IFEM - introduction and overview

Arne Morten Kvarving*, Yared Bekele, Eivind Fonn, Runar Holdahl, Kjetil Johannessen, Knut Morten Okstad, Timo van Opstal

Department of Applied Mathematics
SINTEF ICT

OPM user meeting, Oslo, June 1-2, 2016

Why is IFEM relevant at OUM16?

- (Parts of) IFEM has been put in repositories under the OPM organization on github.
- Currently 4 modules:
  - **IFEM:** Main library.
  - **IFEM-Elasticity:** Linear-elastic application and library.
  - **IFEM-PoroElasticity:** Poro-elastic application and library.
  - **IFEM-OpenFrac:** Fracture dynamics application.

What is IFEM?

- Finite element based framework for solving PDEs.

$$-\nabla^2 u = f \text{ in } \Omega$$
$$u|_{\delta\Omega} = 0$$

$$\downarrow$$

Find u in $H_0^1(\Omega)$ such that

$$\int_\Omega \nabla u \cdot \nabla v \, \mathrm{d}\Omega = \int_\Omega f v \, \mathrm{d}\Omega \; \forall v \text{ in } H_0^1(\Omega)$$

$$\downarrow$$

$$\underline{\mathbf{A}x} = \underline{b}.$$

What is IFEM?

- Standard FEM:
  - Ω approximated using tetrahedral or quadrilateral elements.
  - Polynomial subspaces on the elements.
  - C^0 continuity across element interfaces.
  - Easy access to h refinement.
- IFEM: Isogeometric analysis
  - Ω defined through splines.
  - Quadrilateral elements (knotspans).
  - Any (possibly varying) continuity across element interfaces.
  - Easy access to h refinement (knot insertion).
  - Easy access to p refinement (order elevation).
  - Easy access to k refinement (knot insertion and order elevation not commutative).

What is IFEM?

- Standard FEM design loop:
  - Design in a CAD tool (using splines).
  - Approximate CAD geometry as a mesh.
  - Apply finite element solver.
  - Analyze results and modify CAD model.
  - Rinse, repeat.
- Isogeometric FEM design loop:
  - Design in a CAD tool (using splines).
  - Apply finite element solver.
  - Analyze results.
  - Rinse, repeat.

What is IFEM?

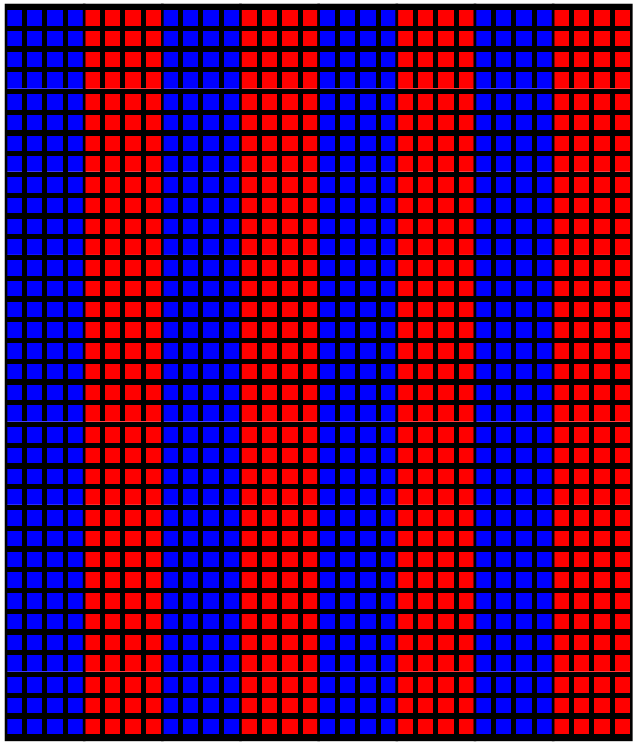
- Standard FEM design loop:
  - Design in a CAD tool (using splines).
  - Approximate CAD geometry as a mesh.
  - Apply finite element solver.
  - Analyze results and modify CAD model.
  - Rinse, repeat.
- Isogeometric FEM design loop:
  - Design in a CAD tool (using splines).
  - Apply finite element solver.
  - Analyze results.
  - Rinse, repeat.

What is IFEM?

- Standard FEM design loop:
  - Design in a CAD tool (using splines).
  - Approximate CAD geometry as a mesh.
  - Apply finite element solver.
  - Analyze results and modify CAD model.
  - Rinse, repeat.
- Isogeometric FEM design loop:
  - Design in a CAD tool (using splines).
  - Apply finite element solver.
  - Analyze results.
  - Rinse, repeat.

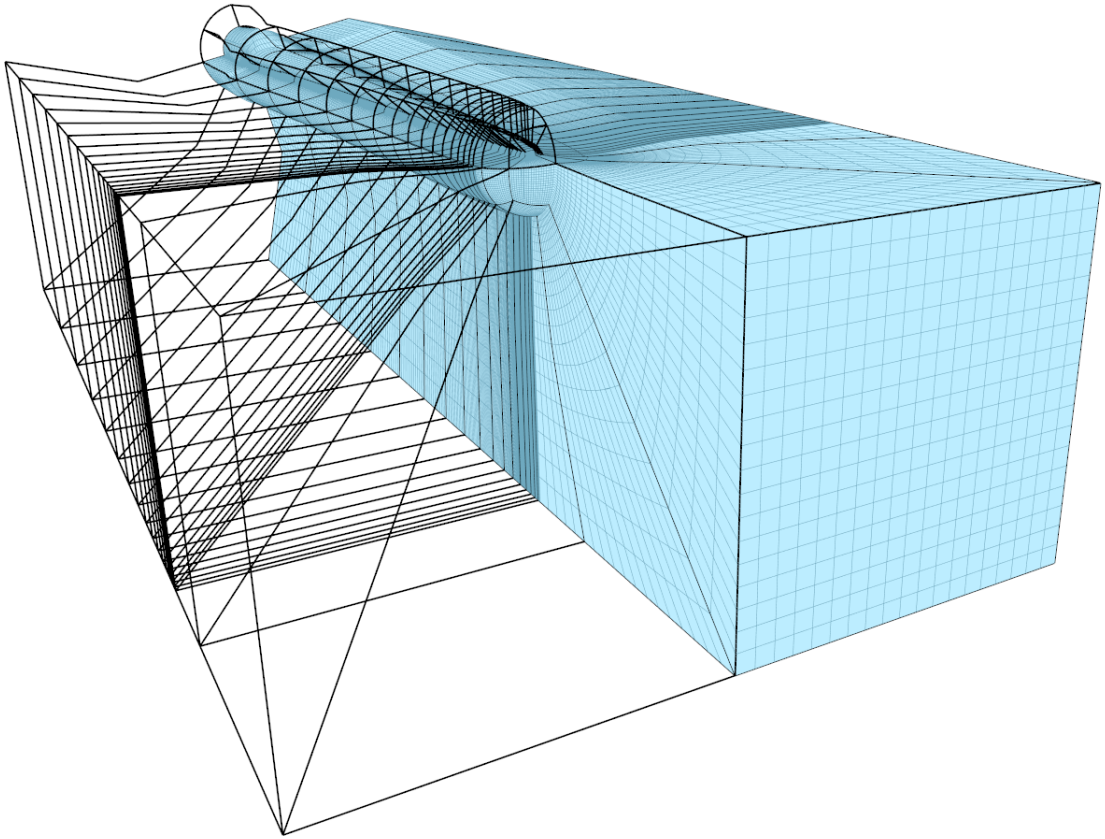
Parallelization.

- IFEM is a structured code which grants easy access to parallelization.
- Mesh coloring used within each patch to enable multi-threaded assembly.



Parallelization

- Domain decomposition based on spline patches to enable multi-process parallelization.



- Parallel linear solvers supplied by PETSc or dune-istl (WIP).

Mixed problems

- IFEM support any number of bases and thus mixed formulations.
  - Full continuity mixed $\left(S^{p}_{p-1}, S^{p-1}_{p-2}\right)$.
  - Taylor-Hood $\left(S^{p}_{p-1}, S^{p-1}_{0}\right)$.
  - Raviart-Thomas $\left(S^{p,p-1}_{p-1,p-2}, S^{p-1,p}_{p-2,p-1}, S^{p-1,p-1}_{p-2,p-2}\right)$.
- Block matrices for multi-physics and saddle point problem preconditioning based on approximated Schur-complements;

$$P\left(\begin{bmatrix} A & B \\ C & D \end{bmatrix}\right)^{-1} = \begin{bmatrix} P\left(A\right)^{-1} & B \\ 0 & P\left(\tilde{S}(A)\right)^{-1} \end{bmatrix}$$

Adaptivity

- Adaptive simulations are supported through the use of locally-refined B-splines (LRB-splines, available at VikingScientist/LRsplines).
- Norm-based adaptivity (guided by an exact solution).
- Recovery-based adaptivity (super-convergent patch recovery).
- Goal-oriented adaptivity (fracture dynamics).
- Immersed boundaries available for simple hole geometries ('quadrature adaptivity').
- Adaptive time-step control through CFL, cutback, embedded schemes, ...

