# Refactoring a research code for multi-physics simulations: the 4C experience

**Georg Hammerl [1], Sven Berger [1], Sebastian D. Proell [2], Matthias Mayr [3], Erik Faulhaber [4], Niklas Neher [5], Ingo Scheider [1], Hoang-Giang Bui [1], Martin Kronbichler [6], Alexander Popp [3], Wolfgang A. Wall [2], Daniel Höche [1], Christian Cyron [1]**

[1] Helmholtz-Zentrum Hereon
[2] Technical University of Munich
[3] University of the Bundeswehr Munich
[4] University of Cologne
[5] University Stuttgart
[6] Ruhr University Bochum

Helmholtz-Zentrum hereon

# Outline

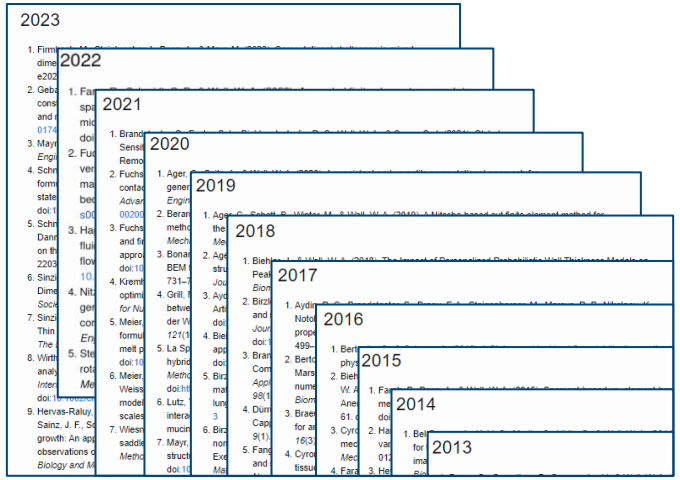
Multi-physics code project 4C

Lessons learned from
- Restarting from scratch
- Refactoring

Summary

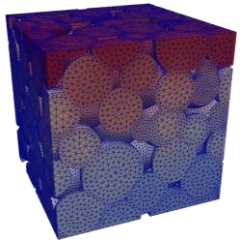
# Code project 4C – Overview

- Multi-physics simulations on the continuum scale
- 1.1 Mio lines of code developed over 20 years
- **> 400 peer-reviewed publications**, > 60 PhD theses

