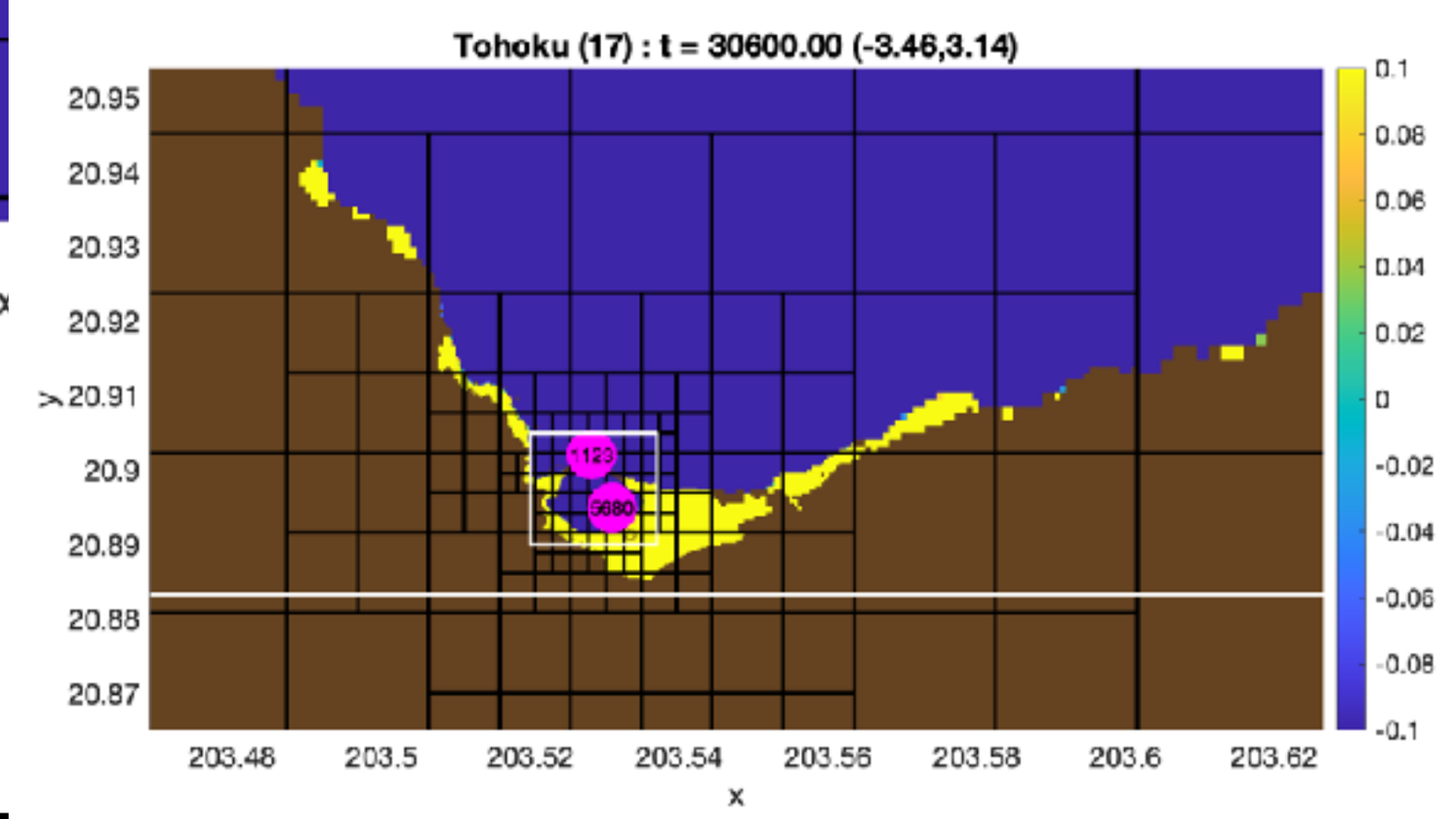
Tohoku 2011


Tohoku (17) : t = 30600.00 (-3.46,3.14)
21.4
21.3
21.2
21.1
21
20.9
20.8
20.7
20.6
20.5
20.4
y
202.4
202.6
202.8
203
203.2
203.4
203.6
203.8
204
204.2
x
0.1
0.08
0.06
0.04
0.02
0
-0.02
-0.04
-0.06
-0.08
-0.1


Tohoku (17) : t = 30600.00 (-3.46,3.14)
21.5
21
20.5
y
203.5
204
204.5
205
205.5
206
206.5
x
0.1
0.08
0.06
0.04
0.02
0
-0.02
-0.04
-0.06
-0.08
-0.1
Tohoku (17) : t = 30600.00 (-3.46,3.14)
20.8
20.75
20.7
0.1
0.08
0.06
0.04
0.02
0
-0.02
-0.04
-0.06
-0.08