Testing

- All applications have regression tests. Output comparison using grep.
- Unit tests based on gtest for library and applications.
- Commit-by-commit testing built into the build system (like in the rest of OPM).
- Pre-merge testing with downstream support through github + jenkins triggers (like in the rest of OPM).
- Automatic post-commit testing.
- Tests run in 3 configurations: gcc, clang and gcc+mpi.
- Difference from rest of OPM: tests are not built by default, need to use 'check' target.

Build system

- Build system is based on CMake.
- Operates slightly different than the rest of OPM.
- Main library built separately.
- Applications use sibling-directory logic to locate dependencies.
- Dependencies not using separate build steps but required bits built in application build system.

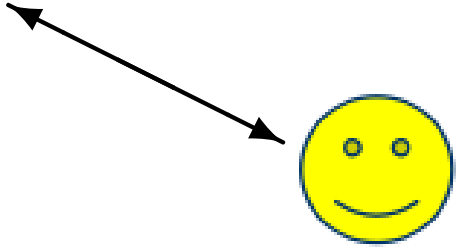
Build system

```
git clone https://github.com/OPM/IFEM
cd IFEM; mkdir build; cd build
cmake ..&&  make && make check && make install
cd /some/where/else
mkdir PoroElasticity
cd PoroElasticity
git clone https://github.com/OPM/IFEM-Elasticity
git clone https://github.com/OPM/IFEM-PoroElasticity
cd IFEM-PoroElasticity; mkdir build; cd build
cmake .. && make && make check
```

IFEM module overview

GEOM – Geometry Modeller

- Edit g2-files (GoTools) by hand
- Using Rhino and small C++ tools
- Python-based modeling tool



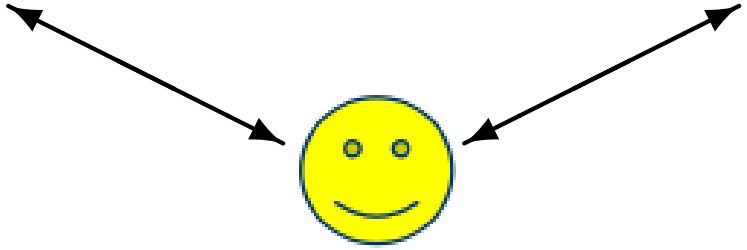
IFEM module overview

GEOM – Geometry Modeller

- Edit g2-files (GoTools) by hand
- Using Rhino and small C++ tools
- Python-based modeling tool

GPM – Grid/Property Modeller

- Automatic global node numbering for multi-patch models
- Interactive assignment of property codes to topological entities



IFEM module overview

GEOM – Geometry Modeller

- Edit g2-files (GoTools) by hand
- Using Rhino and small C++ tools
- Python-based modeling tool

GPM – Grid/Property Modeller

- Automatic global node numbering for multi-patch models
- Interactive assignment of property codes to topological entities

user

SIM – Numerical Simulation

- Object-oriented framework for Isogeometric Finite Element Analysis
- The user has to program his/her own application
- A few sample applications are provided (Poisson, Linear elasticity)

IFEM module overview

GEOM – Geometry Modeller

- Edit g2-files (GoTools) by hand
- Using Rhino and small C++ tools
- Python-based modeling tool

GPM – Grid/Property Modeller

- Automatic global node numbering for multi-patch models
- Interactive assignment of property codes to topological entities

RESV – Result Visualization

- Currently, GLview Inova (Ceetron) is used based on VTF file format
- HDF5 file format is used for result storage and restart, from which converters to any preferred visualization tool can be made

SIM – Numerical Simulation

- Object-oriented framework for Isogeometric Finite Element Analysis
- The user has to program his/her own application
- A few sample applications are provided (Poisson, Linear elasticity)

Major class hierarchies of the SIMulation environment

1. SystemMatrix/Vector - Linear algebra system level
   - Interface to equation solvers (direct/iterative, serial/parallel)
   - Sub-classes for various linear equation solver packages

Major class hierarchies of the SIMulation environment

1. SystemMatrix/Vector - Linear algebra system level
   - Interface to equation solvers (direct/iterative, serial/parallel)
   - Sub-classes for various linear equation solver packages
2. ASMbase - Block/patch level drivers
   - Administers the element loop and numerical integration loop within a block (spline patch)
   - Sub-classes depending on discretization (Splines/NURBS, Lagrange, Spectral)

Major class hierarchies of the SIMulation environment

1. SystemMatrix/Vector - Linear algebra system level
   - Interface to equation solvers (direct/iterative, serial/parallel)
   - Sub-classes for various linear equation solver packages
2. ASMbase - Block/patch level drivers
   - Administers the element loop and numerical integration loop within a block (spline patch)
   - Sub-classes depending on discretization (Splines/NURBS, Lagrange, Spectral)
3. SIMbase - System level drivers
   - Administers an assembly of spline patches (blocks)
   - Sub-classes for problem-specific input and setup

Major class hierarchies of the SIMulation environment

1. SystemMatrix/Vector - Linear algebra system level
   - Interface to equation solvers (direct/iterative, serial/parallel)
   - Sub-classes for various linear equation solver packages
2. ASMbase - Block/patch level drivers
   - Administers the element loop and numerical integration loop within a block (spline patch)
   - Sub-classes depending on discretization (Splines/NURBS, Lagrange, Spectral)
3. SIMbase - System level drivers
   - Administers an assembly of spline patches (blocks)
   - Sub-classes for problem-specific input and setup
4. SIMFoo - Application simulation driver
   - Administers time/load step loop of the solution algorithm
   - Possibly encapsulates algorithms like a Newton iteration loop.

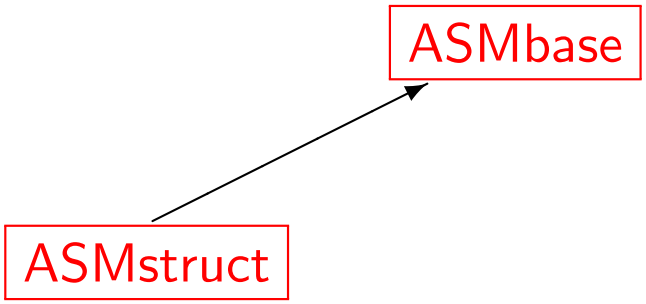
Major class hierarchies of the SIMulation environment

1. SystemMatrix/Vector - Linear algebra system level
   - Interface to equation solvers (direct/iterative, serial/parallel)
   - Sub-classes for various linear equation solver packages
2. ASMbase - Block/patch level drivers
   - Administers the element loop and numerical integration loop within a block (spline patch)
   - Sub-classes depending on discretization (Splines/NURBS, Lagrange, Spectral)
3. SIMbase - System level drivers
   - Administers an assembly of spline patches (blocks)
   - Sub-classes for problem-specific input and setup
4. SIMFoo - Application simulation driver
   - Administers time/load step loop of the solution algorithm
   - Possibly encapsulates algorithms like a Newton iteration loop.
5. Integrand - Integration point level
   - Administers the problem-dependent calculations at an integration point (interior and boundary integrals)
   - Problem-specific sub-classes

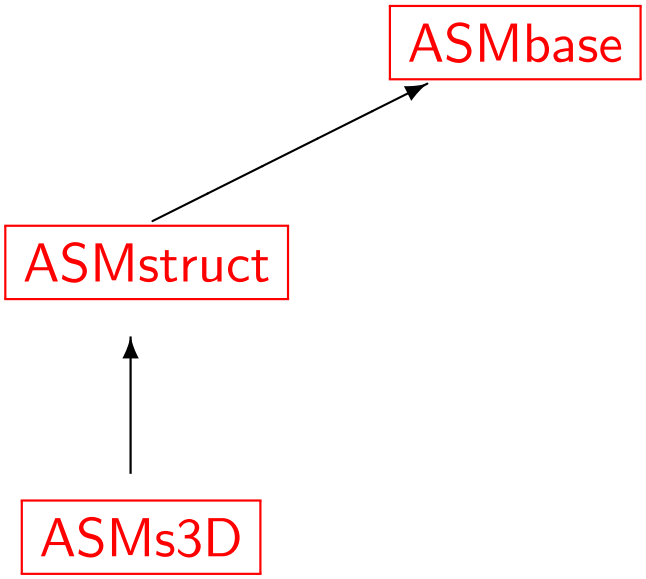
ASM class hierarchy - *the Isogeometry level*

