Adaptive Mesh Refinement for Geodynamics Applications

Wolfgang Bangerth
Department of Mathematics
Texas A&M University

Martin Kronbichler
Uppsala University
Sweden

Carsten Burstedde, Omar Ghattas, Georg Stadler, T-K Tu, Lucas Wilcox
Institute for Computational Engineering and Sciences
University of Texas at Austin

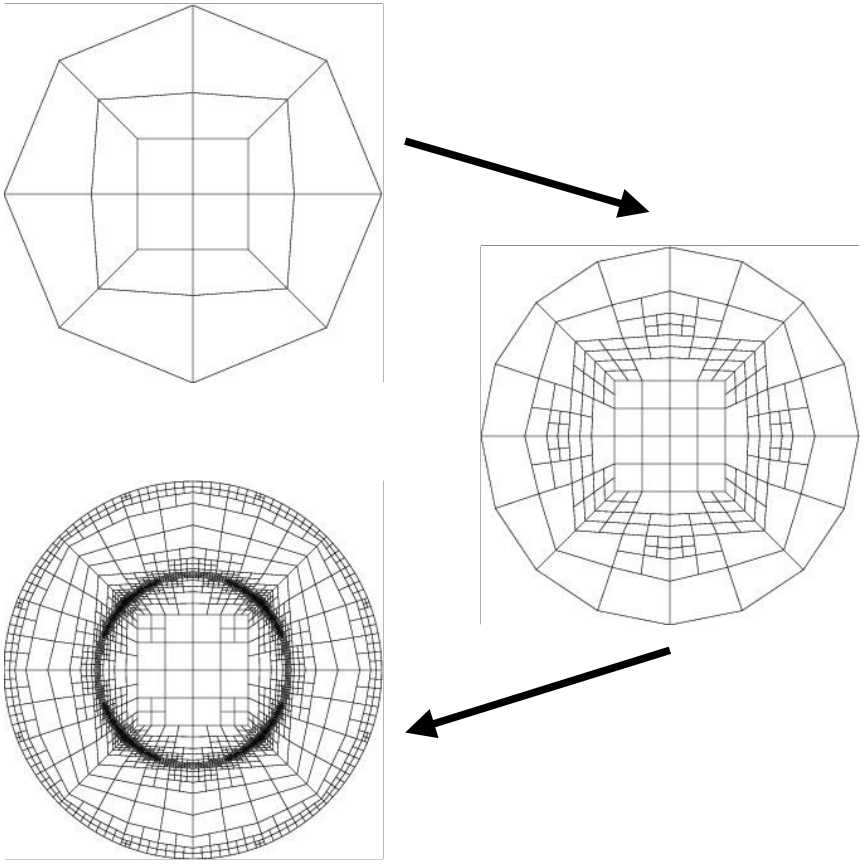
Mike Gurnis, Eh Tan
CalTech

Shijie Zhong
University of Colorado, Boulder

The Adaptive Paradigm

Philosophy of local mesh refinement:

- Solve on a rather coarse grid
- Compute an error criterion
- If error < tolerance, then stop
- Otherwise refine mesh
- Solve again on finer grid



Advantage: We can use meshes adapted to the solution and/or what we are interested in

Disadvantage: We have to solve more than once, and we need more sophisticated algorithms

Questions About Adaptivity

- **Will we gain anything?** This depends on
  - whether we need meshes fitted to geometric features
  - whether we need fully adapted time varying meshes
  - the type of the equation

- **How can we generate adaptive meshes?**
  - mesh generators
  - adaptive mesh refinement using error estimators and indicators

- **How to use them in our codes?**
  - What do we need for existing codes?
  - What do we need for new codes?

- **Parallelization and load balancing issues for adaptive meshes**

When do we gain by using Adaptivity?

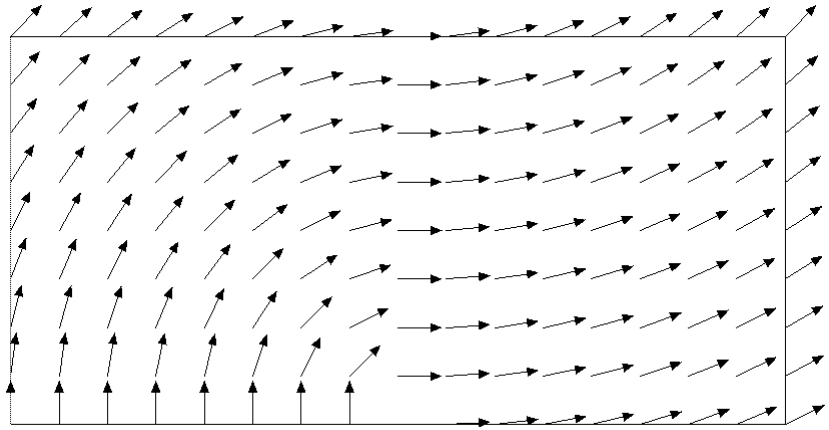
- Adaptive meshes can be beneficial because they promise to reach the same accuracy with (orders of magnitude) less effort
- Can focus degrees of freedom where the solution shows significant variation, achieving resolution otherwise impossible
- Avoid generation of fine meshes (avoids scalability, bad shapes) by generating them as necessary from coarse meshes

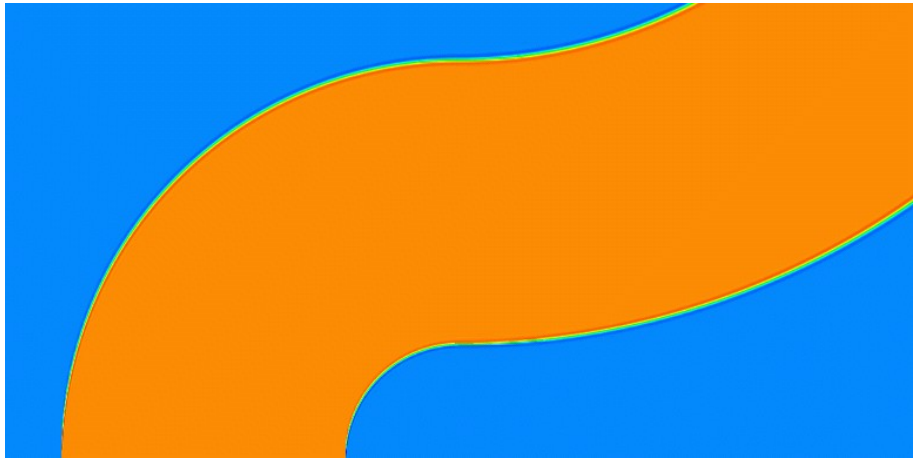
Adaptivity is good if and only if
"the action is localized"

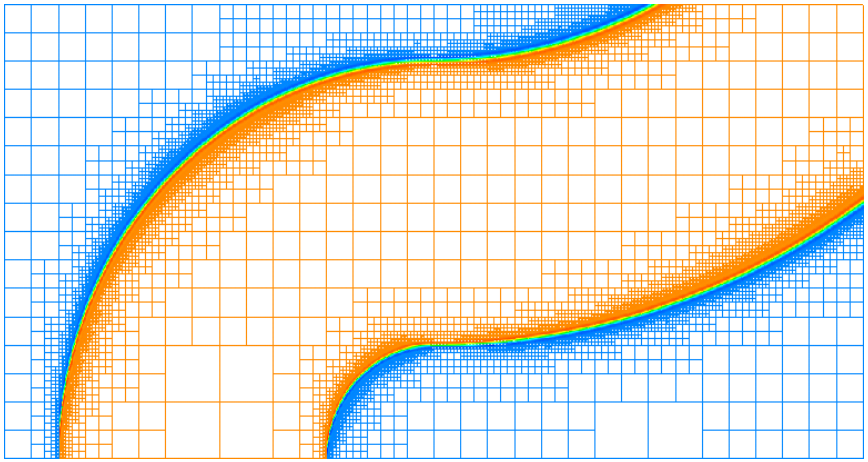
Fortunately, that's frequently the case in geodynamics!

