ICTP - 12 October 2016

Luca Heltai

luca.heltai@sissa.it

…with many contributors around the world…

Problem solving at realistic complexities using the **deal.II** library.

A big part of it boils down to…

Software

Yet, we never talk about it!

- In our students' education
- In our papers
- In our professional interactions

How to write computational software for "real problems"?

...considering differences to "model problems" in:

- size
- complexity
- the way we develop it
- the way we teach it

In this talk:

- Some of our objectives (with examples)
- Our experience (with statistical data)
- Conclusions

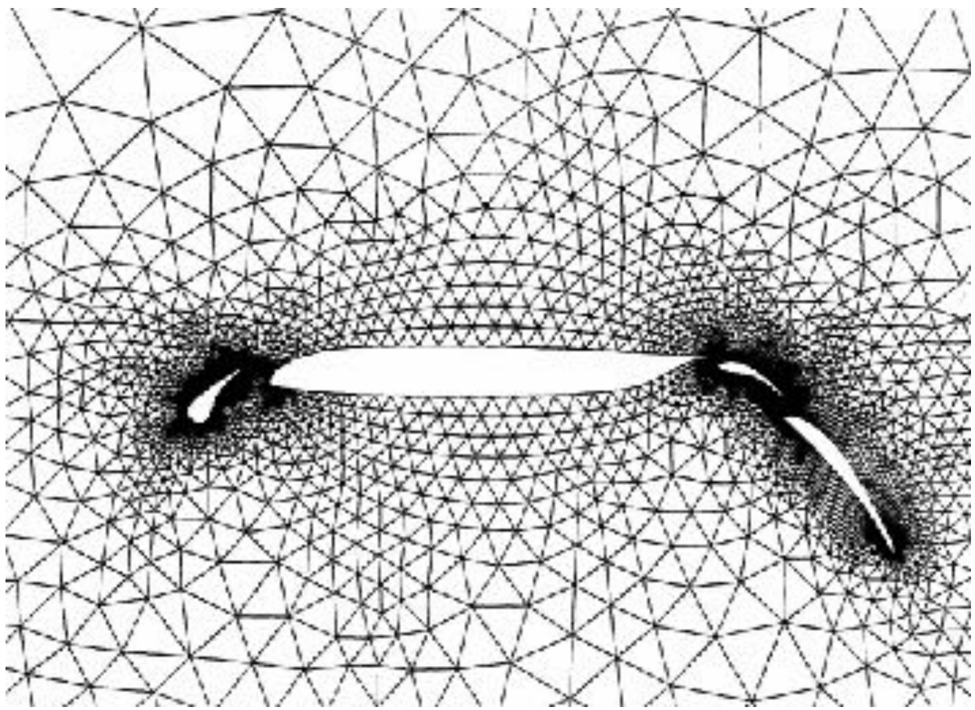
Workflow for HPC in PDEs

Step 1: Identify geometry and details of the model

May involve tens of thousands of pieces, very labor intensive, interface to designers and to manufacturing

Workflow for HPC in PDEs

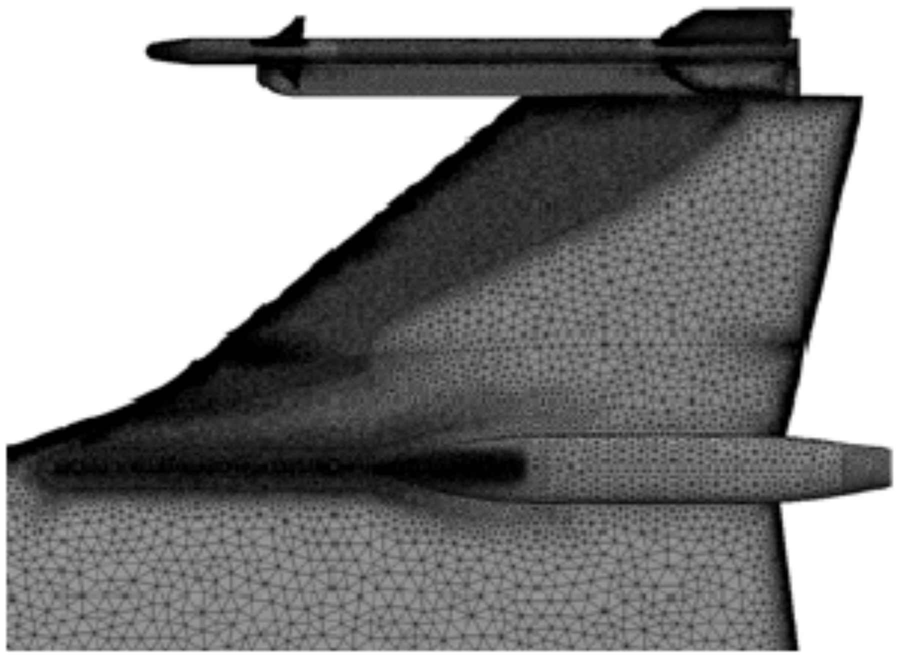
Step 2: Mesh generation and maybe partitioning (preprocessing)



May involve 10s of millions or more of cells; requires lots of memory; very difficult to parallelize

Workflow for HPC in PDEs

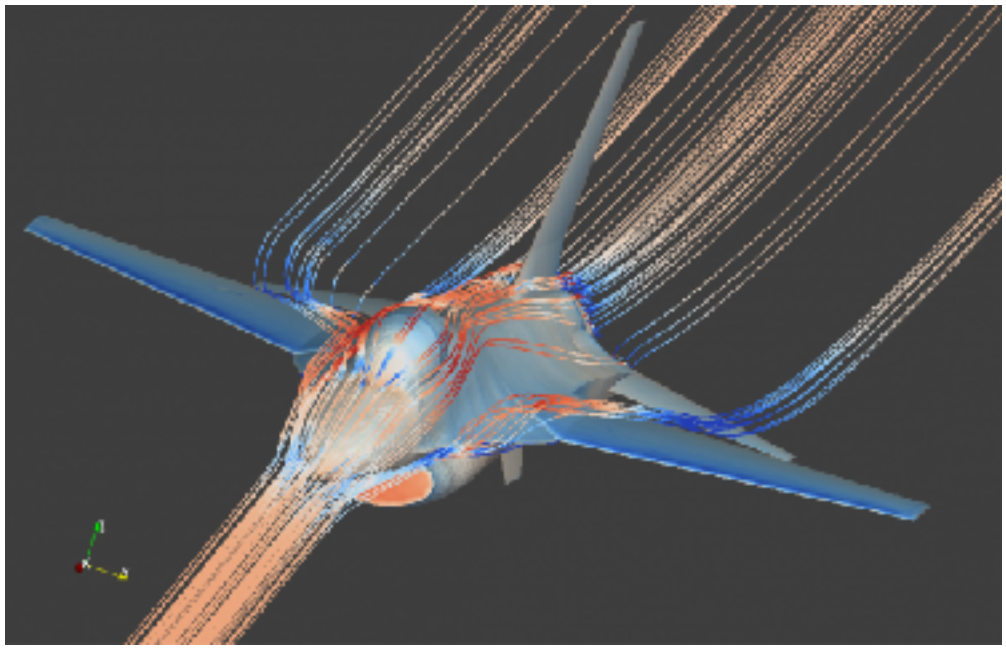
Step 2: Mesh generation and maybe partitioning (preprocessing)



May involve 10s of millions or more of cells; requires lots of memory; very difficult to parallelize

Workflow for HPC in PDEs

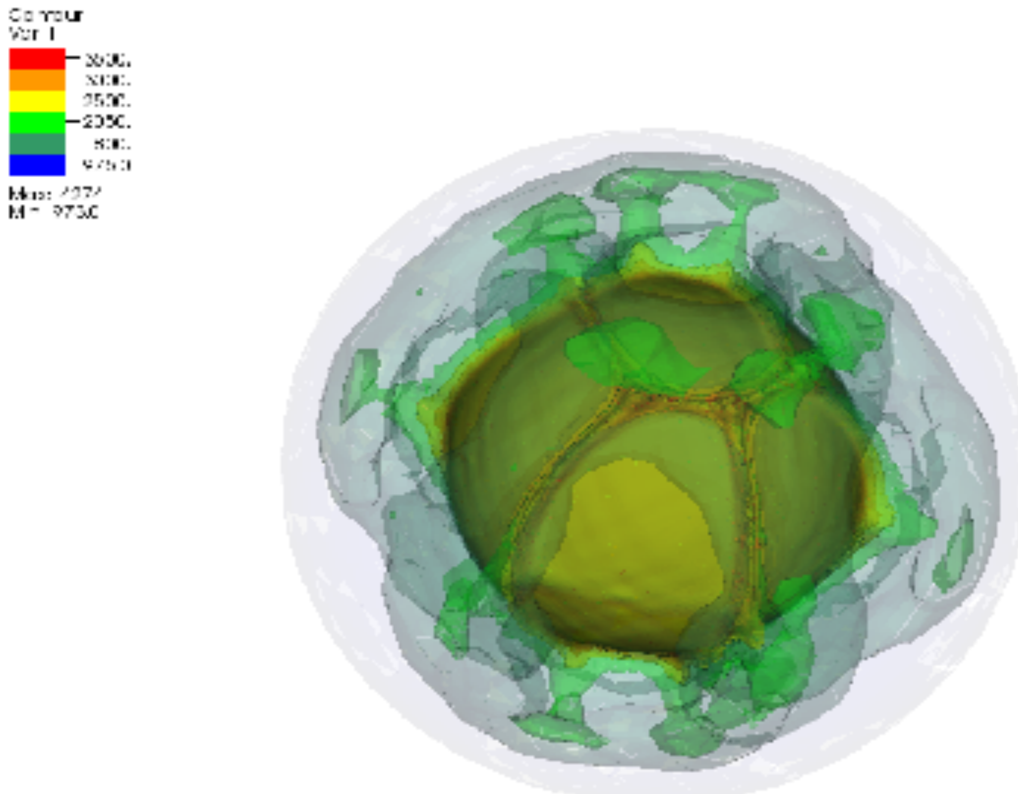
Step 3: Solve model on this mesh using finite elements, finite volumes, finite differences, …



Involves some of the biggest computations ever done, 10,000s of processors, millions of CPU hours, wide variety of algorithms

Workflow for HPC in PDEs

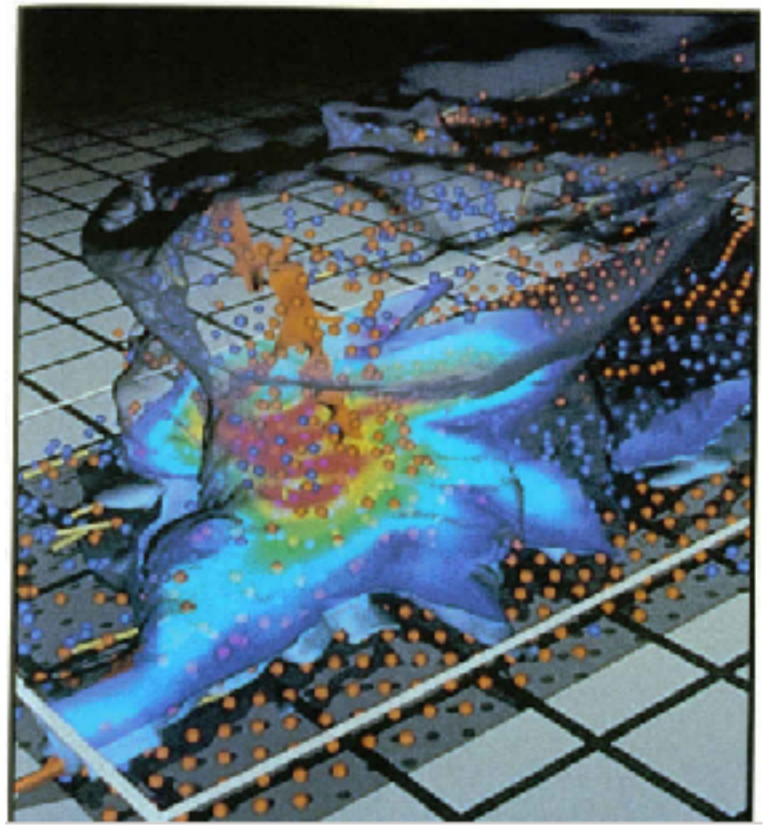
Step 4: Visualization to learn from the numerical results



Can be done in parallel, main difficulty is the amount of data.

Workflow for HPC in PDEs

Step 4: Visualization to learn from the numerical results



Goal: Not to *plot data*, but to *provide insight*!

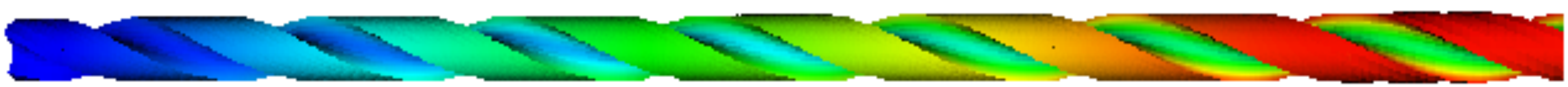
Workflow for HPC in PDEs

Step 5: Repeat

- To improve on the design
- To investigate different conditions (speed, altitude, angle of attack, …)
- To vary physical parameters that may not be known exactly
- To vary parameters of the numerical model (e.g. mesh size)
- To improve match with experiments

A complete example

Goal: Simulating the deformation of a drill



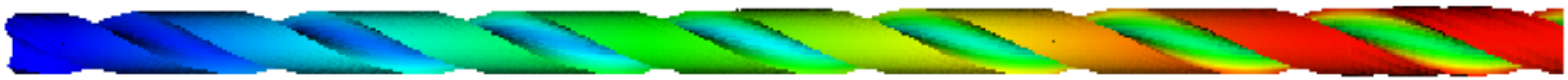
Data produced by *Patrik Boettcher*:

- Created during a 2-week deal.II course
- Time needed: approximately 50 hours, including learning deal.II

Geometry and mesh provided by *Hannah Ludwig*.

A complete example

Goal: Simulating the deformation of a drill



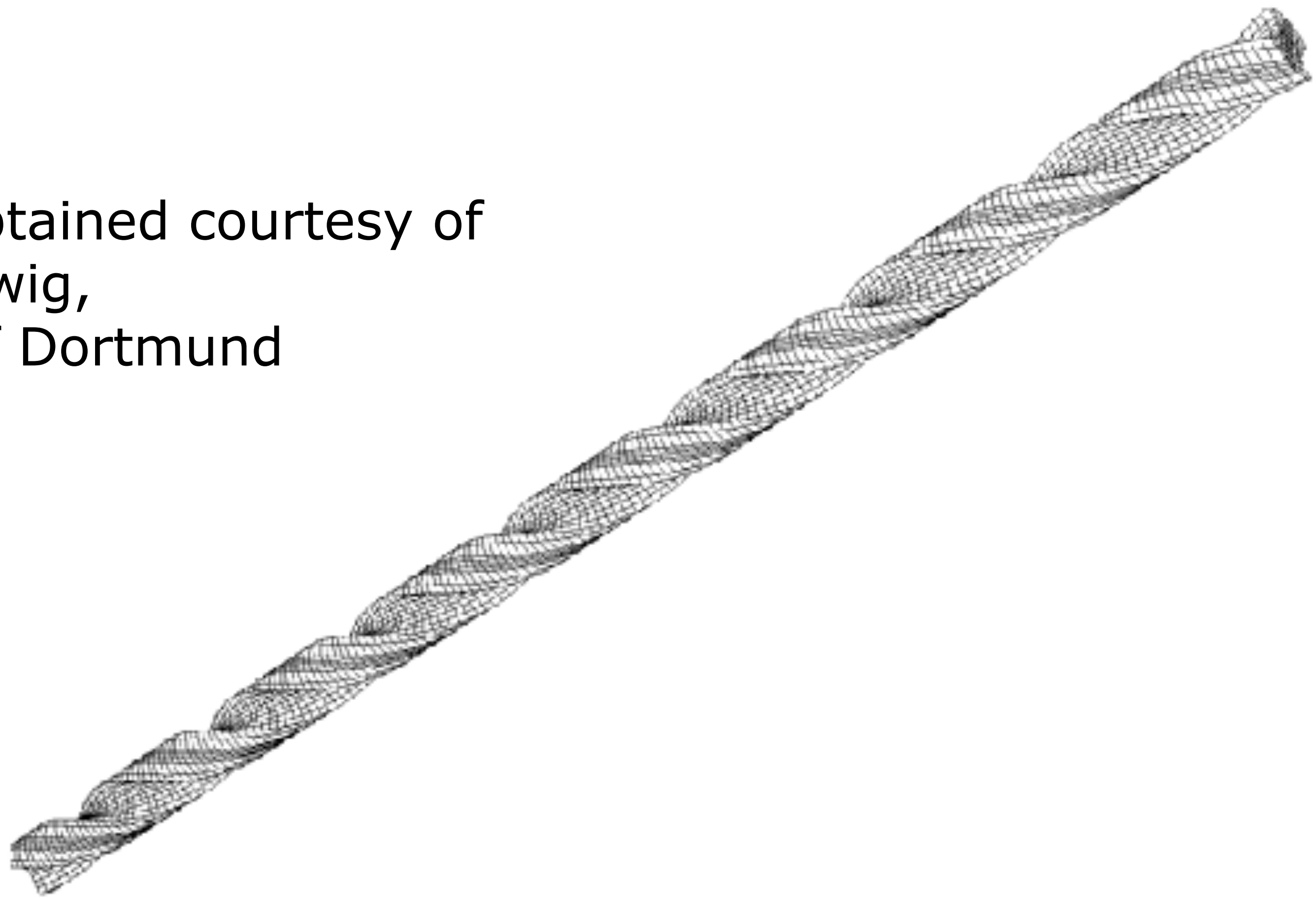
Steps:

(1) Create or obtain a coarse mesh
(2) Identify the model (elasticity) and implement a solver
(3) Obtain material parameters for steel used in the drill
(4) Mark up geometry: Where do which forces act
(5) Identify magnitude of forces
(6) Mark up geometry: Describe boundary approximation
(7) Postprocess for quantities of interest
(8) Visualize
(9) Start over: Optimization of drill and validation

A complete example

Step 1: Create or obtain a coarse mesh

Here:
Mesh was obtained courtesy of
Hannah Ludwig,
University of Dortmund



Project by Patrik Boettcher,
University of Heidelberg, 2012

A complete example

Step 2: Identify the model

Here:

- Linear, small deformation elasticity model 3d:

$$\begin{aligned} -\nabla(\lambda \nabla \cdot u) - 2\nabla \cdot (\mu \varepsilon(u)) &= f && \text{in } \Omega \\ u &= g_D && \text{on } \Gamma_D \\ n \cdot (\lambda (\nabla \cdot u) I + 2\mu \varepsilon(u)) &= g_N && \text{on } \Gamma_N \end{aligned}$$

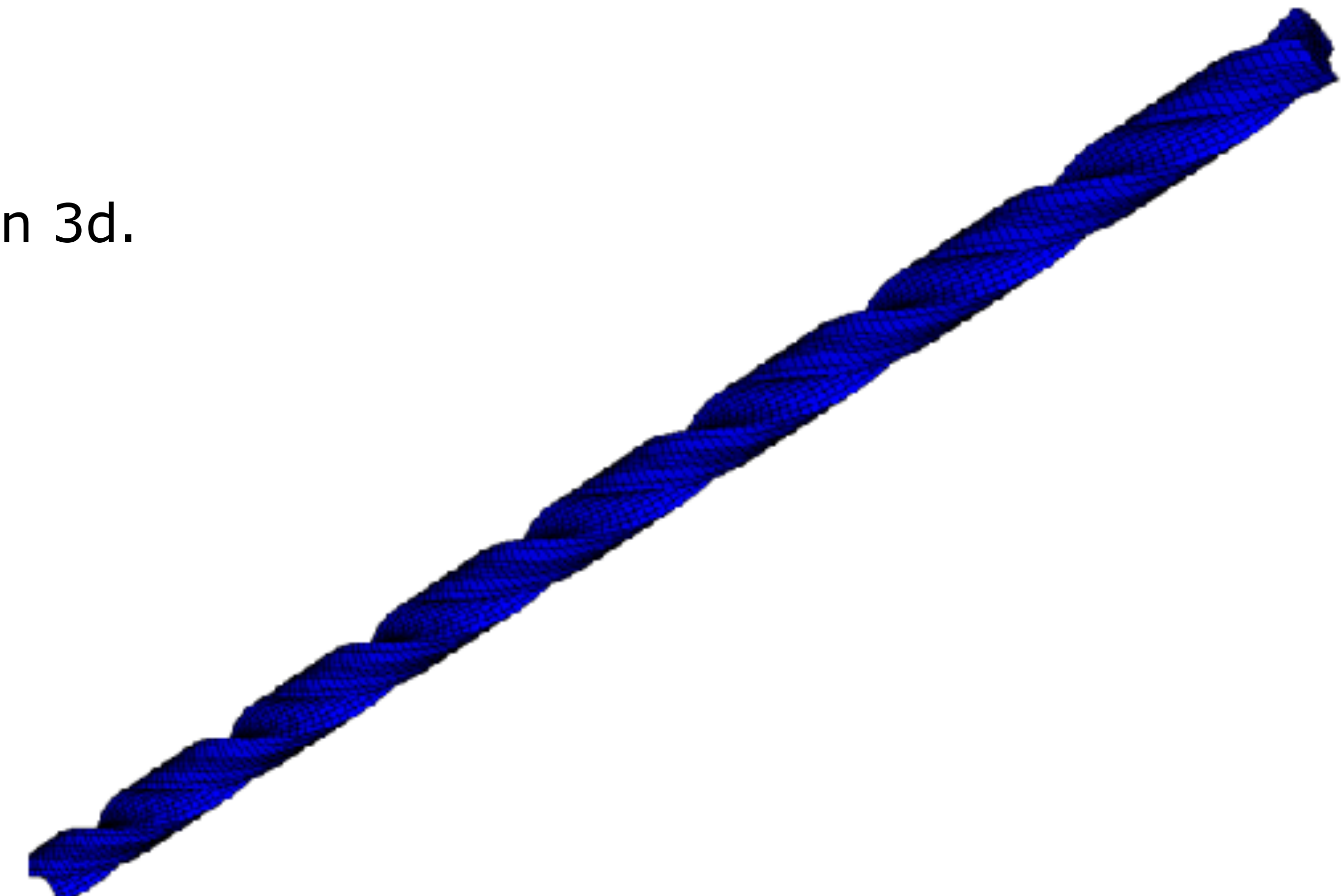
- Justified because displacements will be <0.3mm on domain sizes of >20mm

Project by Patrik Boettcher,
University of Heidelberg, 2012

A complete example

Step 2: Implement an elasticity solver

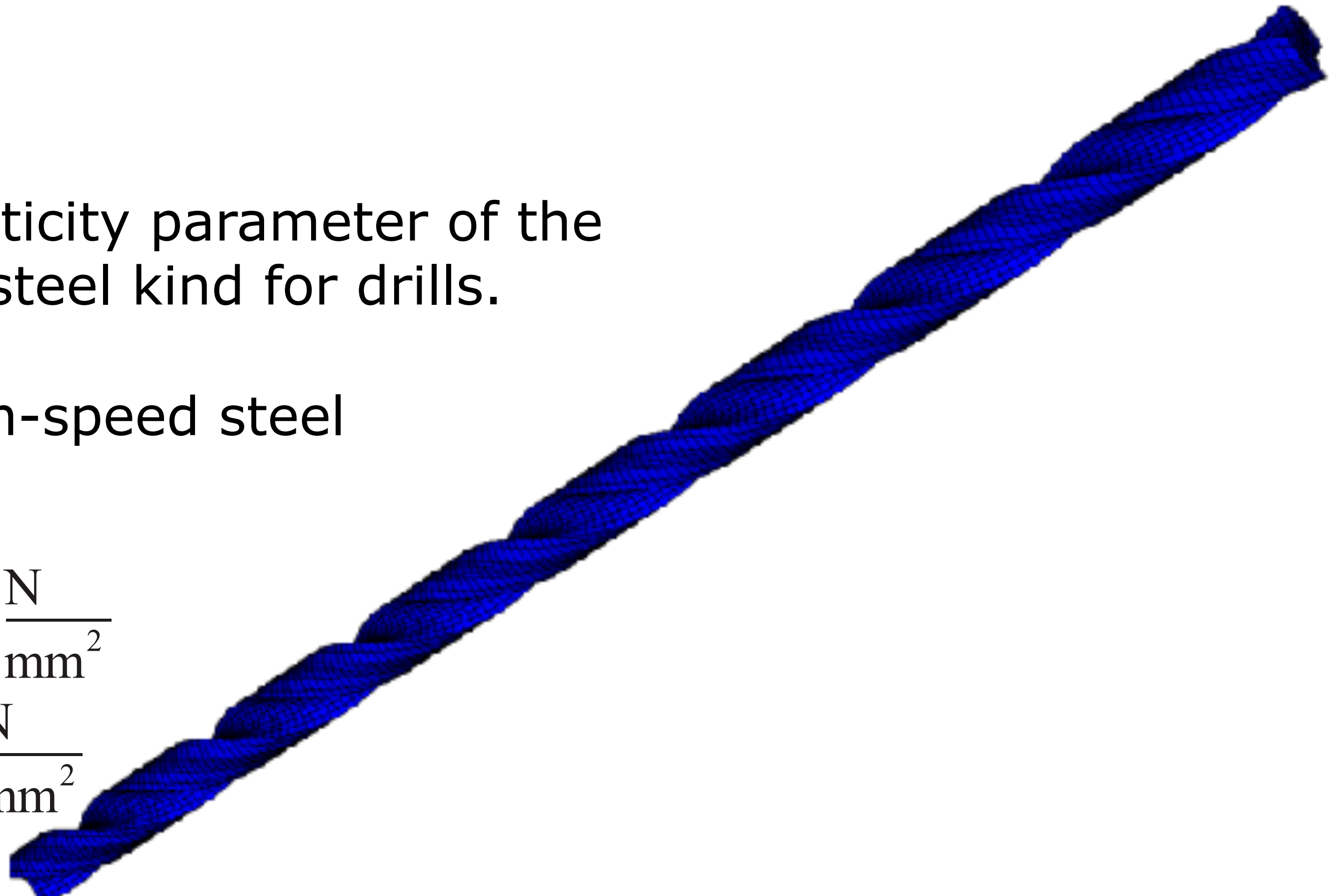
Here:
Use step-8 in 3d.



Project by Patrik Boettcher,
University of Heidelberg, 2012

A complete example

Step 3: Identify material parameters



Here:
Find the elasticity parameter of the appropriate steel kind for drills.

Choose: High-speed steel HS-30 with

$$\lambda = 207{,}000 \frac{\mathrm{N}}{\mathrm{mm}^2}$$

$$\mu = 82{,}800 \frac{\mathrm{N}}{\mathrm{mm}^2}$$

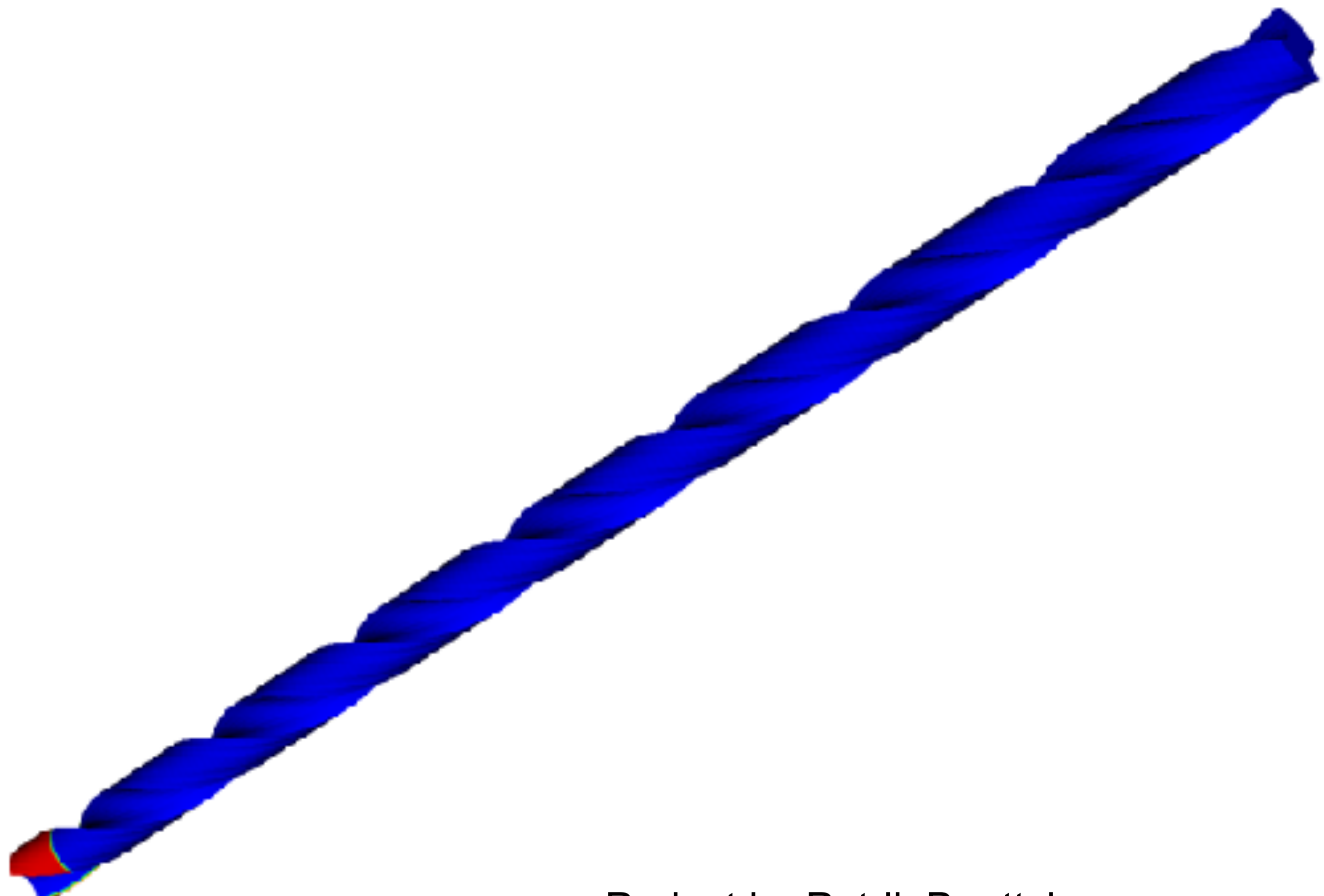
Project by Patrik Boettcher,
University of Heidelberg, 2012

A complete example

Step 4: Mark up geometry – where does which force act?

Here:

- Clamped



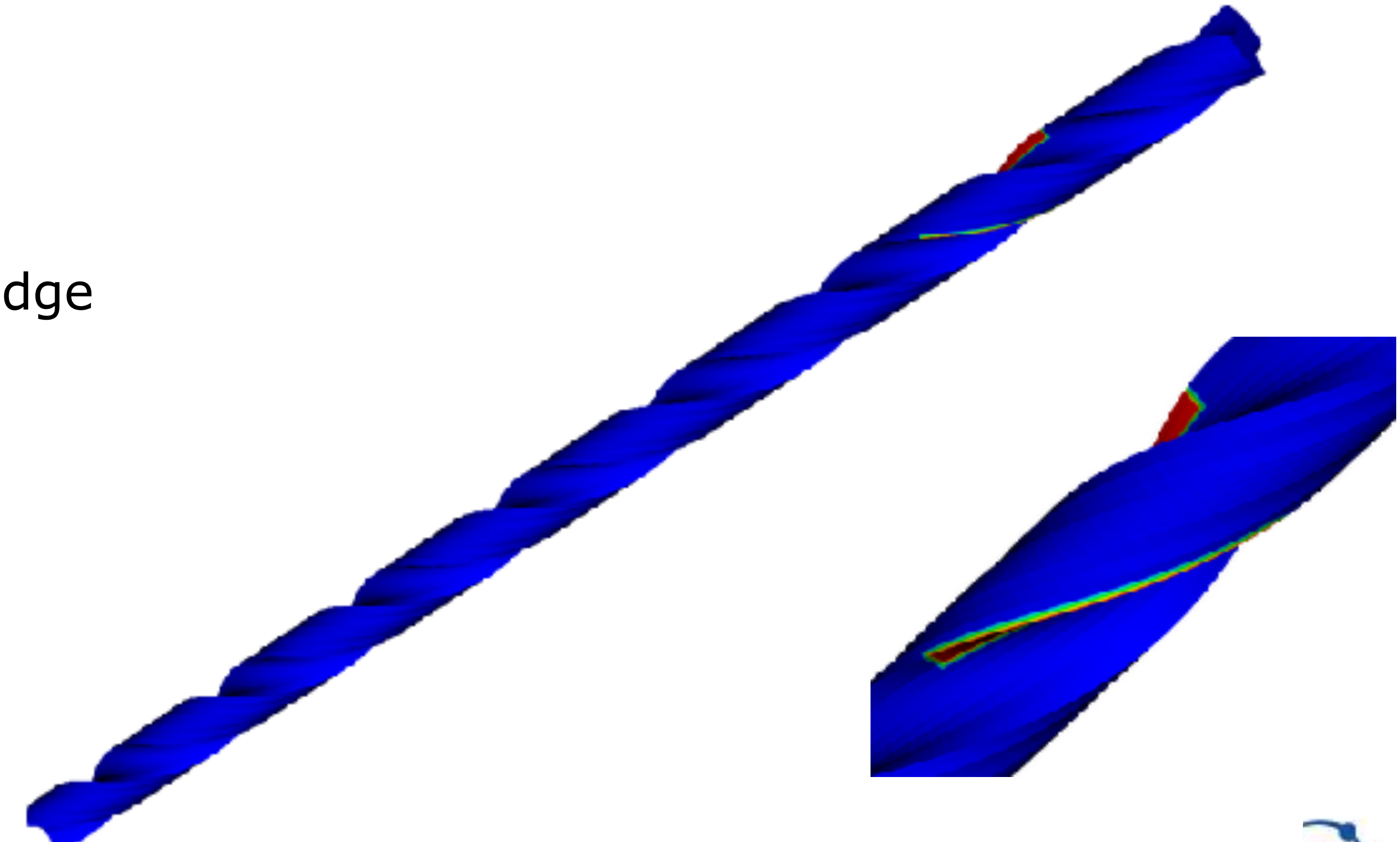
Project by Patrik Boettcher,
University of Heidelberg, 2012

A complete example

Step 4: Mark up geometry – where does which force act?

