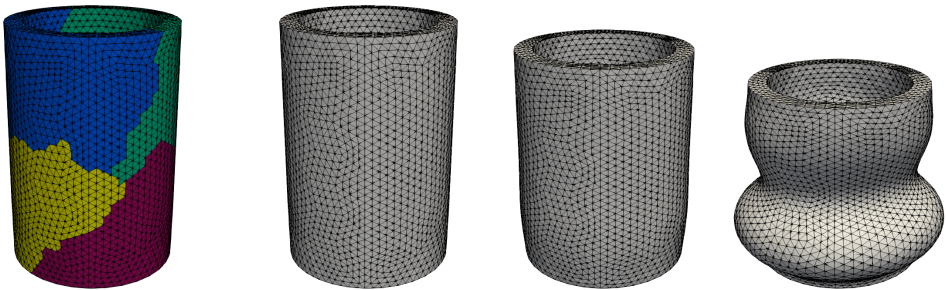
deal.II “simplex” workshop 2021

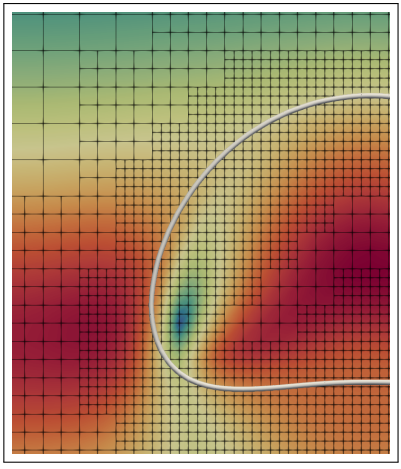
Day 1: organization, overview, introduction



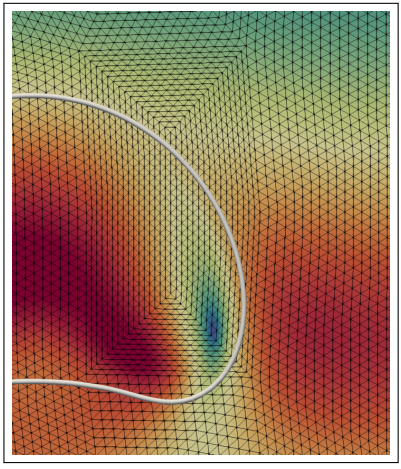
January 13, 2021

Goal: *"The best of both worlds"*

Example: rising bubble (incompressible NS + LS): *by @kronbichler, @mschreter, @peterrum*

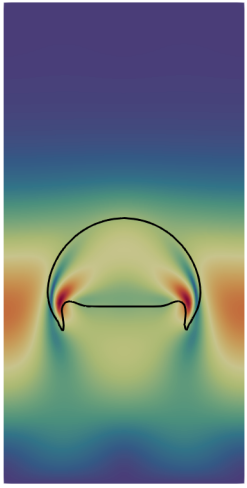


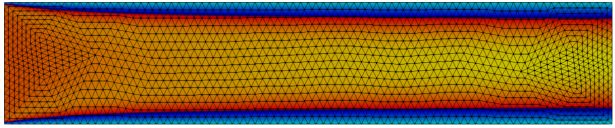
Hypercube mesh + AMR

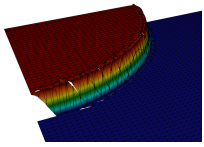


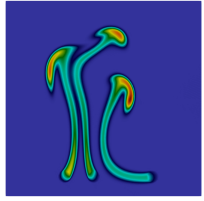
Simplex mesh

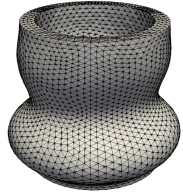
Goal: *"The best of both worlds"* (cont.)



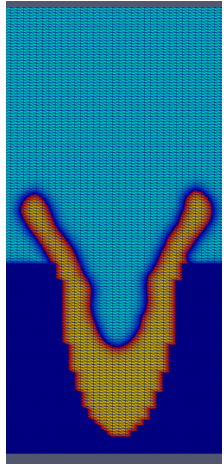








by ©mschreter



see also: adaflo and MeltPoolDG

Goal: *"The best of both worlds"* (cont.)

Goals of this workshop:

- to present the current status of simplex support in deal.II (work of the last 9 months)
- hands-on: using the new infrastructure
- identifying bugs and work packages
- improving documentation
- implementing new simplex features

... the perspective of non-developers is needed!