When do we gain by using Adaptivity?

Positive example: Advective transport in a given wind field (Geophysical analogy: advection by subducting slabs)

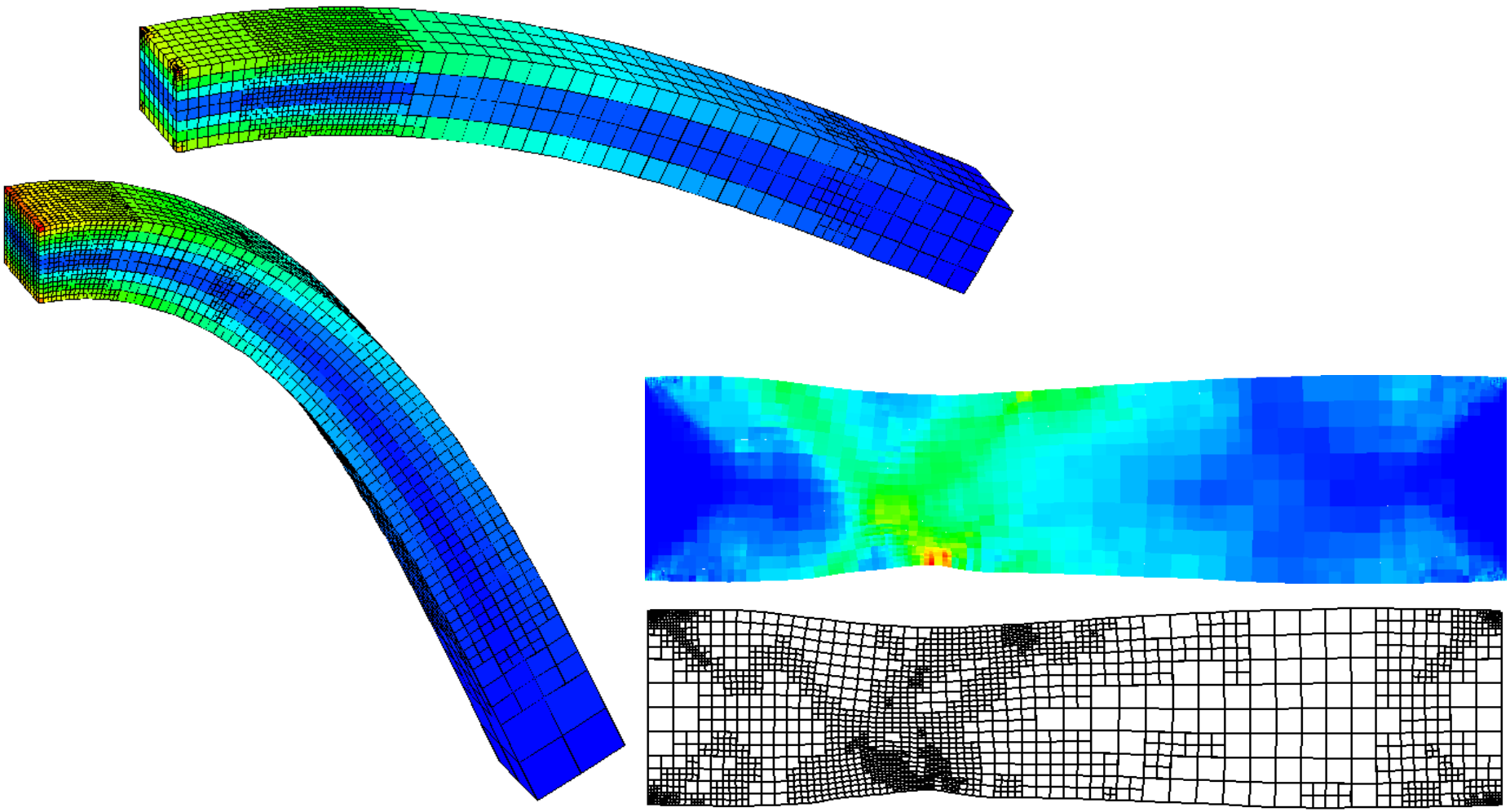






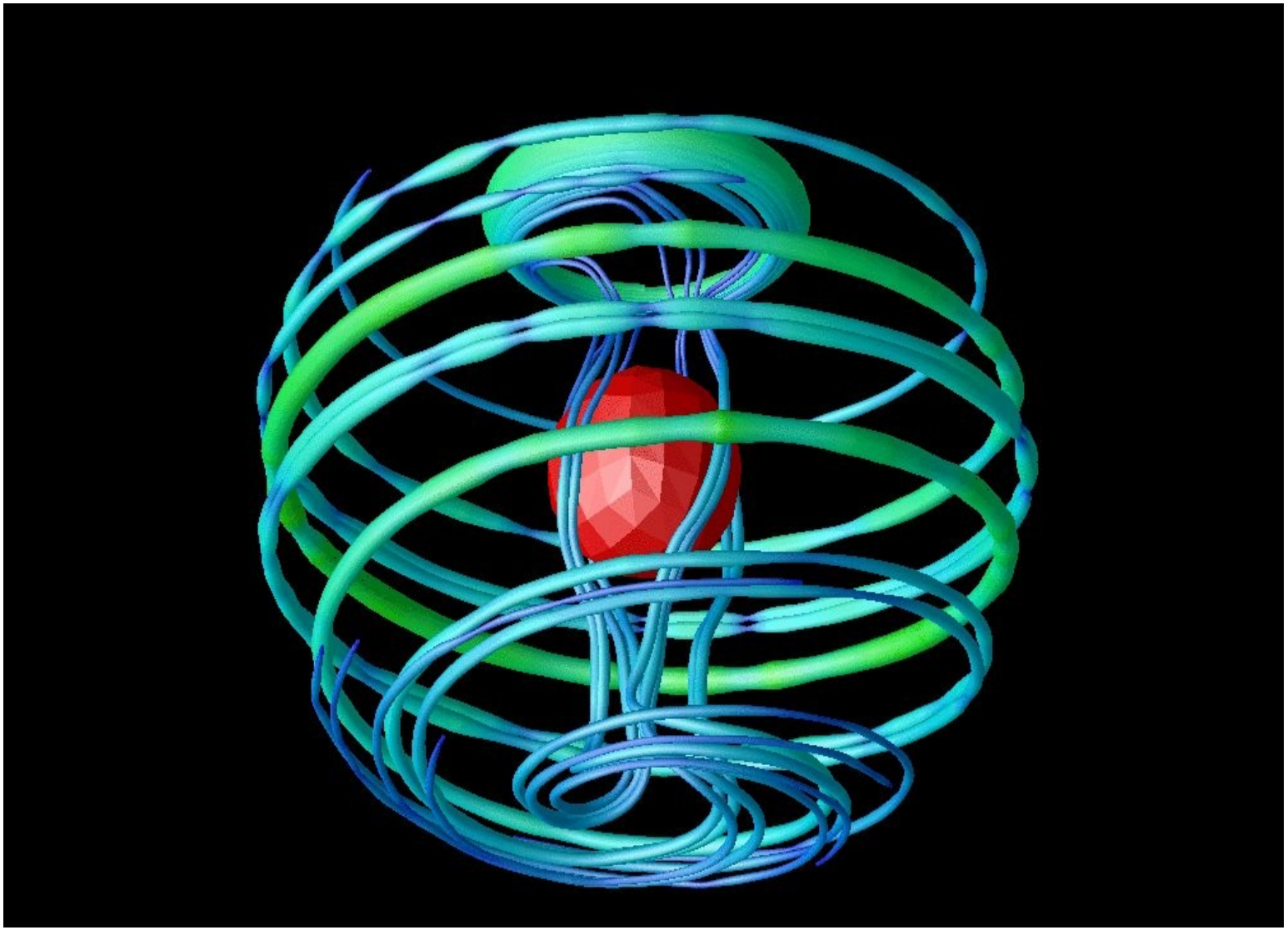
When do we gain by using Adaptivity?

Positive example: Elastoplastic deformation
(Geophysical analogy: Long-term continental deformation, subduction bending of continental plates)



When do we gain by using Adaptivity?

