

deal.II: challenges and opportunities of developing an open-source FEM library

Swedish e-Science Academy 2025

Satellite: Common challenges in large-scale code development

Peter Munch[†]

[†]Institute of Mathematics, Technical University of Berlin, Germany

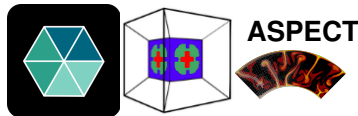
16. October, 2025

Overview



- ▷ FEM library; C++
- ▷ origin in Heidelberg 1998; 3 developers
- ▷ 13 principal developers; 418+ contributors
 - ▷ approx. 2,700 publications
- ▷ free (transition to Apache-2.0 with LLVM-exception)
 - ▷ extensive docu & approx. 90 tutorials

open-source application codes



Lethes

+DFT-FE, OpenFCST, life^x, ...

About myself: postdoctoral researcher at TU Berlin (2024–, before: Uppsala University), deal.II principal developer (2020–)

Questions to be answered in this presentation:

- ▶ **Who** are the library developing? **How** do they work? Which **resources** are used?
- ▶ Who are the users? How do the developers **interact** with them?
- ▶ What are **challenges/opportunities** of developing an open-source library (**in academia**)?

Part 1:

deal.II: introduction

Introduction

- ▶ deal.II¹: mathematical software for finite-element analysis, written in C++
- ▶ origin in Heidelberg 1998: Wolfgang Bangerth, Ralf Hartmann, Guido Kanschat
- ▶ 418+ contributors + principal developer team with 13 active members
- ▶ more than 2,700 publications (on and with deal.II)
- ▶ freely available under Apache-2.0 with LLVM-exception or LGPL-2.1-or-later
- ▶ yearly releases; current release: 9.7
- ▶ **features comprise (supported by third-party libraries)**: matrix-free implementations, parallelization (MPI, threading via [TBB & Taskflow](#), SIMD, GPU support), discontinuous Galerkin methods, AMR via [p4est](#), particles, wrappers for [PETSc](#) and [Trilinos](#), ...
- ▶ similar libraries: e.g., MFEM, DUNE, FEniCS, libMesh, ...



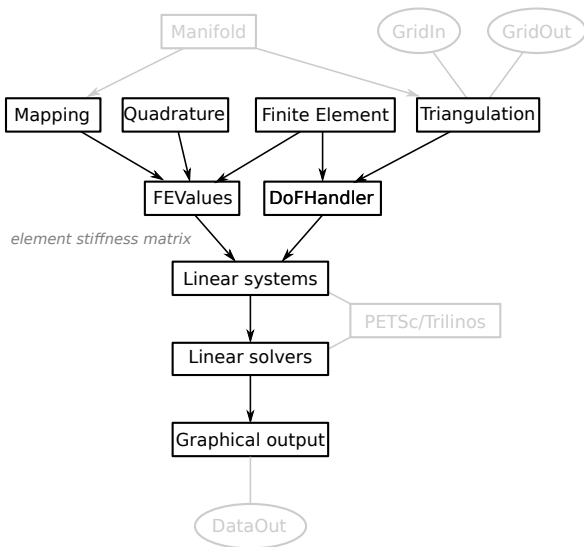
¹ successor of DEAL: **D**ifferential **E**quations **A**nalysis **L**ibrary

Main modules

needed from a FEM library:

- ▶ mesh handling
- ▶ finite elements
- ▶ quadrature rules
- ▶ mapping rules
- ▶ assembly procedure
- ▶ linear solver

deal.II main modules →



Main modules (cont.)

```
const unsigned int dim = 2, degree = 3;

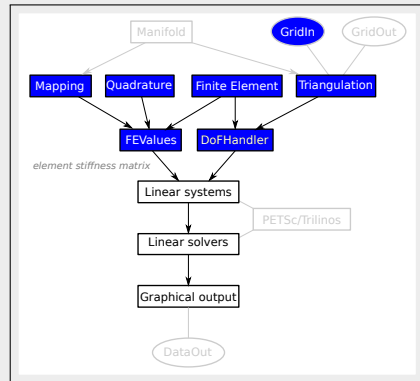
parallel::distributed::Triangulation<dim> tria(MPI_COMM_WORLD);
util::create_reentrant_corner(tria);

FE_Q<dim>          fe(degree);
QGauss<dim>        quad(degree + 1);
MappingQGeneric<dim> mapping(1);

DoFHandler<dim> dof_handler(tria);
dof_handler.distribute_dofs(fe);

// deal with boundary conditions
AffineConstraints<double> constraints;
VectorTools::interpolate_boundary_values(mapping, dof_handler, 0,
    Functions::ZeroFunction<dim>(), constraints);
constraints.close();

// initialize vectors and system matrix
LinearAlgebra::distributed::Vector<double> x, b;
TrilinosWrappers::SparseMatrix A;
util::initialize_dof_vector(dof_handler, x); util::initialize_dof_vector(dof_handler, b);
util::initialize_system_matrix(dof_handler, constraints, A);
```



```
// assemble right-hand side and system matrix
FullMatrix<double>          cell_matrix;
Vector<double>              cell_rhs;
std::vector<types::global_dof_index> local_dof_indices;
FEValues<dim> fe_values(mapping, fe, quad, update_values | update_gradients | update_JxW_values);
```

$$\sum_q (\nabla N_{iq}, \nabla N_{jq}) \cdot |J_q| \times w_q, \quad \sum_q (N_{iq}, f) \cdot |J_q| \times w_q$$

Main modules (cont.)

```
for (const auto &cell : dof_handler.active_cell_iterators()) // loop over all locally-owned cells
{
    if (cell->is_locally_owned() == false) continue;
    fe_values.reinit(cell);

    const auto dofs_per_cell = cell->get_fe().n_dofs_per_cell(); // allocate memory for element matrix/vector
    cell_matrix.reinit(dofs_per_cell, dofs_per_cell);           //
    cell_rhs.reinit(dofs_per_cell);                             //

    for (const auto q : fe_values.quadrature_point_indices()) // compute element matrix/vector
        for (const auto i : fe_values.dof_indices())           //
            for (const auto j : fe_values.dof_indices())       //
                cell_matrix(i, j) += (fe_values.shape_grad(i, q) * //  $\sum_q (\nabla N_{iq}, \nabla N_{jq}) \cdot |J_q| \times w_q \rightarrow \mathbf{K}_{ij}^{(e)}$ 
                                     fe_values.shape_grad(j, q) * //
                                     fe_values.JxW(q));           //

    for (const auto q : fe_values.quadrature_point_indices()) //
        for (const auto i : fe_values.dof_indices())           //
            cell_rhs(i) += (fe_values.shape_value(i, q) * //  $\sum_q (N_{iq}, f) \cdot |J_q| \times w_q \rightarrow \mathbf{f}_i^{(e)}$ 
                            1. * //
                            fe_values.JxW(q));           //

    local_dof_indices.resize(cell->get_fe().dofs_per_cell); // assembly
    cell->get_dof_indices(local_dof_indices); //
    constraints.distribute_local_to_global(cell_matrix, cell_rhs, local_dof_indices, A, b); //  $\mathbf{A}_e \mathbf{K}_e^{(e)}, \mathbf{A}_e \mathbf{f}_e^{(e)}$ 
} //
b.compress(VectorOperation::add); A.compress(VectorOperation::add); //
```