Part 1:

Overview

Organization: timetable

- every day 8am-3pm and 4/5pm-10pm (CEWT): supervised programming session
- Wednesday 3pm-5pm: kick-off meeting
  - presentation “Current status of simplex support in deal.II” by Peter Munch
  - presentation of gained user experience by Pasquale Africa
  - introduction of participants and presentation of projects
- Thursday/Friday: round meeting
  - update on projects + sharing of experience

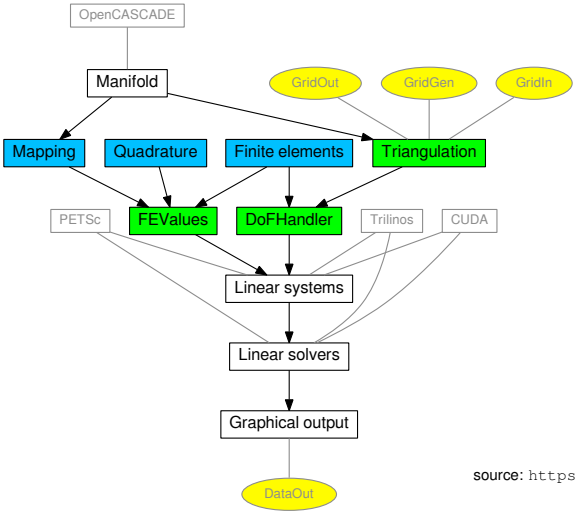
... dedicated discussion rounds?

Part 2:

Current status of simplex support in deal.II[1]

[1] with contributions, i.a., by Wolfgang Bangerth, Elias Dejene, Marc Fehling, Katharina Kormann, Martin Kronbichler, Pasquale Africa, Timo Heister, Peter Munch, Daniel Pauckner, Magdalena Schreter, David Wells

Simplex/mixed meshes and the deal.II main modules



cell-type specific classes (new classes)

input/output

user interaction with existing modules

source: `https://www.dealii.org/developer/doxygen/deal.II/index.html`

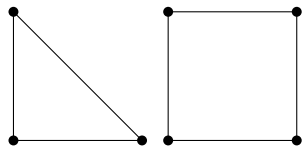
Configuration

Simplex support has to be explicitly enabled during configuration with:

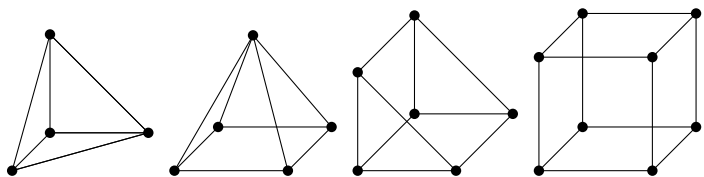
```
cmake ...  -D DEAL_II_WITH_SIMPLEX_SUPPORT="ON" ...
```

Reference-cell types

- reference-cell types in 0D: vertex
- reference-cell types in 1D: line
- reference-cell types in 2D: triangle, quadrilateral



- reference-cell types in 3D: tetrahedron, pyramid, wedge/prism, hexahedron



Finite element/quadrature/mapping

Supported finite elements[2]:

- `Simplex::FE_P`: scalar Lagrange finite element $\mathcal{P}_p$ that yields the finite-element space of continuous, piecewise polynomials of degree $p \leq 2$:

```
Simplex::FE_P<2> fe(p);
```

- `Simplex::FE_DGP`: scalar Lagrange finite element $\mathcal{P}_p$ that yields the finite-element space of discontinuous, piecewise polynomials of degree $p \leq 2$
- `FESystem`: to construct new vector-valued simplex elements

[2] see also: `Simplex::FE_PyramidP/DGP` and `Simplex::FE_WedgeP/DGP`

Finite element/quadrature/mapping (cont.)

Supported quadrature rules[3]:

- `QDuffy`: Duffy transformation from a square to a triangle to integrate singularities in the origin of the reference simplex
- `Simplex::QGauss`: Gauss-like quadrature rules with:

```
Simplex::QGauss<2> quadrature(p);
```

... with $p \leq 3$ as an indication of what polynomial degree to be integrated exactly

[3] see also: `Simplex::QGauss_Pyramid` and `Simplex::QGauss_Wedge`

Finite element/quadrature/mapping (cont.)

