deal.II Workshop @ hereon/TUHH

Day 1: Overview, FEM basics, mesh handling

Peter Munch[1,2,3], Daniel Paukner[3], Ingo Scheider[3]

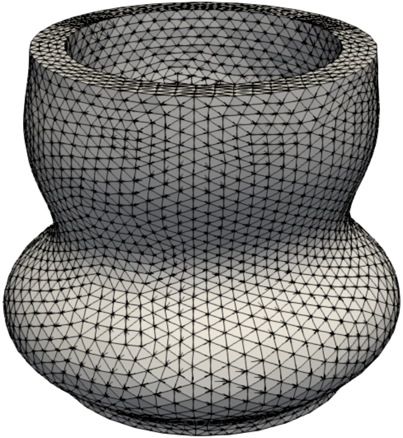
[1]deal.II developer

[2]Institute for Computational Mechanics, Technical University of Munich, Germany

[3]Kontinuumssimulationen, Institut für Werkstoffsystem-Modellierung, Helmholtz-Zentrum hereon, Germany

April 19, 2021

Motivation



$$\mathrm{Div}(\underline{\boldsymbol{F}} \cdot \underline{\boldsymbol{S}}(\underline{\boldsymbol{E}})) + \rho_0 \underline{\hat{\boldsymbol{b}}} = 0$$

FEM

How can deal.II help?

Organization: timetable

Organization:

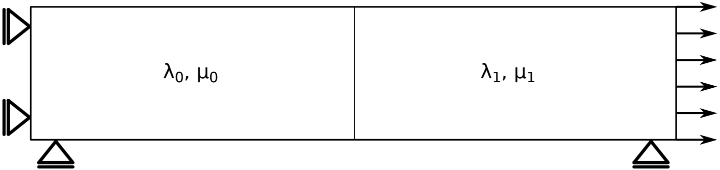
- 9:00-12:00 (Monday-Wednesday): presentation/workshop with open end
- 9:00-9:30 (Monday): presentation round
- 9:00-9:30 (Tuesday, Wednesday): question time

Topics:

- Monday: introduction into FEM, overview of deal.II, mesh handling
- Tuesday: Poisson problem
- Wednesday: solid mechanics

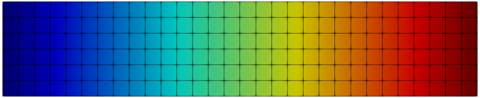
Organization: goal at the end of the second day

Tension rod:

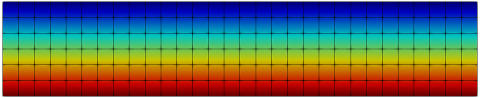


... with the result:

x displacement:



y displacement:



Part 1:

Introduction round

... other slide set

Part 2:

A short introduction into finite-element methods

Model problem: Poisson problem

Strong form of the Poisson problem:

$$
\begin{aligned}
-\nabla \cdot \nabla u &= f && \text{in } \Omega = (0,1) \times (0,1), \\
u &= h && \text{on } \Gamma_D = \{x = 0, y \in (0,1)\}, \\
\nabla u(x,y) \cdot \underline{\boldsymbol{n}} &= g && \text{on } \Gamma_N = \{x = 1, y \in (0,1)\}, \\
\nabla u(x,y) \cdot \underline{\boldsymbol{n}} &= 0 && \text{else.}
\end{aligned}
$$

Steps:

a. definition of the function spaces
b. derivation of the weak form
c. spatial discretization + computation of the element stiffness matrix
d. assembly and set-up of the linear equation system

Definition of the function spaces

- $\mathcal{V}_{\hat{u}}(t,\Omega) = \{u(\cdot,t) \in \mathcal{H}^1(\Omega) \,:\, u = \hat{u} \text{ on } \Gamma_D\}$ for the solution
- $\mathcal{V}_0(\Omega) = \{v \in \mathcal{H}^1(\Omega) \,:\, v = 0 \text{ on } \Gamma_D\}$ for the test function

Derivation of the weak form

1. step: multiplication with the test function v and integration over Ω

$$-\int_{\Omega} v(\underline{\boldsymbol{x}})\nabla\cdot(\nabla u(\underline{\boldsymbol{x}},t))\mathrm{d}\underline{\boldsymbol{x}} = \int_{\Omega} v(\underline{\boldsymbol{x}}) f \mathrm{d}\underline{\boldsymbol{x}}$$