Main modules (cont.)

```
// solve linear equation system
ReductionControl reduction_control;
SolverCG<LinearAlgebra::distributed::Vector<double>> solver(reduction_control);
solver.solve(A, x, b, PreconditionIdentity());

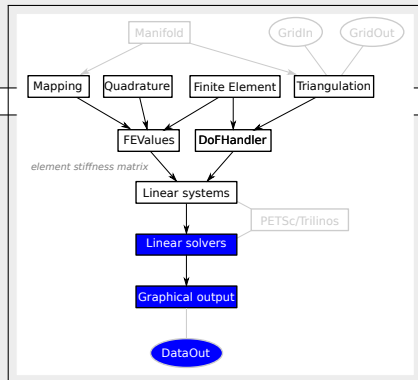
if (Utilities::MPI::this_mpi_process(util::get_mpi_comm(tria)) == 0)
    printf("Solved in %d iterations.\n", reduction_control.last_step());

constraints.distribute(x);
```

```
// output results (e.g. VTK, VTU, Tecplot, HDF5, svg, gnuplot, ...)
DataOutBase::VtkFlags flags;
flags.write_higher_order_cells = true;

DataOut<dim> data_out;
data_out.set_flags(flags);
data_out.attach_dof_handler(dof_handler);
x.update_ghost_values();
data_out.add_data_vector(dof_handler, x, "solution");
data_out.build_patches(mapping, degree + 1);
data_out.write_vtu_with_pvtu_record("./", "result", 0, MPI_COMM_WORLD);
```

$$\mathbf{K}\mathbf{x} = \mathbf{f} \rightarrow \mathbf{x} = \mathbf{K}^{-1}\mathbf{f}$$

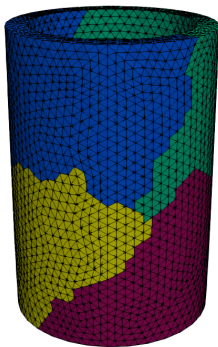


Full code: <https://github.com/peterrum/dealii-examples/blob/master/poisson.cc>

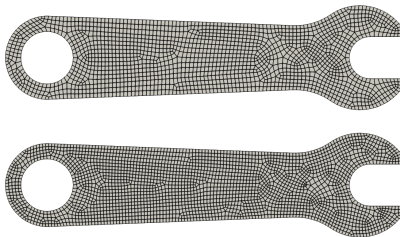
Active development

- ▶ yearly release (current release: 9.7)
- ▶ major additions in the last years, e.g., particles,

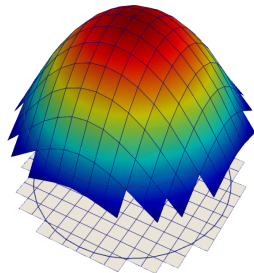
simplex meshes



non-nested multigrid



CutFEM



... and 24 new tutorials over the last 5 years

- ▶ each release is accompanied by a [release paper](#) (mostly in Journal of Numerical Mathematics)

Successful?

Winner of “SIAM/ACM Prize in Computational Science and Engineering 2025”



Successful?

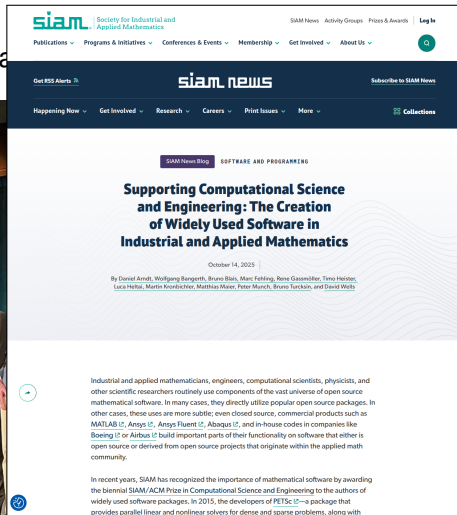
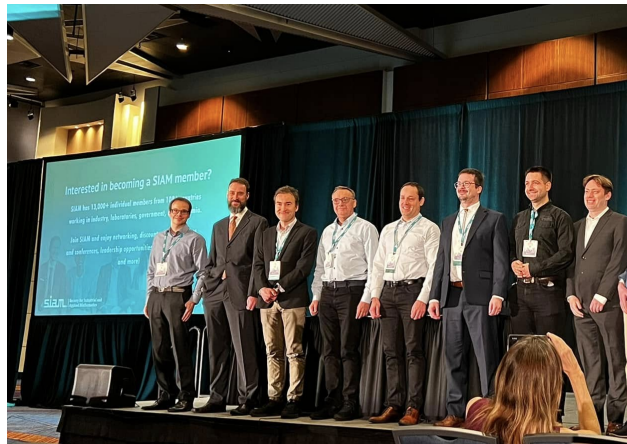
Winner of “SIAM/ACM Prize in Computational Science and Engineering 2025”



Wolfgang Bangerth's acceptance speech [\[1\]](#)

Successful?

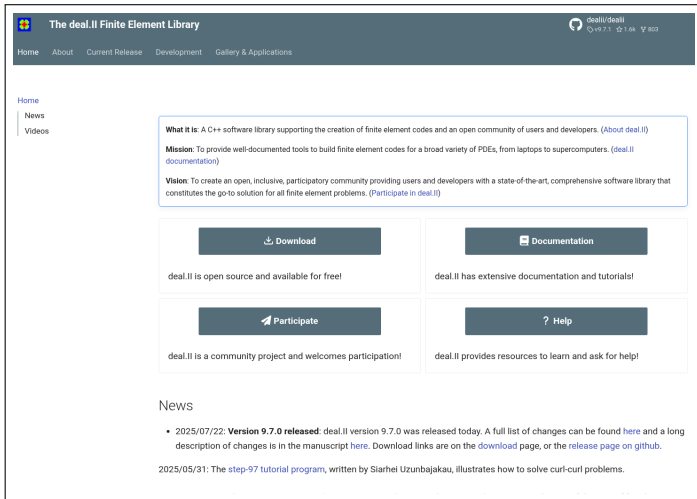
Winner of “SIAM/ACM Prize in Computational Science and Engineering”



Wolfgang Bangerth's acceptance speech [1], SIAM News Blog [2].

