

# The general-purpose finite-element library deal.II

## Overview and basic concepts

@peterrum - Peter Munch<sup>123</sup>

<sup>1</sup>deal.II developer

<sup>2</sup>Institute for Computational Mechanics, Technical University of Munich, Germany

<sup>3</sup>Institute of Materials Research, Materials Mechanics, Helmholtz-Zentrum Geesthacht, Germany

November 18, 2020

1. Overview
2. Poisson problem + introduction into the [main modules](#)
3. [Advanced topics](#): elasticity, volume/surface coupling, adaptivity, simplex support

*... feel free to ask questions at any time you want!*

Part 1:

# Overview

- ▶ deal.II<sup>1</sup>: mathematical software for finite-element analysis, written in C++
- ▶ origin in Heidelberg 1998: Wolfgang Bangerth, Ralf Hartmann, Guido Kanschat
- ▶ 275 contributors + principal developer team with 11 active members
- ▶ more than 1,600 publications (on and with deal.II)
- ▶ freely available under LGPL 2.1 license
- ▶ yearly releases; current release: 9.2
- ▶ features comprise: matrix-free implementations, parallelization (MPI, threading via TBB & Taskflow, SIMD, GPU support), discontinuous Galerkin methods, AMR via p4est, particles, wrappers for PETSc and Trilinos, ...



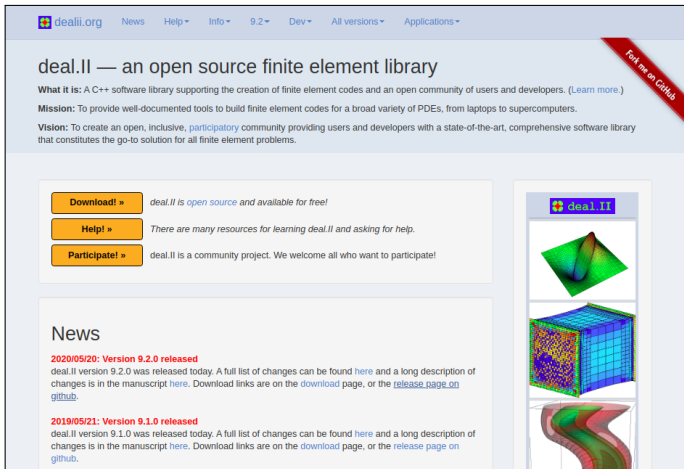
---

<sup>1</sup> successor of DEAL: Differential Equations Analysis Library

### **publications describing the design of and recent development in deal.II:**

D. Arndt, W. Bangerth, D. Davydov, T. Heister, L. Heltai, M. Kronbichler, M. Maier, J.-P. Pelteret, B. Turcksin, and D. Wells. [The deal.II finite element library: Design, features, and insights](#). *Computers and Mathematics with Applications*. 2020. DOI: <https://doi.org/10.1016/j.camwa.2020.02.022>

D. Arndt, W. Bangerth, B. Blais, T. C. Clevenger, M. Fehling, A. V. Grayver, T. Heister, L. Heltai, M. Kronbichler, M. Maier, P. Munch, J.-P. Pelteret, R. Rastak, I. Thomas, B. Turcksin, Z. Wang, and D. Wells. [The deal.II Library, Version 9.2](#). *Journal of Numerical Mathematics*. 2020. DOI: <https://doi.org/10.1515/jnma-2020-0043>



The screenshot shows the official dealii.org website. At the top is a navigation bar with links for News, Help, Info, 9.2, Dev, All versions, and Applications. The main heading is "deal.II — an open source finite element library". Below this, it states "What it is: A C++ software library supporting the creation of finite element codes and an open community of users and developers." It also lists the "Mission" and "Vision" of the project. A red banner on the right says "Fork me on GitHub". In the center, there are three buttons: "Download!", "Help!", and "Participate!", each with a brief description. To the right, there are three 3D visualizations: a surface plot, a mesh of a cube, and a cross-section of a cylinder. Below the buttons, there is a "News" section with two entries: "2020/05/20: Version 9.2.0 released" and "2019/05/21: Version 9.1.0 released", each with a link to the release page on GitHub.

dealii.org News Help Info 9.2 Dev All versions Applications

## deal.II — an open source finite element library

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

**Mission:** To provide well-documented tools to build finite element codes for a broad variety of PDEs, from laptops to supercomputers.

**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.

**Download! »** *deal.II is [open source](#) and available for free!*

**Help! »** *There are many resources for learning deal.II and asking for help.*


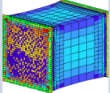

**Participate! »** *deal.II is a community project. We welcome all who want to participate!*

### News

**2020/05/20: Version 9.2.0 released**  
deal.II version 9.2.0 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](#).