2. step: integration by parts:

$$\int_{\Omega} v(\underline{\boldsymbol{x}})\nabla\cdot(\nabla u(\underline{\boldsymbol{x}}))\mathrm{d}\underline{\boldsymbol{x}} = \int_{\Gamma} v(\underline{\boldsymbol{x}})(\nabla u(\underline{\boldsymbol{x}}))\cdot\underline{\boldsymbol{n}}\mathrm{d}\Gamma - \int_{\Omega} \nabla v(\underline{\boldsymbol{x}})\cdot\nabla u(\underline{\boldsymbol{x}})\mathrm{d}\underline{\boldsymbol{x}}$$

3. step: exploitation of the boundary conditions (with: $\Gamma = \Gamma_D \cup \Gamma_N$):

$$\int_{\Gamma} v(\underline{\boldsymbol{x}})\nabla u(\underline{\boldsymbol{x}},t)\cdot\underline{\boldsymbol{n}}\mathrm{d}\Gamma = \overbrace{\int_{\Gamma_D} v(\underline{\boldsymbol{x}})\nabla u(\underline{\boldsymbol{x}},t)\cdot\underline{\boldsymbol{n}}\mathrm{d}\Gamma}^{0} + \int_{\Gamma_N} v(\underline{\boldsymbol{x}}) g \mathrm{d}\Gamma$$

Derivation of the weak form (cont.)

Weak form

Find $u(\underline{\boldsymbol{x}}) \in \mathcal{V}_{\hat{u}}$ such that for all $v(\underline{\boldsymbol{x}}) \in \mathcal{V}_0(\Omega)$:

$$\int\limits_{\Omega} \nabla v(\underline{\boldsymbol{x}}) \cdot \nabla u(\underline{\boldsymbol{x}}) \mathrm{d}\underline{\boldsymbol{x}} = \int\limits_{\Omega} v(\underline{\boldsymbol{x}}) f \mathrm{d}\underline{\boldsymbol{x}} + \int\limits_{\Gamma_N} v(\underline{\boldsymbol{x}}) g \mathrm{d}\Gamma$$

in compact notation:

$$(\nabla v(\underline{\boldsymbol{x}}),\, \nabla u(\underline{\boldsymbol{x}}))_{\Omega} = (v(\underline{\boldsymbol{x}}), f)_{\Omega} + (v(\underline{\boldsymbol{x}}), g)_{\Gamma_N}$$

Discretization

- decompose computational domain into cells $\Omega = \bigcup \Omega^{(e)}$
- use scalar Lagrange finite element $\mathcal{Q}_k$:

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

$$(v, f)_{\Omega_e} \quad \rightarrow \quad \mathbf{f}_i^{(e)} = \sum_q (N_{iq}, f) \cdot |J_q| \times w_q$$

$$(v, g)_{\Gamma_{e,N}} \quad \rightarrow \quad \mathbf{g}_i^{(e)} = \sum_q (N_{iq}, g) \cdot |J_q| \times w_q$$

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

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

$$\boldsymbol{K}\boldsymbol{u} = \boldsymbol{f} + \boldsymbol{g}$$

Requirements to FEM library

The solution of a PDE with an FEM library (like deal.II):

$$\boldsymbol{K}\boldsymbol{u} = \boldsymbol{f} + \boldsymbol{g} \quad \text{with} \quad \begin{aligned} \mathbf{K}_{ij}^{(e)} &= \sum_q (\nabla N_{iq}, \nabla N_{jq}) \cdot |J_q| \times w_q \\ \mathbf{f}_i^{(e)} &= \sum_q (N_{iq}, f) \cdot |J_q| \times w_q \\ \mathbf{g}_i^{(e)} &= \sum_q (N_{iq}, g) \cdot |J_q| \times w_q \end{aligned}$$

requires:

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

Part 3:

Short overview of deal.II

Introduction

- deal.II[1]: 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, ...

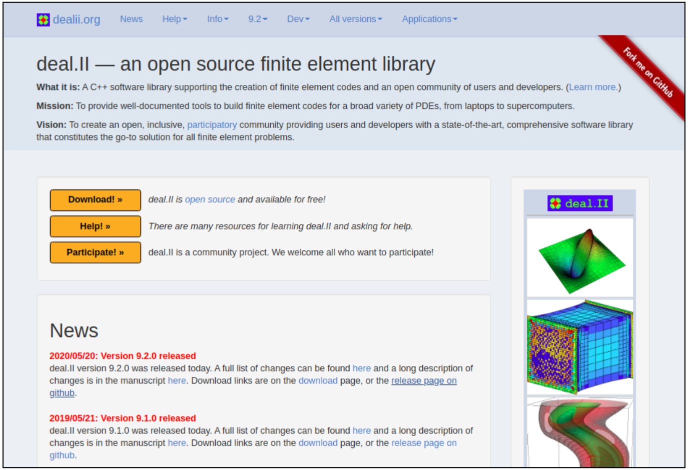
[1] successor of DEAL: **D**ifferential **E**quations **A**nalysis **L**ibrary