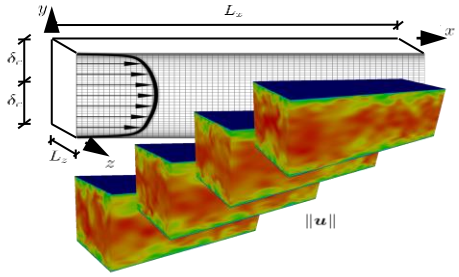



https://baci.pages.gitlab.lrz.de/website/publications.html

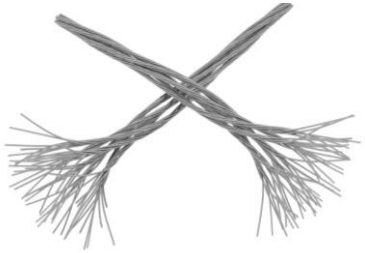
## Solid



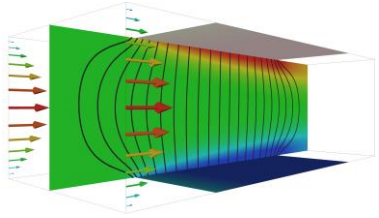
## Fluid



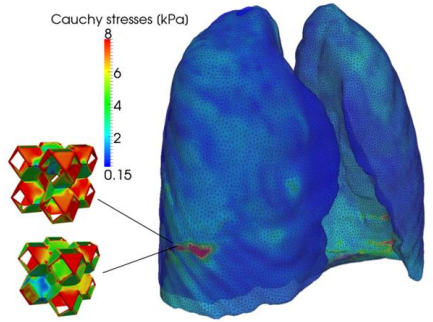
## Structures

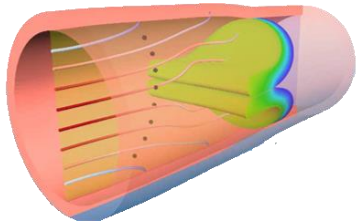


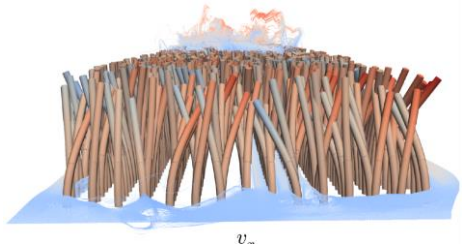
## Scalar transport

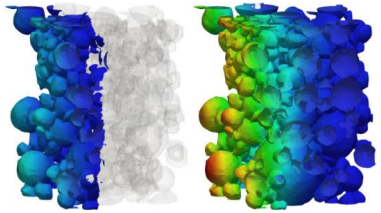


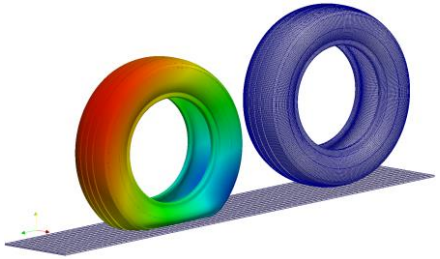
## Multi-physics Applications / Multiscale simulations

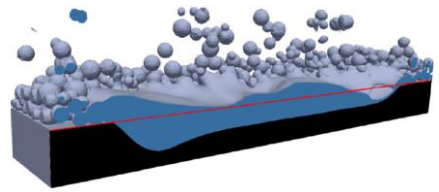










# Code project 4C – Goals

- Multi-physics research code
- Analyze real-world problems by means of computational mechanics
- Efficient use of current High Performance Computing (HPC) systems

**Guiding principle:** Application-motivated fundamental research

**Science**
- Material modeling
- Biopolymer networks
- Mixed-dimensional modeling
- Fluid/structure interaction
- Numerical methods & algorithms
- ...

**Engineering**
- Contact & tribology
- Aerospace engineering
- All-solid-state batteries
- Additive manufacturing
- Structural health monitoring of bridges
- ...

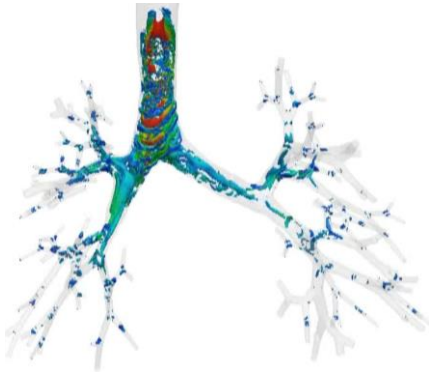
**Biomedicine**
- Respiratory system
- Cardiovascular system
- Musculoskeletal system
- Tumors
- Stomach
- Stents & stent grafts
- Growth & remodeling
- ...

# Code project 4C – Timeline

**Policy:** all development in one code

2006 (10 Devs) **BACI** Uses Trilinos

2017 (8 Devs) **ExaDG** Fluid solver based on deal.II



2019 Successor (prototype) (2 Devs)

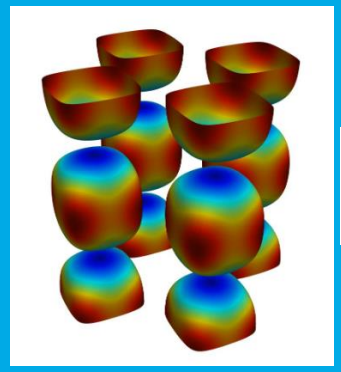
2022 Start of massive refactoring (6 Devs) **TrixiParticles.jl** Julia based solver for particle-based methods

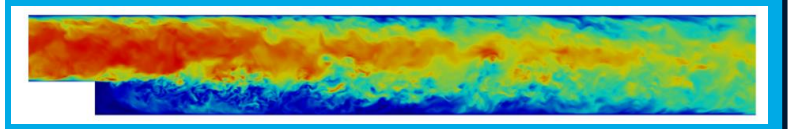
2024 **4C** (30 Devs)