**2019/05/21: Version 9.1.0 released**  
deal.II version 9.1.0 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](#).

deal.II



... [www.dealii.org](http://www.dealii.org)

**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 dealii.org

## 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 >"]
    D["< double >"]
    N["< Number >"]
    VAD["VectorizedArray< double >"]
    VAN["VectorizedArray< Number >"]

    VAB1 -- "< VectorizedArray< Number, width >, 1 >" --> VAB2
    VAB2 --> VA
    VA -- "< double >" --> VAD
    VA -- "< Number >" --> VAN
    
```

Public Types

using `value_type` = Number

Public Member Functions

`VectorizedArray` ()=default  
`VectorizedArray` (const Number scalar)

Extensive Doxygen documentation

dealii / dealii

Unwatch 62 Star 678 Fork 478

<> Code Issues 462 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

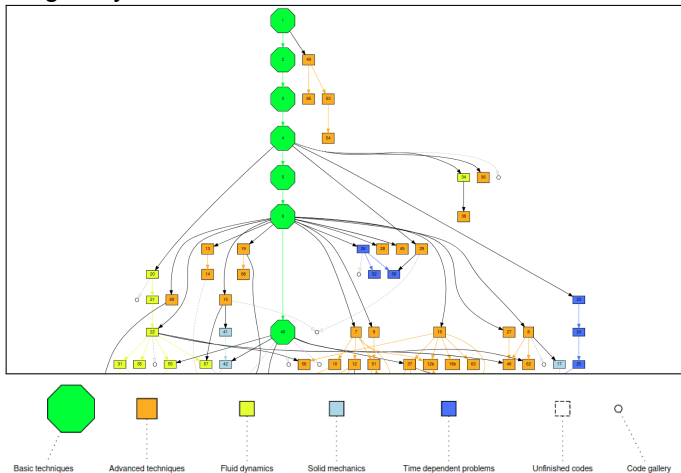
Pages 629

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

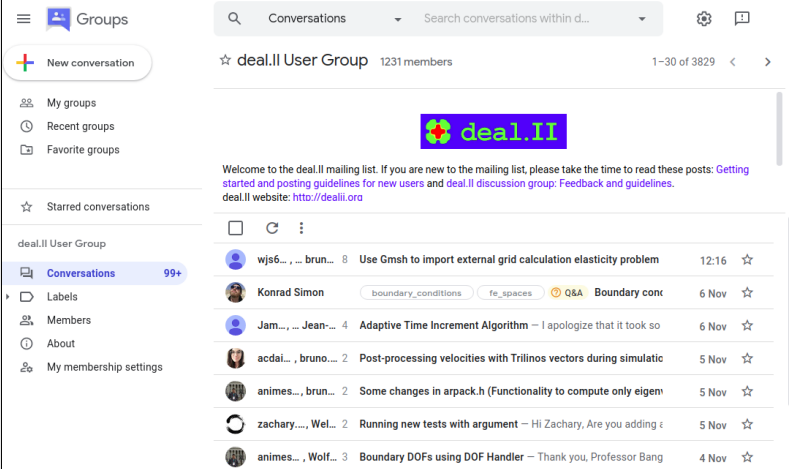
69 tutorials and code gallery:



*... further 6 tutorials: work in progress*



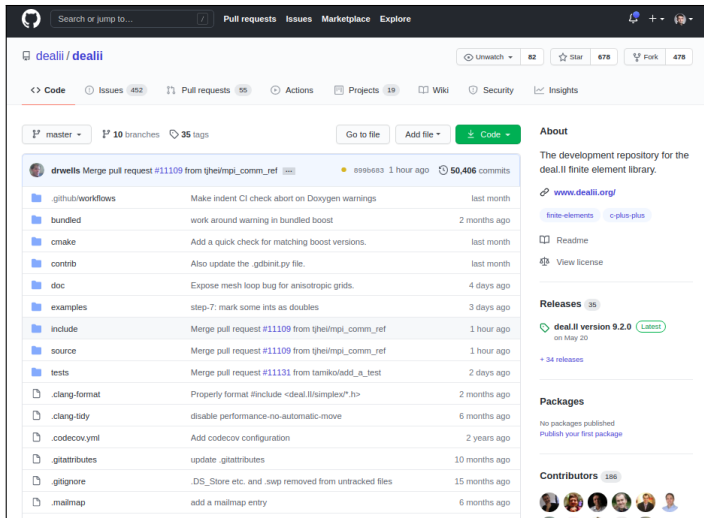
## deal.II user group:



The screenshot shows the deal.II User Group forum page. The left sidebar contains navigation links: 'New conversation', 'My groups', 'Recent groups', 'Favorite groups', 'Starred conversations', and a section for the 'deal.II User Group' with links to 'Conversations' (99+), 'Labels', 'Members', 'About', and 'My membership settings'. The main content area displays the group name 'deal.II User Group' with 1231 members and a search bar. Below the group name is a welcome message and a list of recent discussions. The discussions are listed in a table with columns for user avatars, usernames, post counts, titles, dates, and star icons.

User	Post Count	Title	Date	Star
wjs6..., ... brun...	8	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...	4	Adaptive Time Increment Algorithm – I apologize that it took so	6 Nov	☆
acda..., bruno...	2	Post-processing velocities with Trilinos vectors during simulatio	5 Nov	☆
animes..., brun...	2	Some changes in arpack.h (Functionality to compute only eigen	5 Nov	☆
zachary..., Wel...	2	Running new tests with argument – Hi Zachary, Are you adding ε	5 Nov	☆