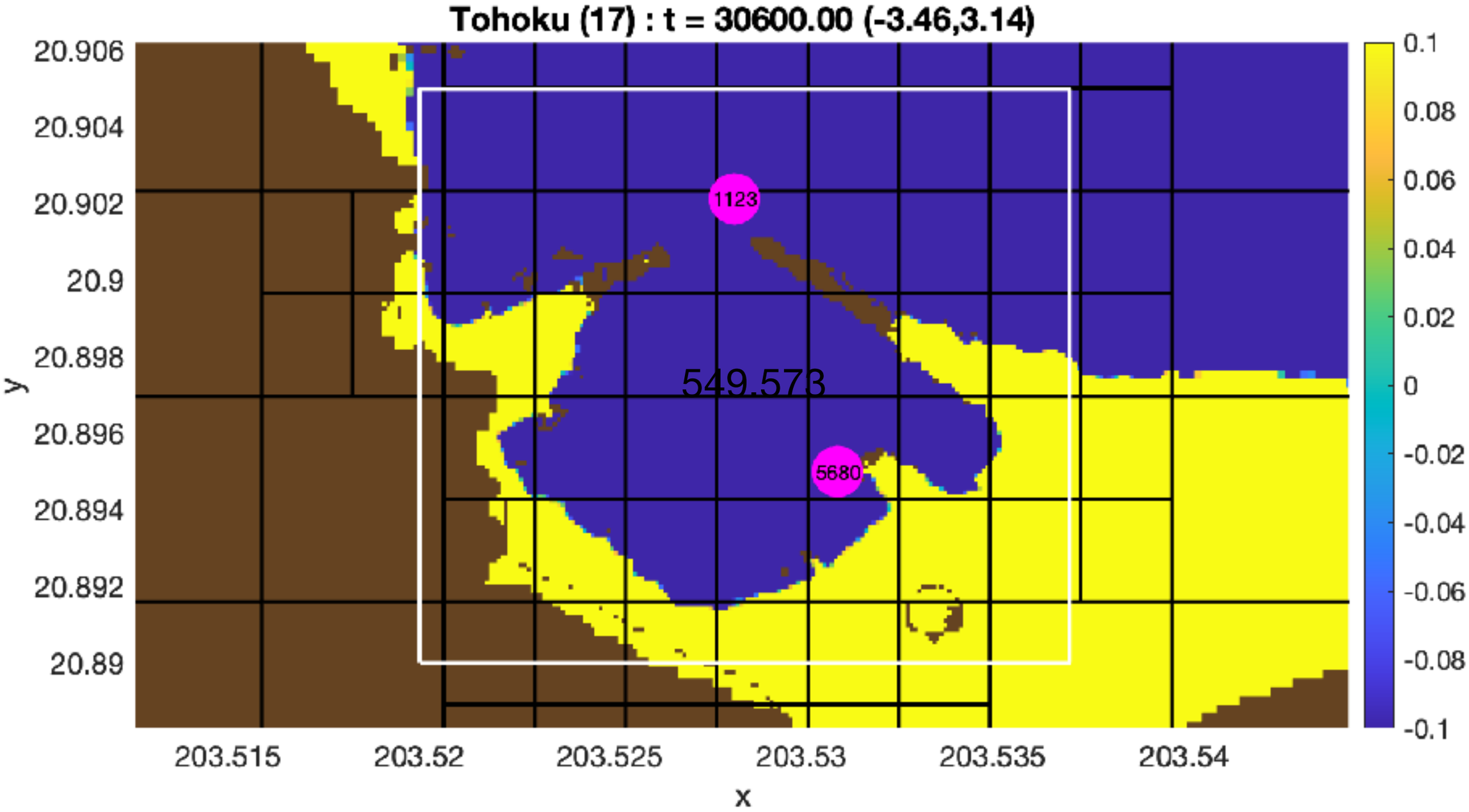
Tohoku 2011





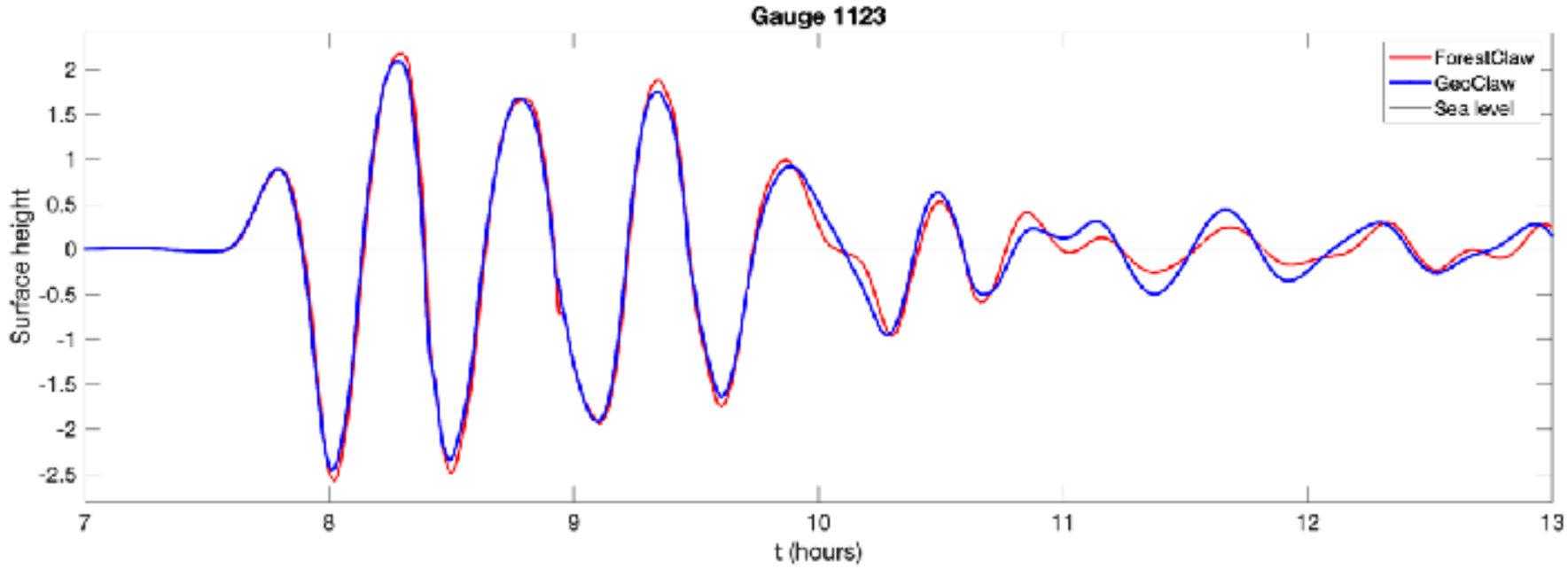
& Zoom 4 (Frame 17)

Tohoku 2011



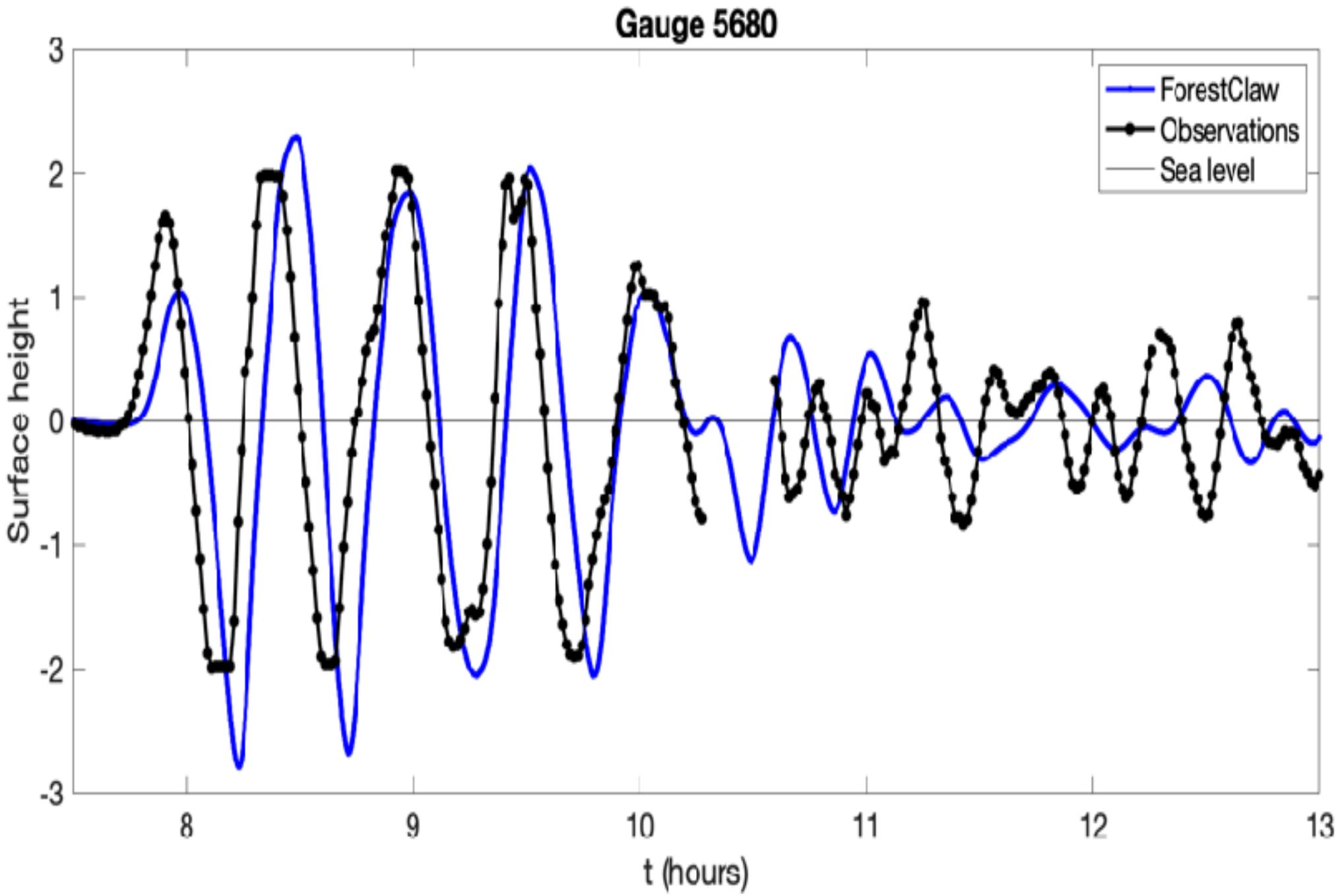
Kahalui Harbor, Maui

Comparison with GeoClaw



Gauges in Kahalui Harbor, Maui

Comparison with observations



Multi-rate time stepping with SWE

Time step is constrained by wave speed $\sim \sqrt{gh}$, where h is the fluid depth.