ASMbase

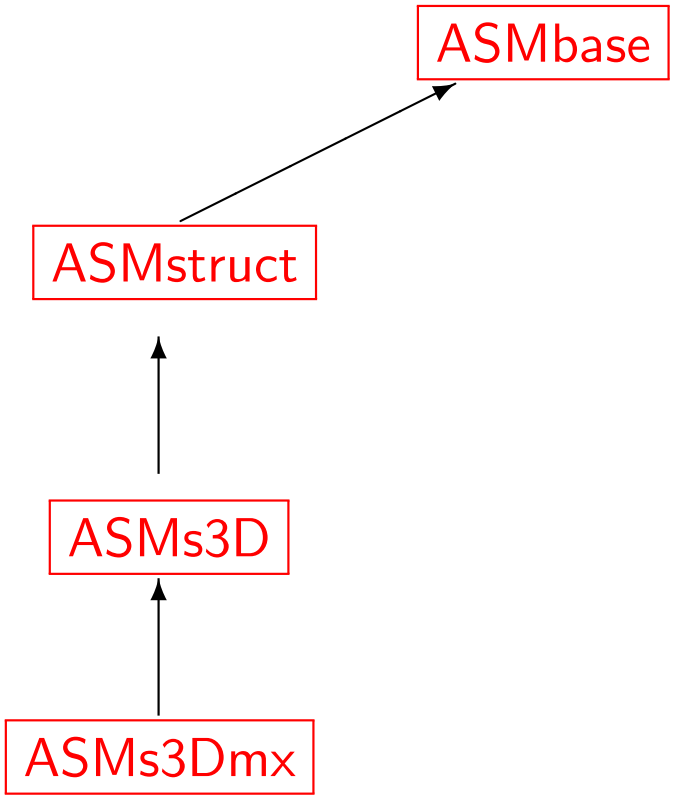
ASM class hierarchy - *the Isogeometry level*



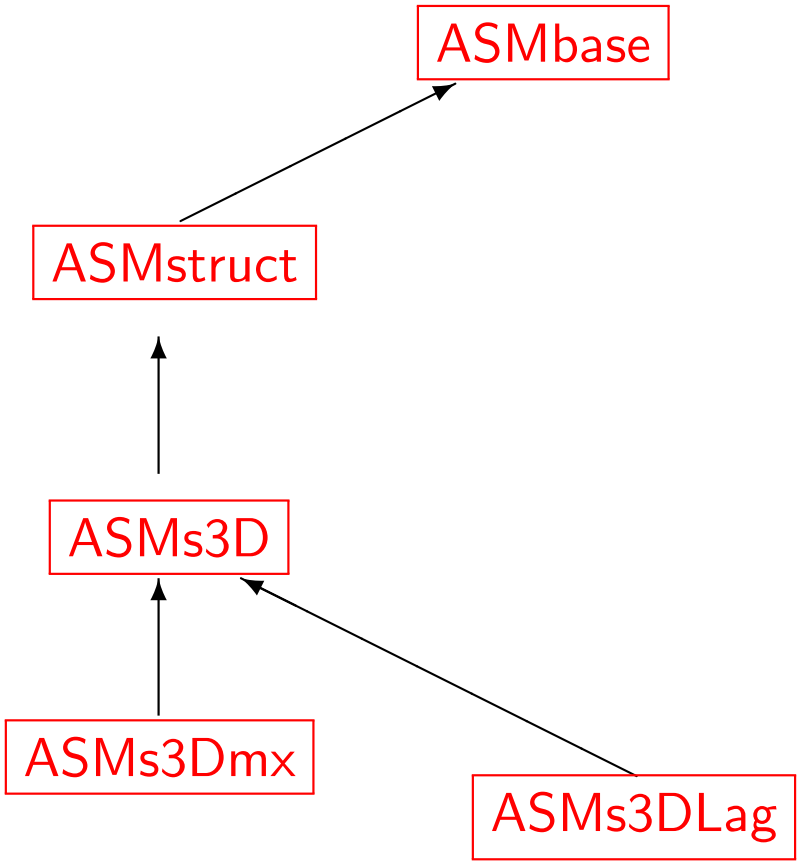
ASM class hierarchy - *the Isogeometry level*



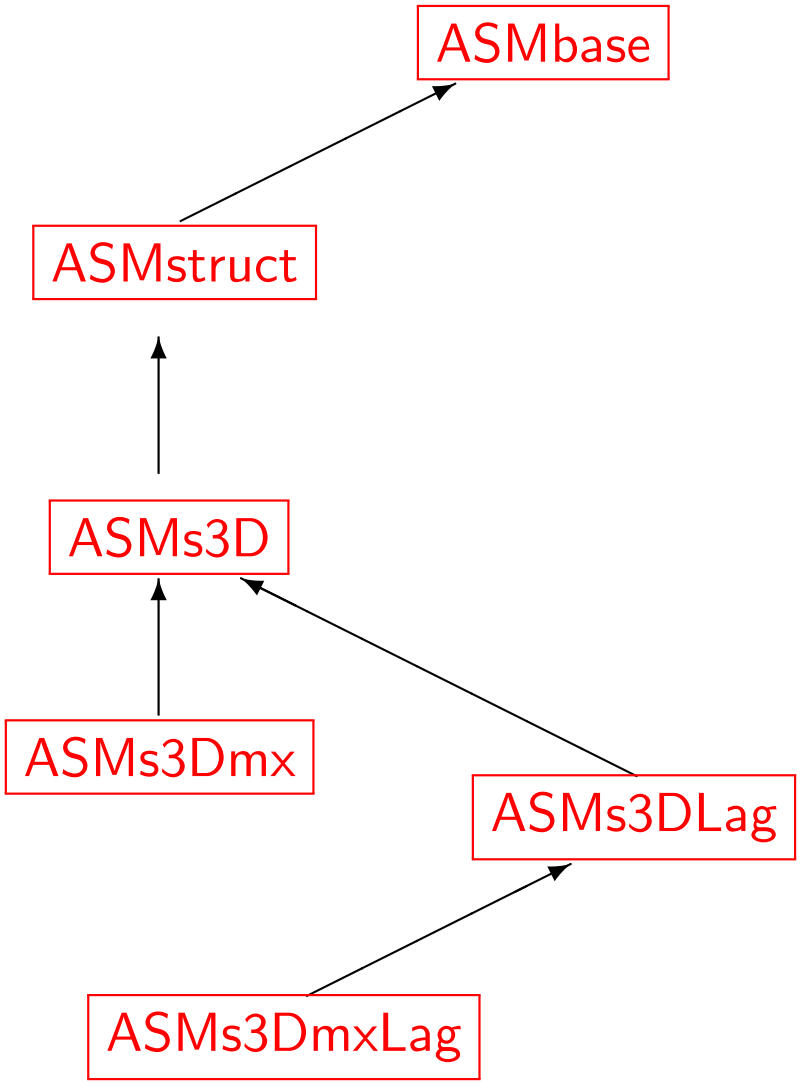
ASM class hierarchy - *the Isogeometry level*