animes..., Wolf...	3	Boundary DOFs using DOF Handler – Thank you, Professor Bang	4 Nov	☆

... Q/A by users and developers!



The screenshot shows the GitHub repository page for `dealii / dealii`. The repository has 82 Unwatch, 678 Stars, and 478 Forks. The main content area displays a list of files and directories, each with a description and the time since the last commit. A pull request #11109 is highlighted, showing it was merged 1 hour ago. The right sidebar contains sections for 'About' (The development repository for the deal.II finite element library), 'Releases' (Latest release: deal.II version 9.2.0 on May 20), 'Packages' (No packages published), and 'Contributors' (196 contributors).

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

- issues
- pull requests
- GitHub actions → CI
- required: approval by  $\geq 1$  principal developer

Login

All Dashboards

Deal.II

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

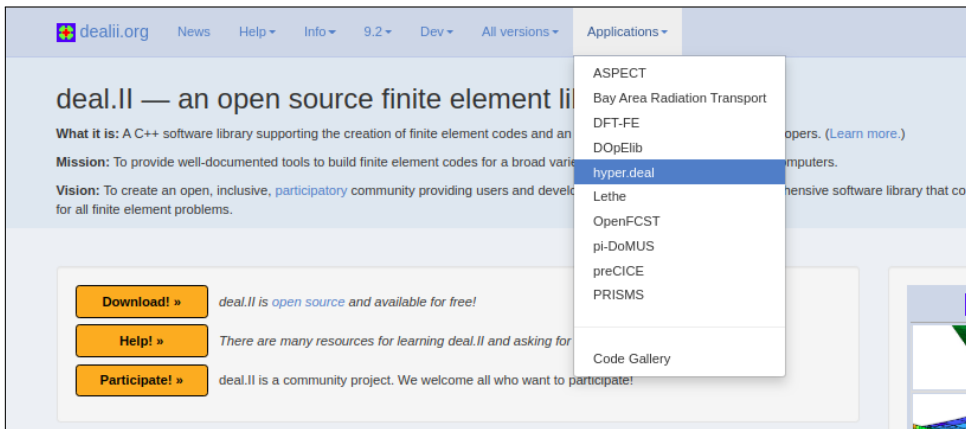
Regression Tests

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	12417	20 hours ago	
tester	Clang-10.0.0-master-avx2-Ofast	414559	0	0	0	0	0	49	12404	Nov 03, 2020 - 07:54 UTC	
tester	Clang-10.0.0-master-avx2-Ofast	2bdb45	0	0	0	0	0	48	12417	Nov 06, 2020 - 01:00 UTC	
tester	Clang-10.0.0-master-avx2-Ofast	cd2846	0	0	0	0	0	48	12415	Nov 05, 2020 - 03:17 UTC	
tester	Clang-10.0.0-master-avx2-Ofast	220f05	0	0	0	0	0	48	12415	Nov 04, 2020 - 05:34 UTC	
tester	GNU-10.1.0-master-tets	10ceee	0	0	0	0	0	38	11866	10 hours ago	
tester	GNU-10.1.0-master-tets	2bdb45	0	0	0	0	0	38	11864	Nov 06, 2020 - 10:56 UTC	

*... more than 5,000 tests run for different compilers/hardware/configurations*

Some deal.II-based user codes/libraries are open source as well:



The screenshot shows the dealii.org website. The navigation bar includes links for News, Help, Info, 9.2, Dev, All versions, and Applications. The Applications dropdown menu is open, displaying a list of user codes/libraries: ASPECT, Bay Area Radiation Transport, DFT-FE, DOpElib, hyper.deal (highlighted), Lethe, OpenFCST, pi-DoMUS, preCICE, and PRISMS. Below the menu, there is a 'Code Gallery' link. The main content area features a large heading 'deal.II — an open source finite element library' and a description of the library. Below this, there are three orange buttons: 'Download! »', 'Help! »', and 'Participate! »', each followed by a short description of the deal.II project.