Counterexample: Geodynamo with its global turbulence and small-scale features



I want to use adaptivity in my code!

What to do with existing codes:

- It is hard to convert existing codes to use adaptive meshes because the changes in data structures and algorithms are so pervasive. These changes involve:
  - mesh data structures
  - finite elements/finite difference stencils
  - handling of hanging node constraints
  - linear solvers/preconditioners
  - top-level logic

- It is believed to be simpler to write a new code from scratch than to adapt it

- Rewrite may be less painful than you think because of experience gained from previous codes (e.g.: what discretization, which solvers work, and which don't) and using libraries that support adaptive finite element codes.

How to use adaptive meshes

What we need for new codes:

In addition to a description of the actual application, we need

- Refinement criteria
- Code that transfers the solution from one mesh to another
- Solvers that are robust against widely varying mesh sizes
- Code that can deal with “hanging nodes”

All of this is readily available as building blocks in libraries such as deal.II, PETSc, or Trilinos!

Writing a new code may not require a huge investment:

- Many building blocks are already available
- There may be a a tutorial program that does what you want!

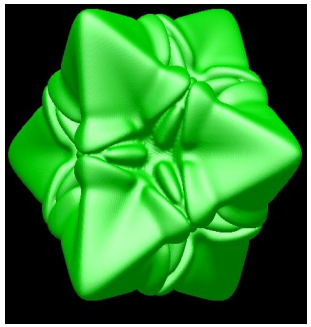
Available resources for new programs

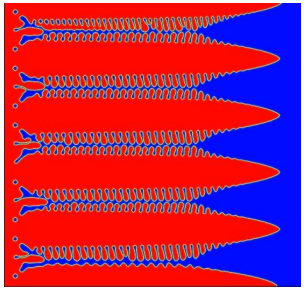
- PETSc (written in C):
  - everything for sequential and parallel linear algebra
  - direct and iterative solvers, algebraic multigrid
  - a bunch of other stuff
- Trilinos (written in C++):
  - everything for sequential and parallel linear algebra
  - direct and iterative solvers, algebraic multigrid
  - nonlinear solvers, automatic differentiation, optimization, ...
- deal.II (written in C++):
  - everything related to meshes, discretizations, etc
  - everything for sequential linear algebra
  - interfaces to PETSc, Trilinos, METIS, UMFPACK, ...
  - huge amount of documentation
  - tutorial programs of realistic complexity

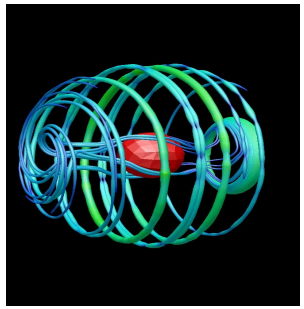
The deal.II library

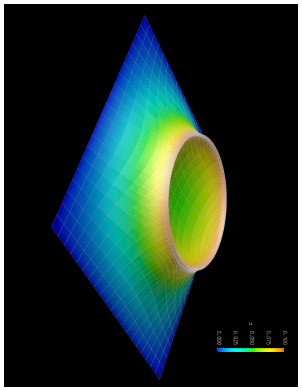
deal.II is a finite element software library:

- Provides support for adaptive meshes in 1d, 2d, and 3d through a unified interface
- Has standard refinement indicators built in
- Provides a variety of different finite element types (continuous, discontinuous, mixed, Raviart-Thomas, ...)
- Low and high order elements available
- Full support for multi-component problems
- Has its own sub-library for dense + sparse linear algebra
- But also comes with interfaces to PETSC, Trilinos, UMFPACK
- Supports SMP + cluster systems (including an interface to METIS)





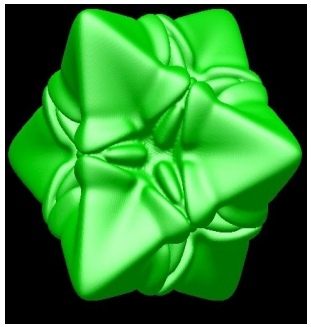


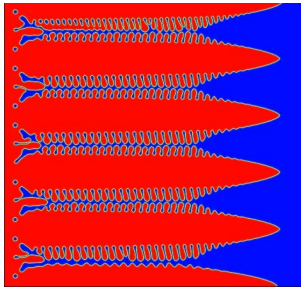


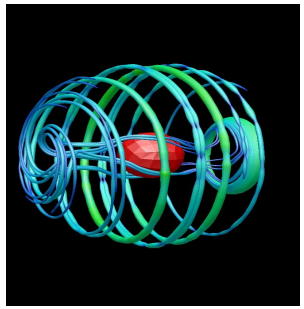
www.dealii.org

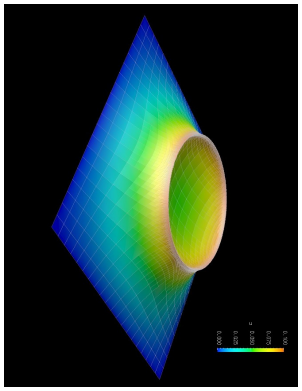
The deal.II library

- Interfaces to all major graphics programs
- Fairly widely distributed in the finite element/adaptivity community:
  - 200 downloads per month
  - 1000+ hits on homepage per month
  - 25-30 publications per year based on deal.II
- Supports a wide variety of applications in all sciences
- Presently over 400,000 lines of C++ code
- More than 5000 pages of documentation
- 32 tutorial programs that explain the use of the library in detail, starting from very simple to parallel quasistatic elasticity/multiphase flow/neutron transport/... applications
- Open Source, active development









www.dealii.org

Examples of recent adaptive applications

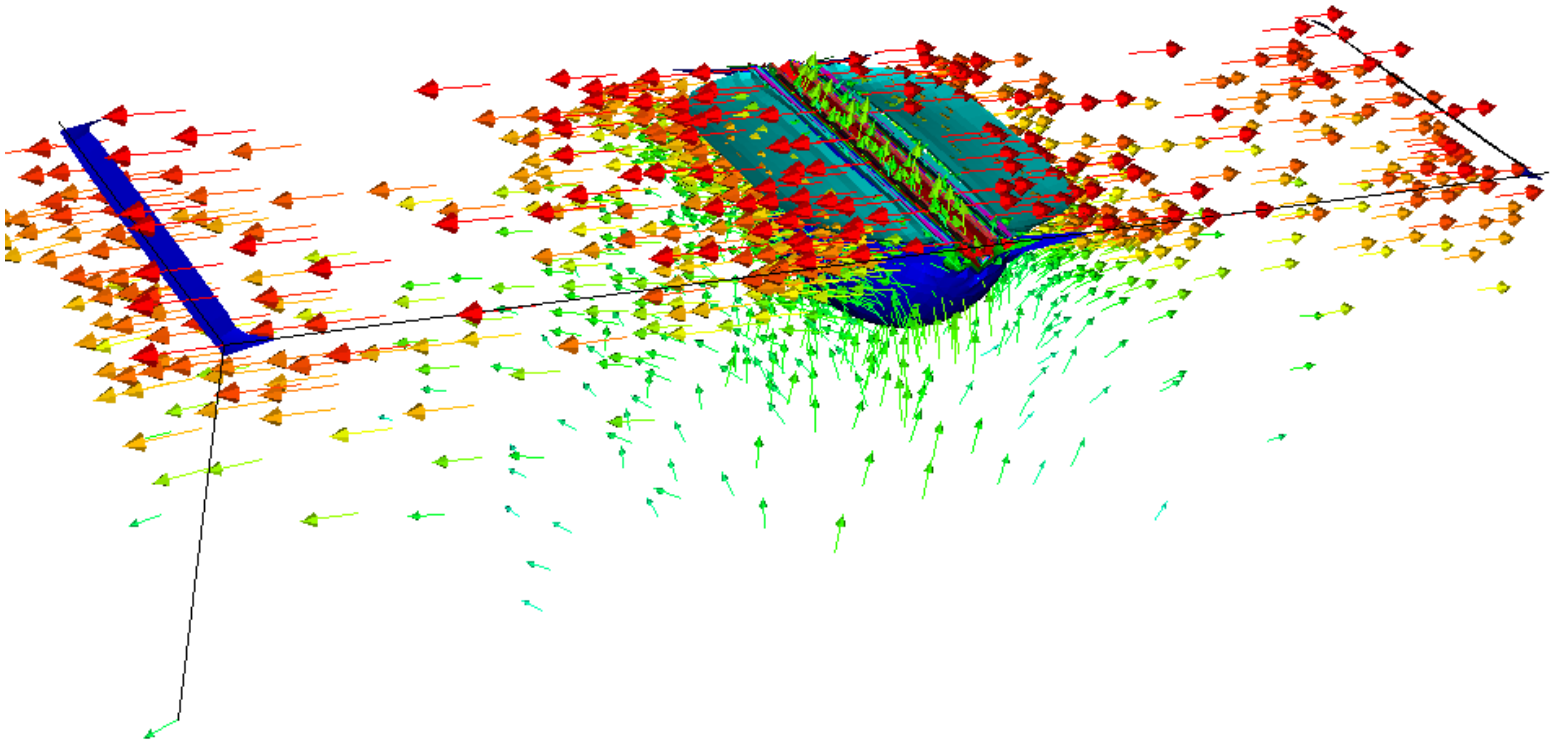
Currently available and underway programs