Supported mapping:

- linear mapping via `MappingFE`:

```
MappingFE<2> mapping(Simplex::FE_P<2>(p));
```

... with $p = 1$ (linear mapping)

Triangulation & GridGenerator/GridIn/GridOut/DataOut

Supported triangulation classes[4]:

- `Triangulation`: serial
- `parallel::shared/fullydistributed::Triangulation`: parallel

... (global) refinement for 2D!

[4] The class `parallel::distributed::Triangulation` will not support simplex meshes!

Triangulation & GridGenerator/GrindIn/GridOut/DataOut (cont.)

processing of a grid:

- `GridGenerator::convert_hypercube_to_simplex_mesh()`
- vtk: `GridIn::read_vtk()`, `GridOut::write_vtk()`
- msh: `GridIn::read_msh()` ▷ *Gmsh*
- ExodusII: `GridIn::read_exodusii()`

visualization of results:

- vtk: `DataOut::write_vtk()` → 1/2 subdivisions, no high-order Lagrange
- vtu: `DataOut::write_vtu()` and `::write_vtu_with_pvtu_record()`
- hdf5: `DataOut::write_hdf5_parallel()`

Implications for usage

RECOMMENDATION: do not use `GeometryInfo`!

The class `TriaAccessor` has been extended with new methods. E.g.:

```
for(const auto & cell : tria)
  for(const auto v : GeometryInfo<dim>::face_indices())
    // do something
```

old

```
for(const auto & cell : tria)
  for(const auto v : cell->face_indices())
    // do something
```

new

... to query the number of subentities and to iterate over them.

Last resort: to get the type of cells (`ReferenceCell::Type`) via:

```
if(cell->reference_cell_type()==ReferenceCell::Type::Hex)
  // do something
```

... and `ReferenceCell::internal::Info`.

Implications for usage (cont.)

WARNING: you cannot rely on default mappings! Be explicit! E.g.:

```
FEValues::FEValues(fe, quad, flags);
```

```
VectorTools::interpolate(dof, function, vec);
```

```
DataOut::build_patches(subdivisions);
```

implicit (MappingQGeneric(1))

```
FEValues::FEValues(mapping, fe, quad, flags);
```

```
VectorTools::interpolate(mapping, dof, function, vec);
```

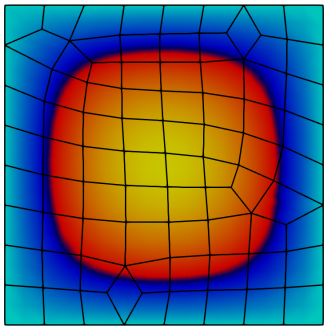
```
DataOut::build_patches(mapping, subdivisions);
```

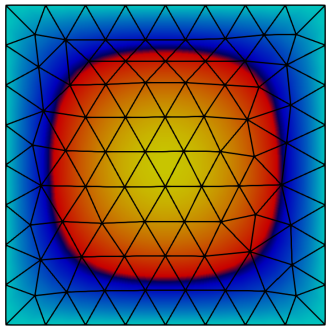
explicit

... there are many functions with default argument mapping!!!

Example

Solution of the Poisson problem on a pure quadrilateral and a pure triangle “Gmsh” mesh:





Gmsh journals:

```
Rectangle(1) = {0, 0, 0, 1, 1, 0};
Recombine Surface{1};
Mesh 2;
Save "box_2D_quad.msh";
```

```
Rectangle(1) = {0, 0, 0, 1, 1, 0};
Mesh 2;
Save "box_2D_tri.msh";
```

... online as example-1 `https://github.com/peterrum/dealii-simplex-workshop-2021`

Mixed mesh: an approach via hp framework

- assign each reference-cell type a finite element:

```
hp::FECollection<2> fes(Simplex::FE_P<2>(2), FE_Q<2>(2));
```

- assign each cell an `active_fe_index`:

```
DoFHandler<2> dof_handler(tria);

for (const auto &cell : dof_handler.active_cell_iterators())
  cell->set_active_fe_index(cell->reference_cell_type() == ReferenceCell::Type::Tri ? 0 : 1);

dof_handler.distribute_dofs (fes);
```

- continue as usual (as if "`hp::DoFHandler`" would be used):