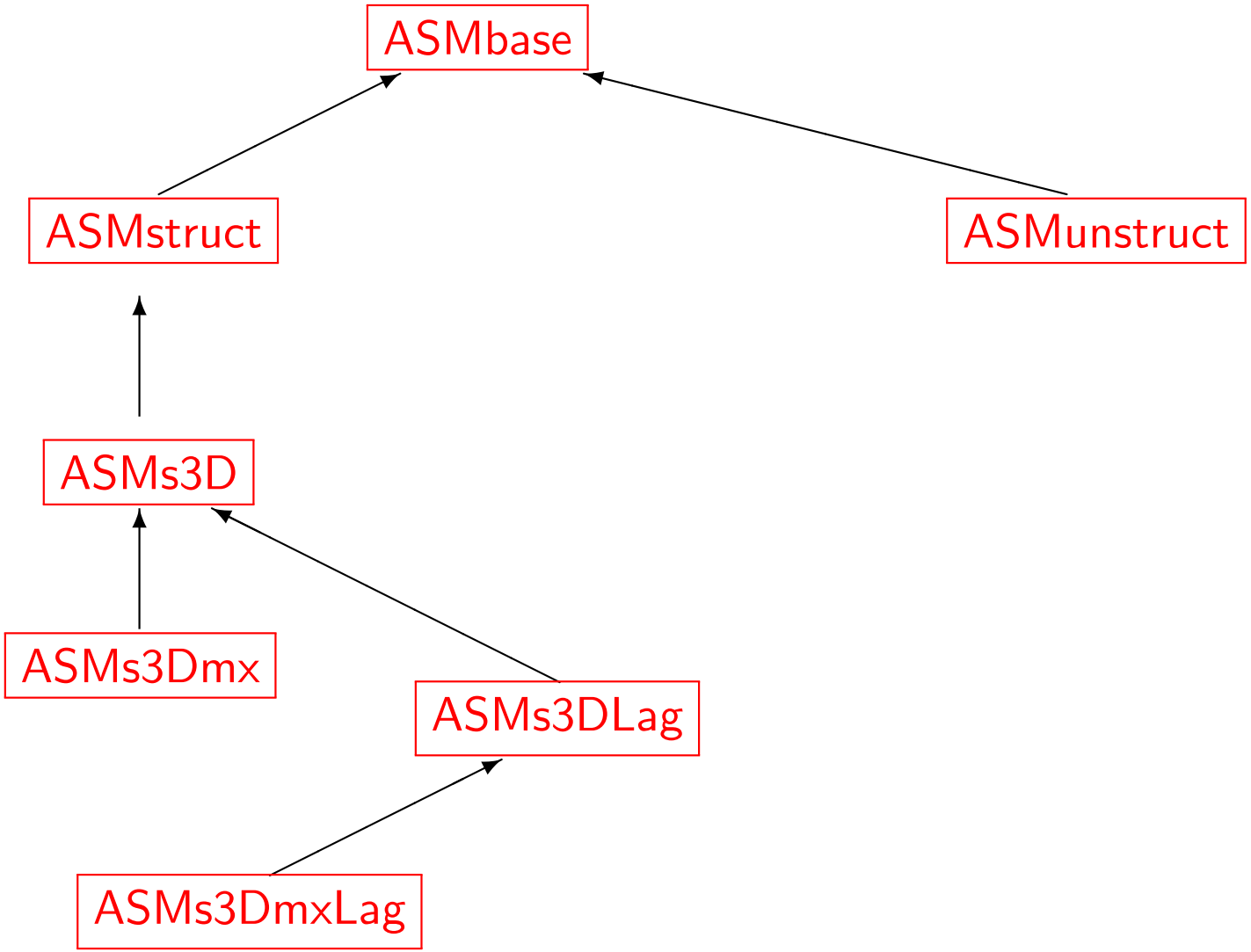
ASM class hierarchy - *the Isogeometry level*



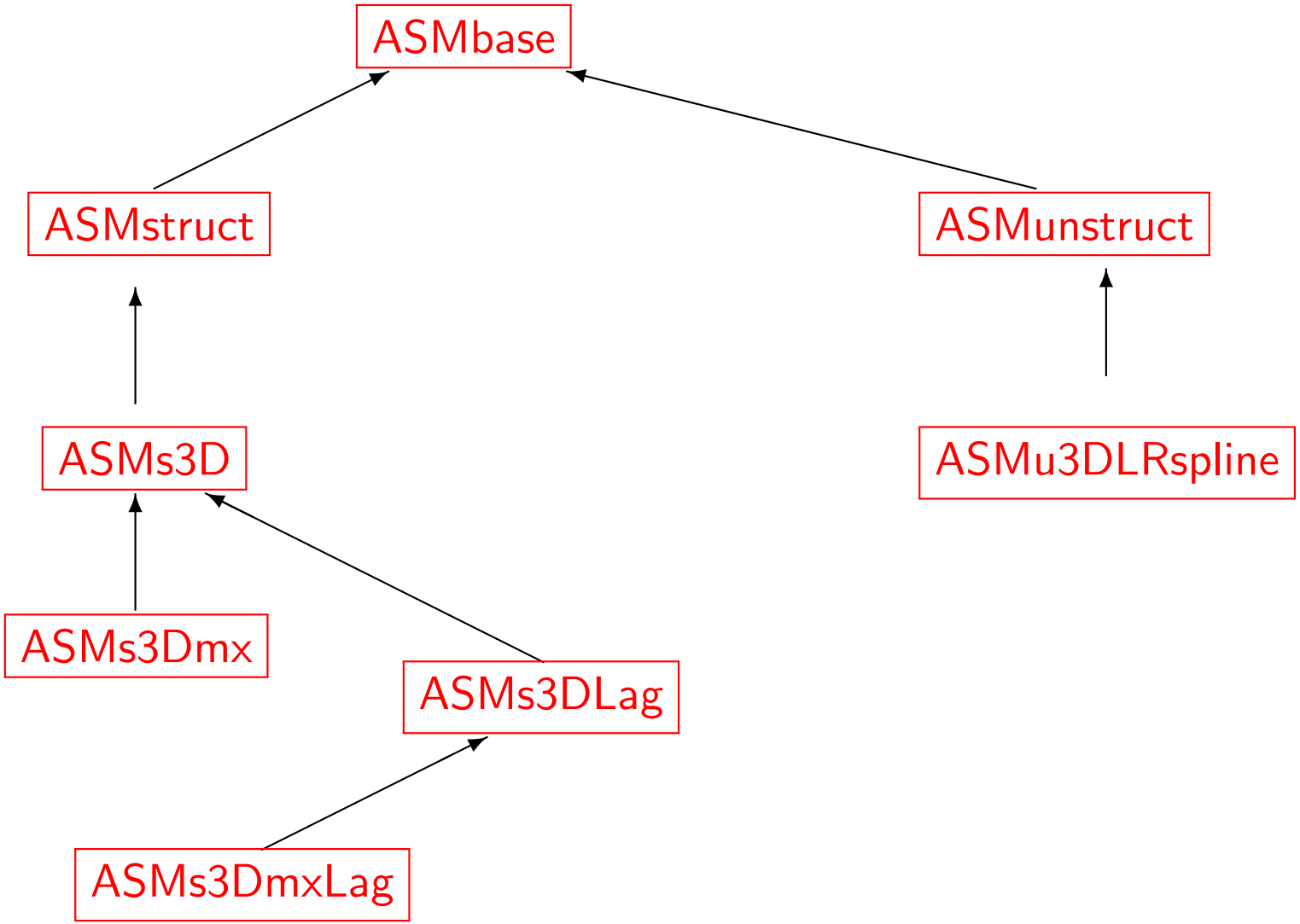
ASM class hierarchy - *the Isogeometry level*



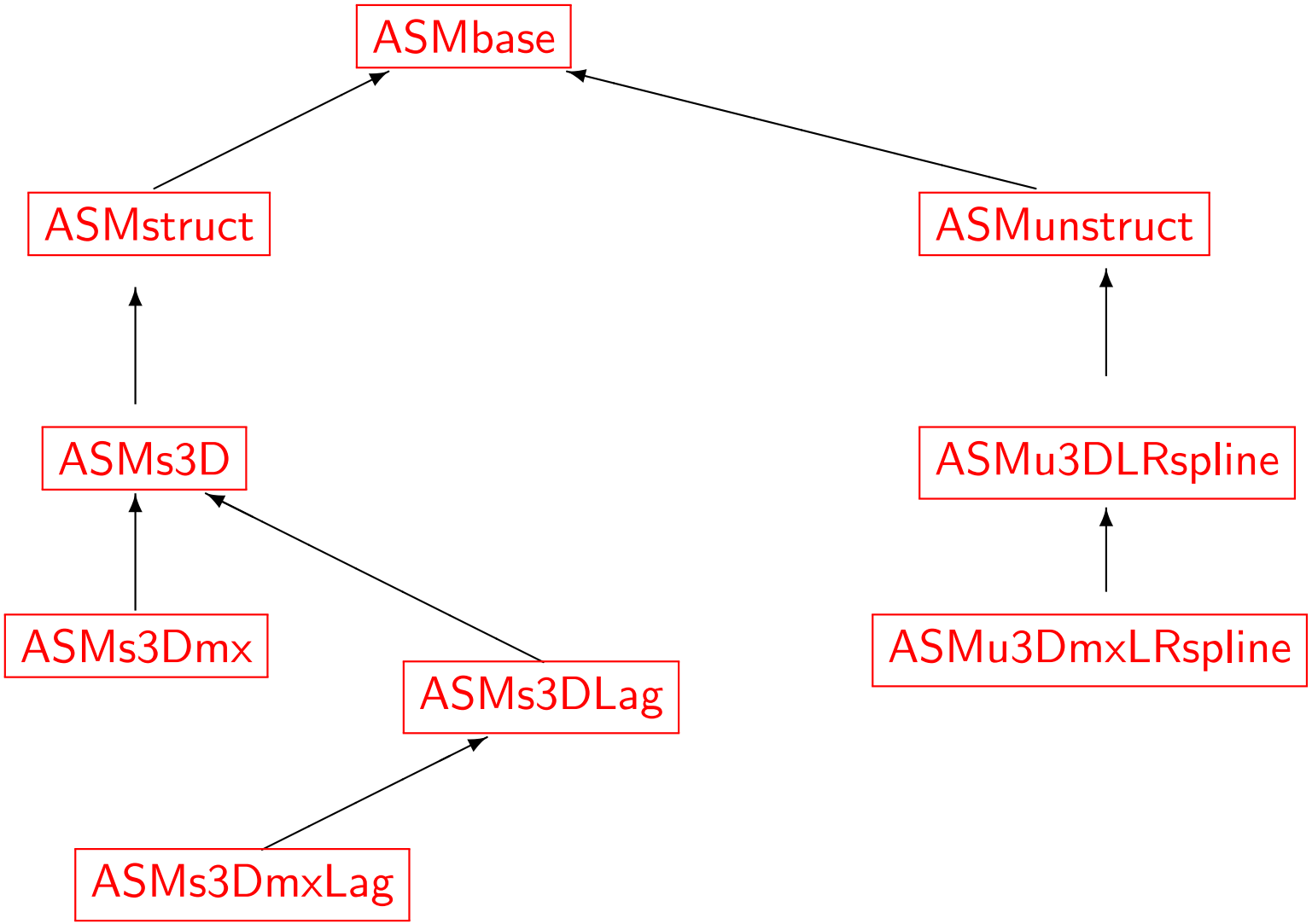
ASM class hierarchy - *the Isogeometry level*