- CIG sponsors Bangerth to work on a suite of tutorial programs for deal.II that illustrate using adaptivity for geodynamics applications:
  . Stokes solver finished
  . Boussinesq solver finished, running on single CPU
  . Boussinesq solver running on 50-100 CPUs almost done
  . Darcy and two-phase flow solver finished
  . 4 more programs coming till 9/2009
- Ghattas, Burstedde, Stadler, Tu, Wilcox, Gurnis, Tan, & Zhong are working on a new massively parallel mantle convection code:
  . cartesian 3d simulations run on up to 32k cores
  . MINRES solver/BoomerAMG preconditioner scale well algorithmically, but suffer from communication
  . extension to more general geometries in the works
- Integration of p4est into deal.II under way

Examples of recent adaptive applications

deal.II's step-22 tutorial program:

An adaptive solver for the Stokes equations (210 lines of code)

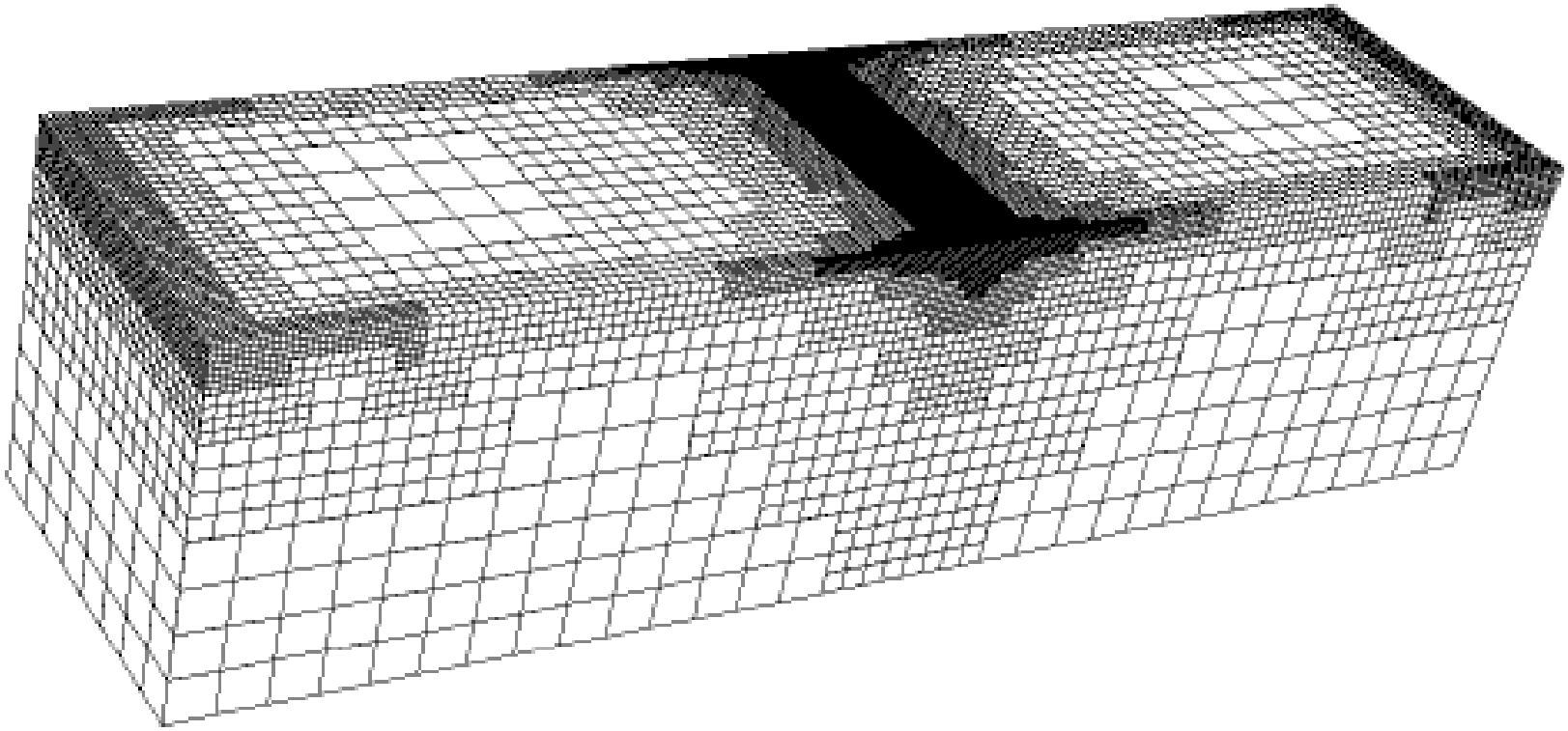


(Kronbichler, Bangerth)

Examples of recent adaptive applications

deal.II's step-22 tutorial program:

An adaptive solver for the Stokes equations (210 lines of code)