# Launching side projects

## ExaDG

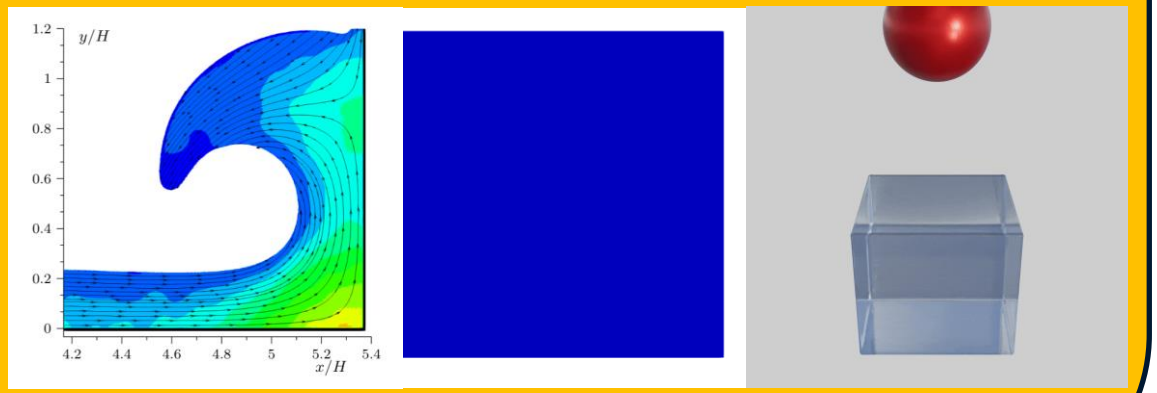
- LES & DNS fluid solver
- **D**iscontinuous **G**alerkin methods
- Fast matrix-free operator evaluation (HPC)
- Based on the HPC library deal.II
- Started with PhD-thesis of Niklas Fehn
- **github.com/exadg/exadg**





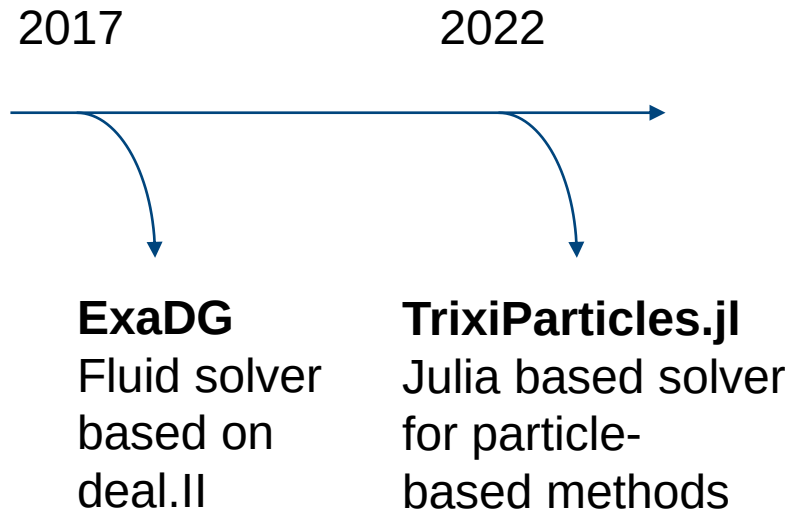
## TrixiParticles.jl

- Wide range of Particle-based methods SPH, DEM*, Lagrange*, PIC**
  * in dev version, ** in development
- Within Trixi-framework of Julia-based solvers
- **github.com/trixi-framework/TrixiParticles.jl**



# Launching side projects

- Desired functionality too far away from existing code base
- Small team starts new code project
- Based on advanced open source libraries to **avoid starting from scratch**