Here:

- Clamped
- Cutting edge



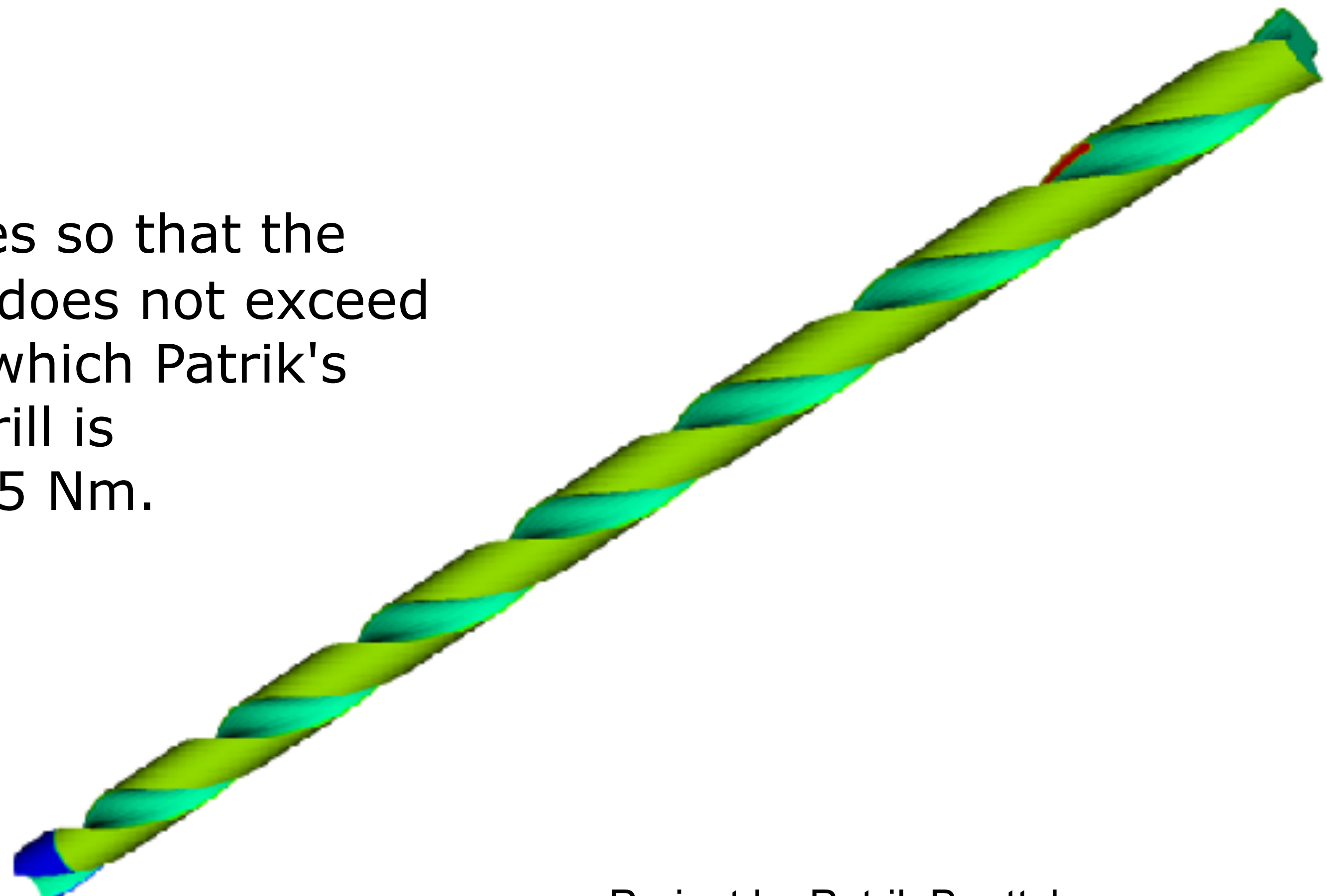
Project by Patrik Boettcher,
University of Heidelberg, 2012

A complete example

Step 5: Identify appropriate magnitude of forces

Here:

Choose forces so that the *total torque* does not exceed the level to which Patrik's household drill is rated, i.e., 25 Nm.



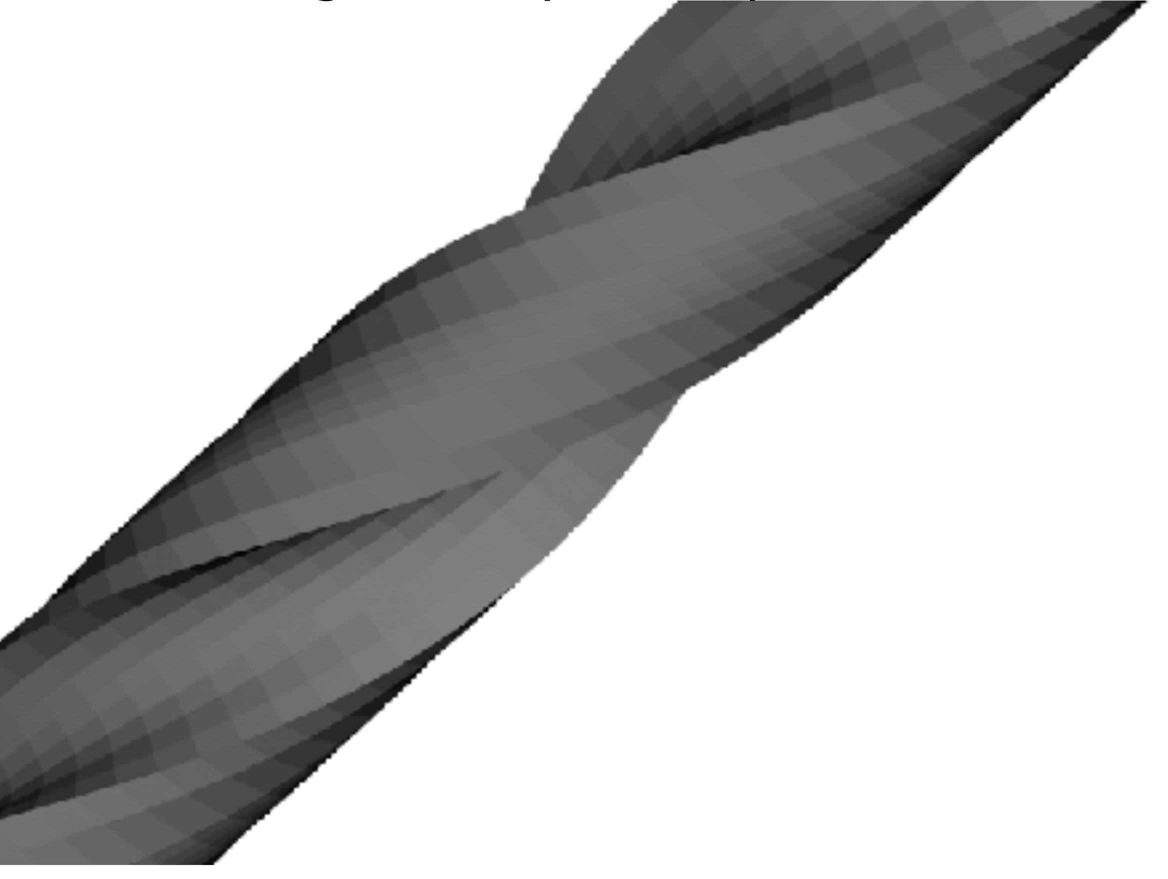
Project by Patrik Boettcher,
University of Heidelberg, 2012

A complete example

Step 6: Mark up boundaries for geometry description

Here:

Without appropriate boundary description

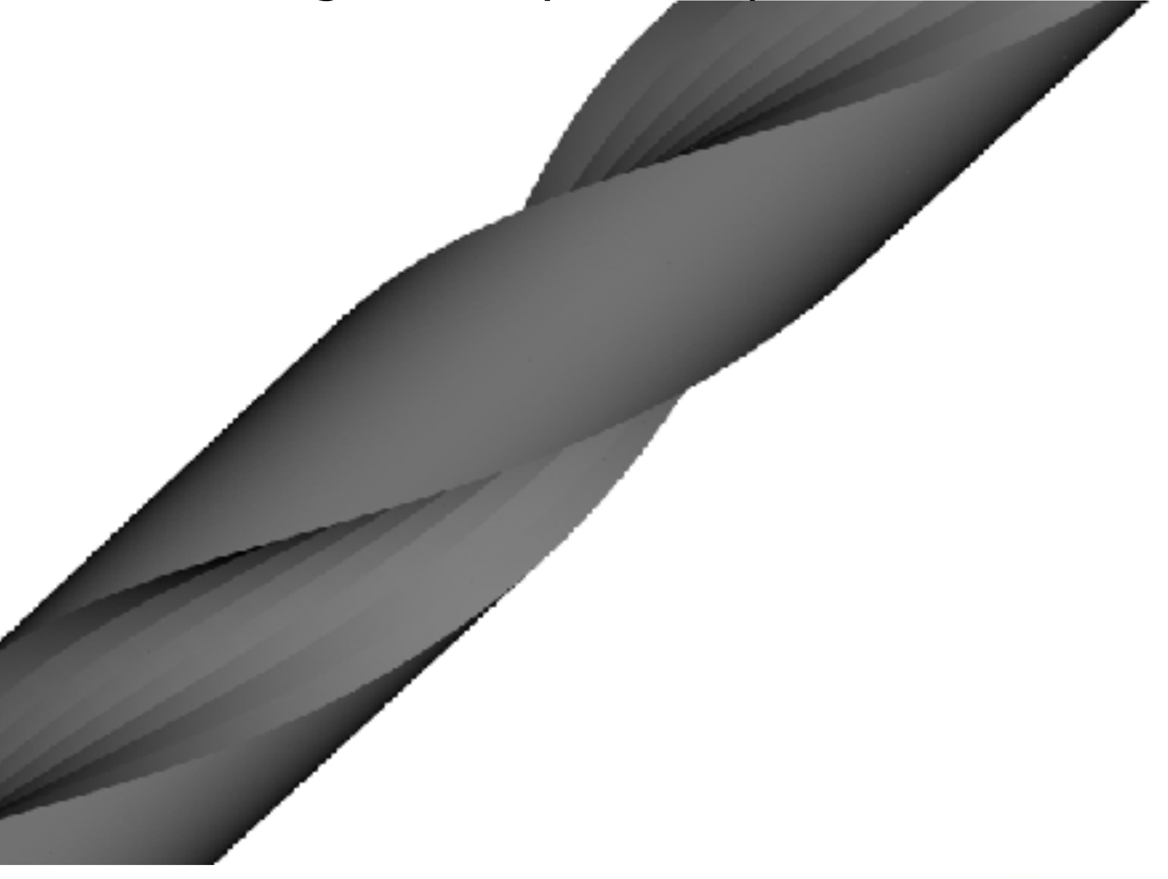


A complete example

Step 6: Mark up boundaries for geometry description

Here:

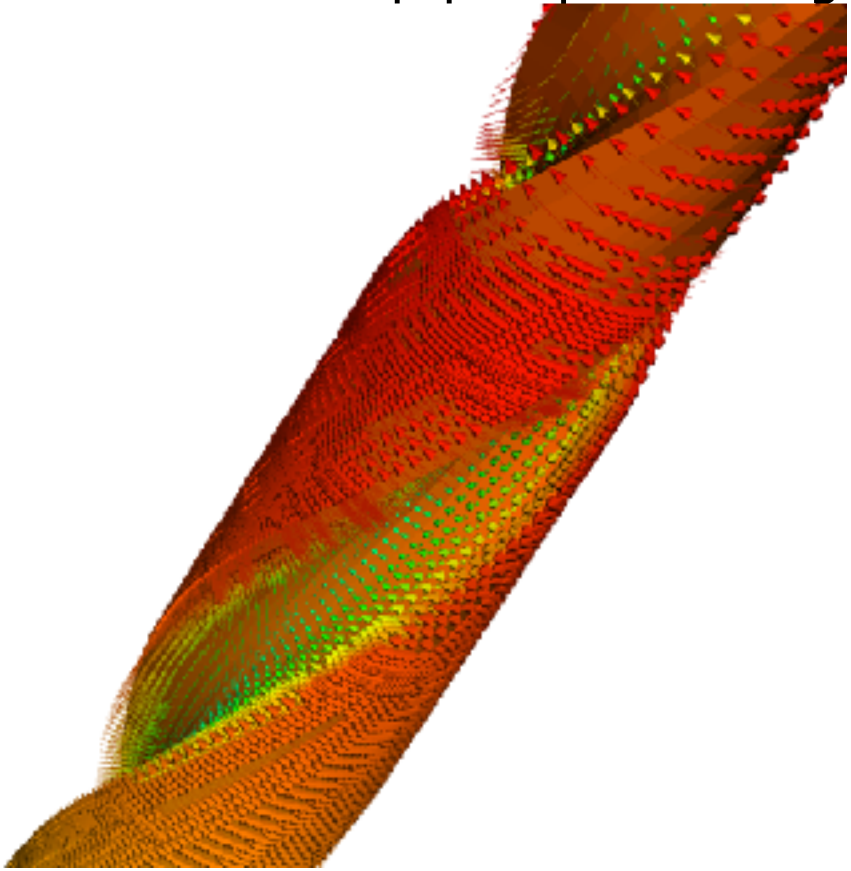
With appropriate boundary description for outer boundary (no description for the inner ones was available)



Project by Patrik Boettcher,
University of Heidelberg, 2012

A complete example

Step 7: Identify goals of simulation and set up postprocessing needs



Here:

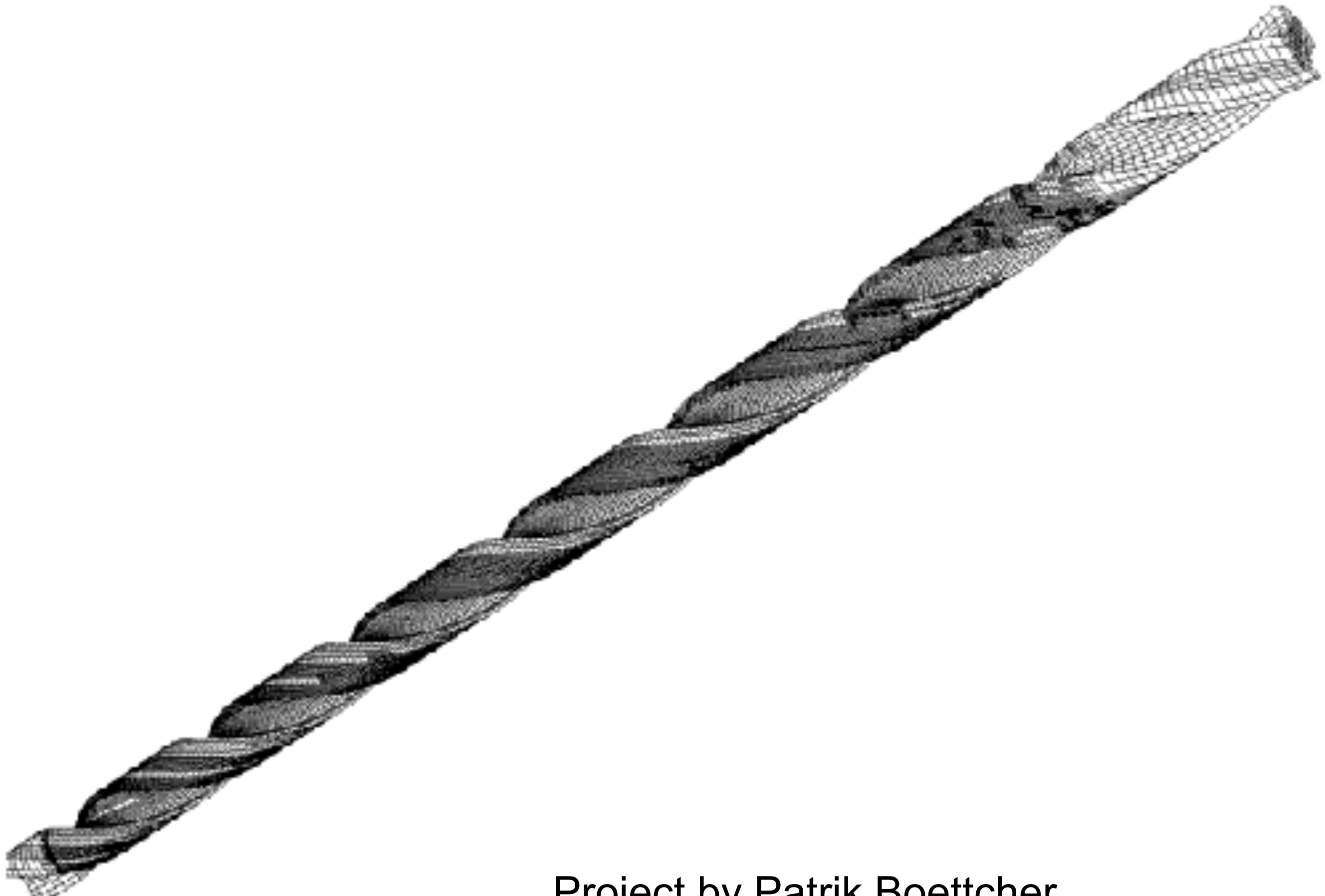
The goal is to determine the torsion angle of the drill from the displacement vector.