Part 2:

Distributed development: resources



The screenshot shows the official webpage of the deal.II Finite Element Library. The header is dark blue with the deal.II logo and name on the left, and version information (v9.7.1, 1.6k, 803) on the right. Below the header is a navigation bar with links: Home, About, Current Release, Development, and Gallery & Applications. On the left side, there is a sidebar with links: Home, News, and Videos. The main content area features a blue-bordered box with the following text:

What it is: A C++ software library supporting the creation of finite element codes and an open community of users and developers. ([About deal.II](#))

Mission: To provide well-documented tools to build finite element codes for a broad variety of PDEs, from laptops to supercomputers. ([deal.II documentation](#))

Vision: To create an open, inclusive, participatory community providing users and developers with a state-of-the-art, comprehensive software library that constitutes the go-to solution for all finite element problems. ([Participate in deal.II](#))

Below this box are four buttons arranged in a 2x2 grid:

- Download**: deal.II is open source and available for free!
- Documentation**: deal.II has extensive documentation and tutorials!
- Participate**: deal.II is a community project and welcomes participation!
- ? Help**: deal.II provides resources to learn and ask for help!

Below the buttons is a "News" section with the following content:

- 2025/07/22: **Version 9.7.0 released:** deal.II version 9.7.0 was released today. A full list of changes can be found [here](#) and a long description of changes is in the manuscript [here](#). Download links are on the [download](#) page, or the [release page on github](#).
- 2025/05/31: The [step-97 tutorial program](#), written by Siarhei Uzunbajakau, illustrates how to solve curl-curl problems.

deal.II Reference documentation for deal.II version Git 899b683bb8 2020-11-07 12:42:48 -0500

Main Page Tutorial Code gallery Modules Namespaces Classes Related Pages Files deali.org Search

Public Types | Public Member Functions | Static Public Member Functions | Public Attributes | Static Public Attributes | Private Member Functions | Friends | Related Functions | List of all members

VectorizedArray< Number, width > Class Template Reference

#include <deal.II/base/vectorization.h>

Inheritance diagram for VectorizedArray< Number, width >:

```
graph TD
    VAB1["VectorizedArrayBase< T, width >"]
    VAB2["VectorizedArrayBase< VectorizedArray< Number, width >, 1 >"]
    VA["VectorizedArray< Number, width >"]
    VAD["VectorizedArray< double >"]
    VAN["VectorizedArray< Number >"]

    VAB1 -.-> VAB2
    VAB2 -.-> VA
    VA -.-> VAD
    VA -.-> VAN
```

Public Types

using value_type = Number

Public Member Functions

VectorizedArray ()=default
VectorizedArray (const Number scalar)

Extensive Doxygen documentation

dealii / dealii

Unwatch 82 Star 678 Fork 478

< Code Issues 452 Pull requests 55 Actions Projects 19 Wiki Security Insights

Frequently Asked Questions

Jens edited this page on Sep 7 · 52 revisions

The deal.II FAQ

This page collects a few answers to questions that have frequently been asked about deal.II and that we thought are worth recording as they may be useful to others as well.

Table of Contents

- The deal.II FAQ
 - Table of Contents
 - General questions on deal.II
 - Can I use/implement triangles/tetrahedra in deal.II?
 - I'm stuck!
 - I'm not sure the mailing list is the right place to ask ...
 - How fast is deal.II?
 - deal.II programs behave differently in 1d than in 2/3d
 - I want to use deal.II for work in my company. Do I need a special license?
 - Supported System Architectures
 - Can I use deal.II on a Windows platform?
 - Run deal.II in the Windows Subsystem for Linux
 - Run deal.II natively on Windows
 - Run deal.II through a virtual box

Pages 69

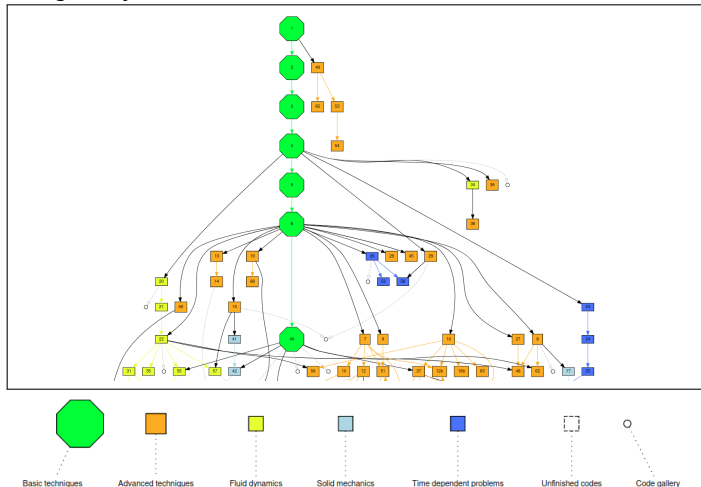
Find a Page...

- Home
- Code DOI best practices
- Contributing
- deal.II in Spack
- deal.II on Homebrew Linuxbrew
- Debugging with GDB
- Docker Images
- Dof-Handler
- Eclipse
- Electromagnetic problem
- Emacs
- Frequently Asked Questions
- Function Plotting Tool
- Gallery

GitHub Wiki

Documentation (cont.)

87 tutorials and code gallery:



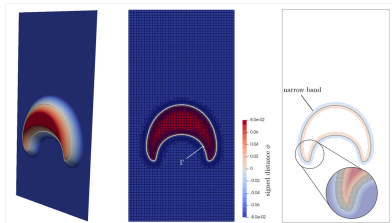
... further 10 tutorials: work in progress

Documentation (cont.)

Example (step-87):

Motivation: two-phase flow

The minimal code examples (short "mini examples") presented in this tutorial are motivated by the application of two-phase flow simulations formulated in a one fluid setting using a Eulerian framework. In diffuse interface methods, the two phases may be implicitly described by a level-set function, here chosen as a signed distance function $\phi(x)$ in Ω and illustrated for a popular benchmark case of a rising bubble in the following figure.



The discrete interface Γ is represented implicitly through a certain isosurface of the level-set function e.g. for the signed distance function $\Gamma = \{x \in \Omega \mid \phi(x) = 0\}$. We would like to note that deal.II provides a set of analytical signed distance functions for simple geometries in the **Functions:SignedDistance** namespace. Those can be combined via Boolean operations to describe more complex geometries [55]. The temporal evolution of the level-set field is obtained by the transport equation

$$\frac{\partial \phi}{\partial t} + \mathbf{u}|_{\Gamma} \cdot \nabla \phi = 0$$

with the transport velocity at the interface $\mathbf{u}|_{\Gamma}$, which might be approximated by the local fluid velocity $\mathbf{u}|_{\Gamma} \approx \mathbf{u}(x)$. To obtain the signed-distance property of the level-set field throughout the numerical solution procedure, PDE-based or, alternatively, also geometric reinitialization methods are used. For the latter, an algorithm for computing the distance from the support points to the discrete interface, e.g., via closest-point projection [124], is needed. This will be part of one of the mini examples, where we describe how to obtain the closest point x^* to the interface Γ for an arbitrary point x . For the simplest case, the former can be computed from the following equation

$$x^* = x - n(x)\phi(x),$$

assuming that the interface normal vector $n(x)$ and $\phi(x)$ represent exact quantities. Typically, this projection is only performed for points located within a narrow band region around the interface, indicated in the right panel of the figure above.