ASM class hierarchy - *the Isogeometry level*



ASM class hierarchy - *the Isogeometry level*



Applications

- Heavy use of generic templates.
  - Avoid spaghetti code such as

```
if (dim == 1)
  // 1D specific code
else if (dim == 2)
  // 2D specific code
else if (dim == 3)
  // 3D specific code
```

using *template specialization* and *templated inheritance*.

Templated inheritance

```
template<class Dim>
class SIMLineEL : public Dim {
  <snip>
  virtual bool parse(const TiXmlElement* elem)
  {
    if (!lineltag(elem))
      return Dim::parse(elem);
    if (isdimspecific(elem))
      return parsedimspecific(elem);
    <general linel parsing code>
  }
};
```

Template specialization

```
template<>
bool SIMLinEl<SIM1D>::parsedimspecific(..);

template<>
bool SIMLinEl<SIM2D>::parsedimspecific(..);

template<>
bool SIMLinEl<SIM3D>::parsedimspecific(..);
```

Applications

- Heavy use of generic templates.
  - Maximum code reuse.

```
template<class Driver> int setupSIM() {
  // instance application driver
  Driver driver;
  // configuration properties for SIMFoo
  typename Driver::SetupProps props;
  // configure driver
  SIM::Configure(driver, props);
  // instance solver object
  SIMSolver<Driver> solver;
  // read time stepping parameters
  solver.read(inputfile);
  // handle application restart
  if (restart)
    SIM::handleRestart(solver, ...);
  // setup data output
  if (dataoutput)
    SIM::handleDataOutput(solver, exporter, ..);

  // solve problem and handle data output
  return solver.solveProblem(exporter);
}
```

Applications

- Heavy use of generic templates.
  - Keep coupling between modules weak to maximize reuse.

```
/*!
  \brief Driver class for thermo-elastic simulators.
  \details A thermo-elastic simulator is a coupling between
  a thermal solver and an elasticity solver.
*/

template<class TempSolver, class SolidSolver,
         template<class T1, class T2> class Coupling=SIMCoupled>
class SIMThermoElasticity : public Coupling<TempSolver,SolidSolver>
{
  (...)
  //! \brief Initializes and sets up field dependencies.
  virtual void setupDependencies()
  {
    this->S2.registerDependency(&this->S1, "temperature1",
                                1, this->S1.getFEModel(), 1);
    if (m_twoway) {
      this->S1.registerDependency(&this->S2, "displacement1",
                                  this->S1.dimension,
                                  this->S2.getFEModel(), 2);
      this->S1.registerDependency(&this->S2, "pressure1", 1,
                                  this->S2.getFEModel(), 2);
    }
  }
(...)
```

Applications

- Heavy use of generic templates.
  - A thermo-linearelastic simulator.

```
// A 2D heat transfer for thermo-linearelastic simulator
typedef SIMHeatEquation<SIM2D, HeatEquation> SIMTE;
// A 2D segregated thermo-linearelastic simulator
typedef SIMThermoElasticity<SIMTE, SIMLinearElasticity<SIM2D>> SIMTEL;
```

$$\nabla \cdot \left(\frac{\mathbf{D}}{2} \left(\nabla \mathbf{u} + \nabla^T \mathbf{u} \right) - \mathbf{D} \alpha \left(T \right) \left(T - T_0 \right) \delta_{ij} \right) = 0$$

$$\rho c \frac{\partial T}{\partial t} - \nabla \cdot \kappa \nabla T = f(\mathbf{u}, T)$$

+ appropriate BCs.

Applications

- Heavy use of generic templates.
  - A segregated thermo-poroelastic simulator.

```
// A 2D heat transfer for thermo-poroelastic simulator
typedef SIMHeatEquation<SIM2D, PoroHeatTransfer> SIMTPE;
// A 2D segregated thermo-poro-elastic simulator
typedef SIMThermoElasticity<SIMPTPE,
                            SIMPoroElasticity<SIM2D>> SIMTEP;
// Time stepping loop for solver
typedef SIMSolver<SIMTEP> TimeStepLoop;
```

$$\nabla \cdot [\boldsymbol{\sigma}' - \alpha p^w \mathbf{I}] + \rho \mathbf{b} = \mathbf{0}$$

$$\alpha \nabla \cdot \mathbf{v}^s + \nabla \cdot \mathbf{w} + \frac{1}{M}\frac{\partial p^w}{\partial t} = 0$$

$$(\rho c)_{eff}\frac{\partial T}{\partial t} + \rho^f c^f \mathbf{w} \cdot \nabla T - \nabla \cdot \kappa \nabla T + \beta T_o \nabla \cdot \mathbf{v}^s = Q$$

+ appropriate BCs.

Applications

- Heavy use of generic templates.
  - A semi-implicitly coupled thermo-poroelastic simulator.

```
// A 2D heat transfer for thermo-poroelastic simulator
typedef SIMHeatEquation<SIM2D, PoroHeatTransfer> SIMTPE;
// A 2D semi-implicitly coupled thermo-poro-elastic simulator
typedef SIMThermoElasticity<SIMPTPE,
                            SIMPoroElasticity<SIM2D>,
                            SIMCoupledSI> SIMTEP;
// Time stepping loop for solver
typedef SIMSolver<SIMTEP> TimeStepLoop;
```

$$\nabla \cdot [\boldsymbol{\sigma}' - \alpha p^w \mathbf{I}] + \rho \mathbf{b} = \mathbf{0}$$

$$\alpha \nabla \cdot \mathbf{v}^s + \nabla \cdot \mathbf{w} + \frac{1}{M}\frac{\partial p^w}{\partial t} = 0$$

$$(\rho c)_{eff}\frac{\partial T}{\partial t} + \rho^f c^f \mathbf{w} \cdot \nabla T - \nabla \cdot \kappa \nabla T + \beta T_o \nabla \cdot \mathbf{v}^s = Q$$

+ appropriate BCs.

Applications

- Heavy use of generic templates.
  - A time-slab adaptive segregated thermo-poroelastic simulator.

```
// A 2D heat transfer for thermo-poroelastic simulator
typedef SIMHeatEquation<SIM2D, PoroHeatTransfer> SIMTPE;
// A 2D segregated thermo-poro-elastic simulator
typedef SIMThermoElasticity<SIMPTPE,
                            SIMPoroElasticity<SIM2D>> SIMTEP;
// A time-slab adaptive time stepping loop
typedef SIMSolverTS<SIMTEP> TimeStepLoop;
```

$$\nabla \cdot [\boldsymbol{\sigma}' - \alpha p^w \mathbf{I}] + \rho \mathbf{b} = \mathbf{0}$$

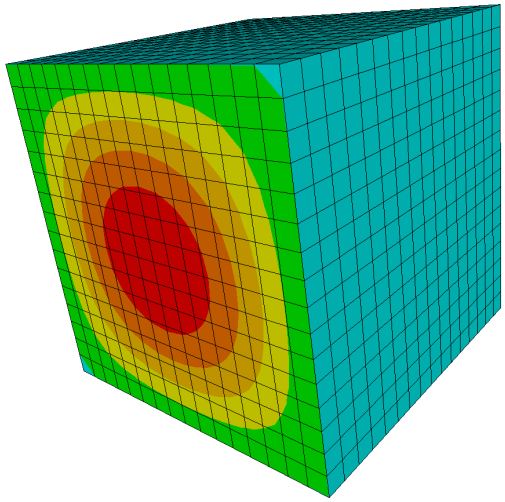
$$\alpha \nabla \cdot \mathbf{v}^s + \nabla \cdot \mathbf{w} + \frac{1}{M}\frac{\partial p^w}{\partial t} = 0$$

$$(\rho c)_{eff}\frac{\partial T}{\partial t} + \rho^f c^f \mathbf{w} \cdot \nabla T - \nabla \cdot \kappa \nabla T + \beta T_o \nabla \cdot \mathbf{v}^s = Q$$

+ appropriate BCs.