- Wave speeds are faster in deep water than in shallow water.
- But shallow water is exactly where we want to place our finest grids.

Competition between fast wave speeds on coarser grids vs slower wave speeds on finer grids. Resulting CFL in shallow and deep water : α_s and α_d

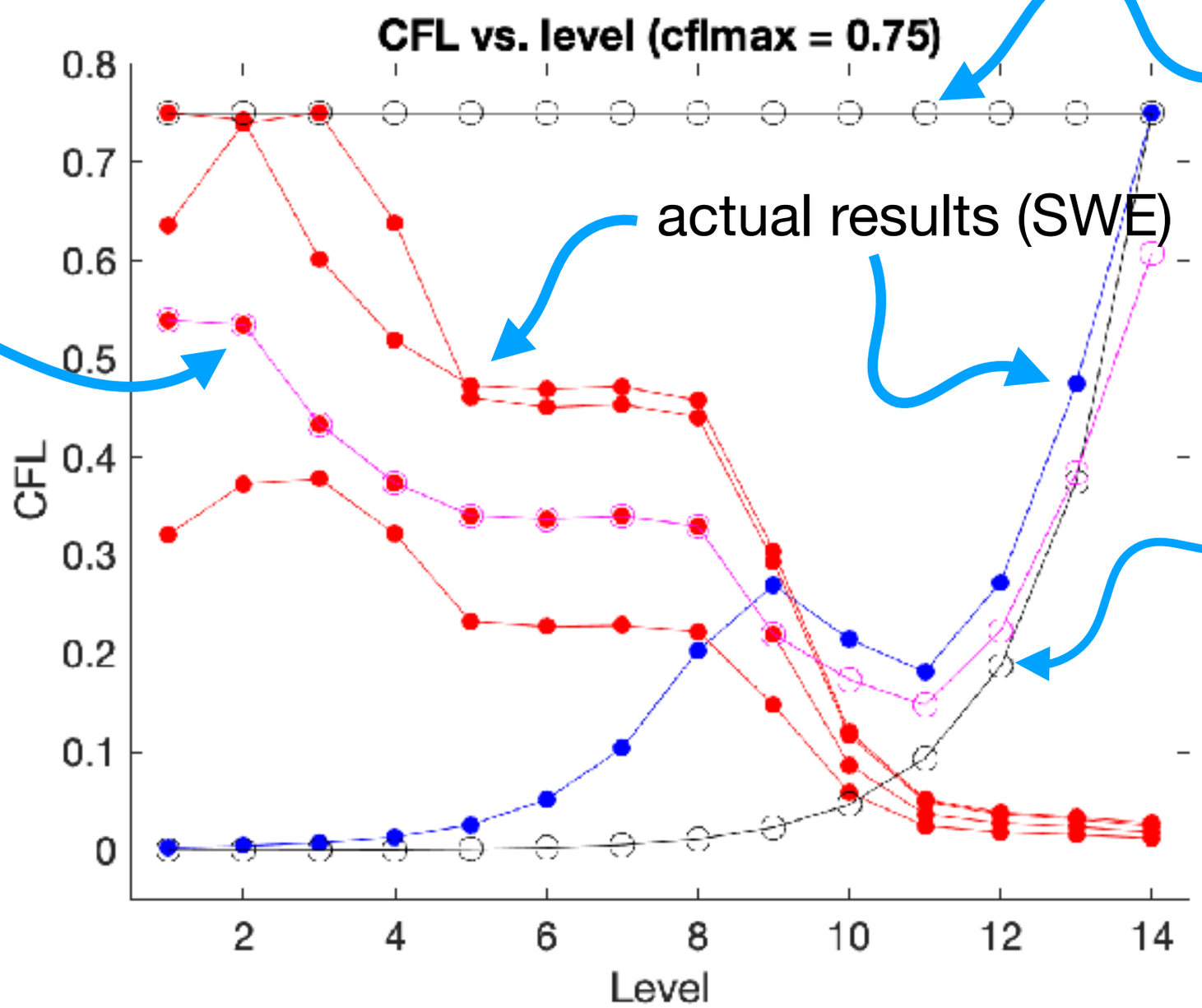
$$2^{\ell_{max}-\ell_{min}}\sqrt{\frac{h_s}{h_d}} < 1 \quad \text{Deep water controls the time step; } \alpha_s \ll 1$$

$$2^{\ell_{max}-\ell_{min}}\sqrt{\frac{h_s}{h_d}} > 1 \quad \text{Shallow water controls the time step; } \alpha_d \ll 1$$

No longer makes sense to simply reduce the spatial and temporal scales by same factors

- A simple multirate strategy may actually be slower than a global time stepping strategy, since many steps may have to be retaken if CFL exceeds 1 in any step.