Project by Patrik Boettcher,
University of Heidelberg, 2012

A complete example

Step 8: Visualize

Here:
Mesh



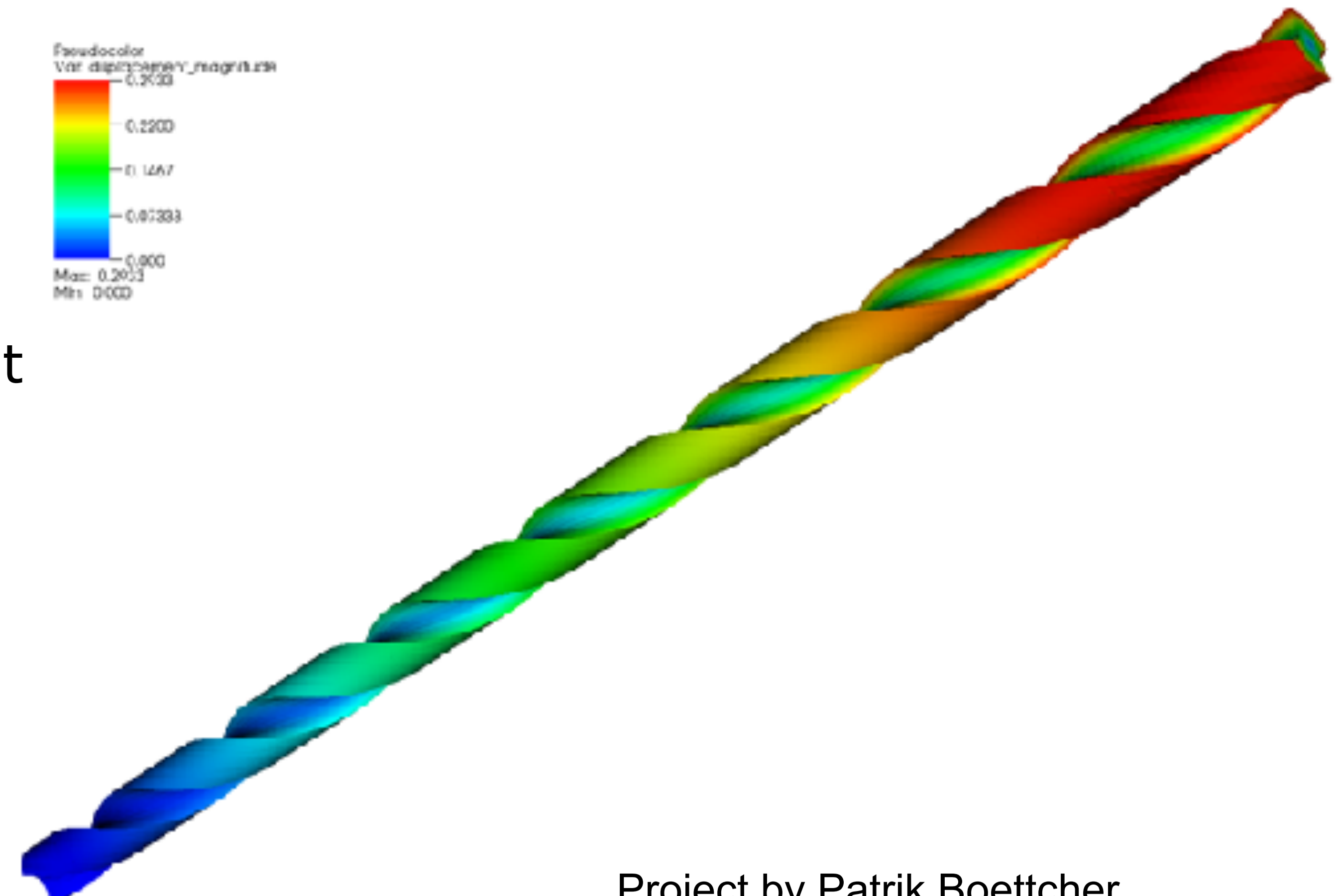
Project by Patrik Boettcher,
University of Heidelberg, 2012

A complete example

Step 8: Visualize

Here:

Magnitude of displacement (in mm)

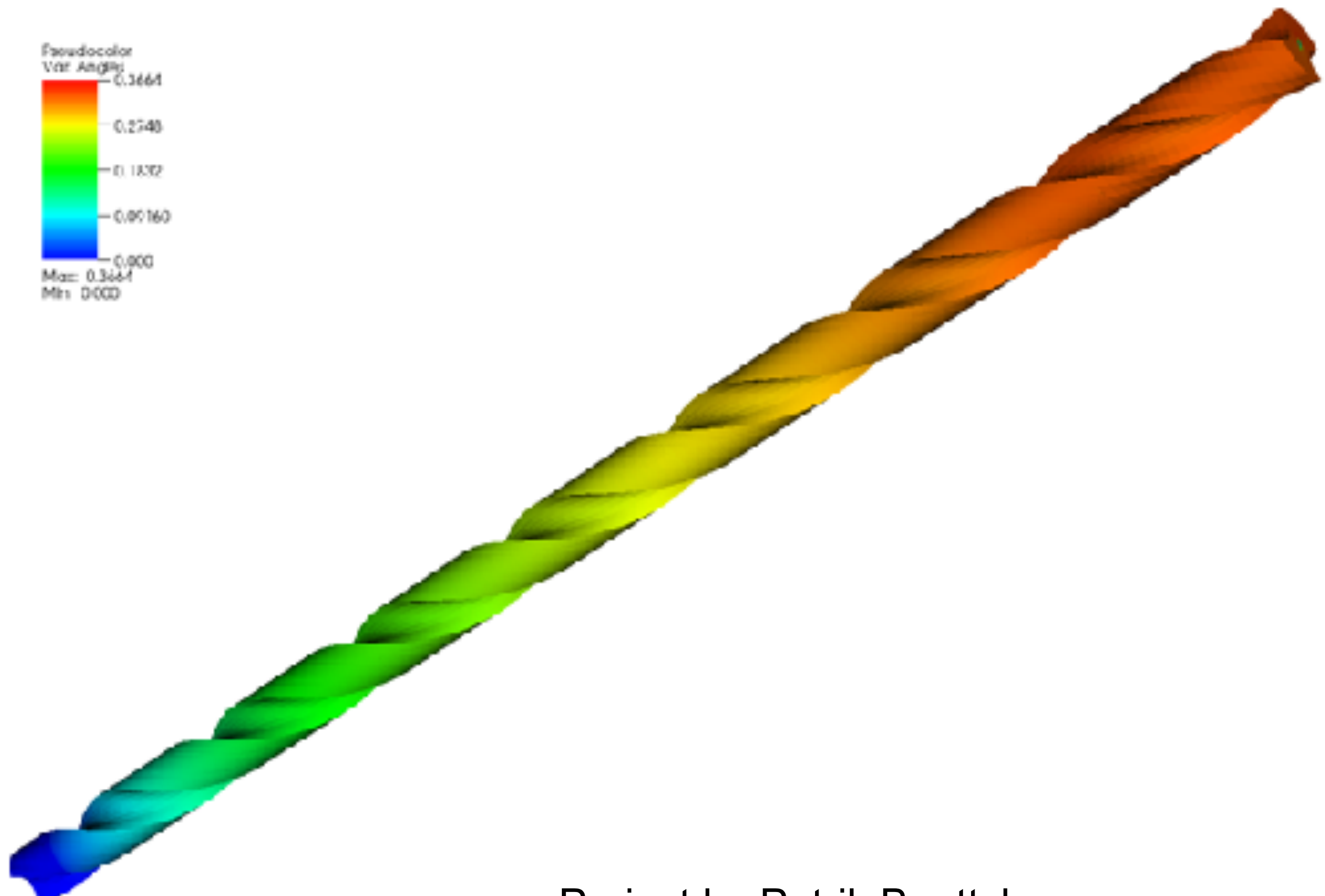


Project by Patrik Boettcher, University of Heidelberg, 2012

A complete example

Step 8: Visualize

Here:
Torsion
angle
(in degrees)

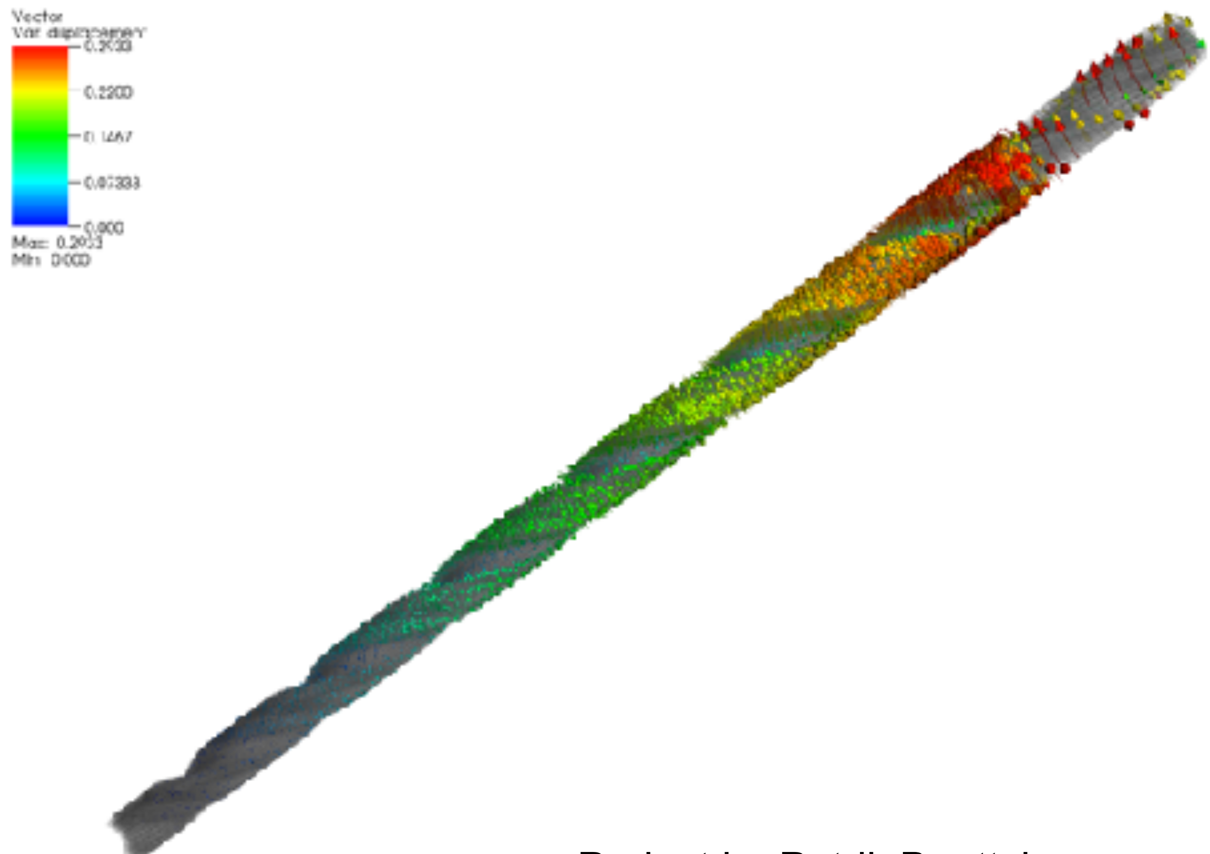


Project by Patrik Boettcher,
University of Heidelberg, 2012

A complete example

Step 8: Visualize

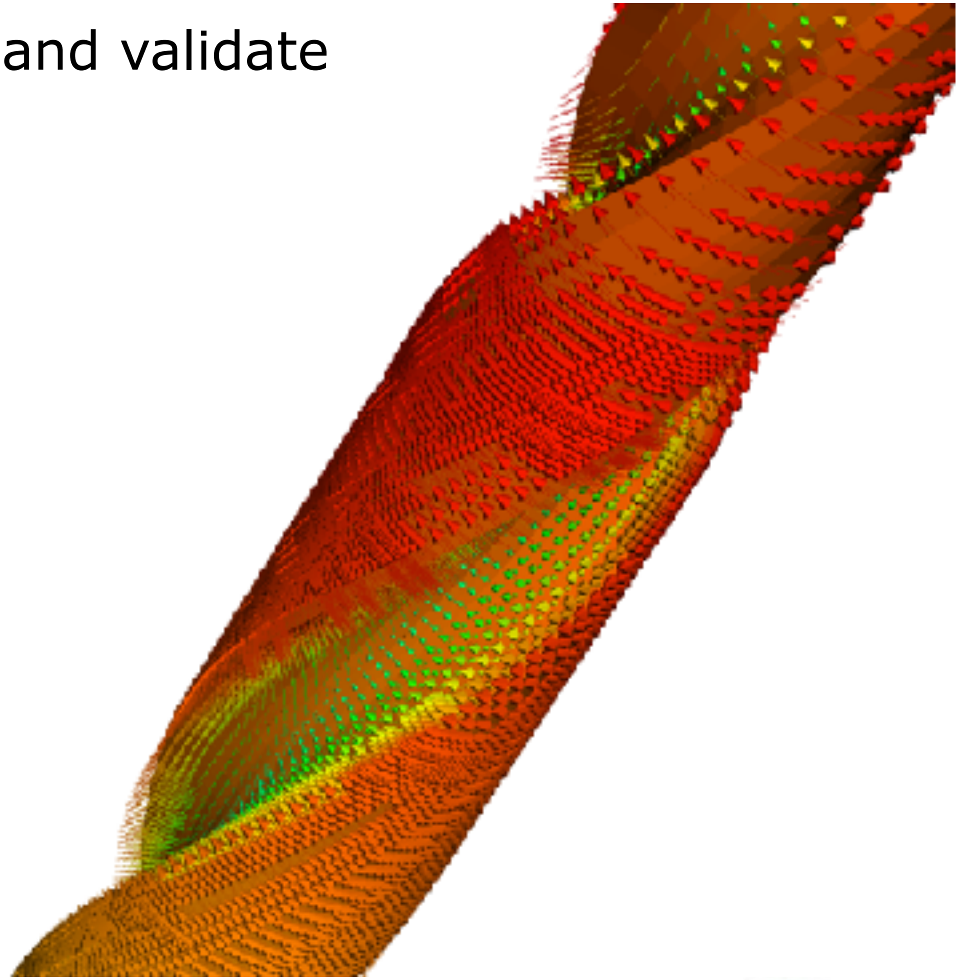
Here:
Disp-
lacement
(in mm)



Project by Patrik Boettcher,
University of Heidelberg, 2012

A complete example

Step 9: Repeat to optimize and validate



Project by Patrik Boettcher,
University of Heidelberg, 2012

Workflow for HPC in PDEs

Each of these steps…

- Identify geometry and details of the model
- Preprocess: Mesh generation
- Solve problem with FEM/FVM/FDM
- Postprocess: Visualize
- Repeat

…needs software that requires:

- domain knowledge
- knowledge of the math. description of the problem
- knowledge of algorithm design
- knowledge of software design and management

A Much More Complex Example from Wolfgang Bangerth

Goal: Simulate convection in Earth's mantle and elsewhere.

The tool of choice:

ASPECT

Advanced Solver for Problems in Earth's ConvecTion

http://aspect.dealii.org/

A Much More Complex Example from Wolfgang Bangerth

Goal: Simulate convection in Earth's mantle and elsewhere.

Questions:

- What drives plate motion?
- What is the thermal history of the earth?
- Do hot spots exist and how do they relate to global convection?
- Interaction with the atmosphere?
- When does mantle convection exist?
- What does that mean for other planets?

ASPECT - Challenges I

For convection in the earth mantle:

- Depth: ~35 – 2890 km
- Volume: $\sim 10^{12}$ km^3
- Resolution required: <10 km
- Uniform mesh: $\sim 10^9$ cells
- Using Taylor-Hood (Q2/Q1) elements: $\sim 3 \cdot 10^{10}$ unknowns
- At 10^5–10^6 DoFs/processor: 30k-300k cores!

Thermal convection is described by the relatively "simple" Boussinesq approximation:

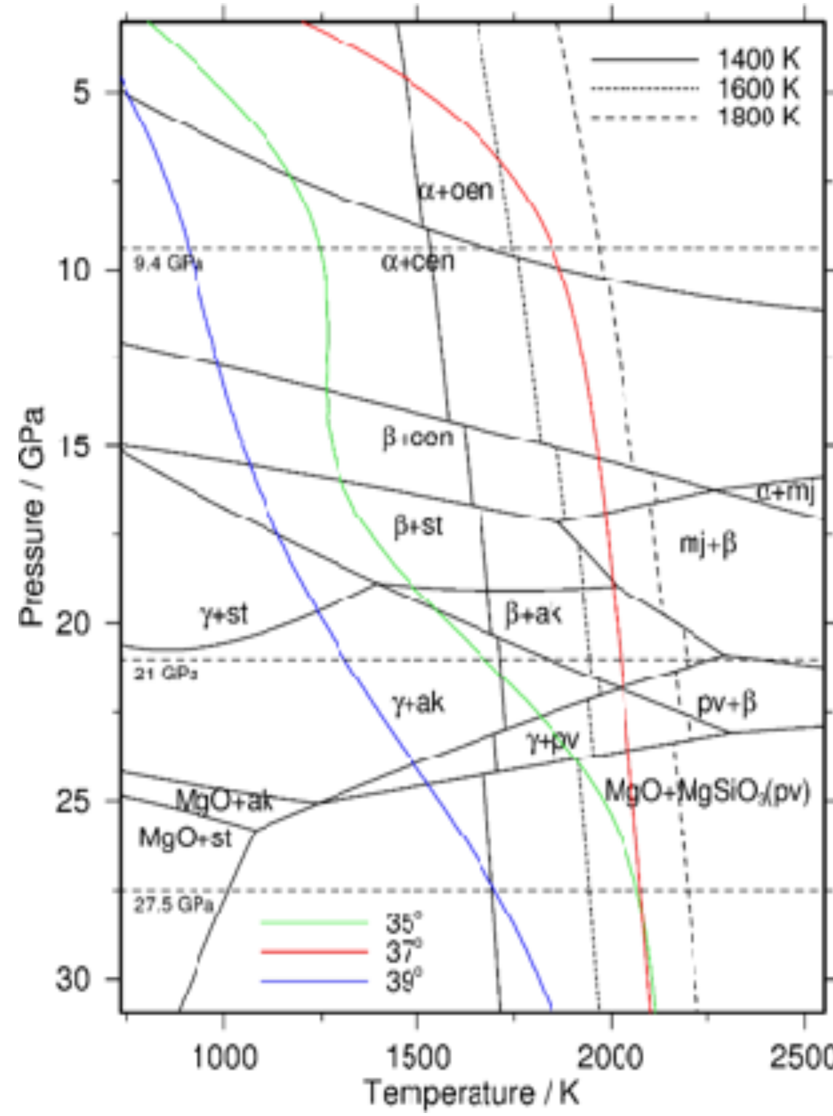
$$
\begin{aligned}
-\nabla \cdot (2\eta\varepsilon(\mathbf{u})) + \nabla p &= \rho(T)\mathbf{g}, \\
\nabla \cdot \mathbf{u} &= 0, \\
\frac{\partial T}{\partial t} + \mathbf{u} \cdot \nabla T - \nabla \cdot \kappa \nabla T &= \gamma,
\end{aligned}
$$