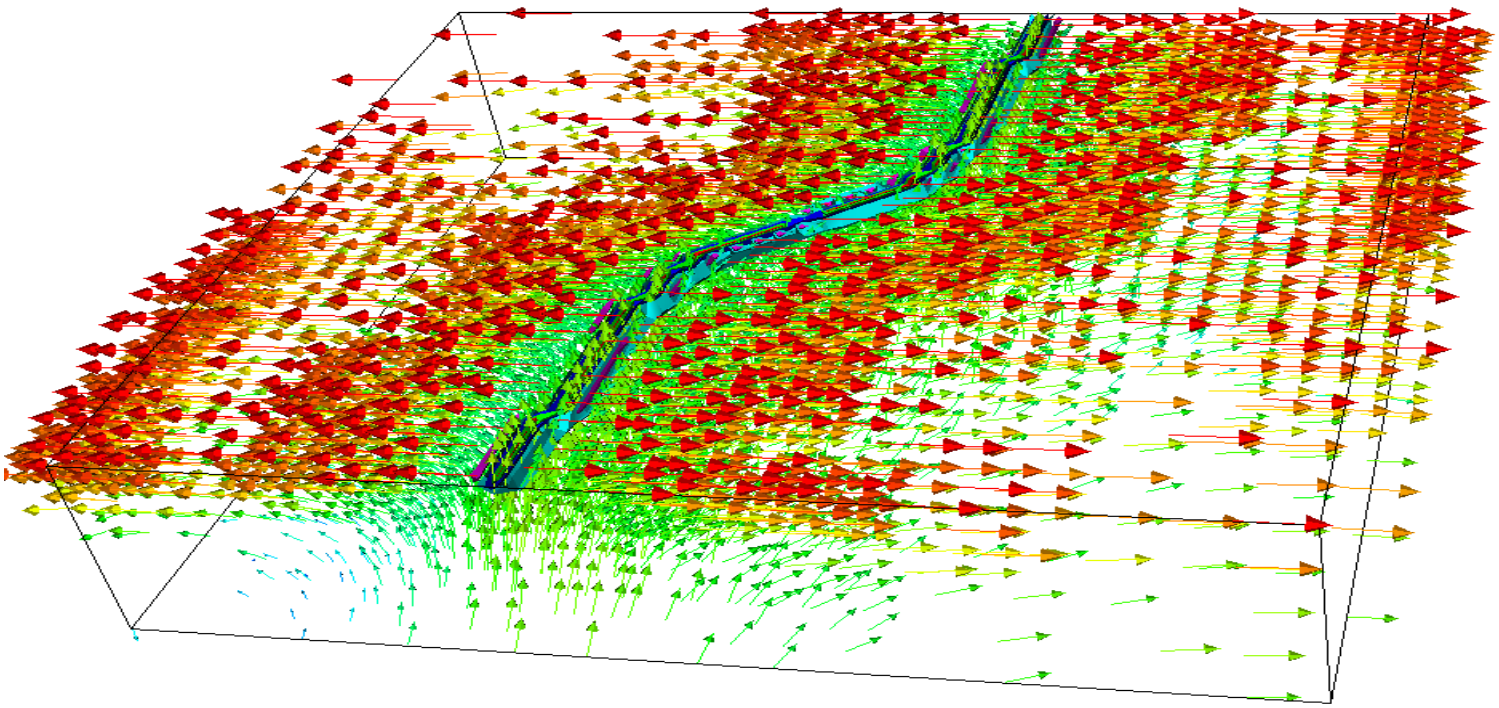


(Kronbichler, Bangerth)

Examples of recent adaptive applications

deal.II's step-22 tutorial program:

An adaptive solver for the Stokes equations (210 lines of code)



(Kronbichler, Bangerth)

Examples of recent adaptive applications

deal.II's step-22 tutorial program:
An adaptive solver for the Stokes equations (210 lines of code)

Code is an extensively documented testbed for numerical methods:

- Uses Q2/Q1 (Taylor-Hood) elements, but Q1/Q1 stabilized takes only changing ~20 lines of code
- Compares solving $\begin{pmatrix} A & B \\ B^T & 0 \end{pmatrix} \begin{pmatrix} U \\ P \end{pmatrix} = \begin{pmatrix} F \\ 0 \end{pmatrix}$

  using the pressure Schur complement $S = B^T A^{-1} B$ with CG

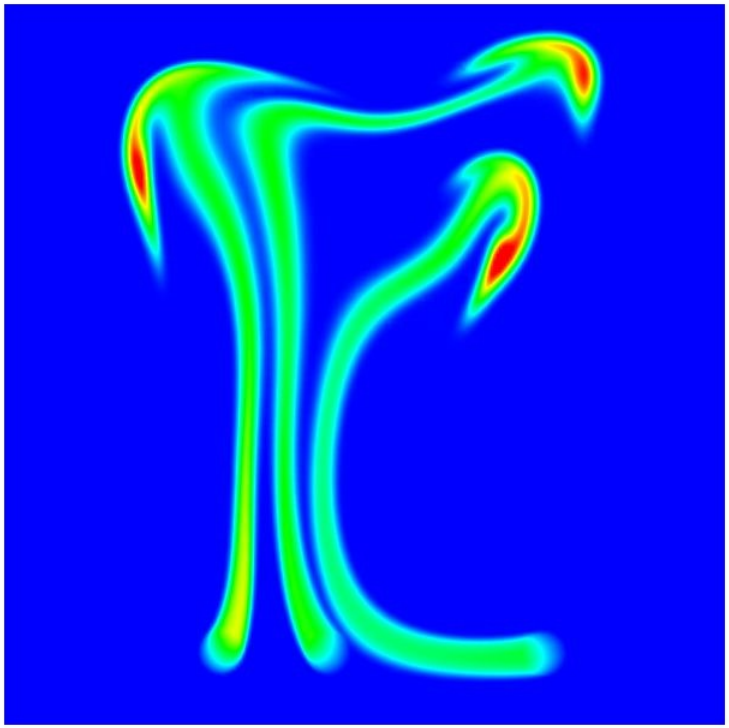
  and GMRES + block preconditioner $\begin{pmatrix} A^{-1} & 0 \\ B^T & S^{-1} \end{pmatrix}$ using Trilinos'

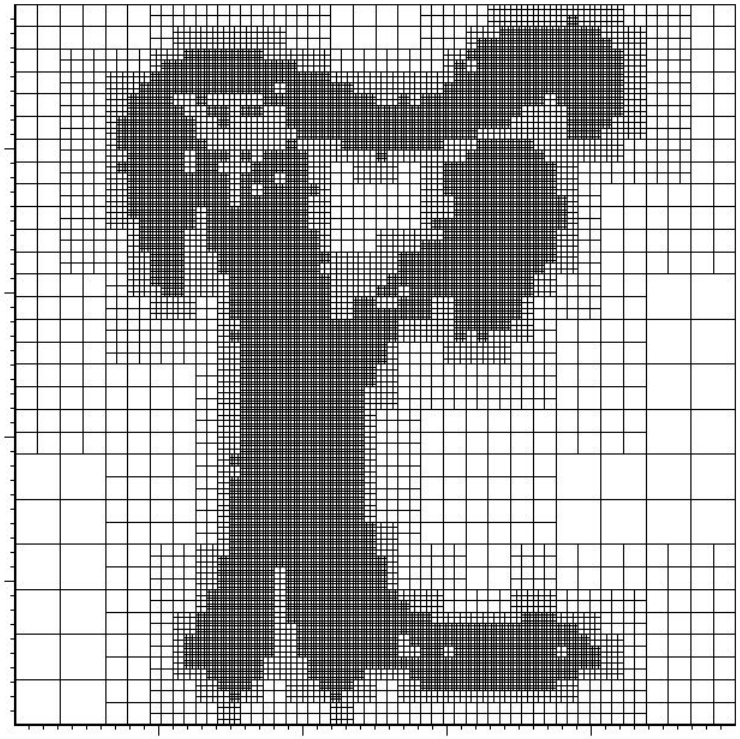
  algebraic multigrid implementation for the Laplace block

Examples of recent adaptive applications

deal.II's step-31 tutorial program:

An adaptive solver for the Boussinesq equations (530 lines of code)