Global time stepping vs. sub cycling



Idealized (constant u) subcycling - CFL is constant and all levels.

actual results (SWE)

Attempt to **switch** from subcycling to global time stepping at level 10 (only moderately successful)

Idealized global time stepping. Full CFL on fine grids; reduced temporal resolution on coarse grids.

Using temporal ratios equal to spatial refinement ratios (2 for ForestClaw) will likely result in reduced temporal resolution (very low CFL) on fine grids. Actual results used both **subcycling** (red) and **global time stepping** (blue)

Flexible multi-rate time stepping

- A naive multirate time stepping strategy can wreak havoc on AMR workflow. Too many steps may need to be retaken.
- GeoClaw (which allows for variable refinement ratios) has some support for a more flexible multirate time stepping.
- ForestClaw implementation of GeoClaw only supports spatial refinement of two, so has less flexibility in multi-rate schemes.

Tohoku timing results (single core/thread)

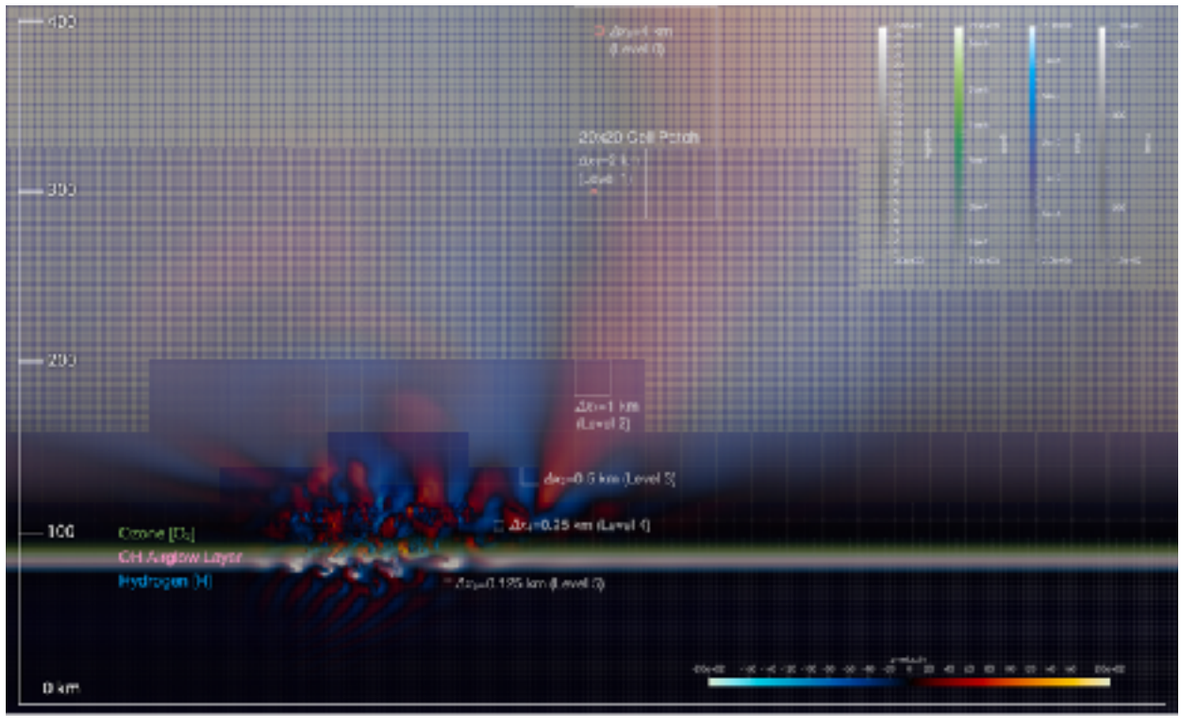
	Number of cells advanced	Walltime	Factor	time/cell
GeoClaw	6.02E+09	5497s	1x	9.13E-07
ForestClaw (global)	1.47E+10	15023s	2.7x	1.02E-06
ForestClaw (MR)	2.33E+10	19635s	3.6x	8.43E-07