dealii.org News Help Info 9.2 Dev All versions Applications

deal.II — an open source finite element library

What it is: A C++ software library supporting the creation of finite element codes and an efficient solver for a broad variety of problems. (Learn more.)

Mission: To provide well-documented tools to build finite element codes for a broad variety of problems on computers.

Vision: To create an open, inclusive, participatory community providing users and developers with a comprehensive software library that covers all finite element problems.

Download! » deal.II is open source and available for free!

Help! » There are many resources for learning deal.II and asking for help.

Participate! » deal.II is a community project. We welcome all who want to participate!

ASPECT  
Bay Area Radiation Transport  
DFT-FE  
DOpElib  
hyper.deal  
Lethe  
OpenFCST  
pi-DoMUS  
preCICE  
PRISMS  
Code Gallery

... motivation for further development

Part 2:

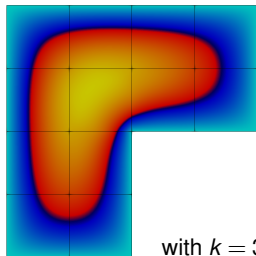
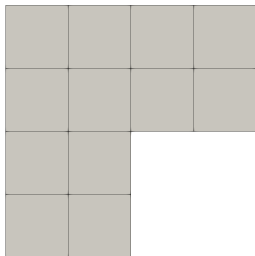
## **Poisson problem/introduction into the main modules**

- ▶ strong form:

$$\nabla^2 u = f \text{ on } \Omega, \quad u|_{\Gamma} = 0$$

- ▶ weak form:

$$(\nabla v, \nabla u)_{\Omega} = (v, f)_{\Omega}, \quad u|_{\Gamma} = 0$$



with  $k = 3$ ,  $f = 1$

- ▶ introduce cells & use scalar Lagrange finite element  $\mathcal{Q}_k$ :

$$(\nabla v, \nabla u)_{\Omega_e} \approx \sum_q (\nabla v, \nabla u) \cdot |J| \times w \quad \rightarrow \quad \mathbf{K}_{ij}^{(e)} = \sum_q (\nabla N_{iq}, \nabla N_{jq}) \cdot |J_q| \times w_q$$

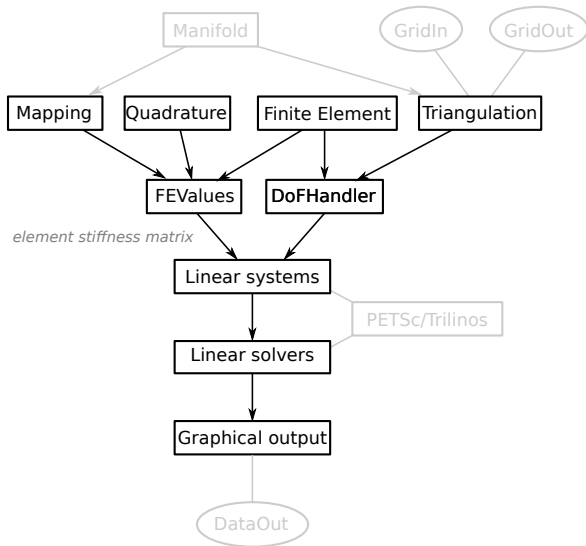
... with  $N$  shape functions in real space, mapping & quadrature

- ▶ loop over all cells, assemble system matrix and right-hand-side vector, and solve system

## needed from a FEM library:

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

**deal.II main modules** →



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

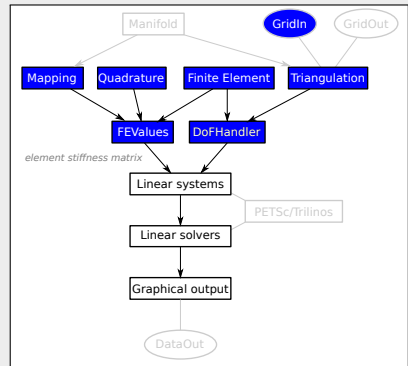
parallel::distributed::Triangulation<dim> tria(MPI_COMM_WORLD);
util::create_reentrant_corner(tria);

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;
```