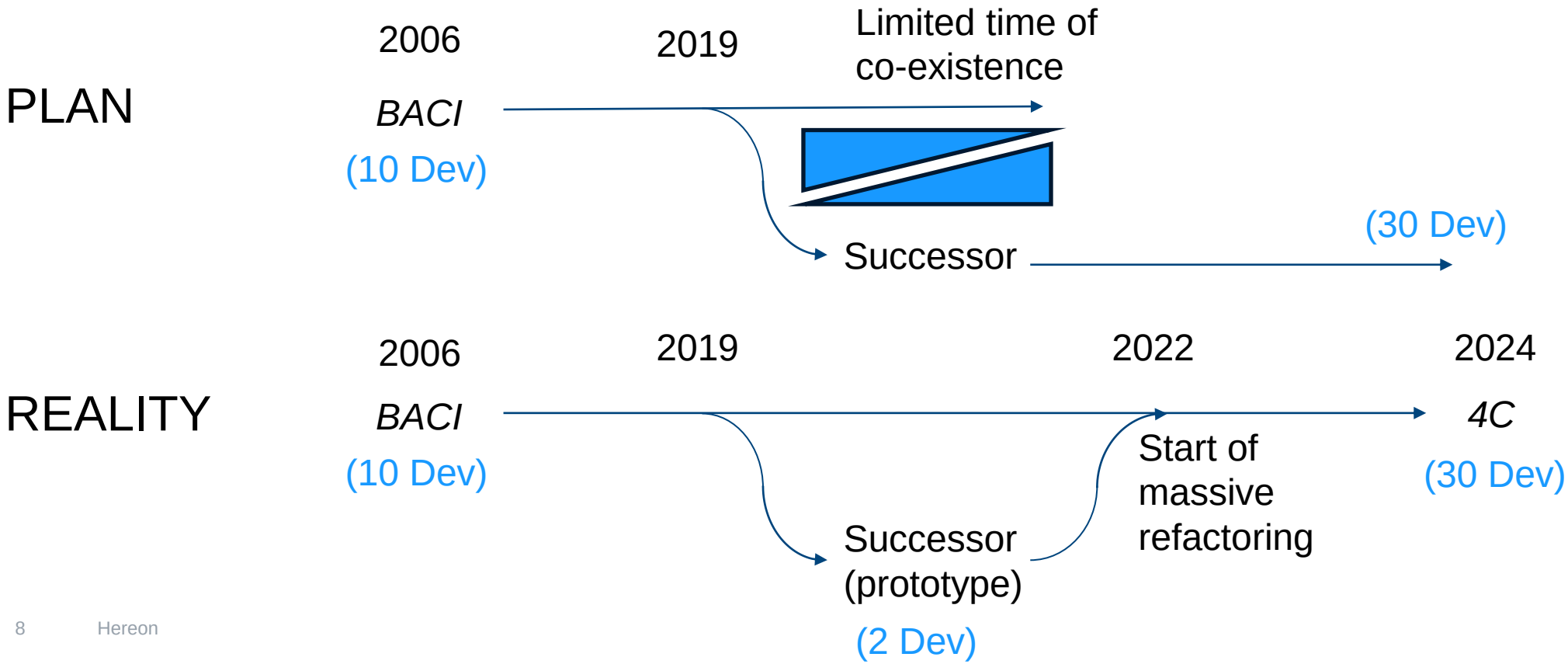



## Lessons learned

- Fast progress of the new projects
- Clean state of the new projects
- Difficult to integrate work of new projects
- Efficient coupling of old and new codes is challenging

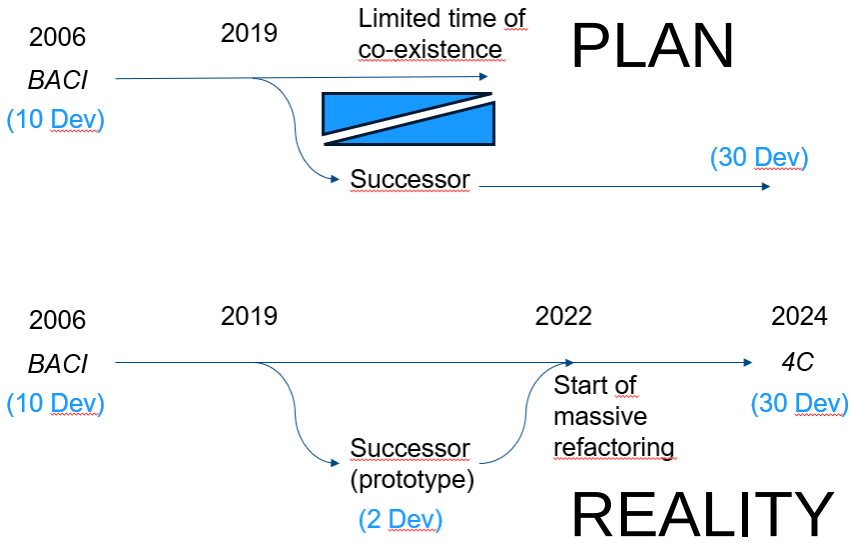
# Main project 4C

- Point of departure: Monolithic code
  - Code was developed as a by-product of PhD/research projects
  - Initial concepts reached their limits
  - Lack of modularity
  - Compilation only in a specific configuration with unclear dependencies (at single institute)