$> 1 =$ slower

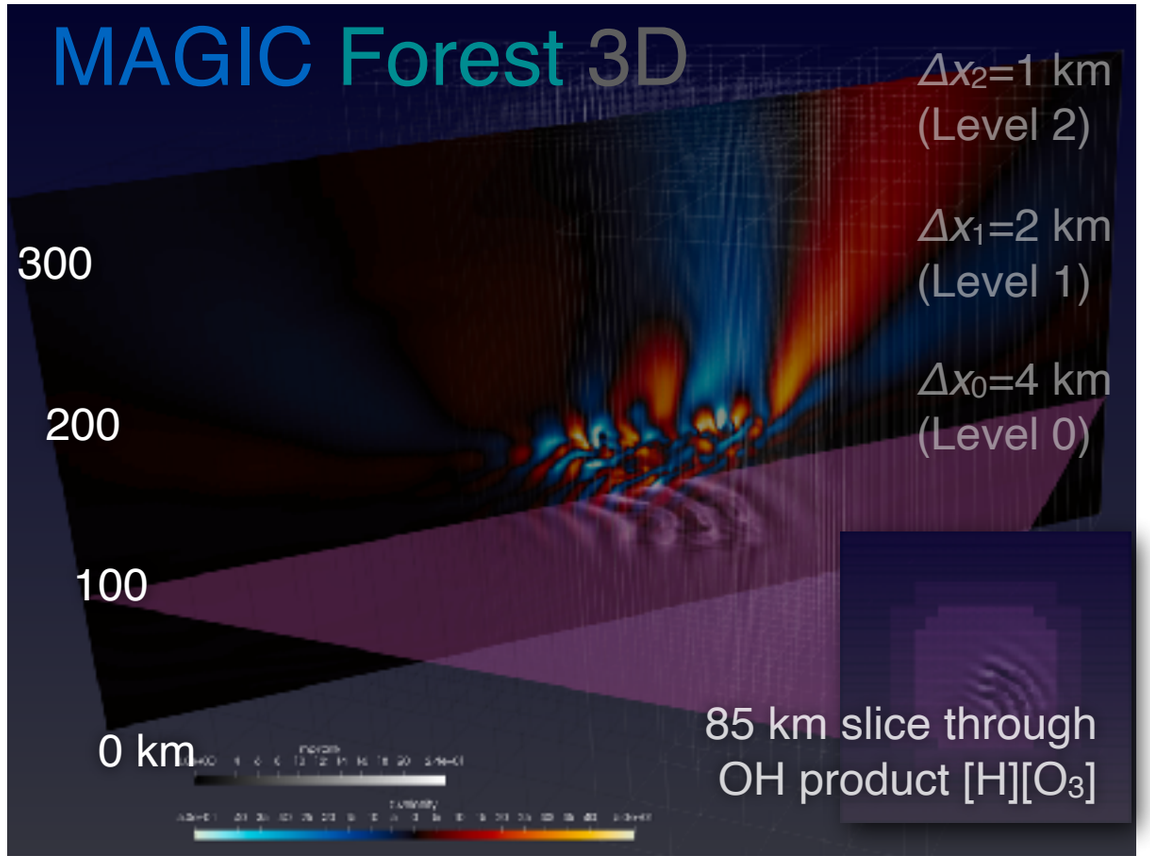
More flexible time stepping for AMR? Discrete Event Simulation (M. Bremer, J. Bachan, C. Chan, C. Dawson)

Case study 3 : Acoustic Gravity Waves

- ForestClaw is now being used as part of a DARPA funded project *AtmosSense* to detect signals in the upper atmosphere triggered by natural hazards (tsunamis, earthquakes, storms).
- Collaboration with Embry-Riddle Aeronautic University. J. Snively (ERAU, PI); Matt Zettergren (ERAU); Michal Hirsch (Boston U); Scott Aiton (BSU) C. Burstedde (Univ. of Bonn) and many others