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

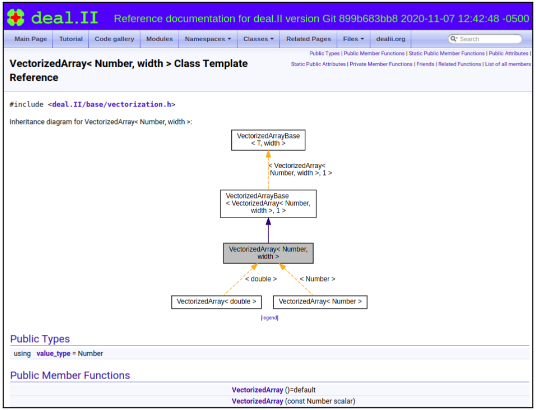
D. Arndt, W. Bangerth, B. Blais, T. C. Clevenger, M. Fehling, A. V. Grayver, T. Heister, L. Heltai, M. Kronbichler, M. Maier, P. Munch, J.-P. Pelteret, R. Rastak, I. Thomas, B. Turcksin, Z. Wang, and D. Wells. The deal.II Library, Version 9.2. *Journal of Numerical Mathematics.* 2020.
DOI: https://doi.org/10.1515/jnma-2020-0043

Official webpage

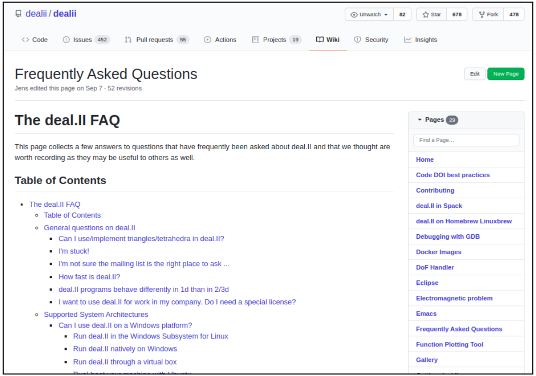


... www.dealii.org

Documentation



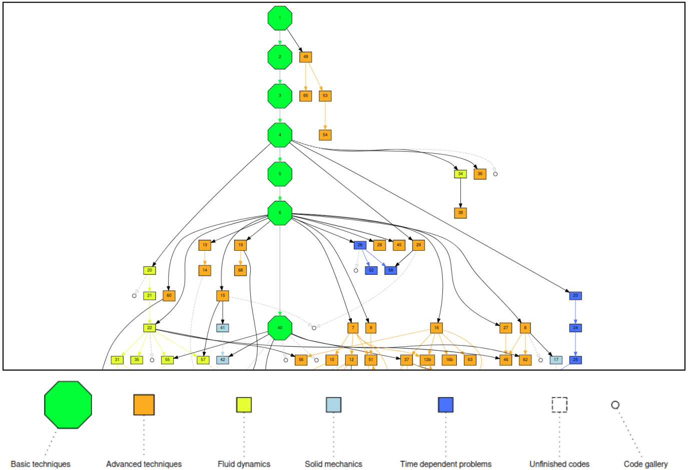
Extensive Doxygen documentation



GitHub Wiki

Documentation (cont.)

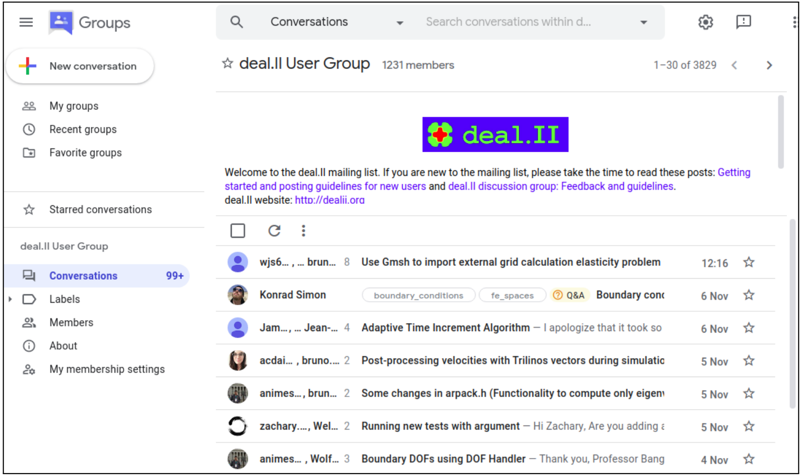
70 tutorials and code gallery:



... further 7 tutorials: work in progress

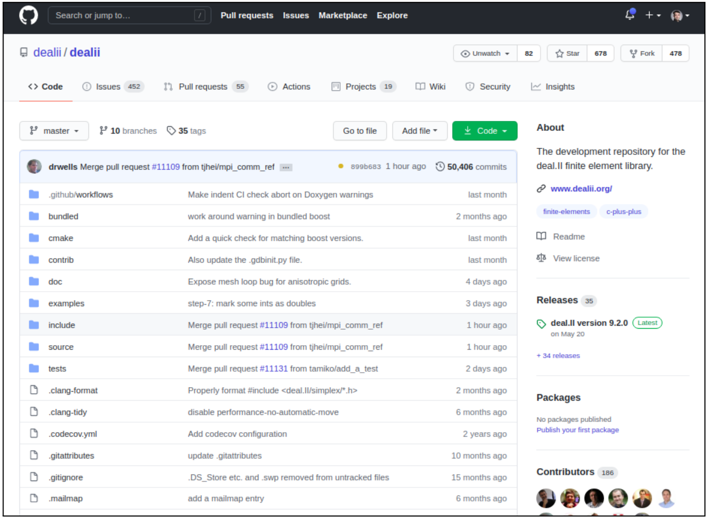
Forum

deal.II user group:



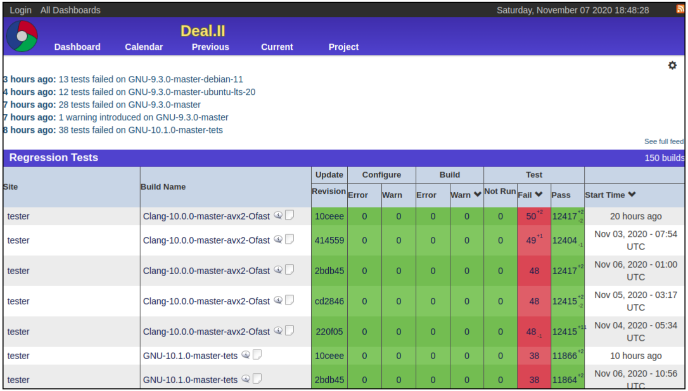
... Q&A by users and developers!

Development on GitHub



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

Continuous integration



Login All Dashboards Saturday, November 07 2020 18:48:28

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

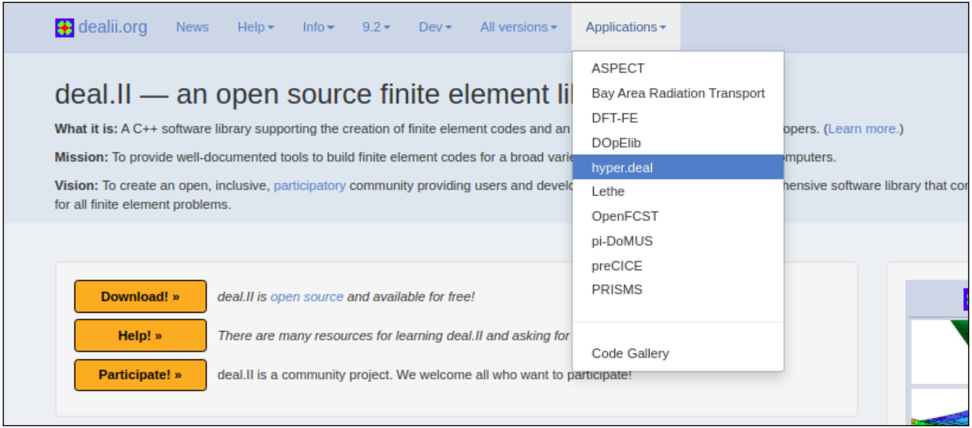
See full feed

Regression Tests 150 builds

Site	Build Name	Update	Configure		Build		Test			
		Revision	Error	Warn	Error	Warn	Not Run	Fail	Pass	Start Time
tester	Clang-10.0.0-master-avx2-Ofast	10ceee	0	0	0	0	0	50^{+2}	12417^{+2}_{-2}	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^{+2}_{-2}	Nov 05, 2020 - 03:17 UTC
tester	Clang-10.0.0-master-avx2-Ofast	220f05	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 5,000 tests run for different compilers/hardware/configurations