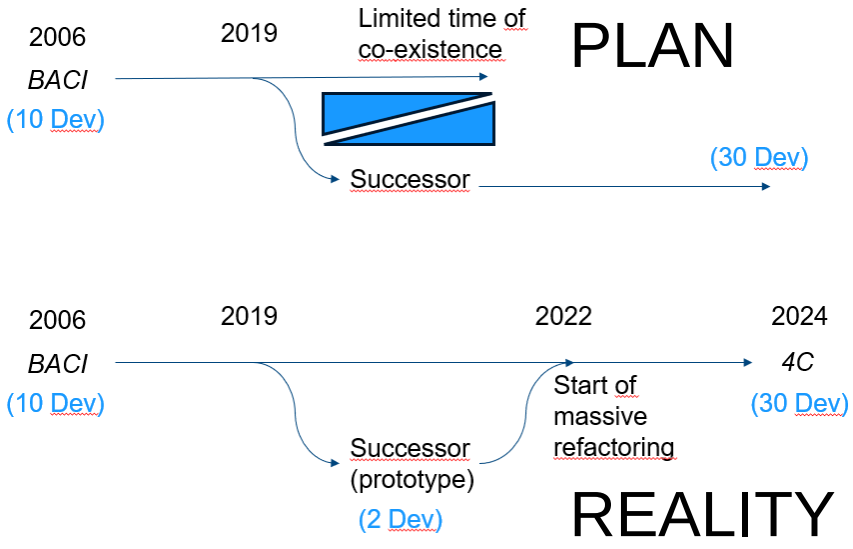
# Main project 4C

- Chicken-and-egg problem
  - The new code is not yet ready for the transition
  - For developers (=PhD students & researchers), specific project goals and teaching obligations always had priority compared to code maintenance
  - Lack of dedicated RSE engineers to drive the new code forward
  - More new features have been integrated in the old code than ported to the new code



# Main project 4C – REALITY

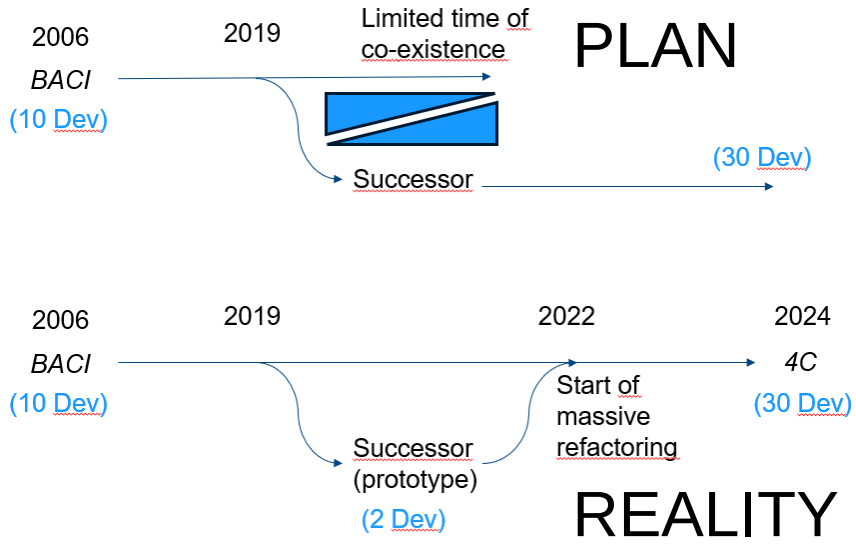
- Successor development is put on hold
- Intensified refactoring was started with the knowledge gained
- Task force groups include all developers




- Code refactoring and modularization
  - Clear external dependencies
  - Clear internal dependencies
  - Code modernization

# Main project 4C – REALITY

- What did we achieve?
  - Robustness (e.g. testing)
  - (Portability)
  - Ease-of-use (understandability and comprehensibility)