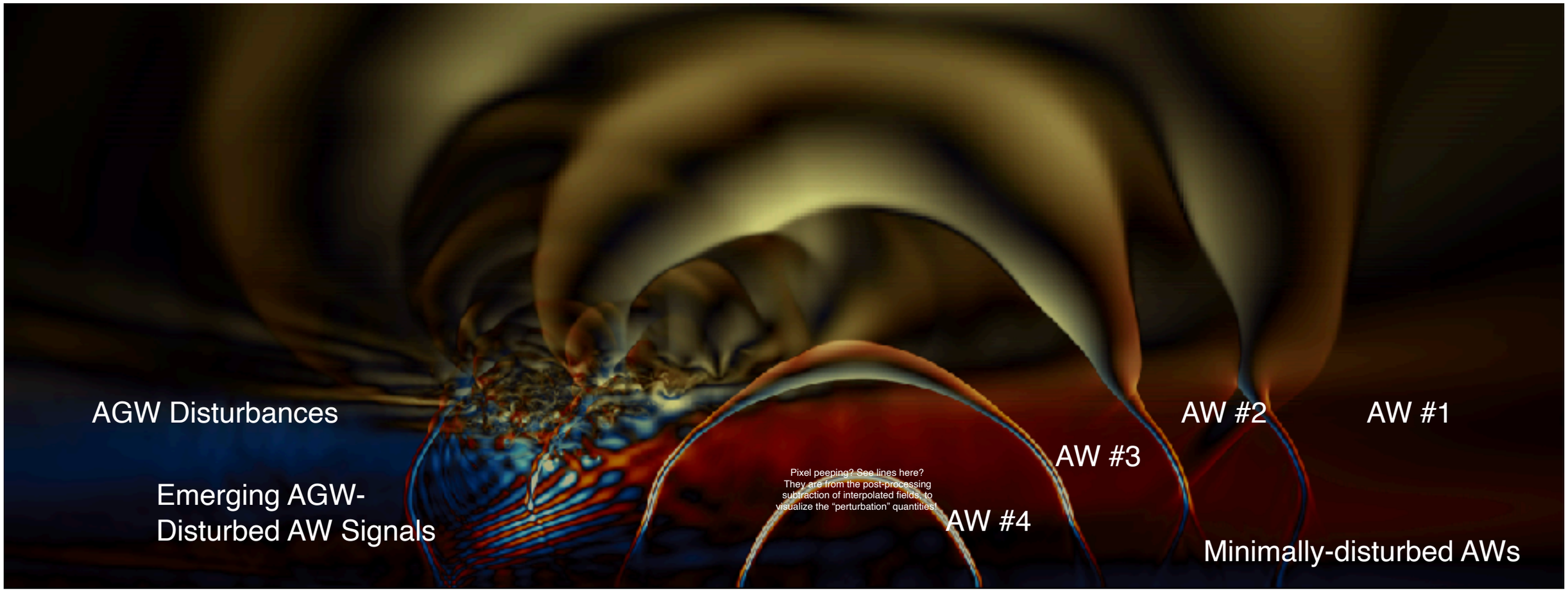
2d: 200x speed-up using MagicForest



3d ForestClaw (extruded mesh)

images : Jonathan Snively (AGU Fall 2021 Meeting)

AirWaves : Acoustic Gravity Wave Interaction



Energy-release source (Sabatini et al., 2019), triggered 4 times at 200 s intervals in Demo AGW field (2D) to address question of “How do ambient wave fields influence AW signals of natural hazard events?”

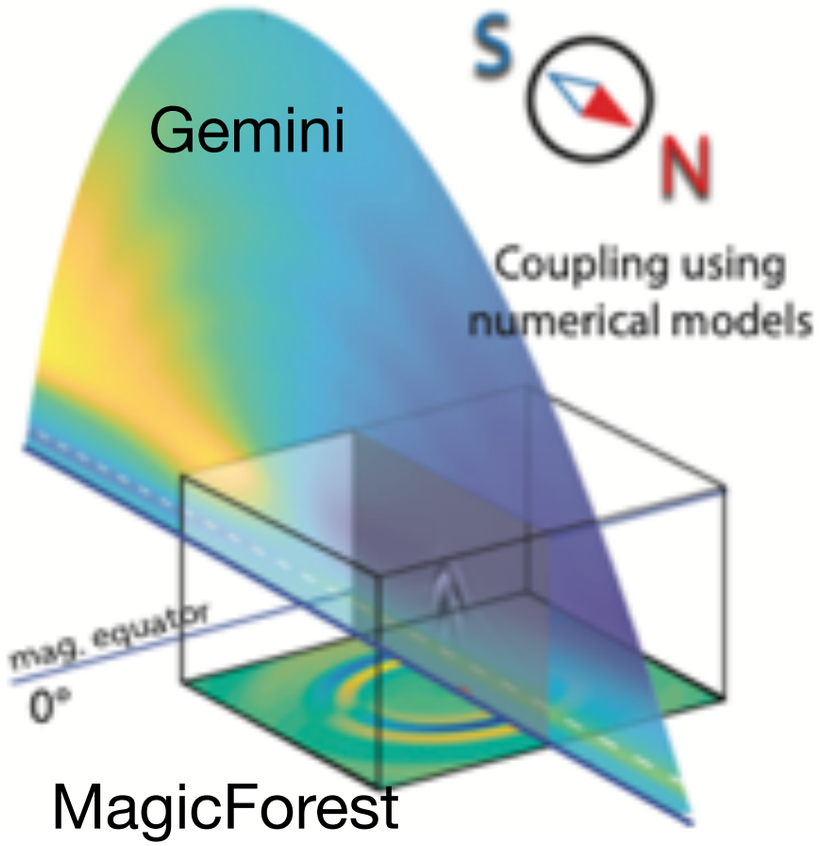
DARPA

images and caption : Jonathan Snively (Poster : AGU Fall 2021 Meeting)

AirWaves : Acoustic Gravity Wave Interaction

Current goal of this project :

- Use ForestClaw to couple MAGIC Forest code (MAGIC = **M**odel for **A**coustic **G**ravity wave **I**nteractions and **C**oupling, J. Snively, ERAU) with ionosphere code GEMINI (**G**eospace **E**nvironment **M**odel of **I**on-**N**eutral **I**nteractions, M. Zettergren, ERAU).



Images : Jonathan Snively

Challenges :

- Factoring GEMINI into well-defined callback functions
- Separate GEMINI MPI from ForestClaw MPI
- Developing build system to two codes can compile the two codes together
- Use ThunderEgg (BICG + multigrid pre-conditioner, S. Aiton) for elliptic solves
- Multiple 2d/3d p4est meshes will need to communicate.
- Need to treat thermal diffusion in MAGIC Forest efficiently.
- Local time stepping will be crucial

Conclusions

Original Berger-Oliger-Collela sub cycling strategy may need to be updated to handle wider variety of problems.