..this is not dissimilar from a typical "model problem"…

ASPECT - Challenges II

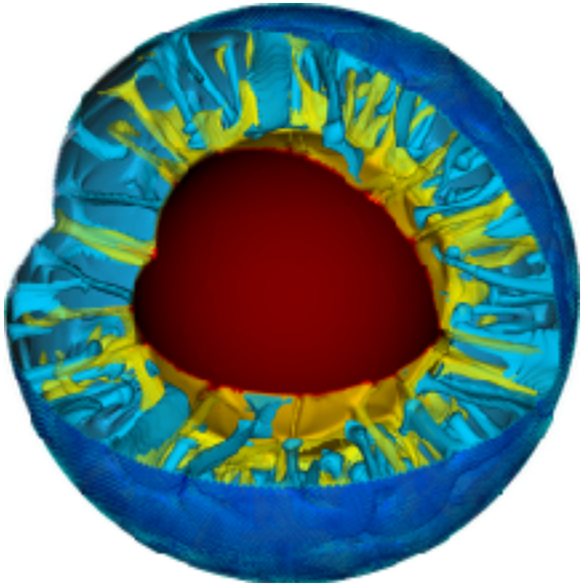
However, in reality:

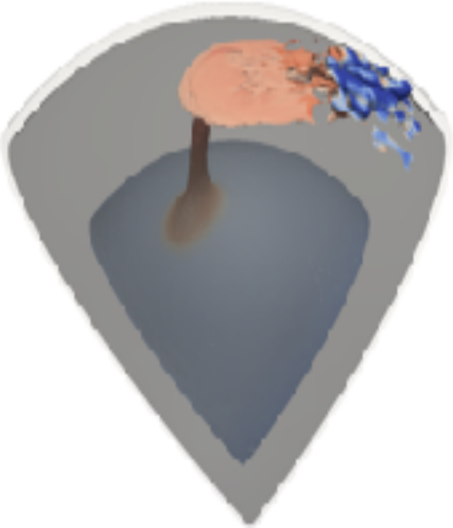
- All coefficients depend nonlinearly on
  - pressure
  - temperature
  - strain rate
  - chemical composition
- Dependence is not continuous
- Viscosity varies by at least 10^{10}
- Material is compressible
- Geometry depends on solution

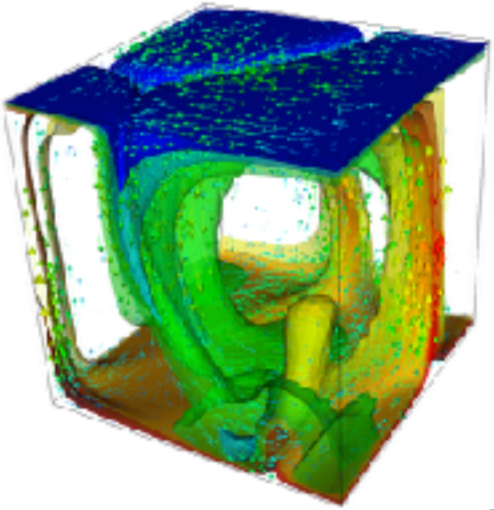


People want to change things:

- Geometries:
  – global
  – regional
  – model problems







People want to change things:

- Geometries:
  - global
  - regional
  - model problems
- Material models:
  - isoviscous vs realistic
  - compressible vs incompressible
- Boundary conditions
- Initial conditions
- Add tracers or compositional fields
- … …

- What happens to the solution: postprocessing

We need to think about the *whole* application:

- Adaptive meshes
- Nonlinear loops
- Efficient preconditioners
- Scalability to 10,000s of cores
- Where we can cut corners to make things faster

If the code is for the community:

- Extensibility
- Ease of use
- Documentation
- Needs to fit into the community workflow

Main QUESTION…

How to write such a Software?

Rough estimate:

From scratch, about 200.000 lines of code

- Realistically: 20.000 lines of code per year/per person
- **About 10 Years of a man's Work**

Will it be good? Well documented?

The Bitter Reality - I

Research software today:

- Typically written by graduate students
  – without a good overview of existing software
  – with little software experience
  – with little incentive to write high quality code

- Often maintained by postdocs
  – with little time
  – need to consider software a tool to write papers

- Advised by faculty
  – with no time
  – oftentimes also with little software experience

The Bitter Reality - II

Most research software is **not** of High Quality

There is a complexity limit to what we can get out of a PhD student

Most research software is never actually released as **OpenSource**

Can we be **inspired** by Existing **OpenSource** Solutions?

- Creating software is both an **art** and a **science**

What makes existing software successful?
(Best practices? Lessons learned?)

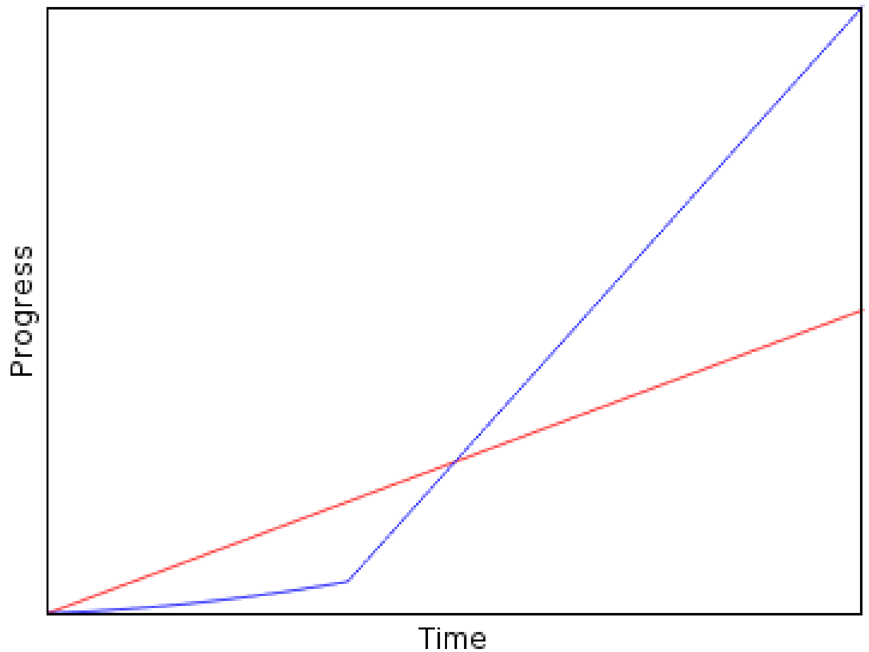
- We could learn from the answers!

- Use what others have already done (and *use for free!*):
  – Linear algebra packages like PETSc, Trilinos
  – Finite element packages like libMesh, FEniCS, deal.II
  – Optimization packages like COIN, CPLEX, SNOPT, …

- On this, build only what is application specific
- Use sound software design principles

An Question of Efficiency…

Progress over time:

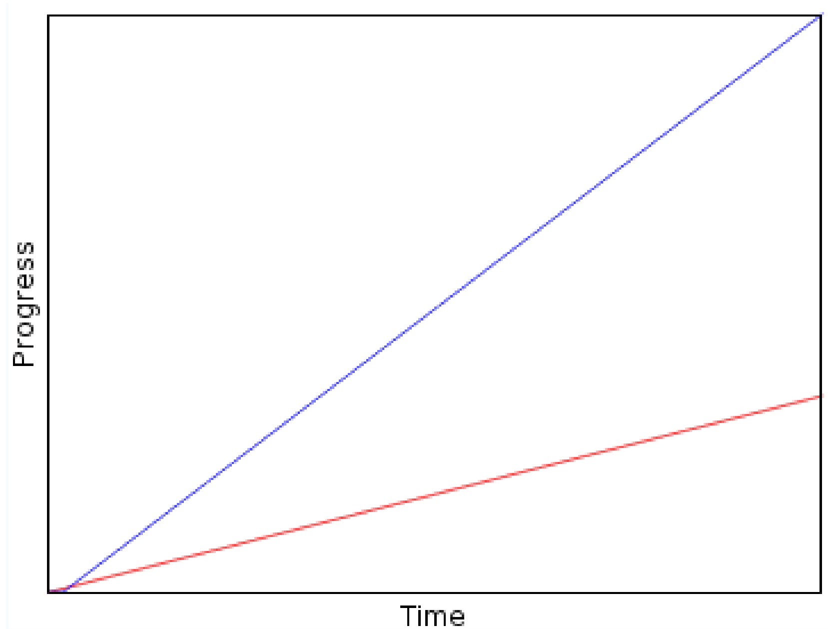


Red: Do it yourself. Blue: Use existing software.

Question: Where is the cross-over point?

An Question of Efficiency…

Progress over time, the real picture:



Red: Do it yourself. Blue: Use existing software.

Answer: Cross-over is after 2–4 weeks! A PhD takes 3–4 years.

…and an Ethical Question!

Would your **math** paper be accept, if you stated a theorem, and provided a graph showing that things work in one particular case?

(Provided you are not Fermat?)

Most computational papers **do not** provide a way to **reproduce** their results…

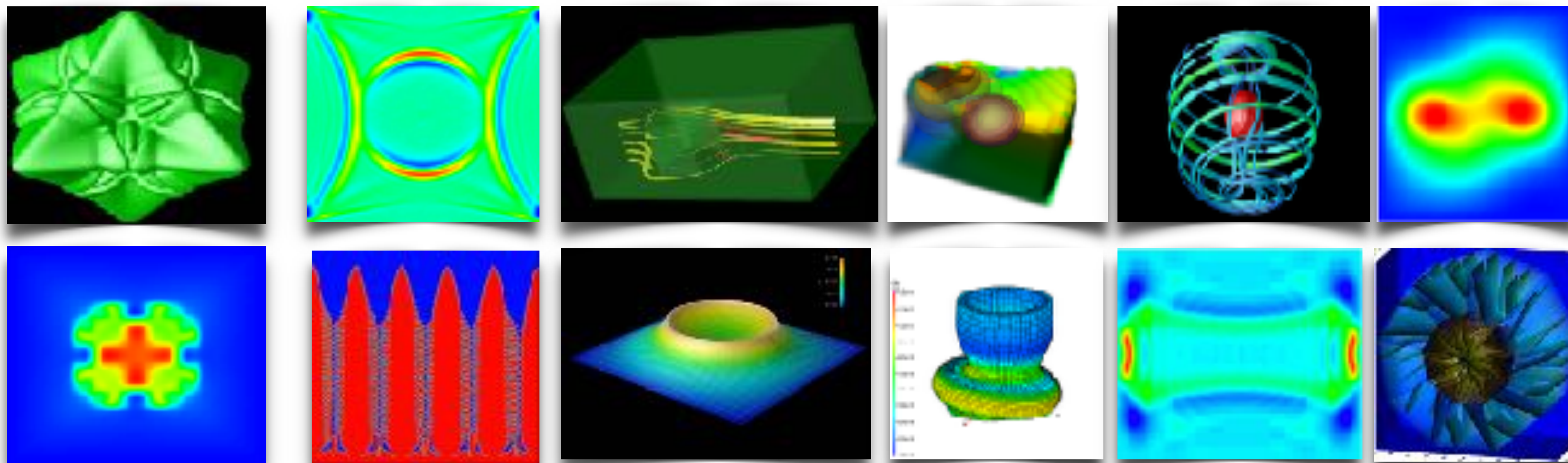
How does this affect our field?

- Reproducibility?
  (rare)
- Archival?
  (very rare)
- “Standing on the shoulders of giants”?
  (extremely rare)

A successful example

Our playground today:

The **deal.II** Library



A library for finite element computations that supports a large variety of PDE applications tailored to non-experts.

Why choose deal.II?

- It supports complex computations in many fields
- It is general (not area-specific)
- It has fully adaptive, dynamically changing 3d meshes
- It scales to 10,000s of processors
- It is efficient on today's multicore machines

Fundamental premise:
Provide building blocks that can be used in many different ways, **not a rigid framework.**

Examples of what can be done with deal.II:

- Biomedical imaging
- Brain biomechanics
- E-M brain stimulation

- Microfluidics
- Oil reservoir flow
- Fuel cells
- Transonic aerodynamics
- Foam modeling
- Fluid-structure interactions
- Atmospheric sciences

- Quantum mechanics
- Neutron transport
- Nuclear reactor modeling

- Numerical methods research

- Fracture mechanics
- Damage models
- Solidification of alloys
- Laser hardening of steel
- Glacier mechanics
- Plasticity
- Contact/lubrication models

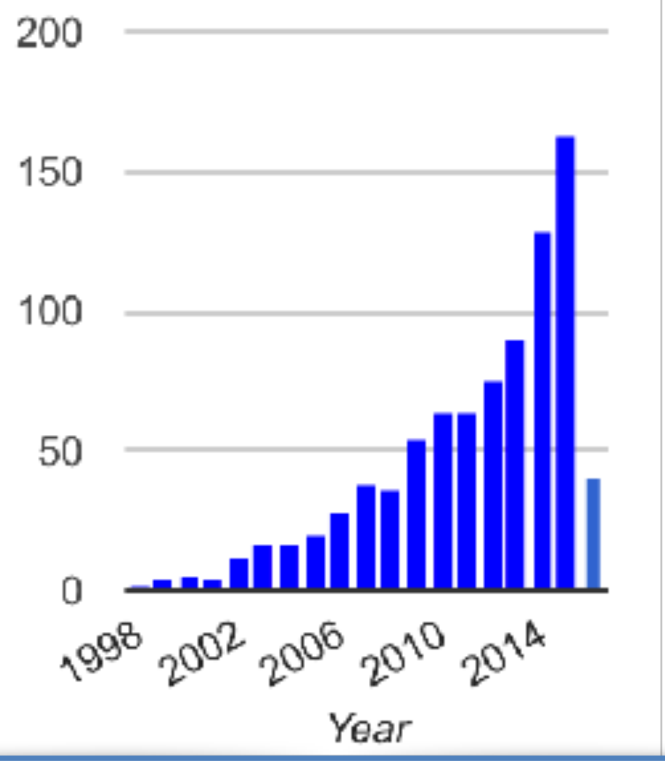
- Electronic structure
- Photonic crystals

- Financial modeling

- Chemically reactive flow
- Flow in the Earth mantle

- Many others…

Publications per year citing deal.II (Total as of today: 857)



What drives people towards deal.II?

What makes it successful?

General observations:

Success or failure of scientific software projects
is **not** decided on **technical** merit alone.

The *true* factors are beyond the code…
It is not enough to be a good programmer

In particular, what **really** counts is:

- **Utility** and **quality**
- **Documentation**
- **Community**

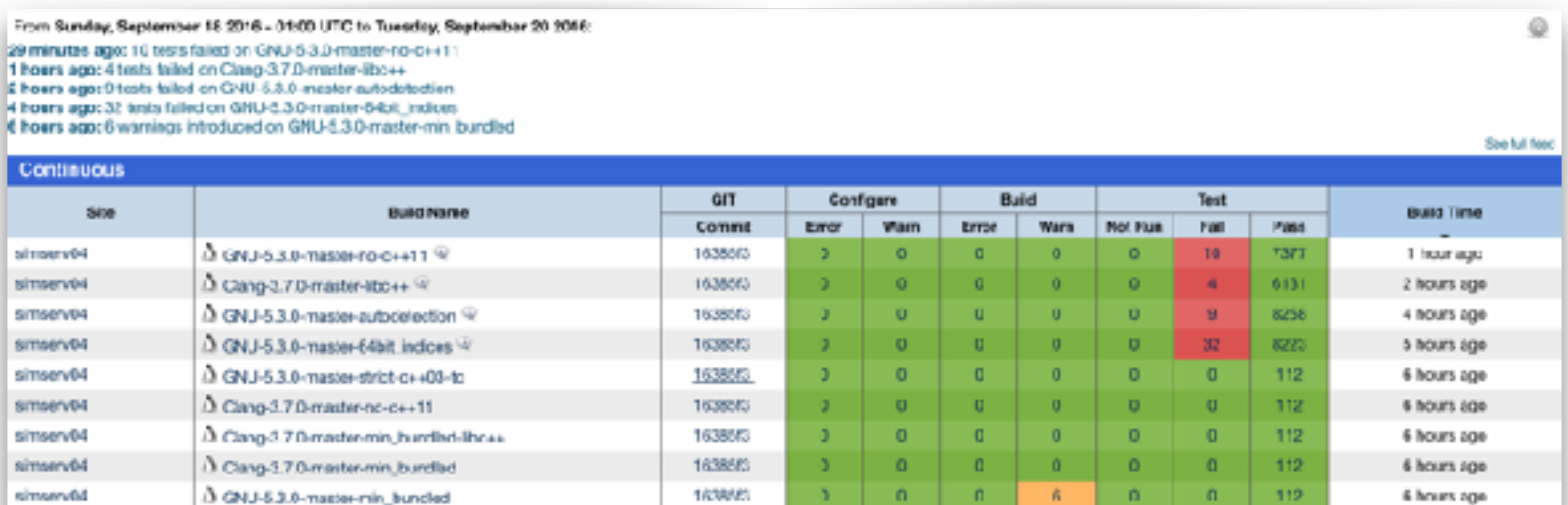
All of the big libraries provide this for their users.