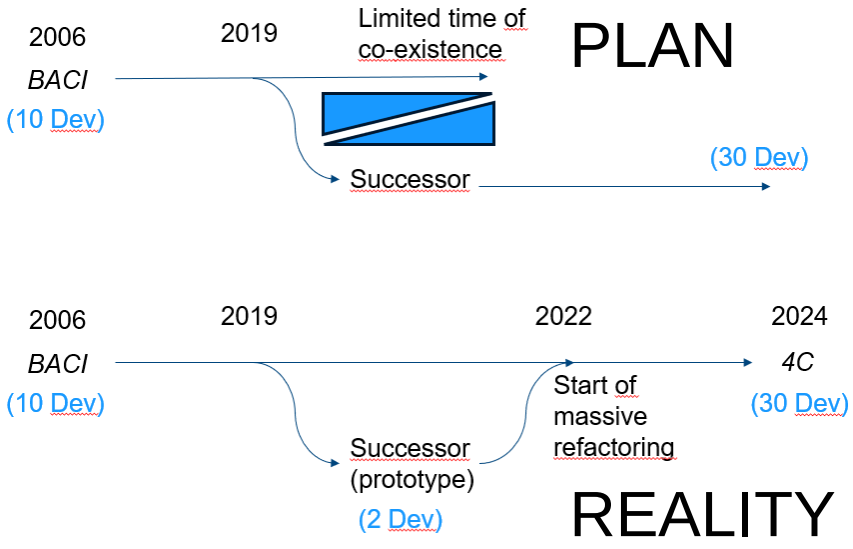



- Aims at FAIR principles

Interoperability

Coupling with other code projects (especially our launched side-projects)

# Main project 4C – Lessons learned




- Restart from scratch demands for sufficient RSE engineers (you need a strong enough basic framework)
- Refactoring allows for more distributed work packages (you still have the big framework running)

- Refactoring and modularization revealed fragile code parts and technical debts
- Use of modern tools very helpful (e.g., modern CMake with targets & preset files, clang-tidy, address-sanitizer)

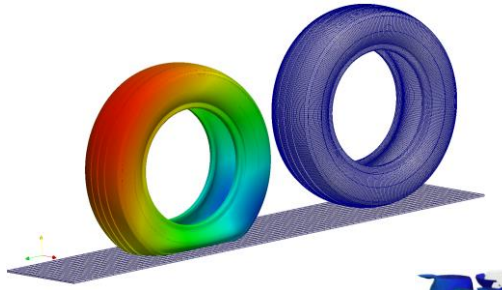
- The question remains which way is faster to obtain an up-to-date code base

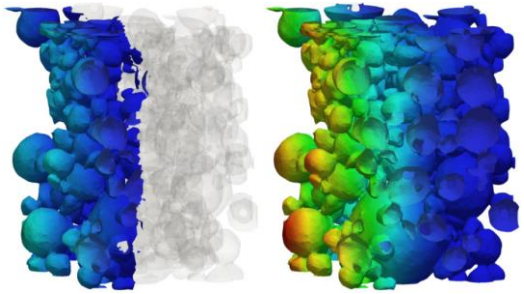
# 4C – Community building

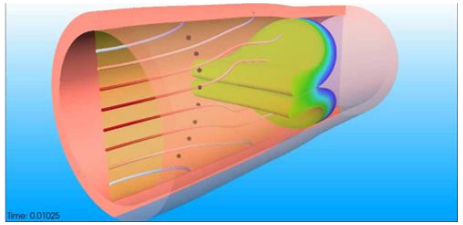
- 1st German Workshop on Next-Generation Software Development in Computational Mechanics
  - held in Raitenhaslach (Feb 1-3, 2023)
  - 37 participants
  - Discussing sustainable coding
  - Keynote lecture by Prof. M. Kronbichler on "Performance optimized programming“
  - Team building

- Preparation for 2nd Workshop
  - September 25-27, 2024 in Herrsching am Ammersee
  - Planned with 49 participants

# Summary







- Legacy code challenge
- We found refactoring of such a big code project with our given resources to be more feasible

- Side-projects have been launched in the meantime to specialize in the HPC direction
- Challenge to hold connection between main project and side projects

- Open source publication of 4C in 2024 https://baci.pages.gitlab.lrz.de/website/index.html (will be forwarded to new domain under 4C)