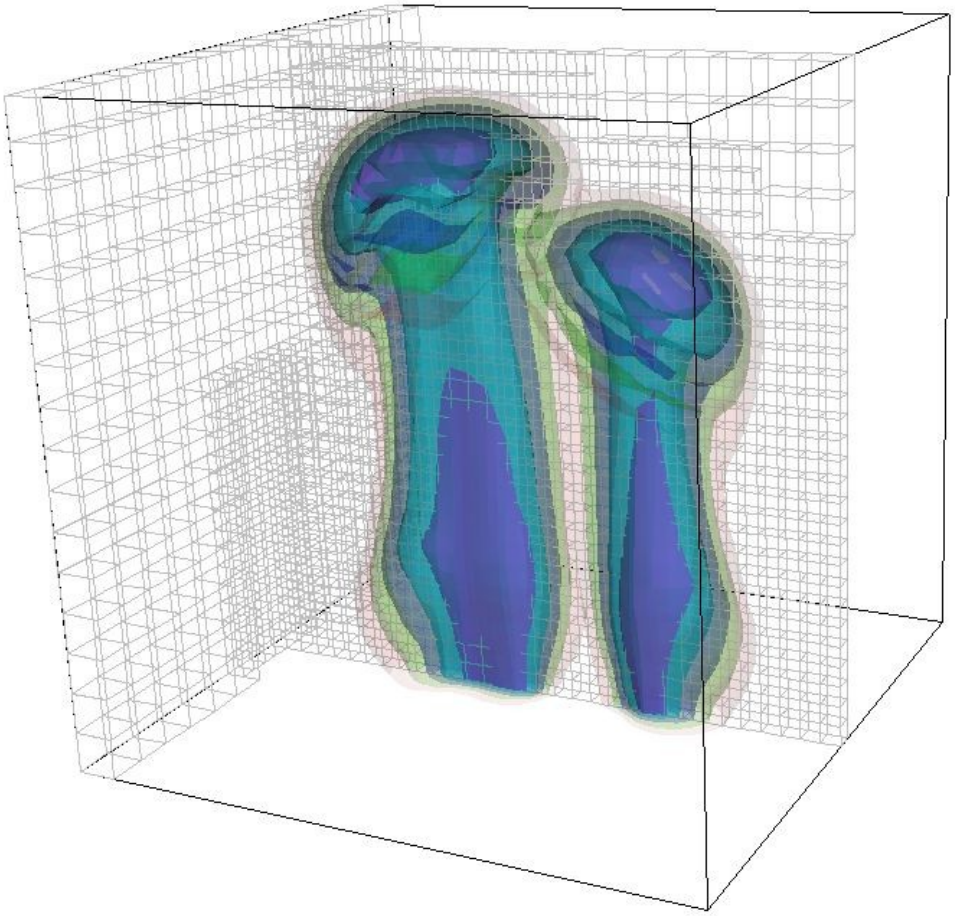


(Kronbichler, Bangerth)

Examples of recent adaptive applications

deal.II's step-31 tutorial program:

An adaptive solver for the Boussinesq equations (530 lines of code)



(Kronbichler, Bangerth)

Examples of recent adaptive applications

deal.II's step-31 tutorial program:
An adaptive solver for the Boussinesq equations (530 lines of code)

Testbed for numerical methods for thermal convection:

- GMRES with block triangular preconditioners, Trilinos' ML as inner preconditioner on the form

$$(\nabla u, \nabla v) \qquad + \text{ constraints from boundary conditions}$$

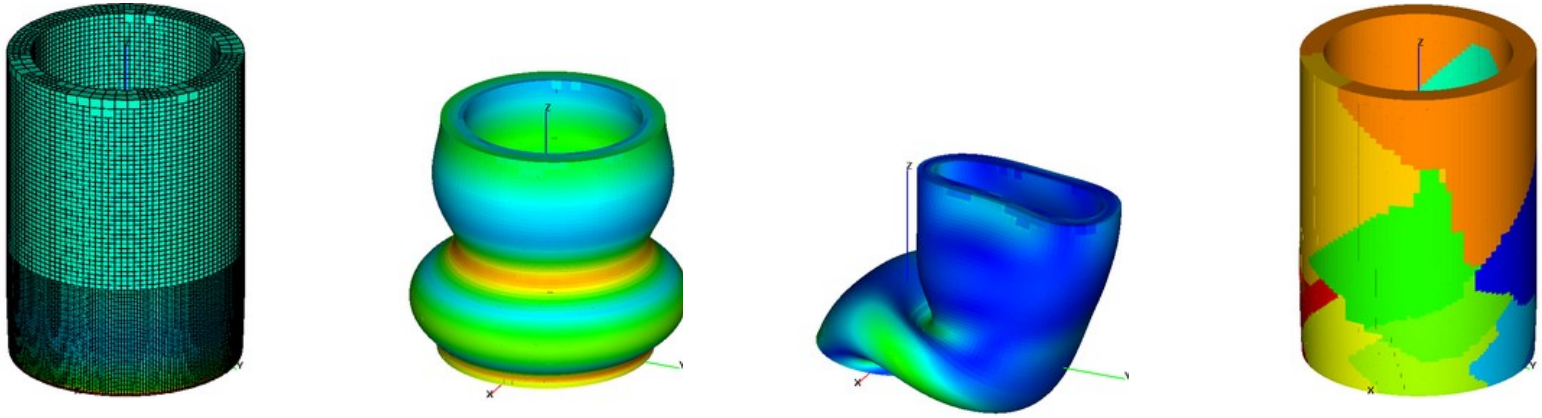
- IMPES-like scheme to decouple Stokes and advection
- Adaptive time step BDF-2 for time discretization
- Nonlinear artificial viscosity to stabilize transport:

$$\frac{\partial T}{\partial t} + u \cdot \nabla T - (\kappa + \nu) \Delta T = q$$

$$\nu_K = C_1 h_K \min\left(C_2, \|R_K\|\right)$$

Challenges in adaptivity

- **Parallelization, partitioning and load balancing**



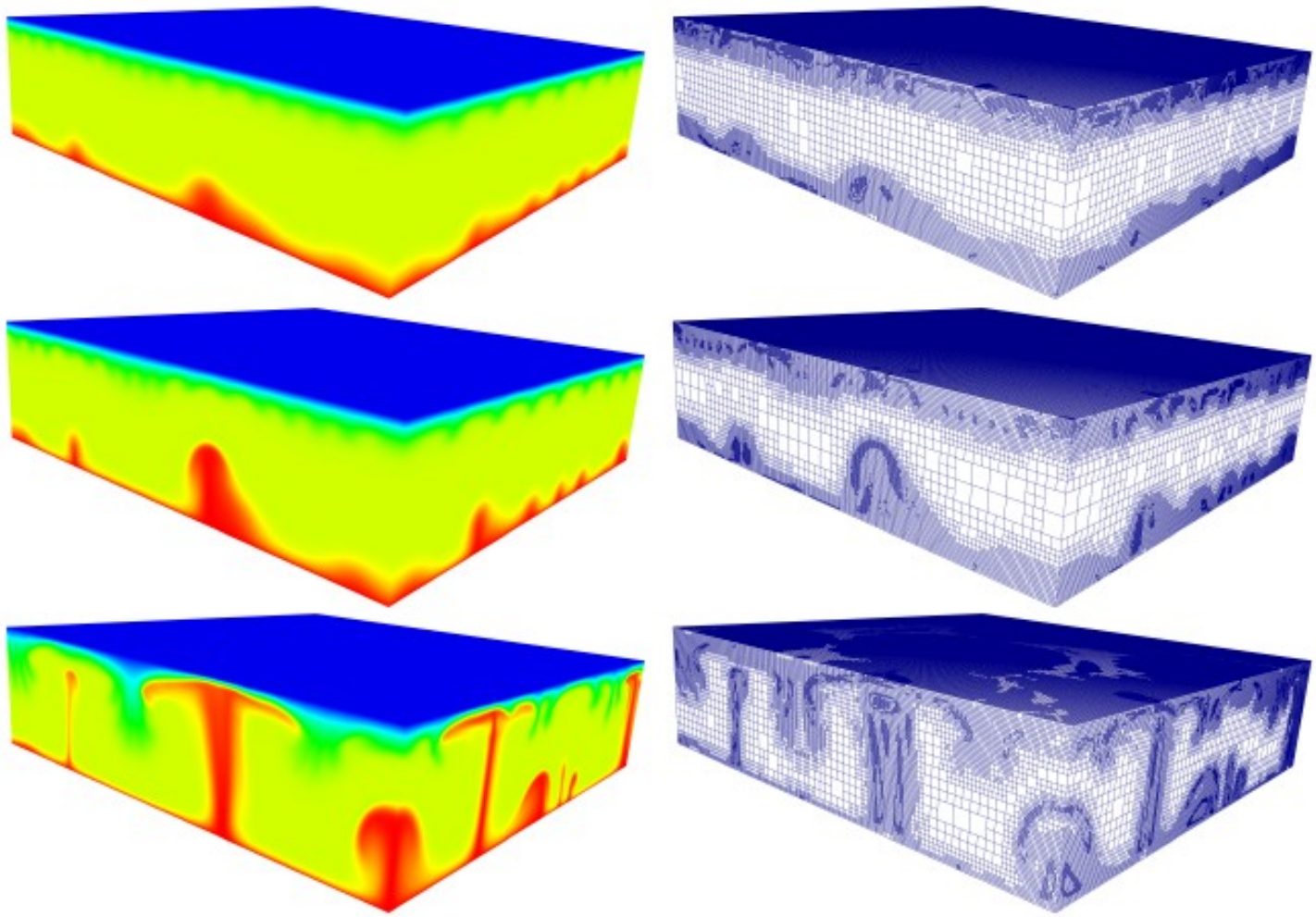
After computing the solution, the refinement indicator tells me which cells to refine

Problem: The individual blocks are now no longer load balanced!
Solution: We need to partition our domain again after refinement.