Applications built using IFEM

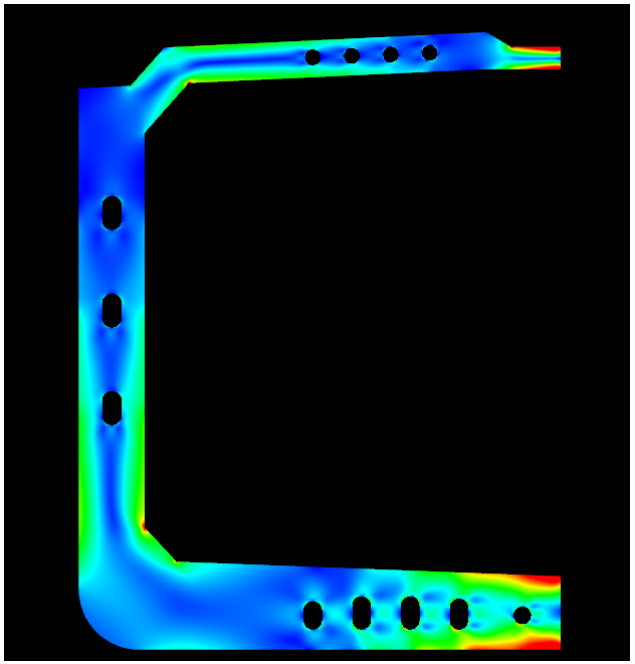
- Poisson of course.



- Not yet public.

Applications built using IFEM

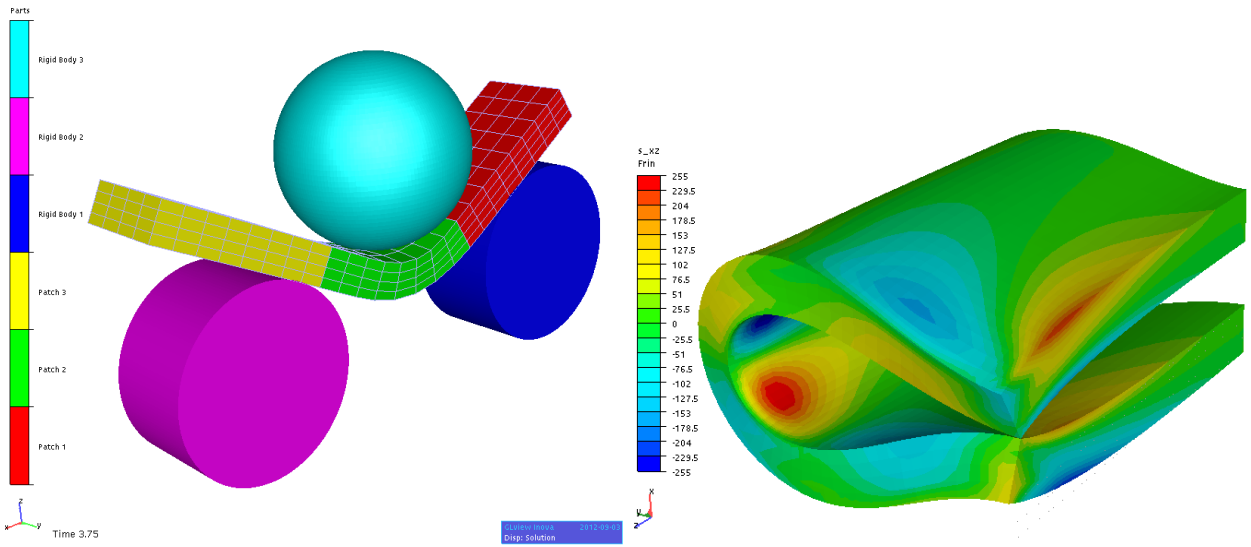
- Linear elasticity.



- Available at OPM/IFEM-Elasticity.

Applications built using IFEM

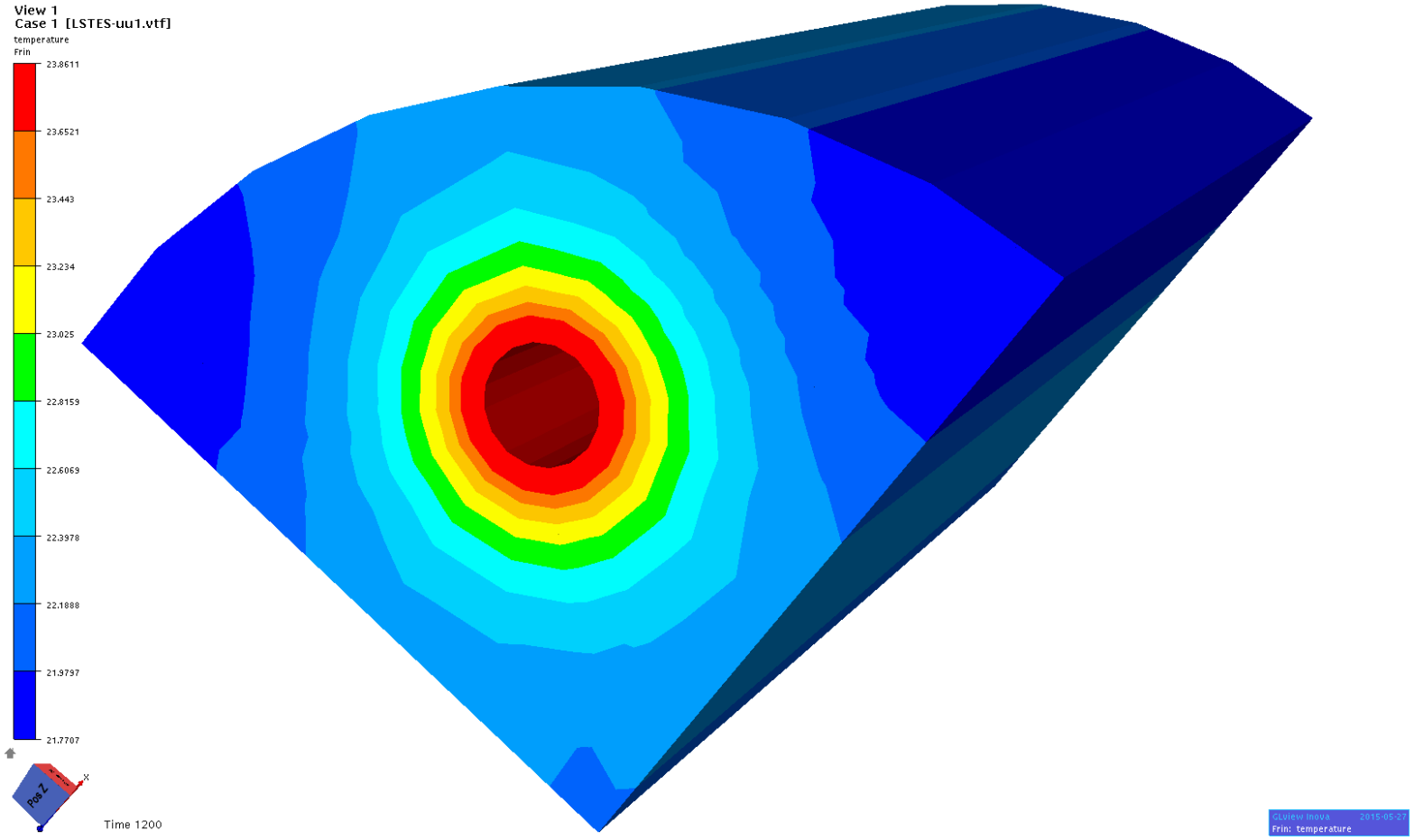
- Finite deformation elasticity.



- Not yet public.