Utility and Quality in deal.II

- **Lots of error checking in the code (with meaningful error messages!)**
  Two versions of the library, one with **debug** code enabled (range checking, internal condition checking, etc.) 30-50 times slower than **release** version.

- **Extensive test-suite**
  8000+ tests are run at each merge to master, with several different configurations. New releases of the library are issued *only* if no tests fail.

- **Code that goes in the the library is always peer reviewed**
  12 people have write access (and they act as code-reviewers). Everybody can make a “pull request”. Nobody merge their own pull request.


From Sunday, September 18 2016 - 01:00 UTC to Tuesday, September 20 2016:

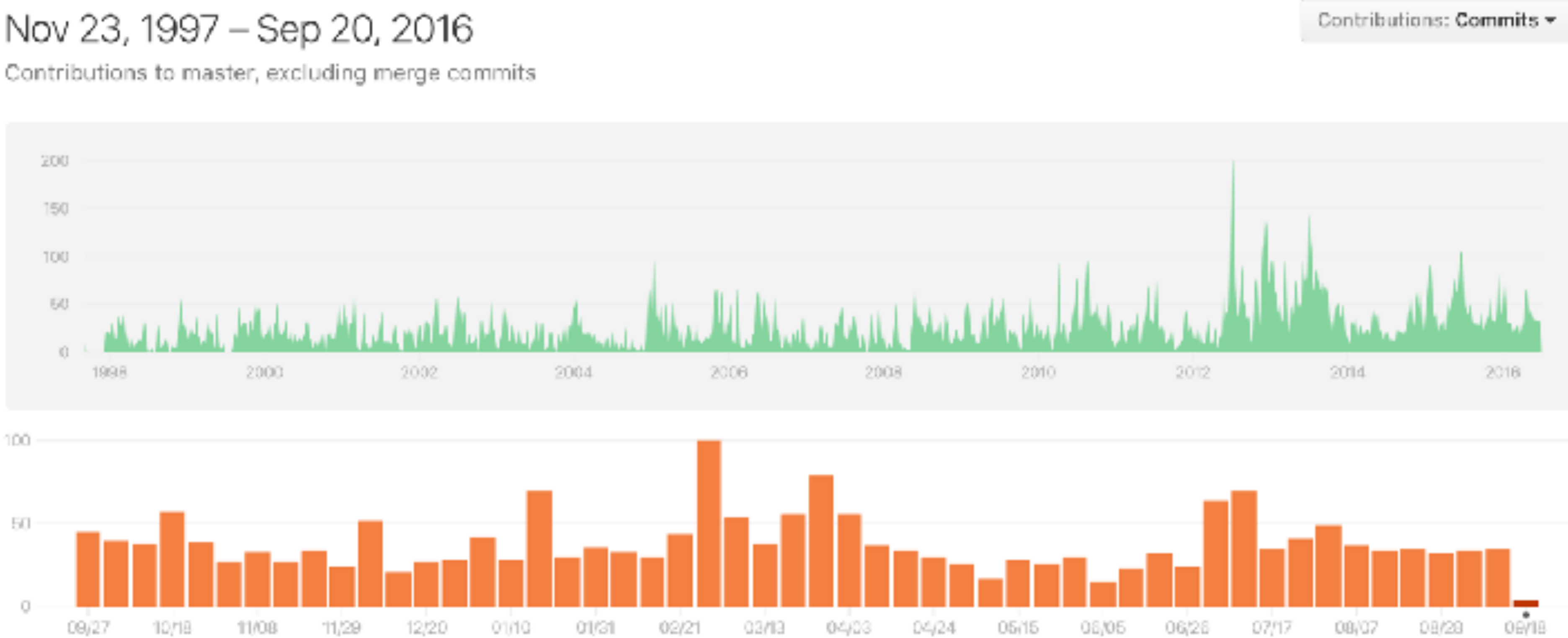
29 minutes ago: 10 tests failed on GNU-5.3.0-master-no-c++11
1 hours ago: 4 tests failed on Clang-3.7.0-master-libc++
4 hours ago: 9 tests failed on GNU-5.3.0-master-autodetection
4 hours ago: 32 tests failed on GNU-5.3.0-master-64bit_indices
6 hours ago: 6 warnings introduced on GNU-5.3.0-master-min_bundled

See full feed

Continuous

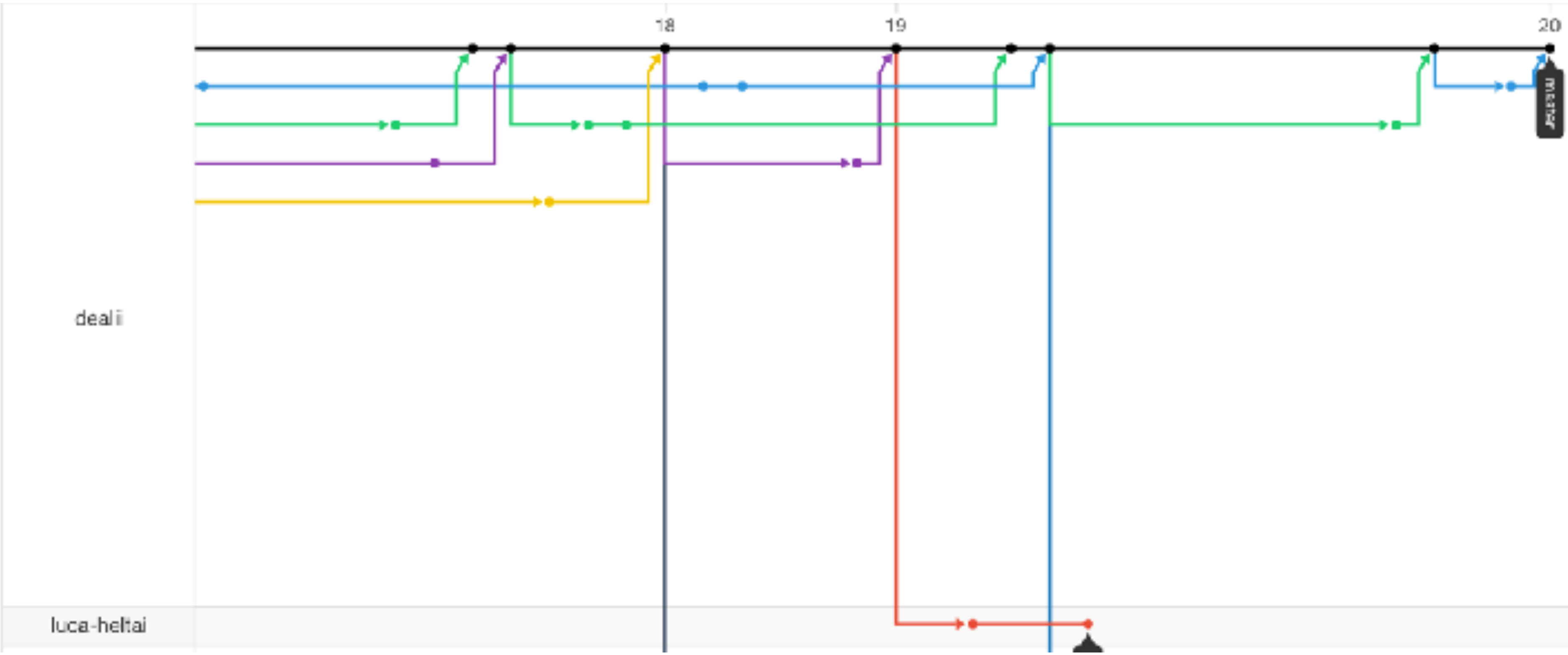
Site	Build Name	GIT	Configure		Build		Test			Build Time
		Commit	Error	Warn	Error	Warn	Not Run	Fail	Pass	
simserv04	GNU-5.3.0-master-no-c++11	[illegible]	0	0	0	0	0	10	7377	1 hour ago
simserv04	Clang-3.7.0-master-libc++	[illegible]	0	0	0	0	0	4	6131	2 hours ago
simserv04	GNU-5.3.0-master-autodetection	[illegible]	0	0	0	0	0	9	8256	4 hours ago
simserv04	GNU-5.3.0-master-64bit_indices	[illegible]	0	0	0	0	0	32	8223	5 hours ago
simserv04	GNU-5.3.0-master-strict-c++03-tc	[illegible]	0	0	0	0	0	0	112	6 hours ago
simserv04	Clang-3.7.0-master-no-c++11	[illegible]	0	0	0	0	0	0	112	6 hours ago
simserv04	Clang-3.7.0-master-min_bundled-libc++	[illegible]	0	0	0	0	0	0	112	6 hours ago
simserv04	Clang-3.7.0-master-min_bundled	[illegible]	0	0	0	0	0	0	112	6 hours ago
simserv04	GNU-5.3.0-master-min_bundled	[illegible]	0	0	0	6	0	0	112	6 hours ago

4 main developers - 8 developers (LH) - 77 contributors



An average of ~40 (peer reviewed and tested) commits per week, ~3 pull requests per day

“Oligarchic” Git Merge Strategy

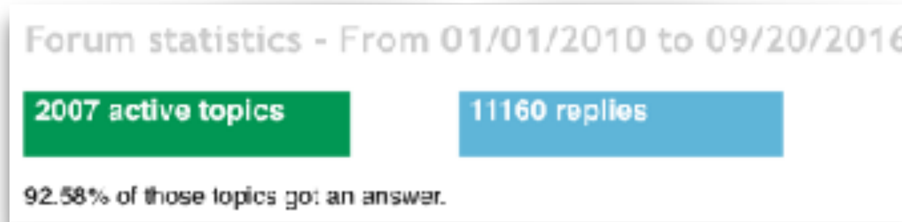


i) every set of changes is **rebased** on master, ii) a **pull request** is opened (on average **3 PR per day**), iii) other developers make a **peer review** on the proposed changes (using github facilities), iv) all comments/ **critiques** are **addressed**, v) an **automatic tester** is run on the proposed changes (Travis CI), vi) the pull request is merged on **master**, and the **full test-suite** is run on master

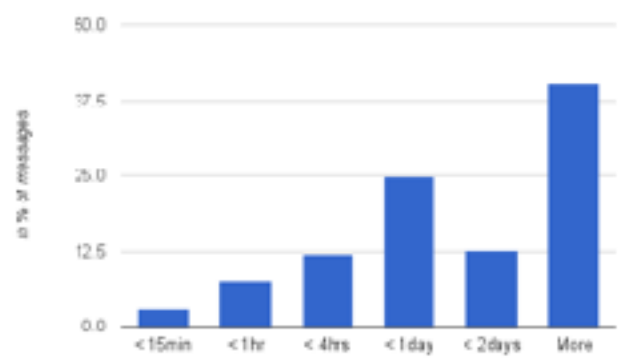
Community

- **Developers Mailing List**
  115 members, 617 topics
- **Users Mailing List**
  733 members, 2007 topics
- **Downloads**
  500+ downloads per month
- **Feedbacks**
  About 150 users give us feedbacks periodically

on average 10 emails per day



Time before first response:

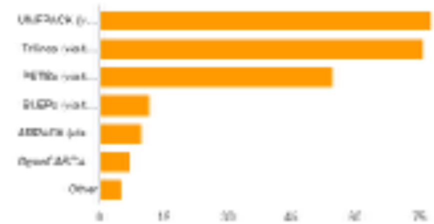


Survey on 150 users

On what operating system(s) do you use deal.II?

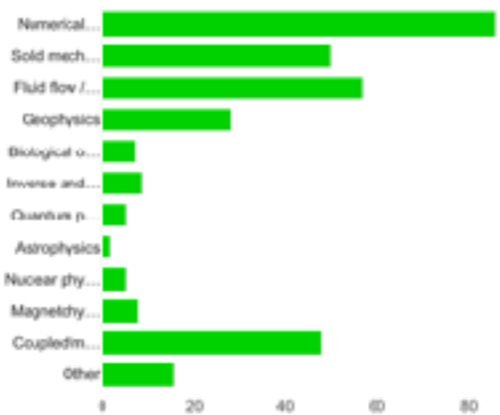
Linux	116	79.5%
Mac OS	35	24%
Windows	30	20.5%

Which of the following interfaces to other packages are you using?



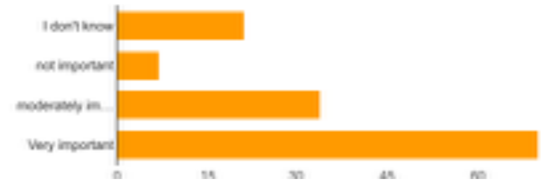
UMFPACK (via the SparseDirectUMFPACK class)	78	62.4%
Trilinos (via the TrilinosWrappers namespace)	76	60.8%
PETSc (via the PETScWrappers namespace)	55	44%
SLEPc (via the SLEPcWrappers namespace)	12	9.6%
ARPACK (via the ArpackSolver/ParpackSolver class)	10	8%
OpenCASCADE (via the OpenCASCADE namespace)	7	5.6%
Other	5	4%

From what area are the problems you typically solve?



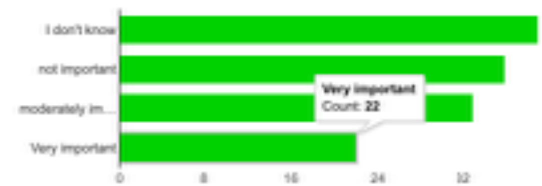
Numerical methods development	86	58.9%
Solid mechanics	50	34.2%
Fluid flow / aerodynamics	57	39%
Geophysics	28	19.2%
Biological or biomedical modeling	7	4.8%
Inverse and imaging problems	9	6.2%
Quantum physics	5	3.4%
Astrophysics	2	1.4%
Nuclear physics	5	3.4%
Magnetohydrodynamics	8	5.5%
Coupled/multiphysics problems	48	32.9%
Other	16	11%

MPI performance [How useful or important would the improvement/implementation of the followi



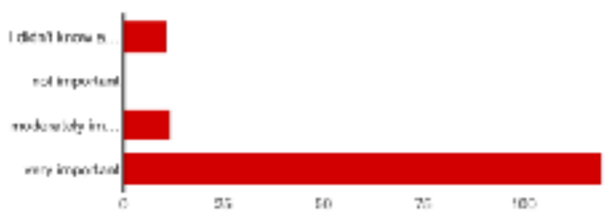
I don't know	21	15.9%
not important	7	5.3%
moderately important	34	25.8%
Very important	70	53%

C1 elements, other nonstandard elements [How useful or important would the improvement/imp



I don't know	39	30%
not important	36	27.7%
moderately important	33	25.4%
Very important	22	16.9%

Tutorial programs (https://dealii.org/developer/doxygen/deal.II/Tutorial.html) [Please rate the importance of the following documentation components:]



I didn't know about it	11	7.7%
not important	0	0%
moderately important	12	8.4%
very important	120	83.9%

SISSA

All Results HERE

Documentation and Education

- Installation instructions/README/FAQs

- Within-function comments
- Function interface documentation
- Class-level documentation
- Module-level documentation
- Worked “tutorial” programs
- Recorded, interactive demonstrations

deal.II has **10,000+ HTML pages**. 170,000 lines of code are actually documentation (**~10 man years of work**).

There are **68** recorded video lectures on YouTube (**Wolfgang Bangerth**).