Alternatively to the implicit representation of the interface, in sharp interface methods, e.g., via front tracking, the interface Γ is explicitly represented by a surface mesh. The latter is immersed into a background mesh, from which the local velocity at the support points of the surface mesh is extracted and leads to a movement of the support points of the immersed mesh as

$$x_i^{(j+1)} = x_i^{(j)} + \Delta t \cdot \mathbf{u}(x_i^{(j)}) \quad \text{for } x_i \in \Gamma$$

which considers an explicit Euler time integration scheme from time step j to $j+1$ with time step-size Δt .

For a two-phase flow model considering the incompressible Navier-Stokes equations, the two phases are usually coupled by a singular surface tension force F_{Γ} , which results, together with the difference in fluid properties, in discontinuities across the interface:

$$F_{\Gamma}(x) = \sigma \kappa(n(x))\delta_{\Gamma}(x).$$

The commented program

Include files

The program starts with including all the relevant header files.

```
#include <deal.II/base/conditional_ostream.h>
#include <deal.II/base/function_lib.h>
#include <deal.II/base/function_signed_distance.h>
#include <deal.II/base/mpl.h>
#include <deal.II/base/mpl_templates.h>

#include <deal.II/distributed/tria.h>

#include <deal.II/dofs/dof_renumbering.h>
#include <deal.II/dofs/dof_tools.h>

#include <deal.II/fe/fe_nothing.h>
#include <deal.II/fe/fe_q.h>
#include <deal.II/fe/fe_system.h>
#include <deal.II/fe/mapping_qt.h>
#include <deal.II/fe/mapping_q_cache.h>

#include <deal.II/grid/grid_generator.h>
#include <deal.II/grid/grid_tools.h>

#include <deal.II/lac/generic_linear_algebra.h>

#include <deal.II/numerics/data_out.h>

#include <iostream>
#include <fstream>
```

The files most relevant for this tutorial are the ones that contain **Utilities:MPI:RemotePointEvaluation** and the distributed evaluation functions in the **VectorTools** namespace, which use **Utilities:MPI:RemotePointEvaluation**.

```
#include <deal.II/base/mpl_remote_point_evaluation.h>
#include <deal.II/numerics/vector_tools.h>
```

The following header file provides the class **FEPointEvaluation**, which allows us to evaluate values of a local solution vector at arbitrary unit points of a cell.

```
#include <deal.II/matrix_free/fe_point_evaluation.h>
```

We pack everything that is specific for this program into a namespace of its own.

```
namespace Step87
{
    using namespace dealii;
```

Utility functions (declaration)

In the following, we declare utility functions that are used in the mini examples below. You find the definitions at the end of the tutorial.

The minimum requirement of this tutorial is MPI. If deal.II is built with p4est, we use **parallel-distributed:Triangulation** as distributed mesh. The class **parallel-shared:Triangulation** is used if deal.II is built without p4est or if the dimension of the triangulation is 1D, e.g., in the case of codim-1 meshes.

```
#ifdef DEAL_II_WITH_P4EST
    template <int dim, int spacedim = dim>
    using DistributedTriangulation = typename std::conditional_t<
```

deal.II user group:

The screenshot shows the deal.II User Group forum interface. The sidebar on the left includes a 'New conversation' button and navigation links for 'My groups', 'Recent groups', 'Favorite groups', 'Starred conversations', and 'Labels'. The main content area features a search bar, a group header for 'deal.II User Group' with 1231 members, and a welcome message. Below the welcome message is a list of recent discussions.

Avatar	Author	Topic	Time	Star
	wjs6... , ... brun...	Use Gmsh to import external grid calculation elasticity problem	12:16	☆
	Konrad Simon	boundary_conditions fe_spaces Q&A Boundary conc	6 Nov	☆
	Jam... , ... Jean...	Adaptive Time Increment Algorithm — I apologize that it took so	6 Nov	☆
	acdai... , bruno...	Post-processing velocities with Trilinos vectors during simulatio	5 Nov	☆
	animes... , brun...	Some changes in arpack.h (Functionality to compute only eigen	5 Nov	☆
	zachary..., Wel...	Running new tests with argument — Hi Zachary, Are you adding ε	5 Nov	☆
	animes... , Wolf...	Boundary DOFs using DOF Handler — Thank you, Professor Bang	4 Nov	☆

... Q&A by users and developers!

Development on GitHub

The screenshot shows the GitHub repository page for `dealii / dealii`. The repository has 82 Unwatch buttons, 678 Stars, and 478 Forks. The main navigation bar includes links for Code, Issues (452), Pull requests (55), Actions, Projects (19), Wiki, Security, and Insights. The repository is currently on the `master` branch, with 10 branches and 35 tags. A table of recent commits is displayed, showing the commit message, author, time ago, and number of commits. The right sidebar contains the 'About' section, which describes the repository as the development repository for the deal.II finite element library, and provides links to the website (www.dealii.org/), README, and license. Below this, the 'Releases' section shows the latest release, `deal.II version 9.2.0`, published on May 20. The 'Packages' section indicates that no packages have been published. Finally, the 'Contributors' section shows 196 contributors.

Commit	Message	Author	Time ago	Commits
drwells	Merge pull request #11109 from tjei/mpi_comm_ref	899b683	1 hour ago	50,406
	.github/workflows	Make indent CI check abort on Doxygen warnings	last month	
	bundled	work around warning in bundled boost	2 months ago	
	cmake	Add a quick check for matching boost versions.	last month	
	contrib	Also update the _gdbinit.py file.	last month	
	doc	Expose mesh loop bug for anisotropic grids.	4 days ago	
	examples	step-7: mark some ints as doubles	3 days ago	
	include	Merge pull request #11109 from tjei/mpi_comm_ref	1 hour ago	
	source	Merge pull request #11109 from tjei/mpi_comm_ref	1 hour ago	
	tests	Merge pull request #11131 from tamiko/add_a_test	2 days ago	
	.clang-format	Properly format #include <deal.II/simplex*.h>	2 months ago	
	.clang-tidy	disable performance-no-automatic-move	6 months ago	
	.codecov.yml	Add codecov configuration	2 years ago	
	.gitattributes	update .gitattributes	10 months ago	
	.gitignore	.DS_Store etc. and .swp removed from untracked files	15 months ago	
	.mailmap	add a mailmap entry	6 months ago	

- issues
- pull requests
- GitHub actions → CI
- required: approval by ≥ 1 principal developer