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

```
FEValues<dim> fe_values(mapping, fe, quad, update_values | update_gradients | update_JxW_values);
```



```

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); //  $\mathbf{A}_e \mathbf{K}^{(e)}, \mathbf{A}_e \mathbf{f}^{(e)}$ 
} //
b.compress(VectorOperation::add); A.compress(VectorOperation::add); //

```

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

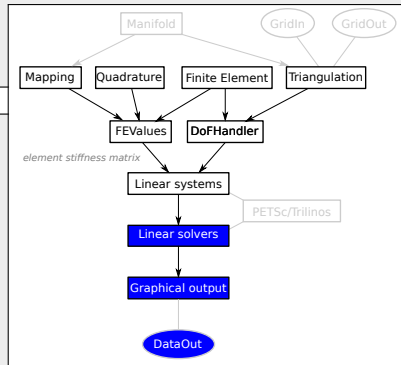
if (Utilities::MPI::this_mpi_process(util::get_mpi_comm(tria)) == 0)
    printf("Solved in %d iterations.\n", reduction_control.last_step());

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);
```

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



### Some general remarks:

- ▶ deal.II provides classes that can be used - parameters have to be handled by user code
- ▶ time integration is not a first-class citizen of deal.II: users need to implement their own time-integration schemes or rely on external packages<sup>2</sup>

---

<sup>2</sup>deal.II has a wrapper to SUNDIALS.

Part 3:

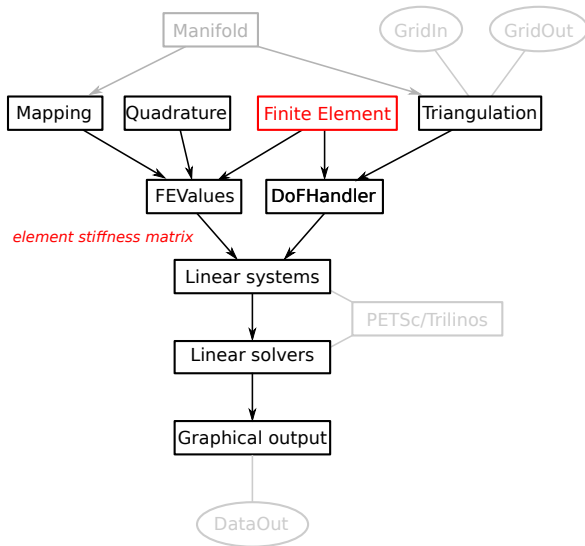
# Elasticity

describe  $\vec{u} \in \mathbb{R}^d$  as a **system of scalar Lagrange finite elements**:

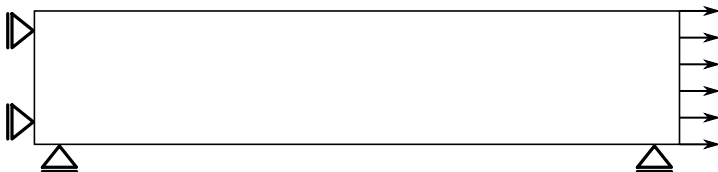
$$\underbrace{[Q_p^d, \dots, Q_p^d]}_{\times d}$$

definition of **element stiffness matrix**  $\mathbf{K}^{(e)}$ :

$$\mathbf{K}^{(e)} = \int_{\Omega^{(e)}} \mathbf{B}^T \mathbf{C} \mathbf{B} d\Omega$$



**Tension rod:**

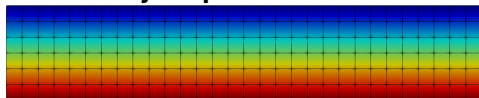


*... with the result:*

**x displacement:**



**y displacement:**



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

**Only relevant differences are shown:**

- ▶ system of scalar finite elements:

```
FESystem<dim> fe(FE_Q<dim>(degree), dim);
```

► computation of  $\mathbf{K}^{(e)} = \int_{\Omega^{(e)}} \mathbf{B}^T \mathbf{C} \mathbf{B} d\Omega$ :

► according to step-18

```
for (const auto i : fe_values.dof_indices()) // with SymmetricTensor<4, dim> C;
for (const auto j : fe_values.dof_indices())
    for (const auto q : fe_values.quadrature_point_indices())
    {
        SymmetricTensor<2, dim> eps_phi_i = get_strain(fe_values, i, q), eps_phi_j = get_strain(fe_values, j, q);

        cell_matrix(i, j) += (eps_phi_i * C * eps_phi_j) * fe_values.JxW(q); // (B^T C B) \cdot |J| x W
    }
```

```
inline SymmetricTensor<2, dim>
get_strain(const FEValues<dim> &fe_values, const unsigned int shape_func, const unsigned int q_point)
{
    SymmetricTensor<2, dim> tmp;

    for (unsigned int i = 0; i < dim; ++i)
        tmp[i][i] = fe_values.shape_grad_component(shape_func, q_point, i)[i];

    for (unsigned int i = 0; i < dim; ++i)
        for (unsigned int j = i + 1; j < dim; ++j)
            tmp[i][j] = (fe_values.shape_grad_component(shape_func, q_point, i)[j] +
                        fe_values.shape_grad_component(shape_func, q_point, j)[i]) / 2;