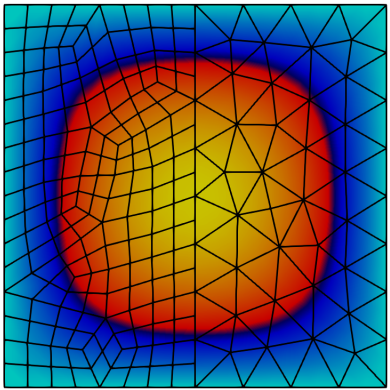
```
hp::MappingCollection<2> mappings(MappingFE<2>(Simplex::FE_P<2>(1)),  MappingFE<dim>(FE_Q<2>(1)));
hp::QCollection<2>       quads(Simplex::PGauss<2>(3), QGauss<2>(3));

hp::FEValues<2> hp_fe_values(mappings, fes, quads, flag);

for (const auto &cell : dof_handler)
  {
    hp_fe_values.reinit(cell);
    const auto &fe_values = hp_fe_values.get_present_fe_values();
    /*... do something useful with fe_values ...*/
  }
```

Mixed mesh: example

Solution of the Poisson problem on a mixed “Gmsh” mesh[5]:



Gmsh journal:

```
Rectangle(1) = {0, 0, 0, 0.5, 1, 0};
Rectangle(2) = {0.5, 0, 0, 0.5, 1, 0};
Recombine Surface{1};
Physical Surface("All") = {1, 2};
Mesh 2;
Coherence Mesh;
Save "box_2D_mixed.msh";
```

... online as example-2 `https://github.com/peterrum/dealii-simplex-workshop-2021`

[5] ”Hanging nodes” are only an artifact of DataOut.

Matrix-free support

The matrix-free framework has been extended. Modifications from user side:

- setup as usual, but passing a d-dimensional quadrature rule:

```
Simplex::QGauss<dim> quad(degree + 1);
matrix_free.reinit(mapping, dof_handler, constraints, quad, additional_data);
```

- integrals as usual, but using non-templated versions of `FEEvaluation`:

```
matrix_free.template cell_loop<VectorType, VectorType>(
  [](const auto &matrix_free, auto &dst, const auto &src, const auto cell_range) {
    FEEvaluation<dim, -1, 0, 1, double> phi(matrix_free, cell_range);
    for (unsigned int cell_range = cells.first; cell_range < cells.second; ++cell_range)
      {
        // ... as usual
      },
  }, dst, src, true);
```

... information regarding active DoF index[6] is extracted form new argument

- see also example-1 and example-2: no difference in the mixed-mesh case!

[6] and face type; see: #11302 and #11416

Summary and upcoming features

Summary: draft of the chapter in the release paper

`https://github.com/peterrum/release-papers/blob/simplex/9.3/paper.tex`

Upcoming features:

- (global) refinement for simplex meshes in 3D #11490
- h- and p-multigrid (via new global-coarsening infrastructure) #11429
- full support for mixed meshes and DG within MatrixFree #11372 #11387 #11387 #11401 #11416

Part 3:

Overview (cont.)

Goal: *"The best of both worlds"* (cont.)

Goals of this workshop:

- to present the current status of simplex support in deal.II (work of the last 9 months)
- hands-on: using the new infrastructure
- identifying bugs and work packages
- improving documentation
- implementing new simplex features

... the perspective of non-developers is needed!

Organization: projects

Some interesting topics:

- writing of ”Step 76: a simplex tutorial”
- implementation of high-order elements
- implementation of periodic boundaries
- validation + profiling
- development of a workflow for distributed and mixed meshes

... any help is welcome!

Organization: projects (cont.)

Converting of existing tutorials to tests (see also `tests/simplex`):

▷ see also: `https://github.com/dealii/dealii/tree/master/tests/simplex`

dealii-simplex-workshop-2021

File Edit View Insert Format Data Tools Add-ons Help Last edit was 2 minutes ago

	A	B	C	D
1		**Ported**	**Ported by**	**Issues**
2	**step-1**	x	Elias Dejene (PR)	
3	**step-2**	x	Peter Munch	GridGenerator::hyper_shell() not working, no hanging-node constraints
4	**step-3**	x	Elias Dejene (PR)	
5	**step-4**			
6	**step-5**			
7	**step-6**			
8	**step-7**			
9	**step-8**			
10	**step-9**			
11	**step-10**			
12	**step-11**			
13	**step-12**	x	Peter Munch	
14	**step-13**			
15	**step-14**			

Registration sheet | Tutorials

... one tutorial per participant would be a great help!

Questions?

Following:

- presentation of gained user experience by Pasquale Africa
- introduction of participants and presentation of projects