Continuous integration

Login

All Dashboards

Saturday, November 07 2020 18:48:28



Dashboard

Calendar

Previous

Current

Project

3 hours ago: 13 tests failed on GNU-9.3.0-master-debian-11

4 hours ago: 12 tests failed on GNU-9.3.0-master-ubuntu-lts-20

7 hours ago: 28 tests failed on GNU-9.3.0-master

7 hours ago: 1 warning introduced on GNU-9.3.0-master

8 hours ago: 38 tests failed on GNU-10.1.0-master-tets

See full feed

Regression Tests

150 builds

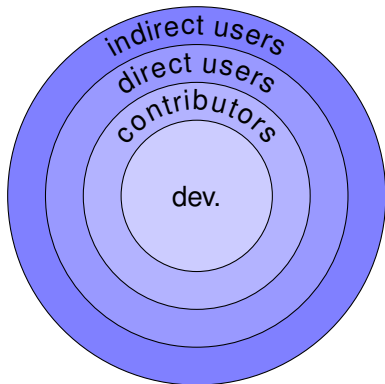
Site	Build Name	Update	Configure		Build		Test			Start Time
		Revision	Error	Warn	Error	Warn	Not Run	Fail	Pass	
tester	Clang-10.0.0-master-avx2-Ofast	10ceee	0	0	0	0	0	50 ⁺³	12417 ⁻³	20 hours ago
tester	Clang-10.0.0-master-avx2-Ofast	414559	0	0	0	0	0	49 ⁺¹	12404 ⁻¹	Nov 03, 2020 - 07:54 UTC
tester	Clang-10.0.0-master-avx2-Ofast	2bdb45	0	0	0	0	0	48	12417 ⁻²	Nov 06, 2020 - 01:00 UTC
tester	Clang-10.0.0-master-avx2-Ofast	cd2846	0	0	0	0	0	48	12415 ⁻³	Nov 05, 2020 - 03:17 UTC
tester	Clang-10.0.0-master-avx2-Ofast	22f0f5	0	0	0	0	0	48 ⁻¹	12415 ⁻¹¹	Nov 04, 2020 - 05:34 UTC
tester	GNU-10.1.0-master-tets	10ceee	0	0	0	0	0	38	11866 ⁻²	10 hours ago
tester	GNU-10.1.0-master-tets	2bdb45	0	0	0	0	0	38	11864 ⁻²	Nov 06, 2020 - 10:56 UTC

... more than 17,000 tests run for different compilers/hardware/configurations

Part 3:

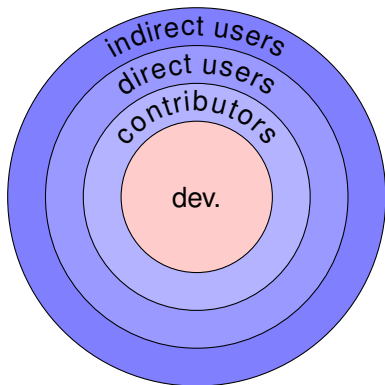
Distributed development: users and developers

User and developer base



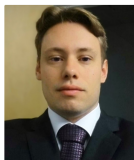
- ▶ deal.II has a large body of developers and users
- ▶ this includes Master students, PhD students, researchers (academia, research centers), industry
- ▶ background: Mathematics, Computer Science, Engineering, ...
- ▶ location: Europe, China, USA, Canada, India, ...

User and developer base: principal developers



- ▶ team of 13 principal developers
- ▶ know a significant fraction of the library
- ▶ review and merge PRs
- ▶ answer questions/issues
- ▶ responsible for infrastructure/testing
- ▶ organization of workshops
- ▶ not paid for work on deal.II
- ▶ discussion via GitHub/Element
- ▶ regular Zoom meetings

User and developer base: principal developers (cont.)



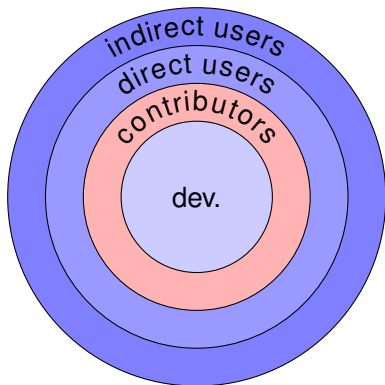
Developers emeriti: Denis Davydov (2015-2020), Ralf Hartmann (1998-2012), Guido Kanschat (1997-2022), Toby D. Young (2013-2017)

User and developer base: principal developers (cont.)

- ▶ Daniel Arndt, Oak Ridge National Laboratory, TN, USA (since 2016)
- ▶ Wolfgang Bangerth, Colorado State University, CO, USA (since 1997)
- ▶ Bruno Blais, Polytechnique Montréal, Canada (since 2023)
- ▶ Marc Fehling, Charles University, Czech Republic (since 2021)
- ▶ Rene Gassmoeller, GEOMAR Helmholtz Centre for Ocean Research, Kiel, Germany (since 2023)
- ▶ Timo Heister, Clemson University, SC, USA (since 2012)
- ▶ Luca Heltai, University of Pisa, Italy (since 2013)
- ▶ Martin Kronbichler, Ruhr University Bochum, Germany (since 2013)
- ▶ Matthias Maier, Texas A&M University, College Station, TX, USA (since 2013)
- ▶ [Peter Munch, Technical University of Berlin, Germany \(since 2020\)](#)
- ▶ Jean-Paul Pelteret, industry, Germany (since 2016)
- ▶ Bruno Turcksin, Oak Ridge National Laboratory, TN, USA (since 2013)
- ▶ David Wells University of North Carolina, Chapel Hill, NC, USA (since 2015)

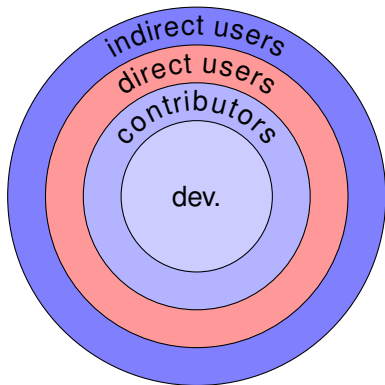
Summary: PhD students (0) vs. postdocs (7) vs. professors (6), Europe (6) vs. USA/Canada (7), male (13) vs. female (0), academia (9) vs. research centers (3) vs. industry (1), math (7) vs non-math (6)

User and developer base: contributors

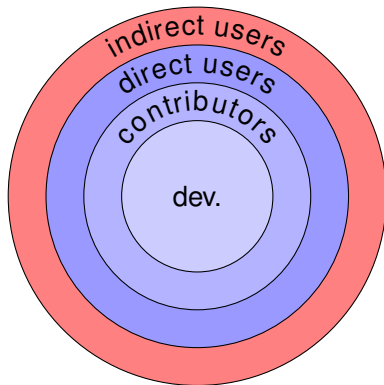


- ▶ use deal.II extensively
- ▶ read/write deal.II code
- ▶ contribute: bug fixes, fixes in documentation, features
- ▶ total: 418+
- ▶ last release: 88 contributors
- ▶ award: co-author of release paper

User and developer base: users



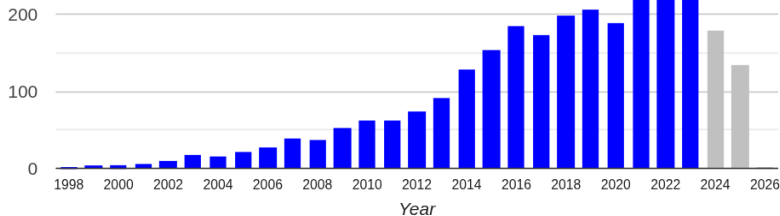
- ▶ write applications using deal.II
- ▶ use documentation
- ▶ potentially start from a tutorial
- ▶ interaction via google group



- ▶ use application/libraries using deal.II (e.g., ASPECT for geosciences)
- ▶ rare interaction
- ▶ citation

Publications

Number of publications (total: 2746) gives an indication of number of users:



... gray bars: incomplete data

Workshops

(Yearly) workshops:



Fort Collins, Colorado, USA, 2024

+ get-together at workshops

Workshops (cont.)