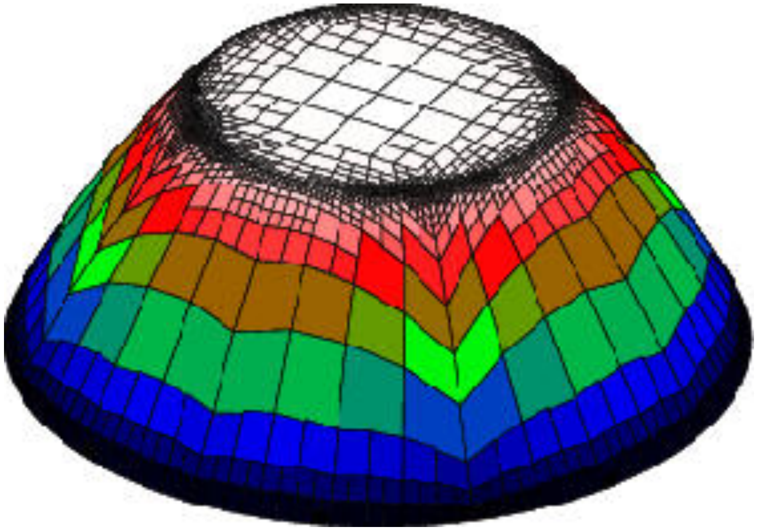
deal.II comes with ~60 extensively documented tutorial programs:

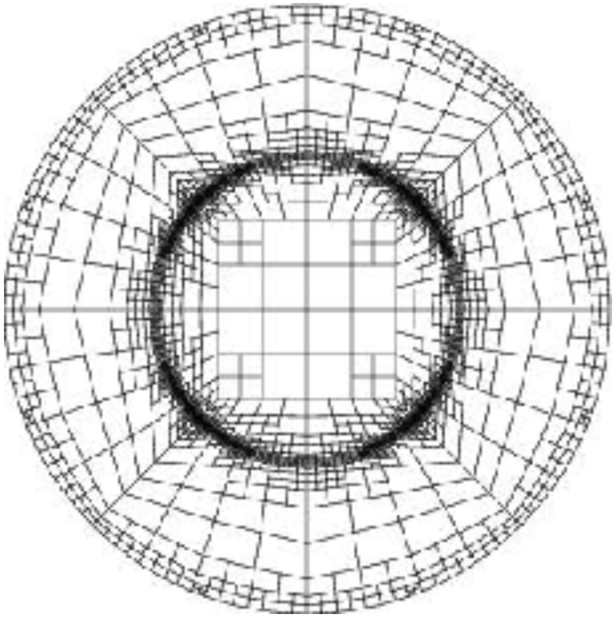
- From small Laplace solvers (~100s of lines)
- To medium-sized applications (~1000s of lines)
- Intent:
  - teach deal.II
  - teach advanced numerical methods
  - teach software development skills

This is what *really* drives users to deal.II

Step-6:

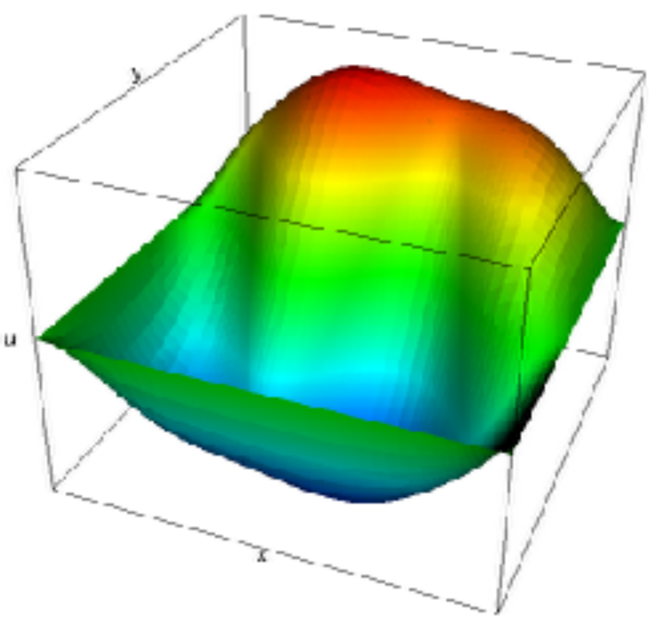
- Laplace equation, variable coefficient
- Adaptive mesh refinement
- 118 lines of code

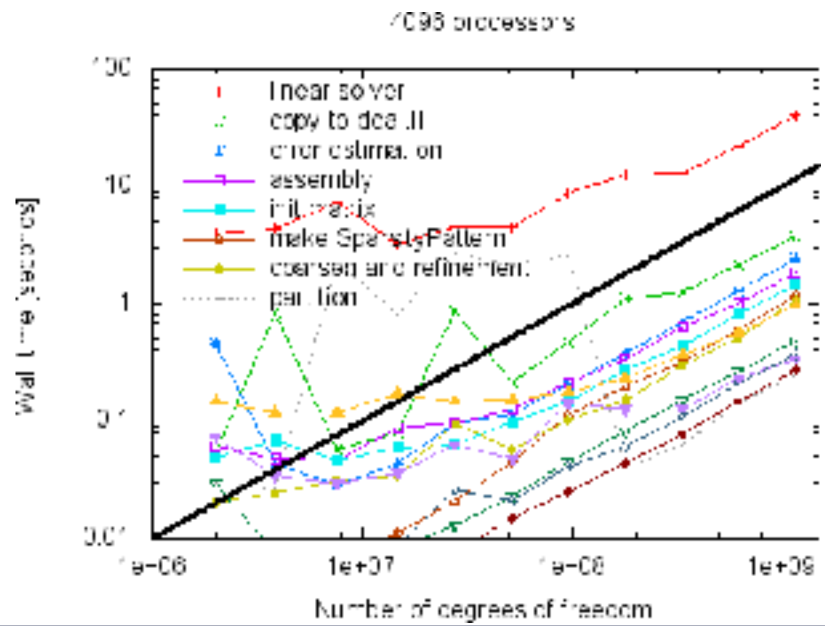




Step-40:

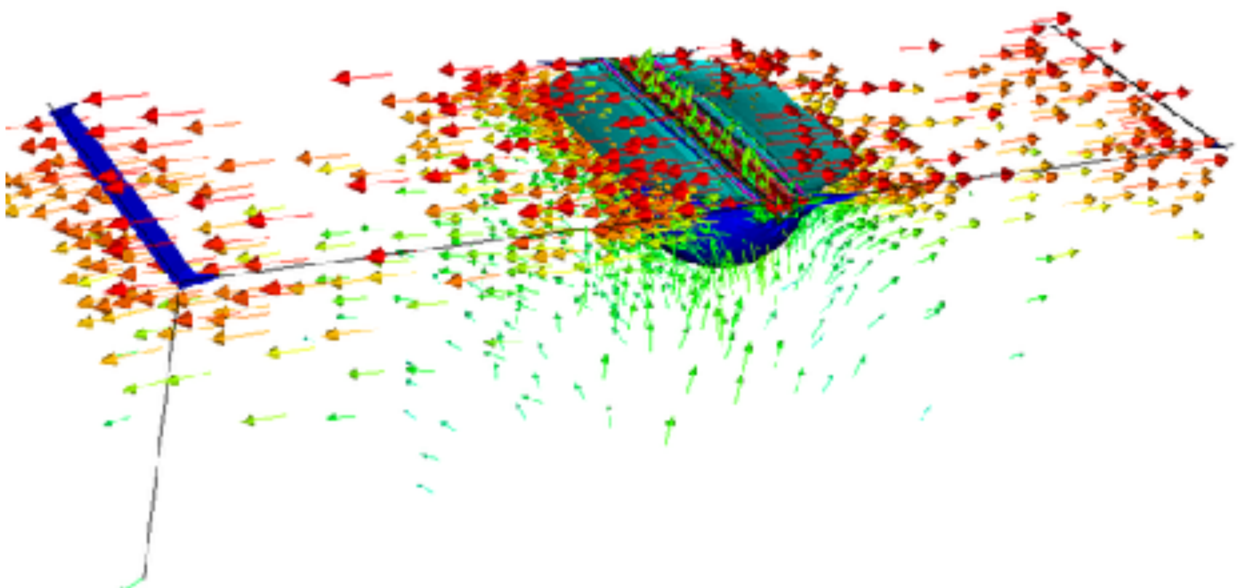
- Laplace equation, variable coefficient, 2d or 3d
- Adaptive mesh refinement
- Massively parallel: runs on 16k cores
- 138 lines of code





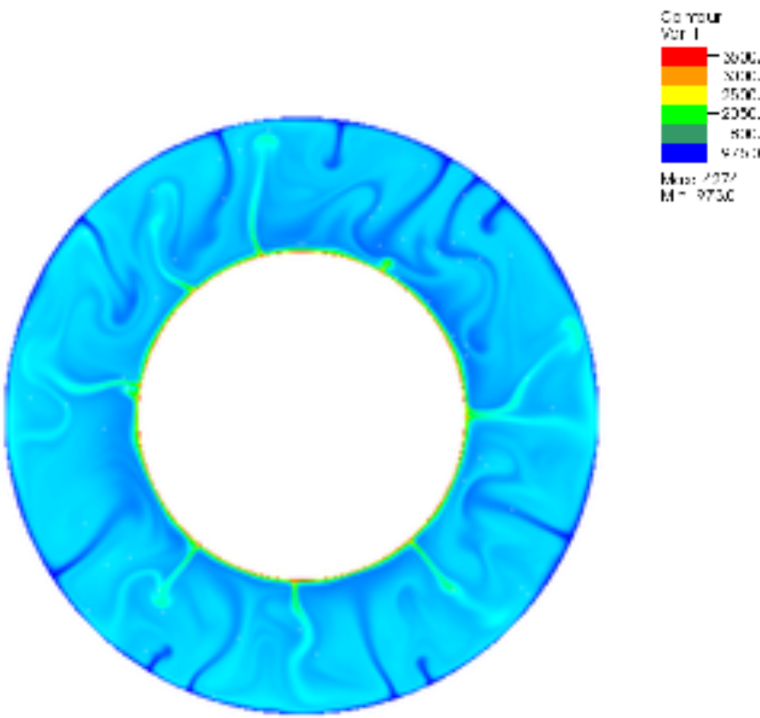
Step-22:

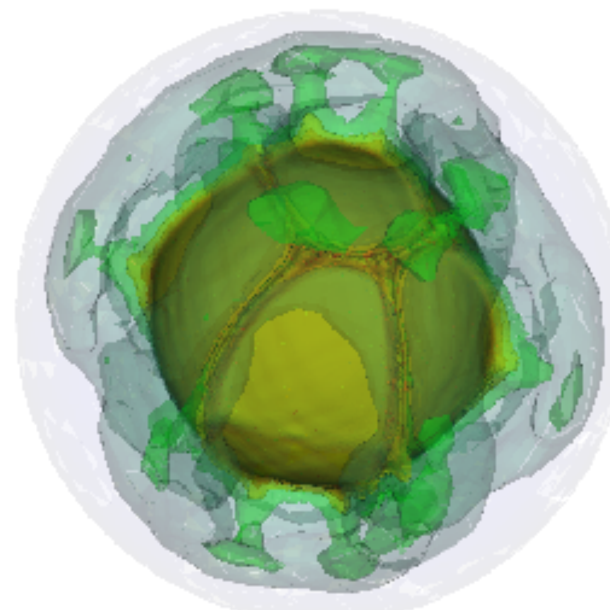
- Stokes equations, “interesting” boundary conditions, 2d/3d
- Adaptive mesh refinement, advanced solvers
- 206 lines of code



Step-31/32:

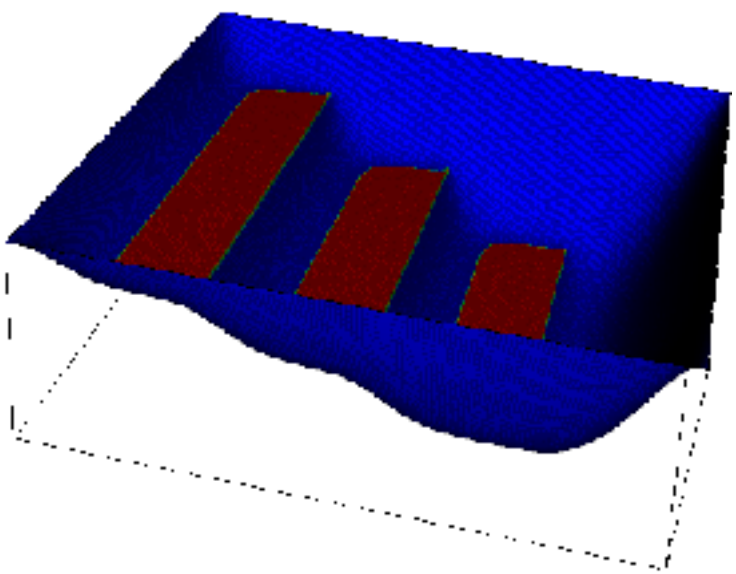
- Boussinesq equations, realistic material models, 2d/3d
- Adaptive mesh refinement, advanced solvers, parallel
- 864 lines of code

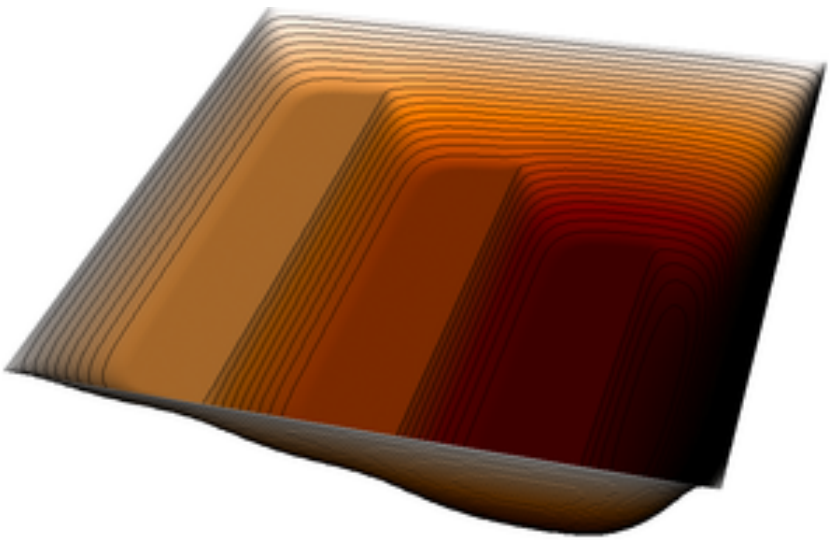




Step-41:

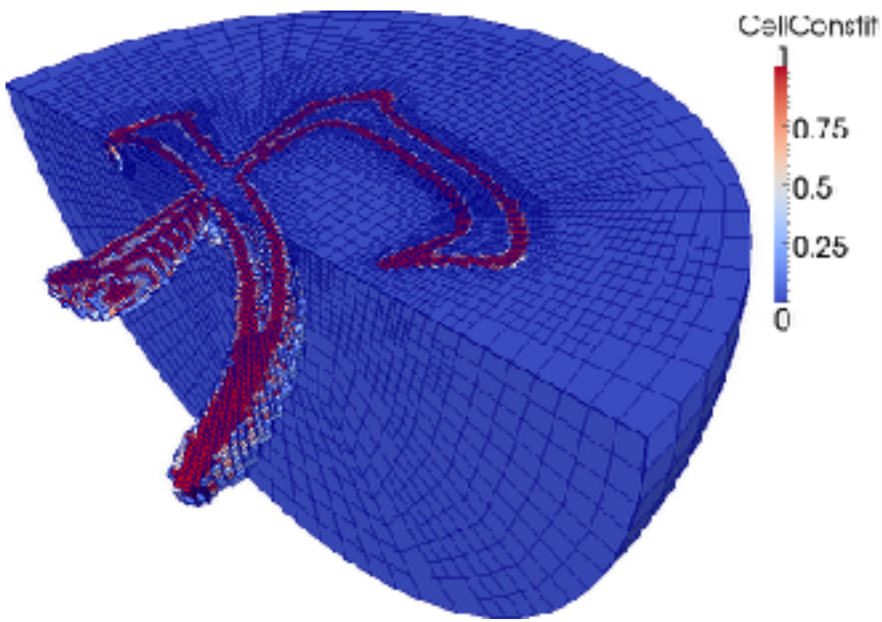
- Contact problem: Membrane over an obstacle
- Active set/Newton solver
- 177 lines of code

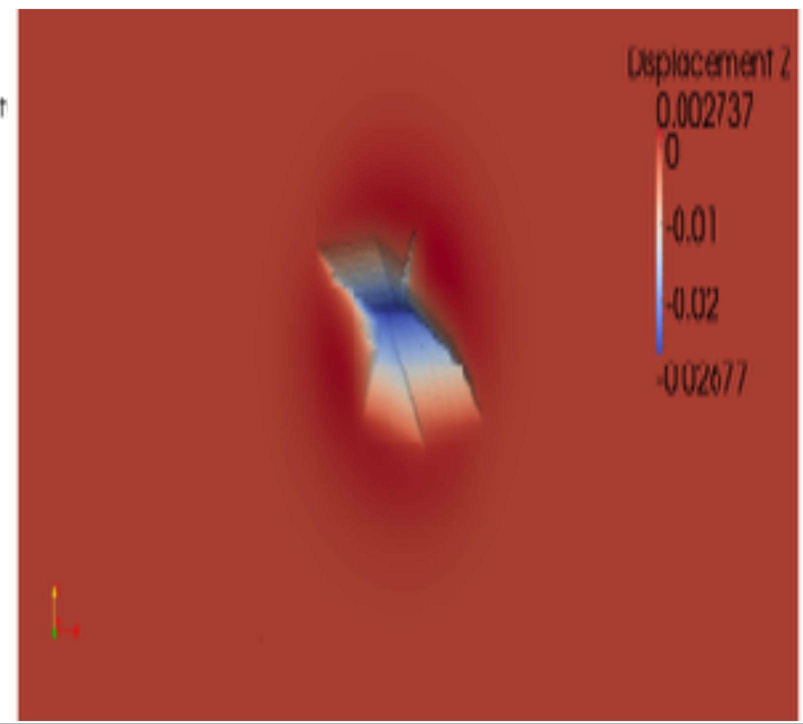




Step-42:

- Elasto-plastic contact problem
- Active set/Newton solver, multigrid, 3d, parallel
- 586 lines of code





What a student can expect

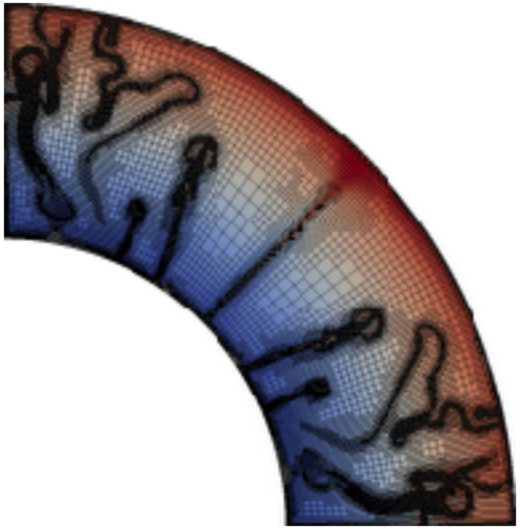
Because they no longer have to write most of their codes, a student can achieve in 3 years with deal.II:

- Solve a complex model
- With realistic geometries, unstructured meshes
- Higher order finite elements
- Multigrid-based solver
- Parallelization
- Output in formats for high-quality graphics
- Results almost from the beginning: a wide variety of tutorials allow a gentle start

SISSA

There are also large applications (not part of deal.II):