Applications built using IFEM

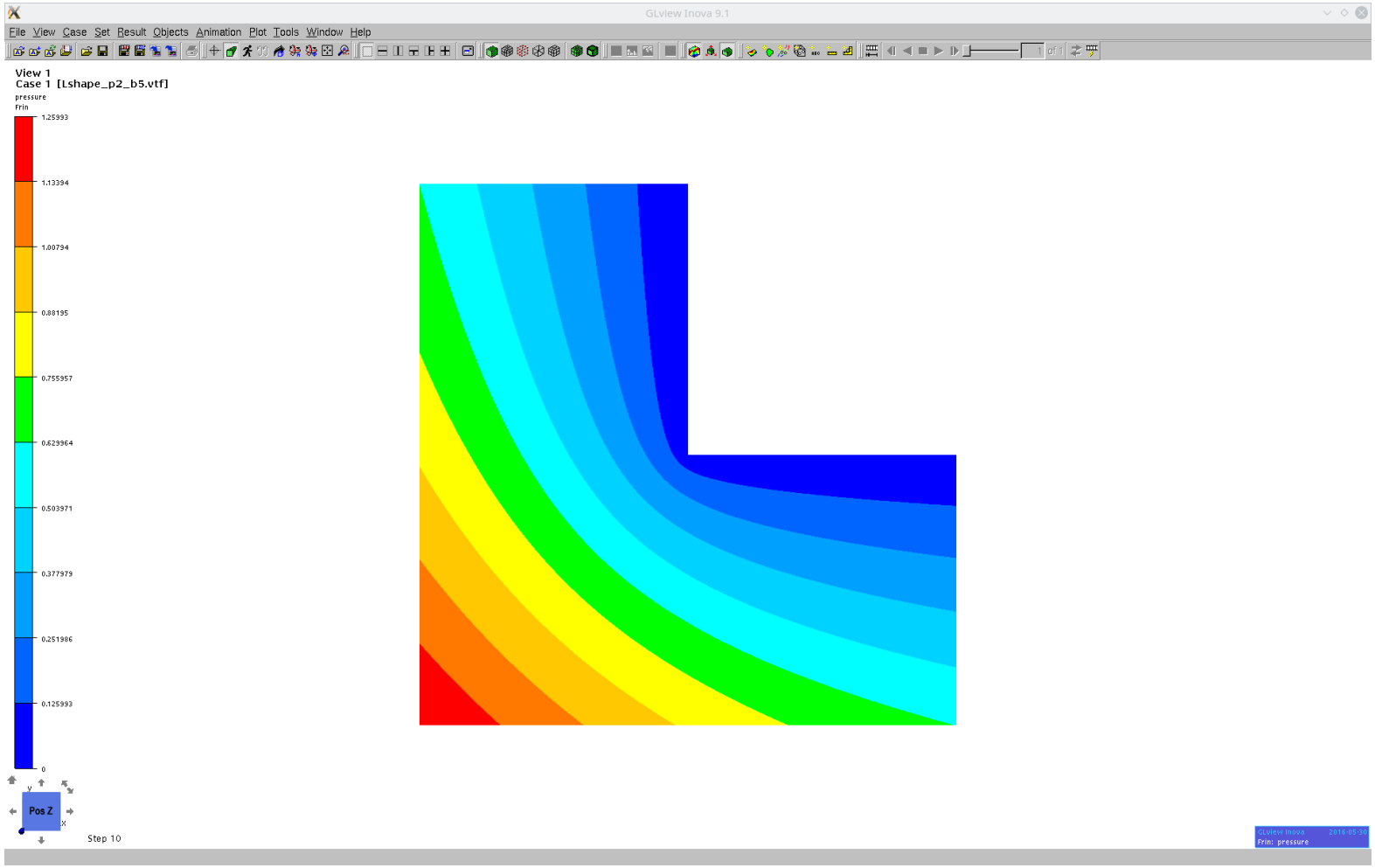
- Thermo-elasticity.



- Not yet public.

Applications built using IFEM

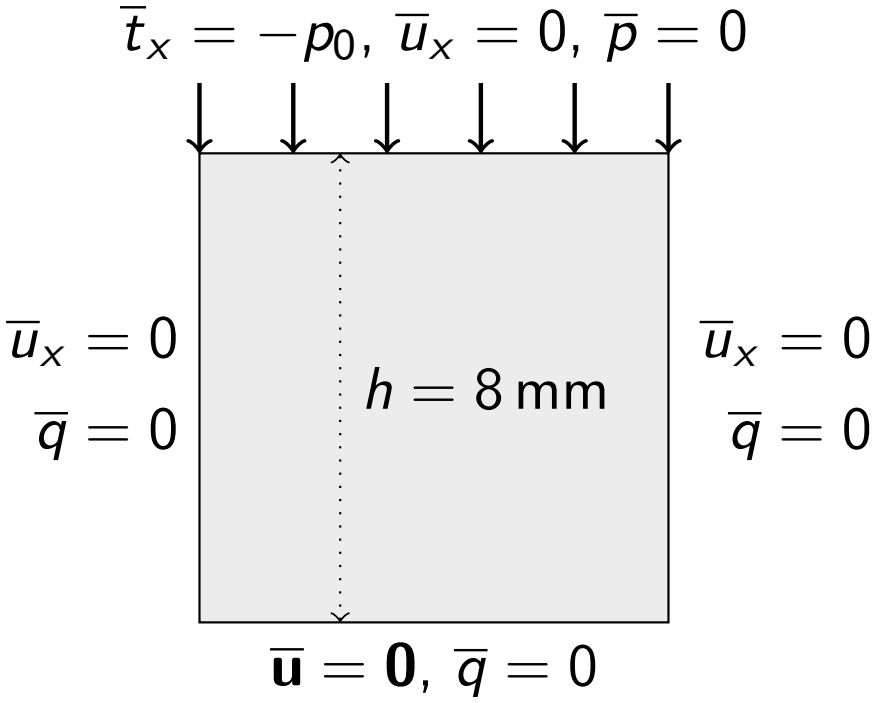
- Simple Darcy flow.

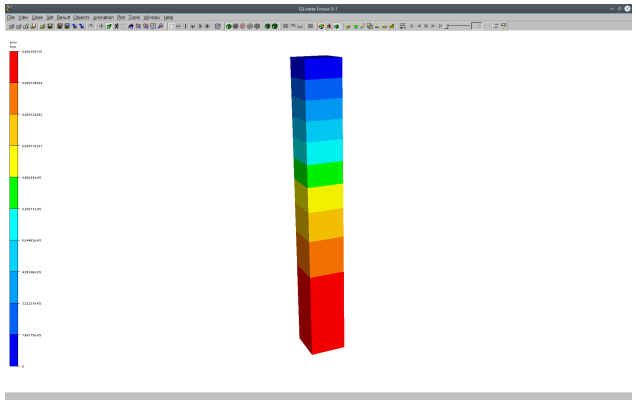


- Not yet public.

Applications built using IFEM

- PoroElasticity.

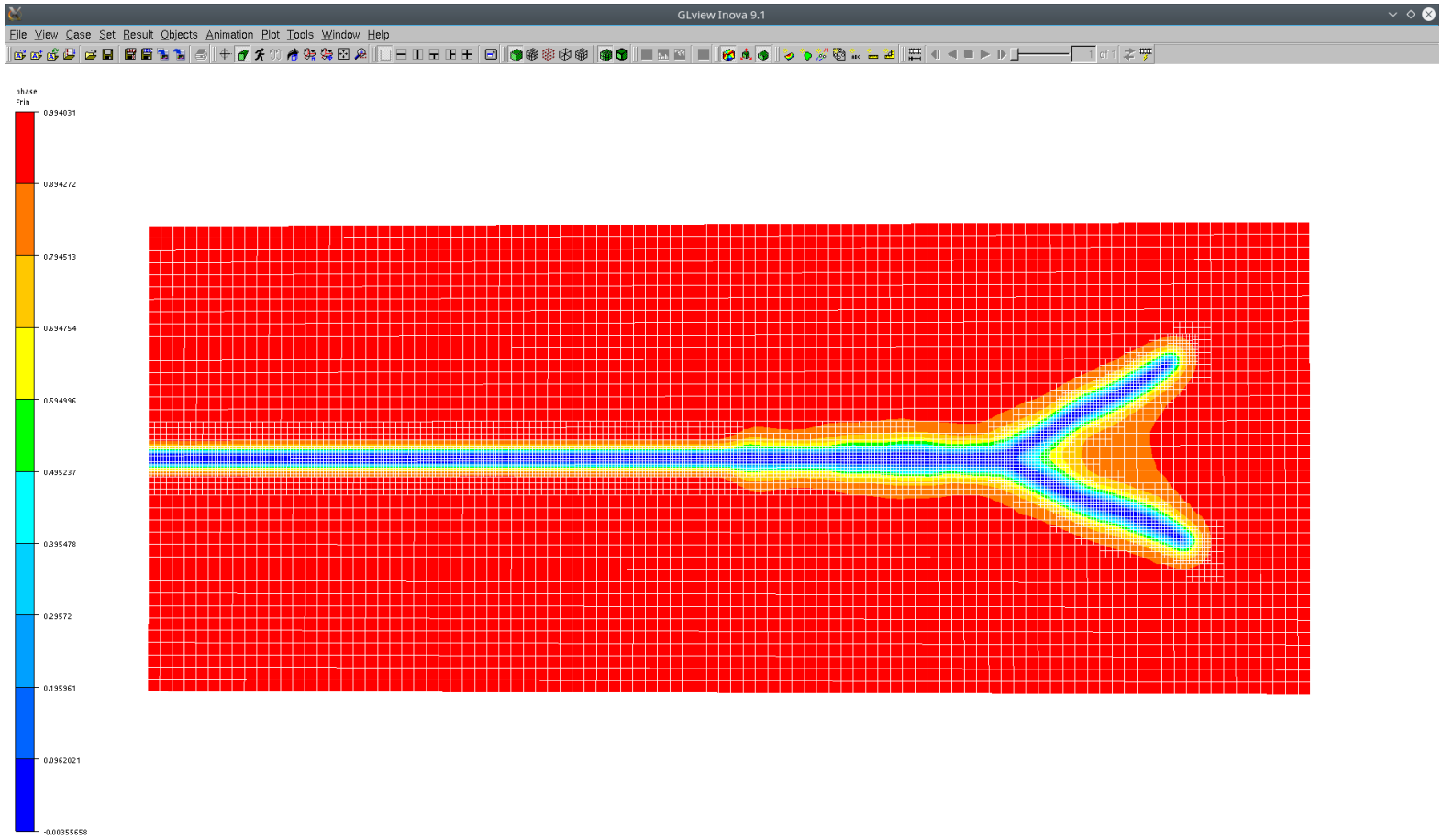




- Available at OPM/IFEM-PoroElasticity.