Best deal.II-based paper award:

A comprehensive and biophysically detailed computational model of the whole human heart electromechanics

Marco Fedele^{*,a}, Roberto Piersanti^a, Francesco Regazzoni^a, Matteo Salvador^a, Pasquale Claudio Africa^a, Michele Bucelli^a, Alberto Zingaro^a, Luca Dede^a, Alfio Quarteroni^{a,b}

^a MOX - Department of Mathematics, Politecnico di Milano, Piazza Leonardo da Vinci, 32, Milano, 20133, Italy

^b Mathematics Institute (Professor Emeritus), École Polytechnique Fédérale de Lausanne, Av. Piccard, Lausanne, CH-1015, Switzerland

Received 24 July 2022; received in revised form 27 February 2023; accepted 27 February 2023

Available online 21 March 2023

2023: Computer Methods in Applied Mechanics and Engineering

Collapse of microbubbles over an elastoplastic wall

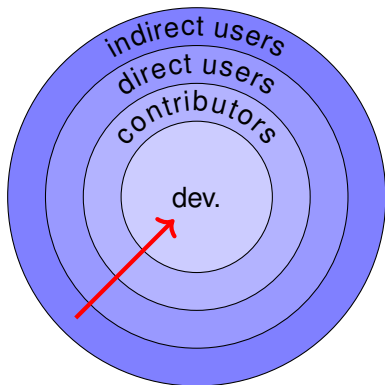
Dario Abbondanza¹, Mirko Gallo¹ and Carlo Massimo Casciola^{1,†}

¹Dipartimento di Ingegneria Meccanica e Aerospaziale, Sapienza Università di Roma, Via Eudossiana 18, 00184 Roma, Italy

(Received 15 February 2024; revised 5 September 2024; accepted 5 September 2024)

2024: Journal of Fluid Mechanics

Becoming more involved



Typical path:

- ▶ start to use deal.II (as student)
- ▶ contribute a lot
- ▶ become co-author of release paper
- ▶ contribute much more and be active in the community
- ▶ become principal developer

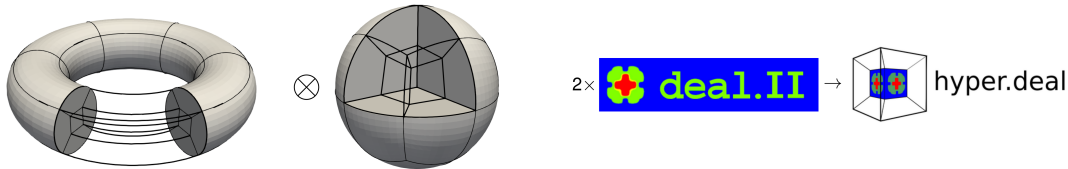
About myself

First project using deal.II (2019, student project): [Vlasov equation](#) (comp. plasma physics):

$$\frac{\partial f}{\partial t} + \mathbf{v} \cdot \nabla_{\mathbf{x}} f + \mathbf{a}(t, f, \mathbf{x}, \mathbf{v}) \cdot \nabla_{\mathbf{v}} f = 0 \quad \text{w.} \quad \mathbf{a}(t, f, \mathbf{x}, \mathbf{v}) = -\mathbf{E}(t, \mathbf{x}) \quad (\text{Poisson problem})$$

Derivative in phase ($\mathbf{v}$)-space implies [high-dimensional nonlinear advection equation](#).

Approach: [high-order discontinuous Galerkin method](#) up to 6D



Challenges:

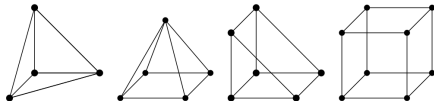
- ▶ mesh generation (open-source packages only support 1D-3D)
- ▶ performance: large working sets, communication pattern/data *curse of dimensionality* ⚡
- ▶ Munch, P., Kormann, K. and Kronbichler, M., 2021. hyper. deal: An efficient, matrix-free finite-element library for high-dimensional partial differential equations. ACM TOMS.

About myself (cont.)

Since 2020 (beginning of second year of PhD), one of the principal developers of deal.II

Projects in deal.II:

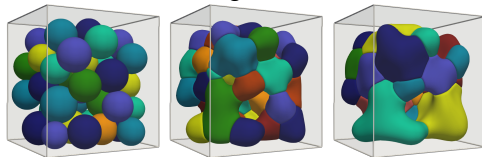
- ▶ simplex/mixed-mesh support



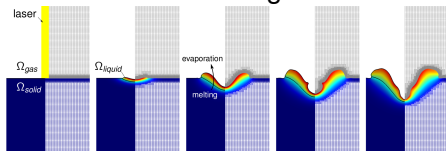
- ▶ (global-coarsening) multigrid
- ▶ hp-adaptivity, non-matching algorithms
- ▶ (fully) distributed triangulation (project 2)
- ▶ large-scale computations → consensus algorithm (project 1)
- ▶ performance optimization, e.g., matrix-free algorithms

Projects using deal.II:

- ▶ solid-state sintering



- ▶ additive manufacturing



- ▶ stage-parallel IRK and space-time FEM
- ▶ Galerkin difference methods
- ▶ direct solvers

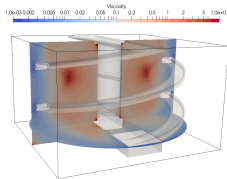
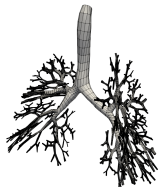
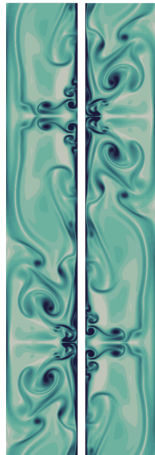
About myself (cont.)