- This presents a programming challenge
- Local time stepping will be critical, though.
- How do we couple elliptic solves with multi-rate?

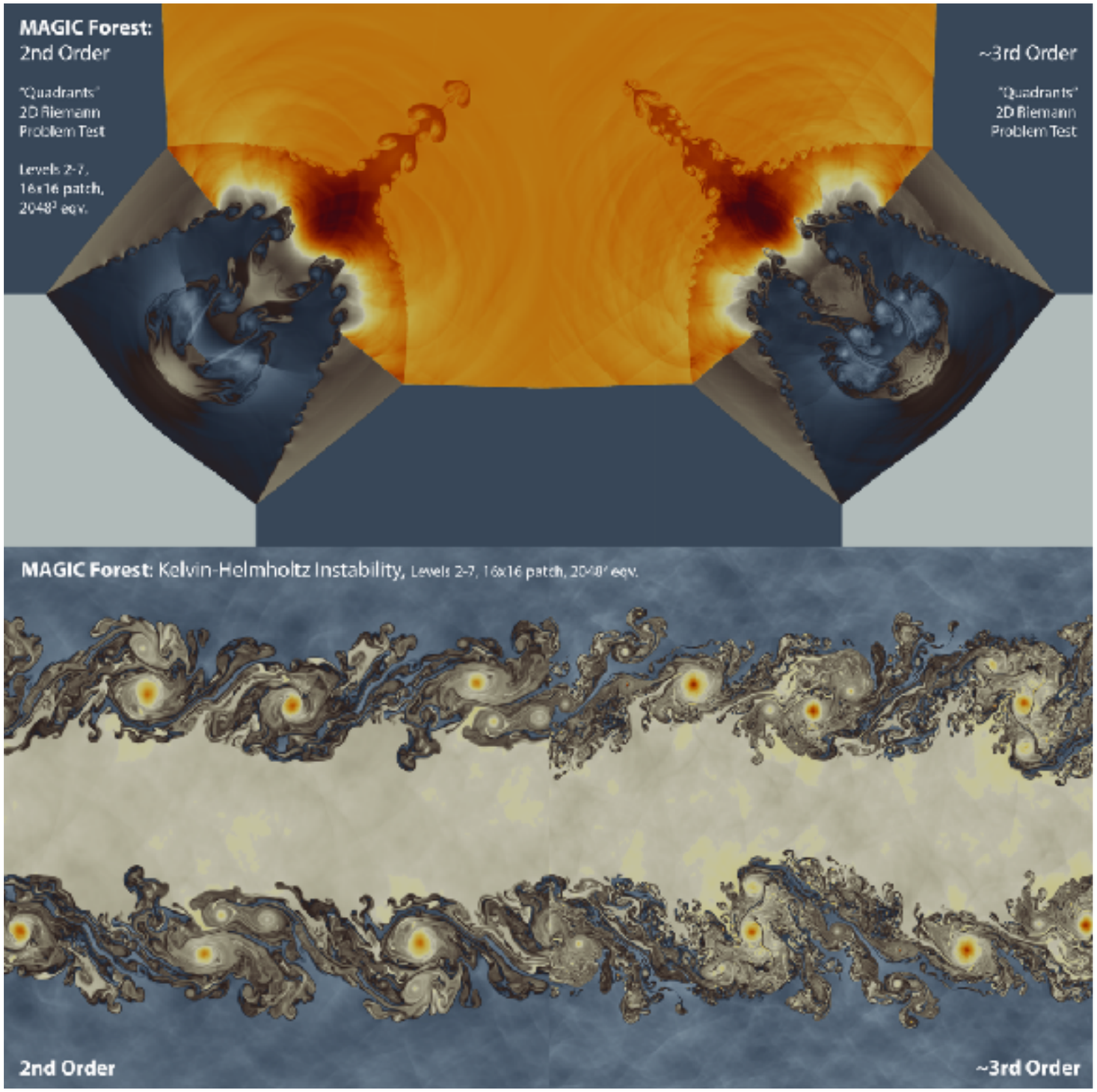
ForestClaw

- Robust funding, especially from DARPA and NASA has allowed for continued development.
- Currently, we are supporting four PhD students, plus a full time research assistant (Scott Aiton)
- Support for full octree AMR is next big project.

Other projects :

- Wildfire smoke modeling (inverse modeling); Patricia Azike and Sandra Babyale (BSU)
- Direct elliptic solver using Henri-Poicare-Steklov (HPS) solver (A. Gillman and G. Martinnson) solver; Damyn Chipman (BSU)
- Serre-Green-Naghdi solver (Collaboration with M. Berger and R. J. LeVeque)
- Coupling GeoClaw with MAGIC ForestClaw : Brian Kyanjo (BSU)
- Iterative elliptic solver (ThunderEgg, S. Aiton).

AirWaves : Magic Forest



Images : Jonathan Snively