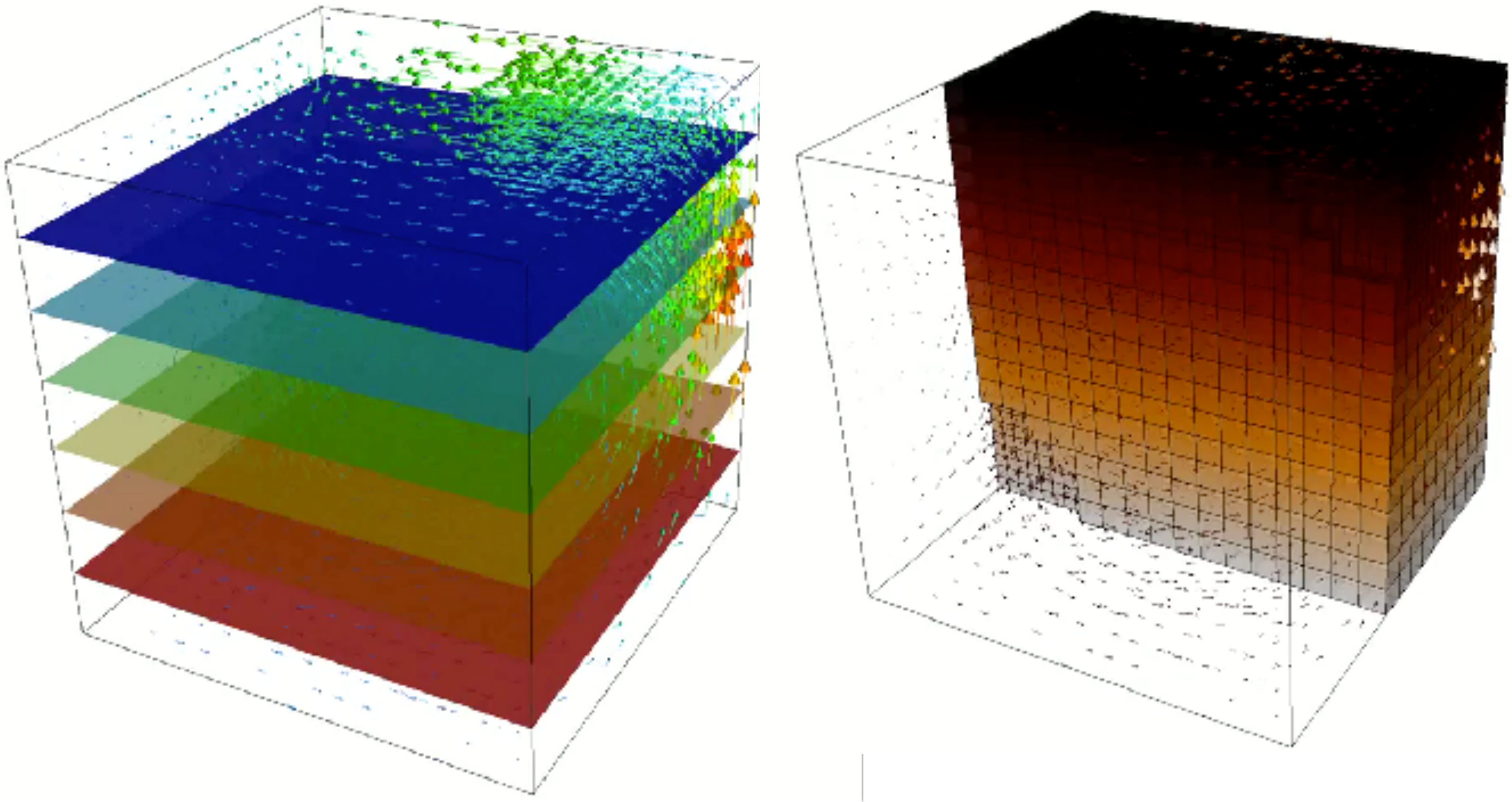
- *Aspect:* Advanced Solver for Problems in Earth Convection
  – ~60,000 lines of code
  – Open source: http://aspect.dealii.org/



- *OpenFCST:* A fuel cell simulation package
  – Supported by an industrial consortium
  – Open source: http://www.openfcst.org/

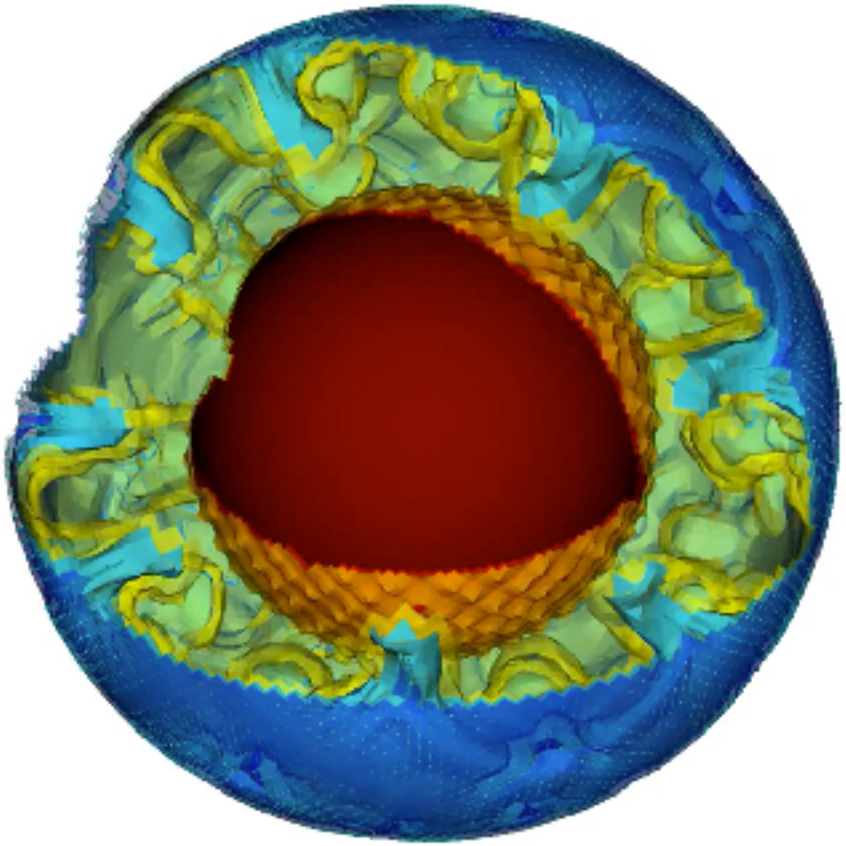
- *WaveBEM*: A nonlinear solver for ship-wave interaction
  – *Supported by a mixed consortium (OpenViewSHIP)*
  – *~80,000 lines of code*
  – Open source*:* http://github.com/mathLab/WaveBEM

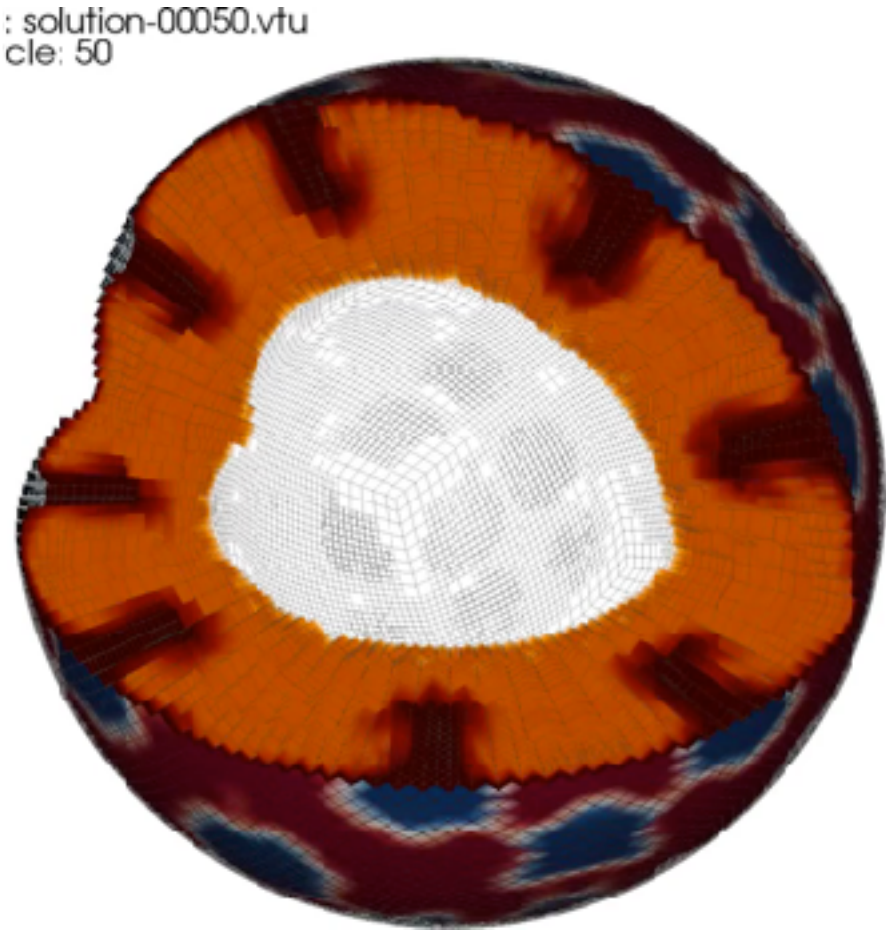
Aspect Example - I



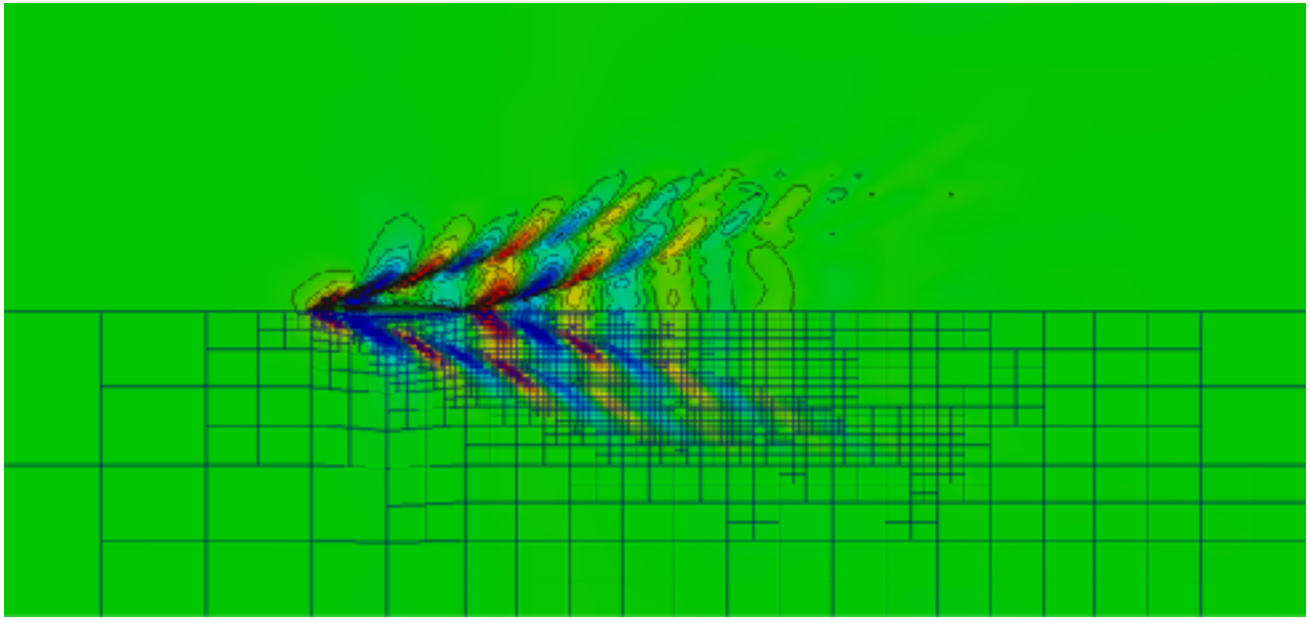
http://aspect.dealii.org/

Aspect Example - II





http://aspect.dealii.org/

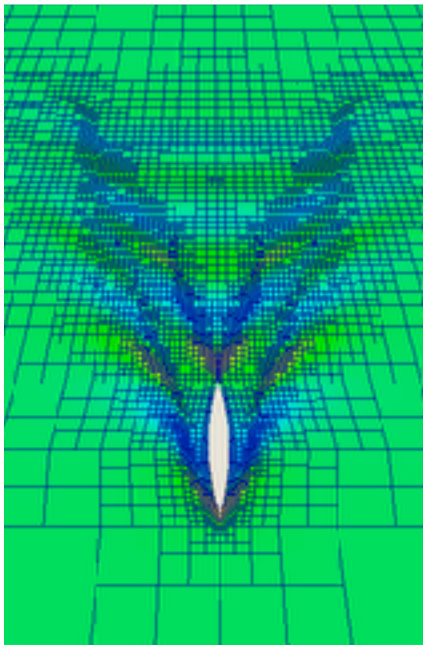


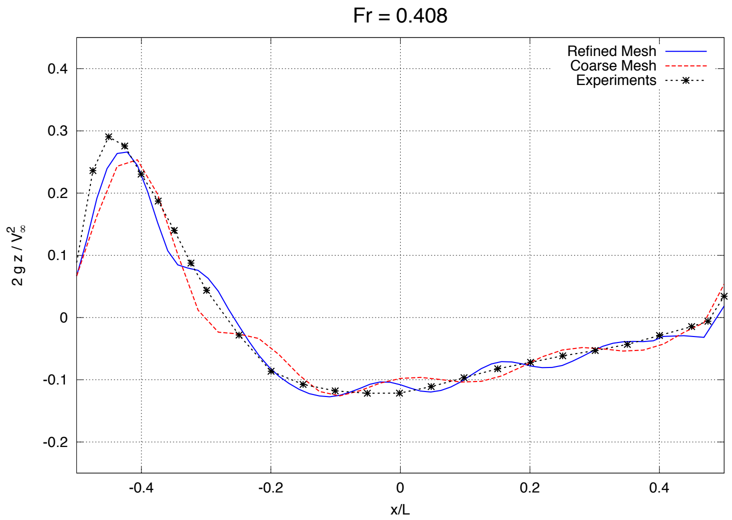
WaveBEM:

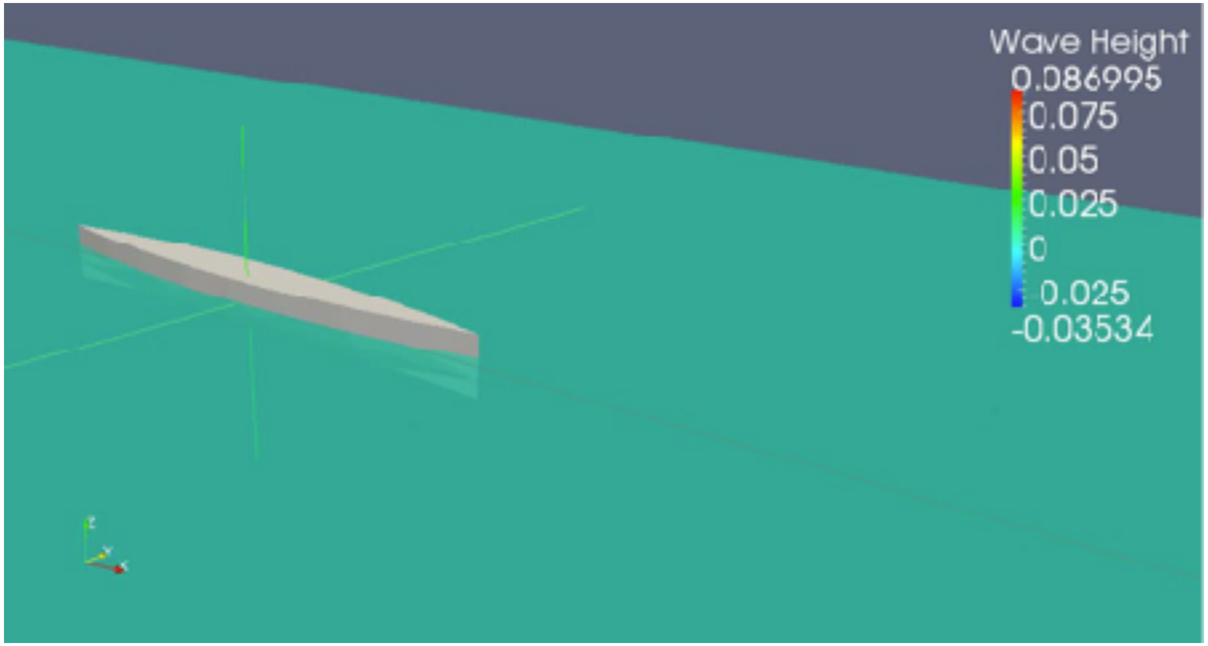
Nonlinear solver for ship-wave interaction using potential flow

(Andrea Mola, Luca Heltai, Nicola Giuliani,

Antonio DeSimone, SISSA)




Fr = 0.408
Refined Mesh
Coarse Mesh
Experiments
0.4
0.3
0.2
0.1
0
-0.1
-0.2
2 g z / V^2_∞
-0.4
-0.2
0
0.2
0.4
x/L


Wave Height
0.086995
0.075
0.05
0.025
0
0.025
-0.03534

Conclusions - I

What this development model means for us:

- We can solve problems that were previously intractable

- Methods developers can demonstrate applicability
- Applications scientists can use state of the art methods

- Our codes become far smaller:
  – less potential for error
  – less need for documentation

  – lower hurdle for “reproducible” research (publishing the code along with the paper)

- **More impact/more citations** when publishing one's code

Conclusions - II

What this development model means for our community:

- Faster progress towards “real” applications
- Leveling the playing field – excellent online resources are there for *all*
- Raising the standard in research:

  – can't get 2d papers published (easily) any more
  – reviewers can require state-of-the-art solvers
  – allows for easier comparison of methods

Conclusions - III

Computational science has spent too much time where everyone writes their own software.

By building on existing, well written and well tested, software packages:

- We build codes *much* faster
- We build better codes
- We can solve more realistic problems