Weak form of (stabilized) incompressible Navier–Stokes equations:

$$\begin{aligned}(\partial_t \mathbf{u}, \mathbf{v}) + (\mathbf{u} \cdot \nabla \mathbf{u}, \mathbf{v}) - (p, \nabla \cdot \mathbf{v}) + (\nu \varepsilon(\mathbf{u}), \varepsilon(\mathbf{v})) \\ + \delta_1 (\partial_t \mathbf{u} + \mathbf{u} \cdot \nabla \mathbf{u} + \nabla p - \nu \Delta \mathbf{u}, \mathbf{u} \cdot \nabla \mathbf{v}) + \delta_2 (\nabla \cdot \mathbf{u}, \nabla \cdot \mathbf{v}) = 0, \\ (\nabla \cdot \mathbf{u}, q) + \delta_1 (\partial_t \mathbf{u} + \mathbf{u} \cdot \nabla \mathbf{u} + \nabla p - \nu \Delta \mathbf{u}, \nabla q) = 0\end{aligned}$$

Solve with BDF2/ST and Newton–Krylov method with Jacobian:

$$J = \begin{bmatrix} A & B^T \\ C & D \end{bmatrix} \rightarrow J' = \begin{bmatrix} A & B^T \\ 0 & S \end{bmatrix} \rightarrow \text{(monolithic) multigrid}$$



Interests:

- preconditioning
- multiphysics
- performance
- hp-adaptivity
- open-source dev.: deal.II, Lethé

Compressible NS (+reaction, cut)

$$\mathbf{U}_t + \nabla \cdot (F_a(\mathbf{U}) + F_v(\mathbf{U})) = G(\mathbf{U})$$

Part 4:

Challenges

① Backwards compatibility

- ▶ during development, mistakes are made
- ▶ correcting them if they are user-visible is not that easy

Example:

- ▶ old interface:

```
template <int dim, int spacedim>
void
extract_locally_active_dofs(const DoFHandler<dim, spacedim> &dof_handler,
                           IndexSet                        &dof_set);
```

- ▶ new interface:

```
template <int dim, int spacedim>
IndexSet
extract_locally_active_dofs(const DoFHandler<dim, spacedim> &dof_handler);
```


① Backwards compatibility (cont.)

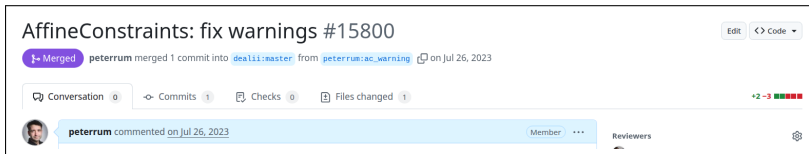
Solution: deprecation warnings

```
template <int dim, int spacedim>
DEAL_II_DEPRECATED void
extract_locally_active_dofs(const DoFHandler<dim, spacedim> &dof_handler,
                           IndexSet &dof_set);
```

- ▶ preprocessing macro
- ▶ deprecation takes 2 years: *early deprecation* → *deprecation*
- ▶ macros can be disabled
- ▶ user experience: “New warning/incompatibility after each update of deal.II!”

② Keeping up to date: new compilers

- ▶ support newest versions of GCC, ICC, Clang
- ▶ keep supporting old versions since not everyone has access to the newest versions
- ▶ tested versions
 - ▶ GCC: 9.4.0 (Ubuntu 20.04 LTS), 10.2.1, 10.4.1, 11.3.1, 11.4.0, 12.2.0, 12.2.1, 13.1.1, 13.3.0, 14.2.0, 14.2.1, 15.1.0, ...
 - ▶ Clang: 13.0.1, 14.0.6, 15.0.6, 16.0.6, 17.0.6, 18.1.8, 19.1.6, 19.1.7, 20.1.7, 21.1.7, ...
- ▶ goal: no warnings!



② Keeping up to date: new versions of C++

- ▶ we want to use/support newest versions of the C++ standard
- ▶ however, many used compilers do not support current features; requirement: C++17

Solutions:

- ▶ check C++ version during configuration
- ▶ provide own implementations

```
#ifndef DEAL_II_HAVE_CXX20
    template <typename T, typename B>
    using iota_view = boost::integer_range<T>;
#else
    using std::ranges::iota_view;
#endif
```

- ▶ modules (C++20) supported!

```
template <int dim, int spacedim>
DEAL_II_CXX20_REQUIRES((concepts::is_valid_dim_spacedim<dim, spacedim>))
class Triangulation;
```

Experience converting a large mathematical software package written in C++ to C++20 modules

WOLFGANG BANGERTH, Colorado State University, USA

Mathematical software has traditionally been built in the form of “packages” that build on each other. A substantial fraction of these packages is written in C++ and, as a consequence, the interface of a package is described in the form of header files that downstream packages and applications can then “#include”. C++ has inherited this approach towards exporting interfaces from C, but the approach is chunky, unreliable, and slow. As a consequence, C++20 has introduced a “module” system in which packages explicitly export declarations and code that compilers then store in machine-readable form and that downstream users can “import” – a system in line with what many other programming languages have used for decades.

Herein, I explore how one can convert large mathematical software packages written in C++ to this system, using the deal.II library with its around 800,000 lines of code as an example. I describe an approach that allows providing both header-based and module-based interfaces from the same code base, discuss the challenges one encounters, and how modules actually work in practice in a variety of technical and human metrics. The results show that with a non-trivial, but also not prohibitive effort, the conversion to modules is possible, resulting in a reduction in compile time for the converted library itself; on the other hand, for downstream projects, compile times show no clear trend. I end with thoughts about long-term strategies for converting the entire ecosystem of mathematical software over the coming years or decades.

CCS Concepts: • Applied computing → Mathematics and statistics; • Software and its engineering → Software libraries and repositories; Modules / packages.

Additional Key Words and Phrases: Mathematical software, C++, C++20 modules

ACM Reference Format:

② Keeping up to date: external libraries

The interfaces of external libraries change over time, and features are added/removed.

Solution: check version during configuration and use preprocessing macros.

Example:

```
# DEAL_II_WITH_PETSC set up with external dependencies
#   PETSC_VERSION = 3.18.6.
#   PETSC_DIR = /homes/numerik/muench/store/sw-candi/petsc-3.18.6
```

```
#if DEAL_II_PETSC_VERSION_LT(3, 8, 0)
  ierr = MatTranspose(tmp, MAT_REUSE_MATRIX, &tmp);
#else
  ierr = MatTranspose(tmp, MAT_INPLACE_MATRIX, &tmp);
#endif
```

② Keeping up to date: new hardware architecture

To support different CPU architectures, large part of the library does not use double/float directly but wraps them in structs:

double	float	ISA
VectorizedArray<double, 1>	VectorizedArray<float, 1>	(auto-vectorization)
VectorizedArray<double, 2>	VectorizedArray<float, 4>	SSE2/Altivec/ARM NEON
VectorizedArray<double, 4>	VectorizedArray<float, 8>	AVX/AVX2
VectorizedArray<double, 8>	VectorizedArray<float, 16>	AVX-512

Use cases: vectorization over cells and quadrature points

With C++23, we will be able to replace (hopefully) our own implementation!