    return tmp;
}
```

Part 4:

## **Volume and surface coupling**



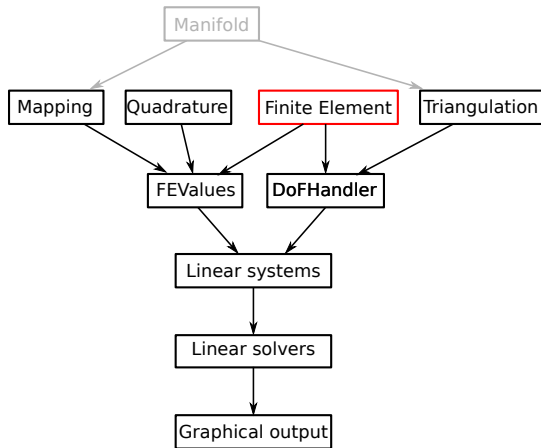
## Example: TSI

- thermo-structural element:

$$\underbrace{[Q_p^d, \dots, Q_p^d]}_{\times d+1}$$

- alternatively with  $p_t \neq p_u$ :

$$[Q_{p_t}^d, \underbrace{[Q_{p_u}^d, \dots, Q_{p_u}^d]}_{\times d}]$$



```
FESystem<dim> fe(FE_Q<dim>(degree), dim + 1);  
// vs.  
FESystem<dim> fe(FE_Q<dim>(degree), 1, FESystem<dim> fe(FE_Q<dim>(degree), dim));
```

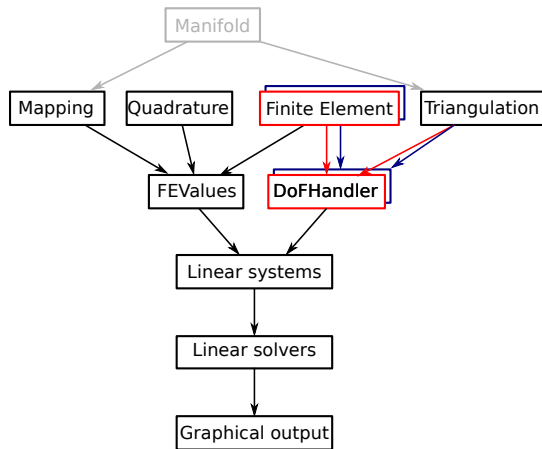
## Example: TSI

- thermoelement:

$$Q_{p_t}^d$$

- structural element:

$$\underbrace{[Q_{p_u}^d, \dots, Q_{p_u}^d]}_{\times d}$$



```
FE_Q<dim>    fe_t(degree);  
FESystem<dim> fe_s(FE_Q<dim>(degree), dim);
```

## Example: FSI with 2 grids

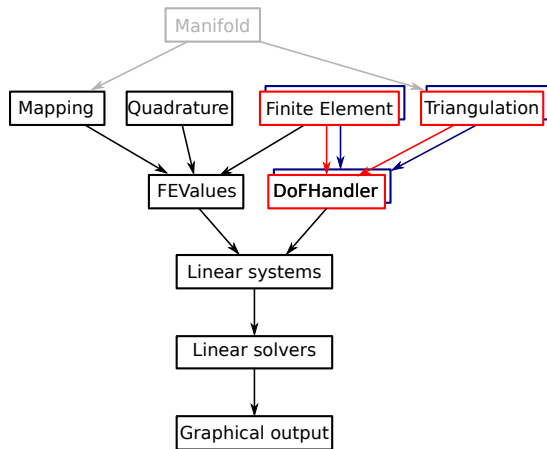
- elasticity:

$$\underbrace{[Q_p^d, \dots, Q_p^d]}_{\times d}$$

- incompressible Navier-Stokes eq.:

$$\underbrace{[Q_{p_v}^d, \dots, Q_{p_v}^d]}_{\times d} \quad \text{and} \quad Q_{p_p}^d$$

for surface coupling, e.g., preCICE  
adapter can be used

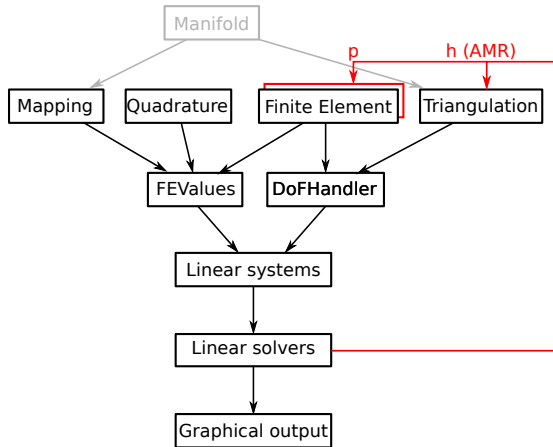


Part 5:

## **h/p/hp-adaptivity**

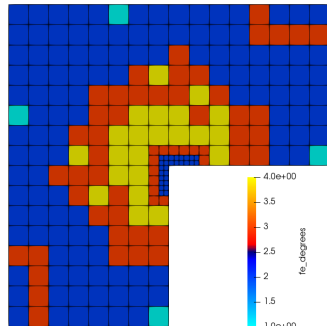
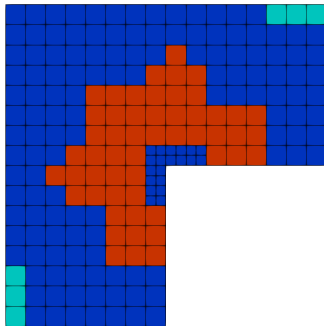
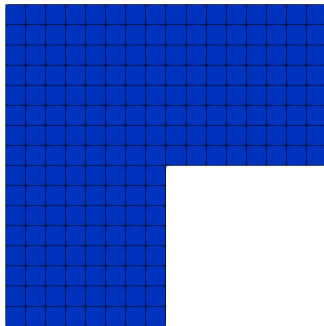
## support of:

- ▶ fe collections
- ▶ hanging nodes
- ▶ distributed hp-adaptivity



3 cycles of hp-adaptivity:

▷ *colors: polynomial degree  $1 \leq k \leq 4$*



... solved with a distributed matrix-free p-multigrid algorithm

Part 6:

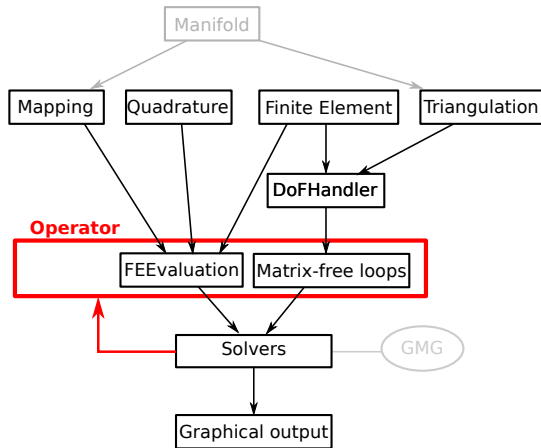
## **Matrix-free operator evaluation**

**idea:** rely on operator evaluations