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



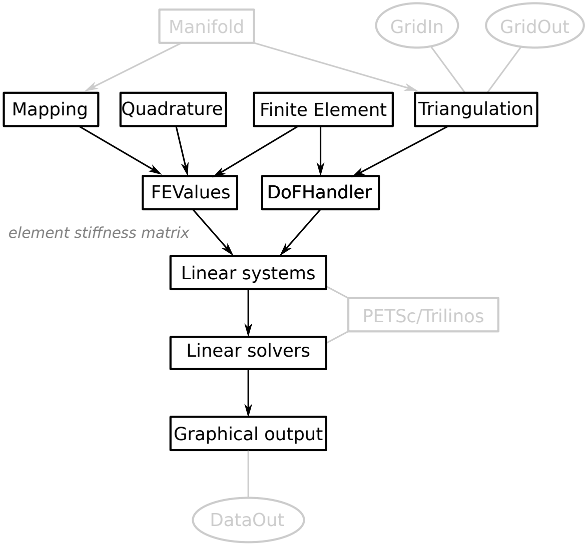
... motivation for further development

Main modules

needed from a FEM library:

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

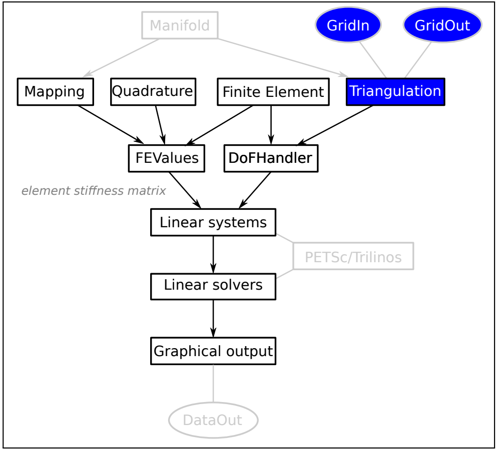
deal.II main modules $\rightarrow$



Part 4:

Mesh handling in deal.II

Mesh handling



Triangulation

- meshes are called `Triangulation`
- meshes can be created with functions in the `GridGenerator` namespace →
- meshes can be read from files
- meshes can be written to files

```
#include <deal.II/grid/tria.h>

int main()
{
  using namespace dealii;

  Triangulation<2> tria;
  GridGenerator::subdivided_hyper_cube(tria, 3);

  // output properties
  std::cout << tria.n_cells() << std::endl;
}
```

more information about `Triangulation` and `GridGenerator`:
https://www.dealii.org/developer/doxygen/deal.II/index.html

Cells

- meshes consist of cells
- it is possible to loop over all cells of a mesh
- cell properties (e.g., material ID) can be get/set →

  see: `CellAccessor`
- vertices, lines, and faces of cells can be accessed

  see: `TriaAccessor`

```
#include <deal.II/grid/tria.h>

int main()
{
  using namespace dealii;

  Triangulation<2> tria;
  // ...

  for(auto & cell : tria.active_cell_iterators())
  {
    std::cout << cell->matrial_id() << std::endl;
    std::cout << cell->face(0)->boundary_id()
                                    << std::endl;
  }
}
```

note "`operator->()`": we are working with iterators (here: `TriaIterator`)

Task 1a: reading and writing meshes

- read the mesh “beam.msh” with `GridIn::read()`
- write the mesh to “task-1a-grid.vtk” with `GridOut::write_vtk()`
- take a look at the mesh in Paraview
- what properties are visualized?

... don’t forget to include the needed header files!

Optional:

- write the mesh to “task-1a-data.vtk” with `DataOut::write_vtk()`
- create a mesh in Gmsh
- try out other mesh formats

Task 1a: reading and writing meshes (cont.)

Getting started with Linux:

- open a terminal and get/compile the code:

```
git clone https://github.com/peterrum/dealii-hereon-workshop.git
cd dealii-hzg-workshop-draft
cmake .
make task-1a-empty
```

- run the program:

```
./task-1a-empty
```

- for visualization use Paraview:

```
paraview
```

... in a new tab or new terminal

Task 1b: working with meshes

Loop over all cells and boundary faces and

- print the center of each cell
- assign material IDs to cells
- assign boundary IDs to faces

... hint: check results in Paraview!