VectorizedArray (deal.II)	std::simd (C++23)
VectorizedArray<Number>	std::experimental::native_simd<Number>
VectorizedArray<Number, size>	std::experimental::fixed_size_simd<Number, size>

Support of mixed precision, e.g., double-float in geometric multigrid.

③ Development by PhD students

Many features of deal.II are developed by PhD students/early-career researchers:

- ▶ distributed meshes → Timo Heister (2011, Göttingen)
- ▶ matrix-free operator evaluation → Martin Kronbichler (2012, Uppsala)
- ▶ matrix-free operator evaluation on the GPU → Karl Ljungkvist (2014, Uppsala)
- ▶ distributed hp-adaptive computation → Marc Fehling (2019, Jülich)
- ▶ simplex/mixed-mesh support → Peter Munch (2021, Geesthacht)

Observations:

- ▶ projects are continued as postdoctoral researchers/professors
- ▶ projects are continued by new PhD students
- ▶ maintenance by others
- ▶ but ...

③ Development by PhD students (cont.)

Karl Ljungkvist (Uppsala University) has implemented matrix-free operator evaluation on the GPU during his PhD. At that time, state-of-the-art implementation.

- ▶ **Ljungkvist, K.**, 2014. Matrix-free finite-element operator application on graphics processing units. In European Conference on Parallel Processing (pp. 450-461). Cham: Springer International Publishing. **16 citations.**
- ▶ **Ljungkvist, K.**, 2017. Matrix-free finite-element computations on graphics processors with adaptively refined unstructured meshes. In Proceedings of the 25th High Performance Computing Symposium (pp. 1-12). **38 citations.**
- ▶ **Ljungkvist, K.** and Kronbichler, M., 2017. Multigrid for matrix-free finite element computations on graphics processors. **11 citations.**
- ▶ Kronbichler, M. and **Ljungkvist, K.**, 2019. Multigrid for matrix-free high-order finite element computations on graphics processors. ACM TOPC. **97 citations.**
- ▶ Munch, P., **Ljungkvist, K.** and Kronbichler, M., 2022. Efficient application of hanging-node constraints for matrix-free high-order FEM computations on CPU and GPU. ISC. **7 citations.**

③ Development by PhD students (cont.)

After the PhD was concluded, the development of GPU features in deal.II stalled. Others group have not rested, e.g., NEKO, libCEED, MFEM, NekRK, GALÆXI, ...

Developments:

- ▶ port Karl Ljungkvist's code to master
- ▶ new tutorial: step-64
- ▶ replacing CUDA backend by Kokkos
- ▶ overintegration and vectorial elements
- ▶ example codes deal.II + libCEED (BP1–BP6 supported)²

²<https://github.com/CEED/libCEED/tree/main/examples/deal.II>

④ Feature requests

Most requested features of deal.II:

- ▶ simplex/mixed-mesh support
- ▶ Python wrappers

④ Feature requests (cont.)

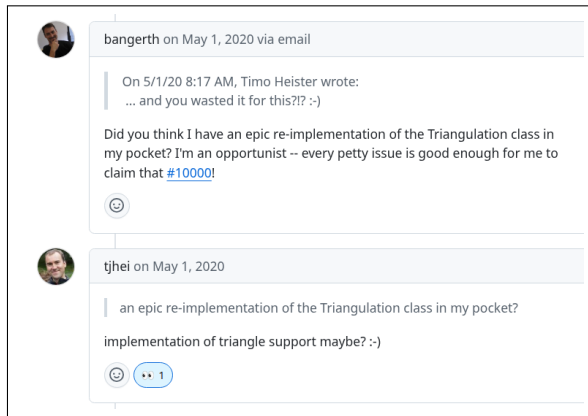
Most requested features of deal.II:

- ▶ simplex/mixed-mesh support ✓
- ▶ Python wrappers

Addition simplex/mixed mesh support:

- ▶ running gag for a long time

- ▶ available since 2021
- ▶ added by PM, financially backed by Helmholtz-Zentrum Hereon, Germany
- ▶ hesitation: long-time maintenance & incomplete feature



④ Feature requests (cont.)

Most requested features of deal.II:

- ▶ simplex/mixed-mesh support
- ▶ Python wrappers ✗

Python wrappers:

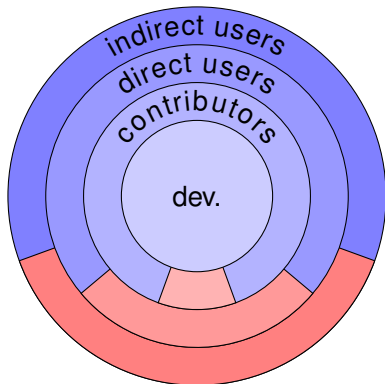
- ▶ basic implementation of Python bindings for geometry generation/manipulation

```
import PyDealII.Release as dealii
triangulation = dealii.Triangulation(dim = '2D')
triangulation.read(filename = 'example.msh', format = 'msh')
```

- ▶ no bindings for the rest of the library; challenge template arguments?
- ▶ observation: C++ difficult for starting when you are used to Python/MATLAB
- ▶ in contrast to: FEniCS, Firedrake, Dune, MFEM, libCEED, ...
- ▶ no interest/no financial support/no (scientific) motivation since no new innovation

5 Diversity

Visualization of **men**/**women** distribution:



Observation:

- ▶ the developer team of deal.II is highly uniform: 13 men!
- ▶ however, large fraction of users are women!

Challenge/question: how can we improve?

⑥ Artificial intelligence

- ▶ LLMs are trained on open-source code projects
- ▶ i.e., can be used to assist in writing code
- ▶ an early example (ChatGPT $\leftrightarrow$ deal.II) is given by G. Orlando in 2023



MOX-Report No. 31/2023

Assessing ChatGPT for coding finite element methods

Orlando, G.

MOX, Dipartimento di Matematica
Politecnico di Milano, Via Bonardi 9 - 20133 Milano (Italy)

mox-dmat@polimi.it

<https://mox.polimi.it>

⑦ (Under)Appreciation

Is software development (under)appreciated ...

- ▶ by the community? which community?
- ▶ by funding agencies?
- ▶ by journal publishers?
- ▶ by hiring committees?

How much time/effort can we spent on developing software?

Part 5:

Opportunities

For users

- ▶ rely on maintained, optimized, well-documented code
- ▶ one can concentrate on scientific questions without having to reimplement FEM
- ▶ benchmark codes can be published to allow reproducibility

For contributors/developers

- ▶ good for the CV (contribution to a big open-source project, visible)
- ▶ meet and interact with interesting people
- ▶ learn *research software engineering* as a side project
- ▶ developed features are tested (+ reviewed and as such have high quality), further developed (also by others), maintained, used (also in contexts where the original developer did not imagine)
- ▶ fun

Part 6:

Conclusions

Conclusions

- ▶ [deal.II](#) is an open-source finite-element library written in C++
- ▶ well-documented and public development on GitHub
- ▶ developer team and active user community
- ▶ opportunities overweigh challenges (e.g., maintenance burden)