```
vmult(dst, src)
```

**features:**

- ▶ sum factorization
- ▶ vectorization via SIMD (CPUs)
- ▶ GPU support





## Matrix-free operator evaluation of $(\nabla v, \nabla u)_\Omega$ :

```
matrix_free.template cell_loop<Vector<Number>, Vector<Number>>({
    [&](const auto & mf, auto &dst, const auto &src, const auto cells) {
        FEEvaluation<dim, -1, 0, n_components, Number, VectorizedArrayType> phi(mf);

        for (unsigned int cell = cells.first; cell < cells.second; ++cell)
        {
            phi.reinit(cell);
            phi.gather_evaluate(src, EvaluationFlags::gradients);

            for (unsigned int q = 0; q < phi.dofs_per_cell; ++q)
                phi.submit_gradient(phi.get_gradient(q), q);

            phi.integrate_scatter(EvaluationFlags::gradients, dst);
        }
    }, dst, src);
```

// on a cell batch  
//  
//  $(\nabla v, \nabla u)_{\Omega(e)}$   
//  
//  $\sum_q (\nabla_\xi v, \mathcal{J}_q^{-1} |\mathcal{J}_q| w_q \mathcal{J}_q^{-T} \nabla_\xi u)$

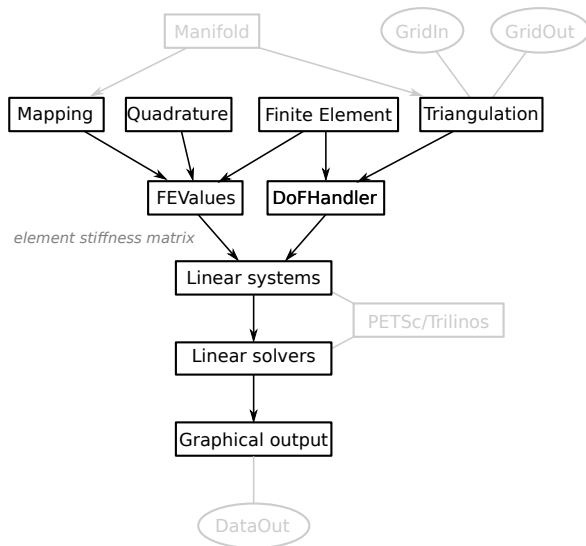
$\rightarrow \mathcal{J}_q^{-1} |\mathcal{J}_q| w_q \mathcal{J}_q^{-T} \nabla \hat{u}_q$   
 $\uparrow$   
 $\sum_q (\nabla_\xi v, \mathcal{J}_q^{-1} |\mathcal{J}_q| w_q \mathcal{J}_q^{-T} \nabla_\xi u)$