Problem: Requires access to the entire mesh to be efficient!
Solution 1: Keep the entire mesh on all all processors
Solution 2: Have distributed data structures that can be load balanced

The future of adaptivity

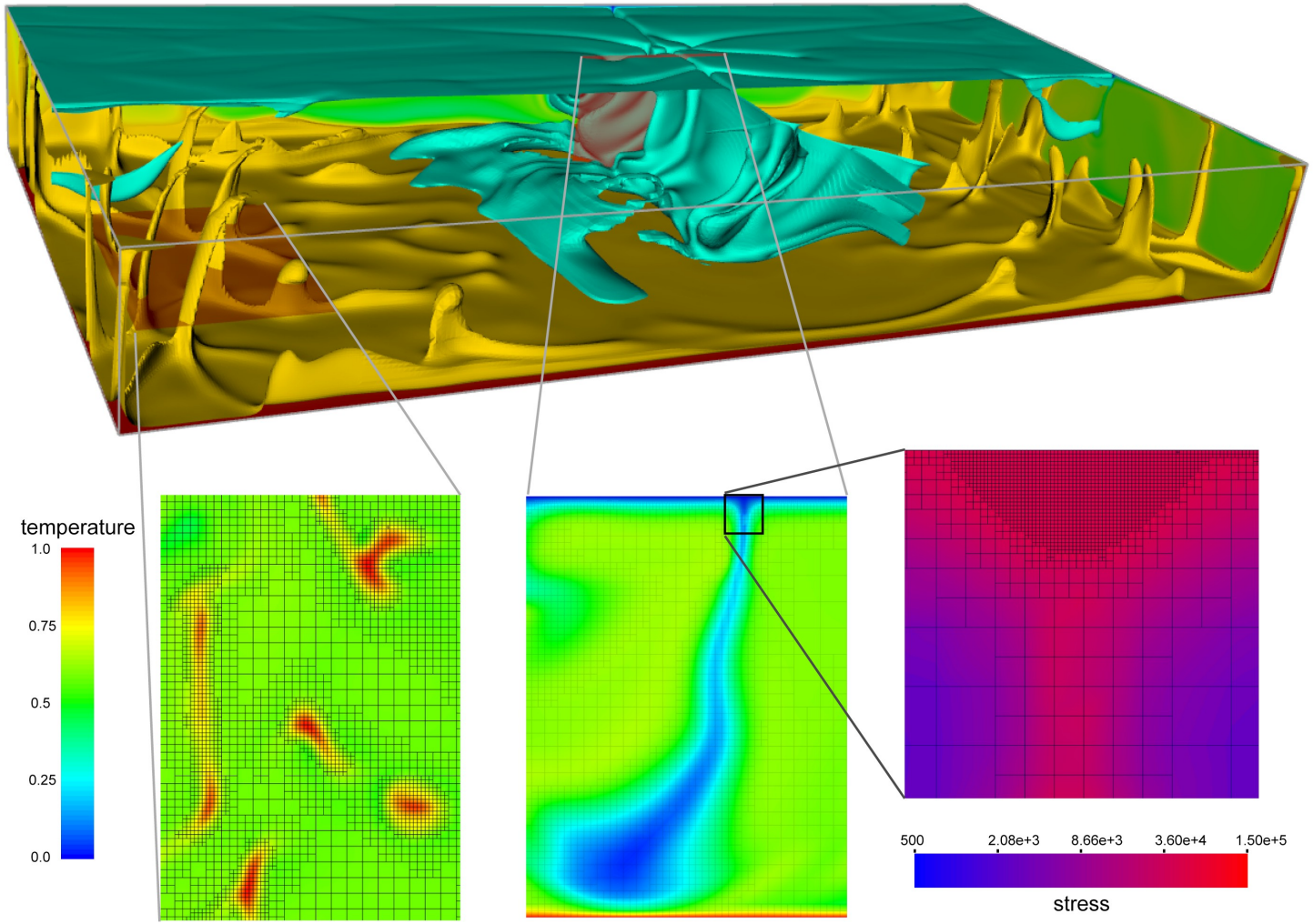
A completely parallel simulator for mantle convection:



(Burstedde, Ghattas, Gurnis, Stadler, Tan, Tu, Wilcox, Zhong)

The future of adaptivity

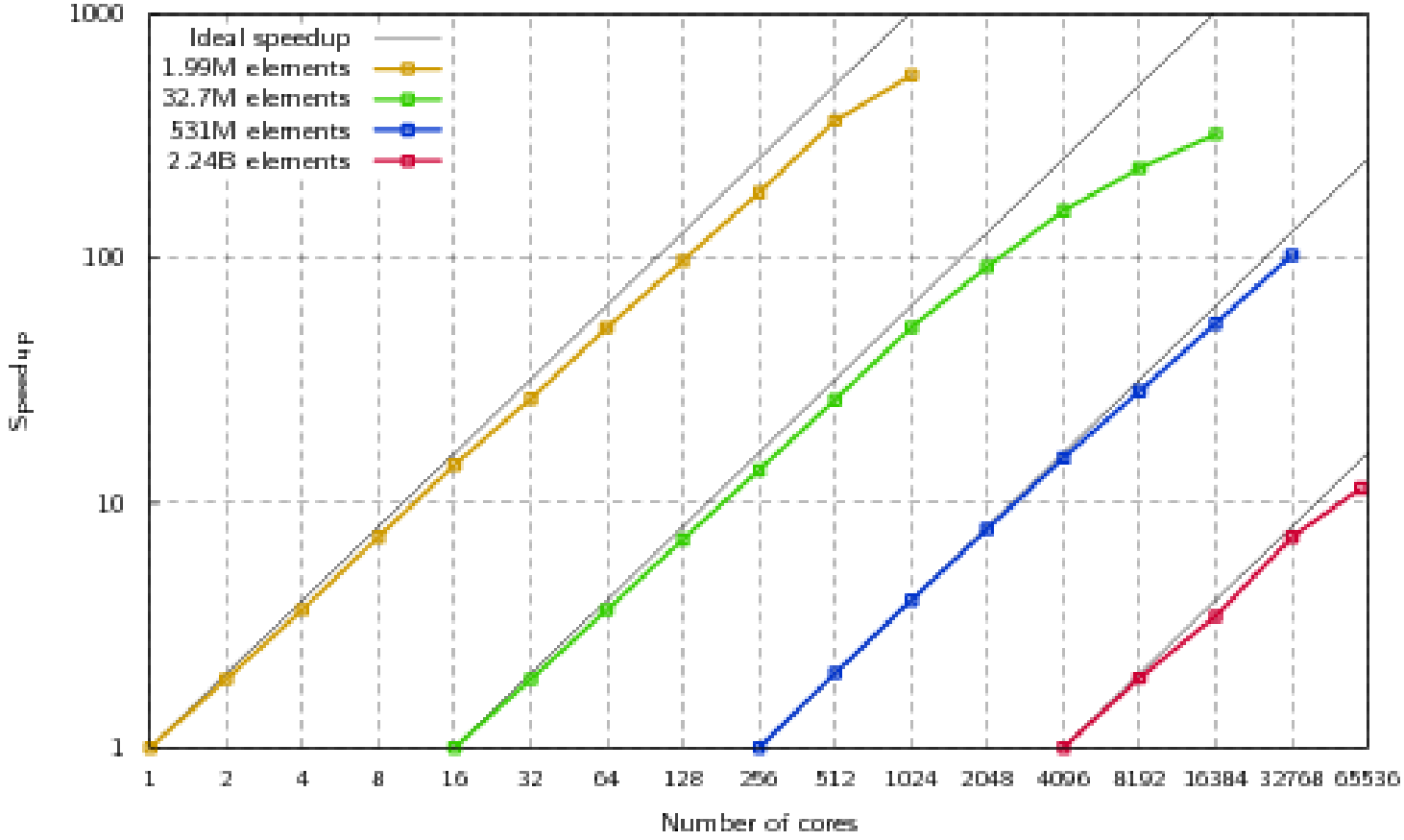
A completely parallel simulator for mantle convection:



(Burstedde, Ghattas, Gurnis, Stadler, Tan, Tu, Wilcox, Zhong)

The future of adaptivity

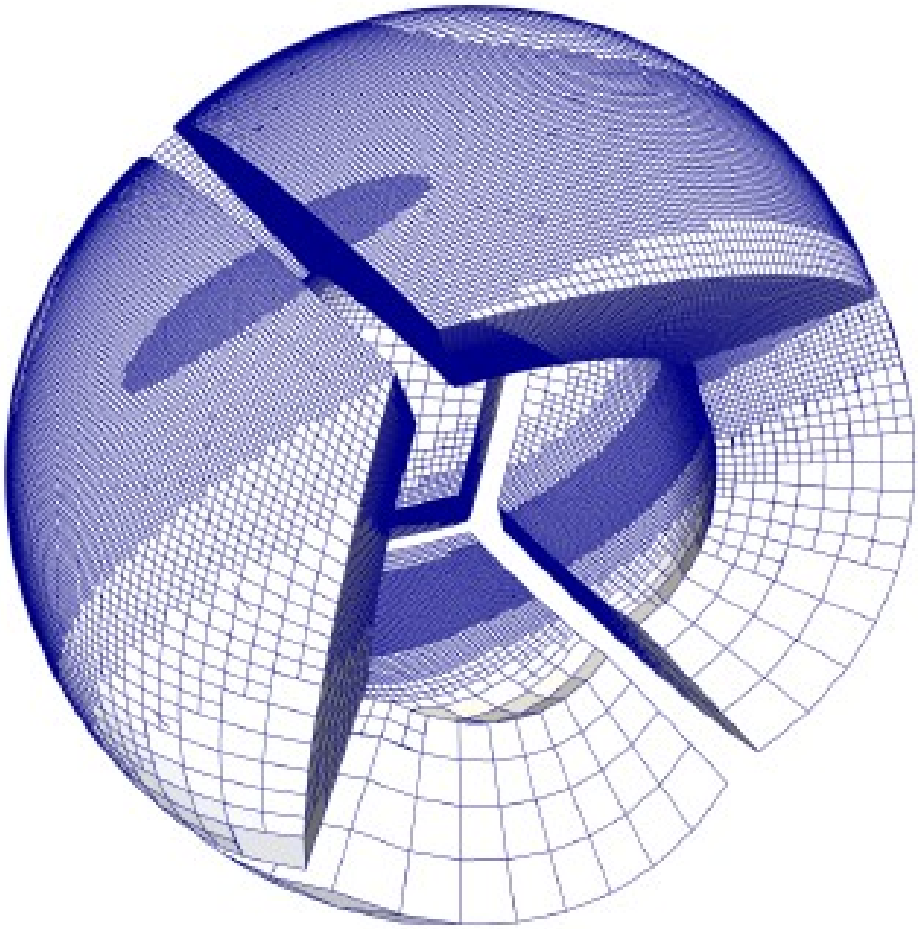
A completely parallel simulator for mantle convection:



This code scales to 1,000s or 10,000s of CPU cores. Its mesh data structures are dynamic and completely distributed. The underlying capabilities are currently being integrated into deal.II.

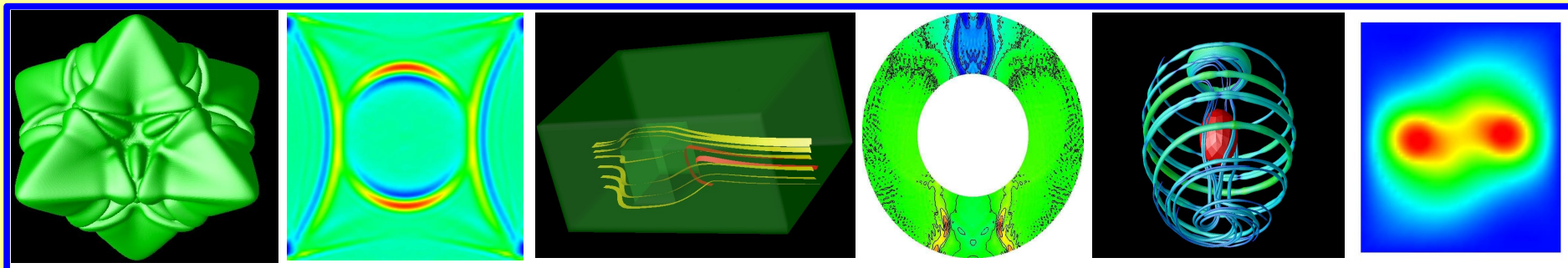
The future of adaptivity

A completely parallel simulator for mantle convection:

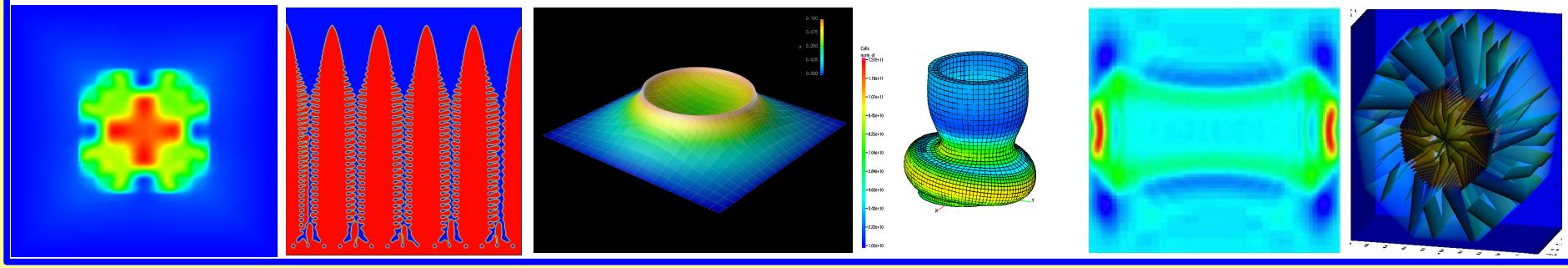


(Burstedde, Ghattas, Gurnis, Stadler, Tan, Tu, Wilcox, Zhong)

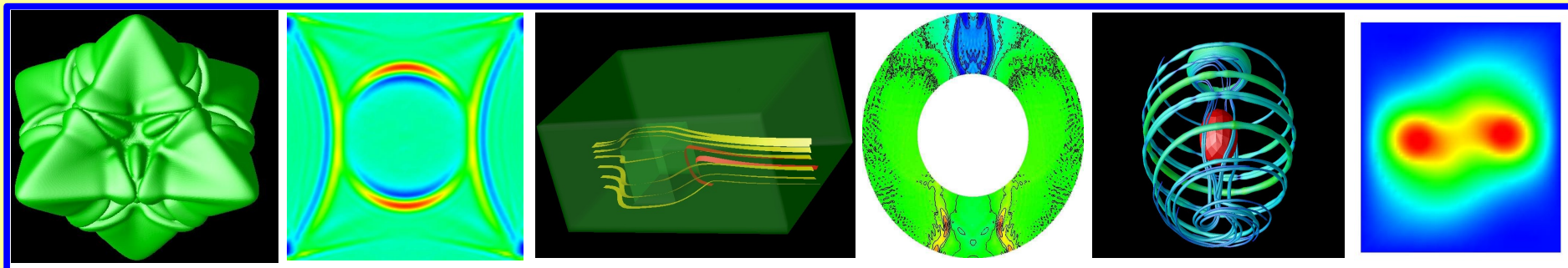
Conclusions



- Adaptivity promises better resolution with less work
- Requires substantial changes to codes
- It may be simpler to re-write a code
- **But:** New programs can draw from very large libraries of building blocks and existing tutorials!



The deal.II library



Visit the deal.II library:

http://www.dealii.org