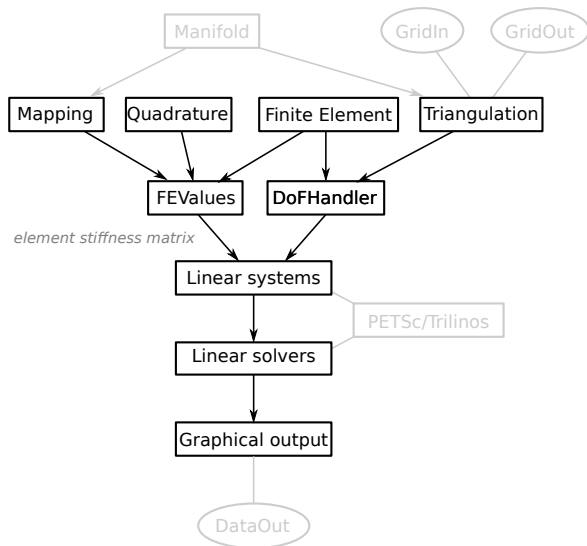
... full code: <https://github.com/peterrum/dealii-examples/blob/master/poisson-mf.cc>

Part 7:

## Summary

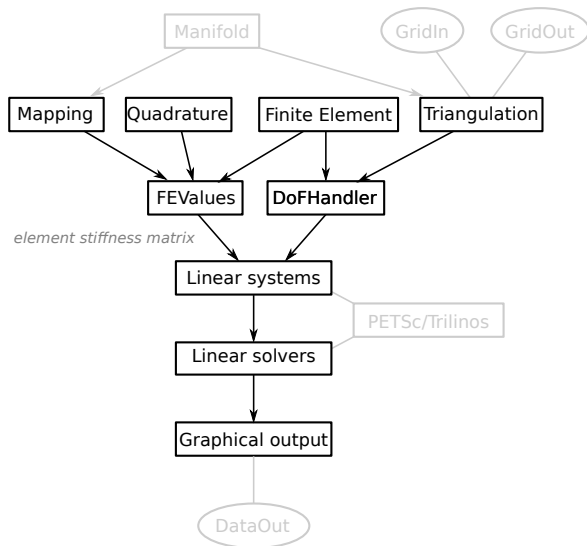
# Triangle and tetrahedron support?





## Question 1:

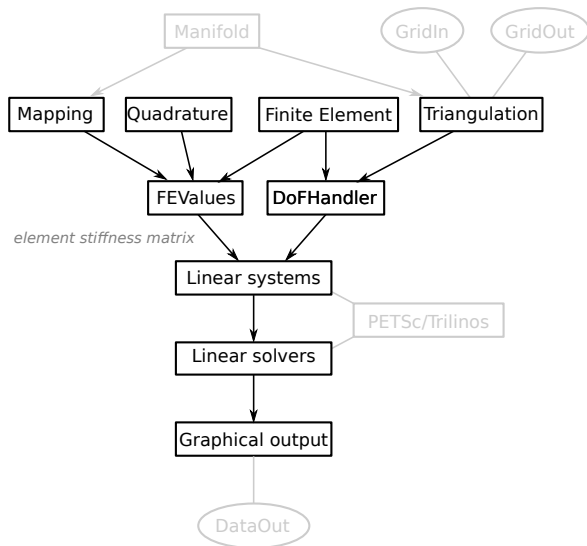
How would you proceed to use simplicities (triangles, tetrahedrons)?



## Question 1:

How would you proceed to use simplicities (triangles, tetrahedrons)?

```
using namespace Simplex;
FE_P<dim>      fe(degree);
QGauss<dim>    quad(degree + 1);
MappingFE<dim> mapping(FE_P<dim>(1));
```



## Question 1:

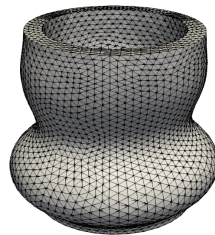
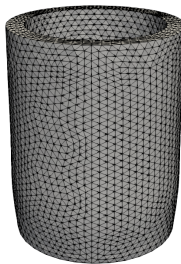
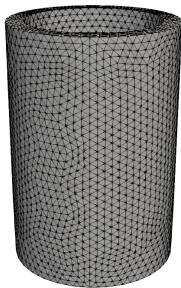
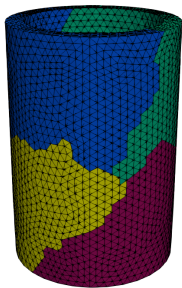
How would you proceed to use simplicities (triangles, tetrahedrons)?

```
using namespace Simplex;
FE_P<dim>      fe(degree);
QGauss<dim>    quad(degree + 1);
MappingFE<dim> mapping(FE_P<dim>(1));
```

## Question 2:

How would you proceed to add simplex support to a library with pure hexahedral mesh support?

step-18 with  $p::s::T$  (5 procs):  
55,168 cells, 251,028 DoFs



Questions/comments?