Applications built using IFEM

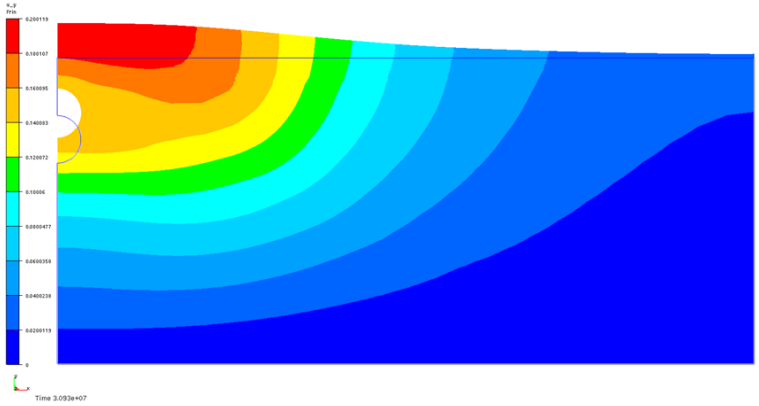
- Fracture dynamics.



- Available at OPM/IFEM-OpenFrac.

Applications built using IFEM

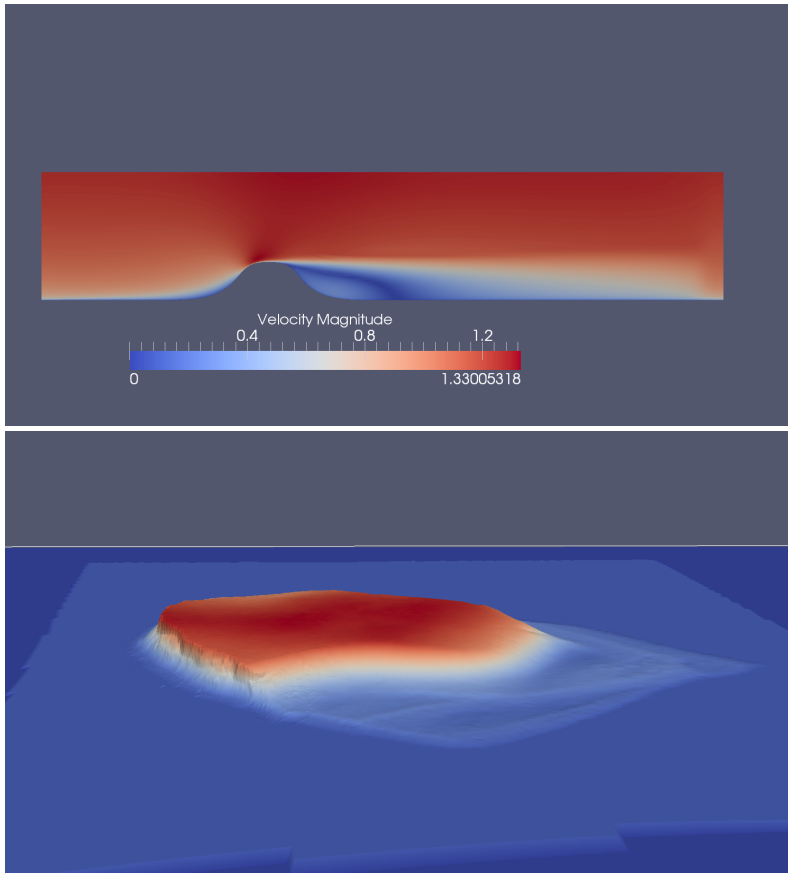
- ThermoHydroMechanics (Three phase poroelasticity w/ phase transitions and temperature effects).



- Not yet public.

Applications built using IFEM

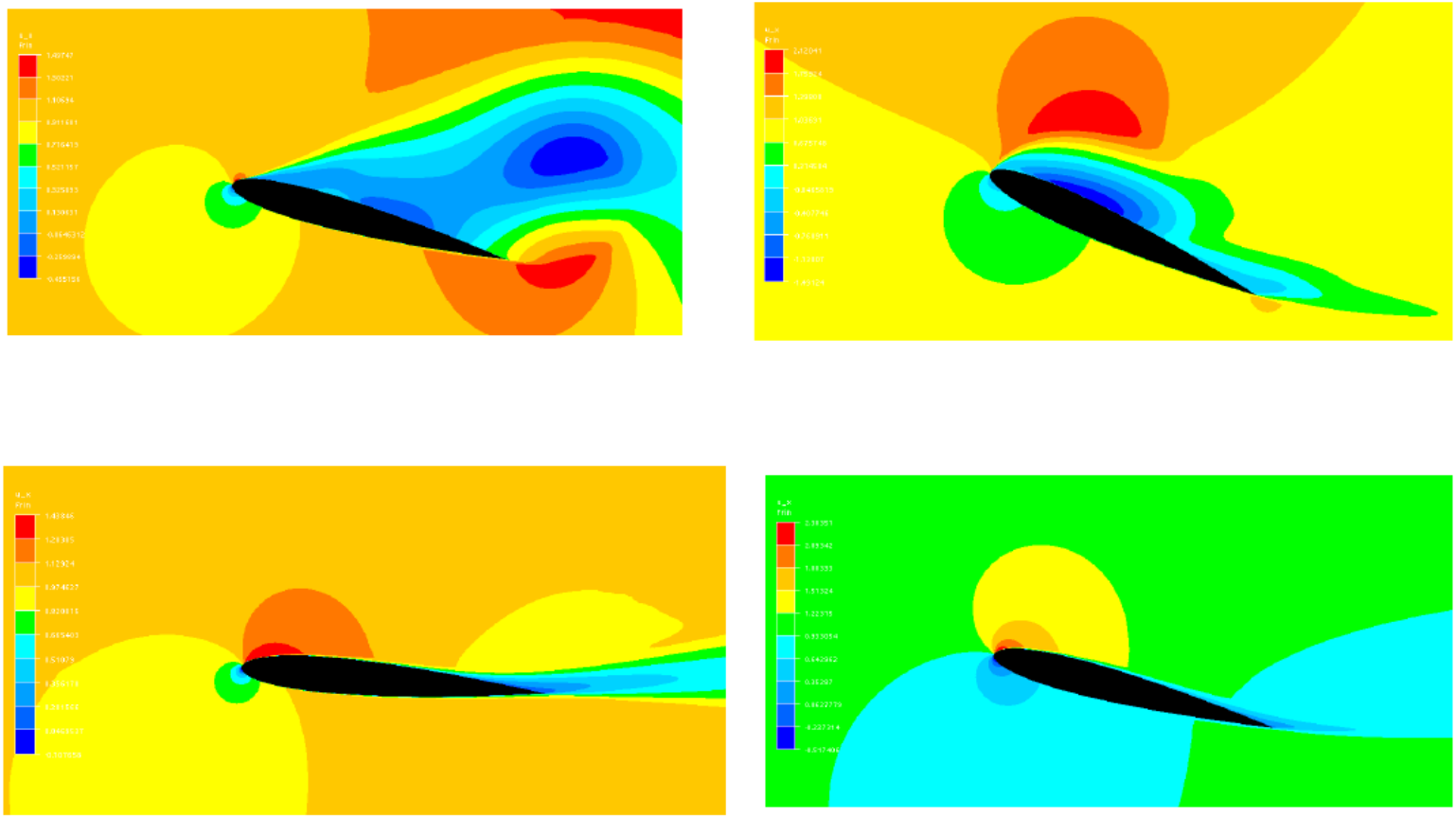
- Navier-Stokes.



- Not yet public.

Applications built using IFEM

- Fluid-structure interaction.



- Not yet public.

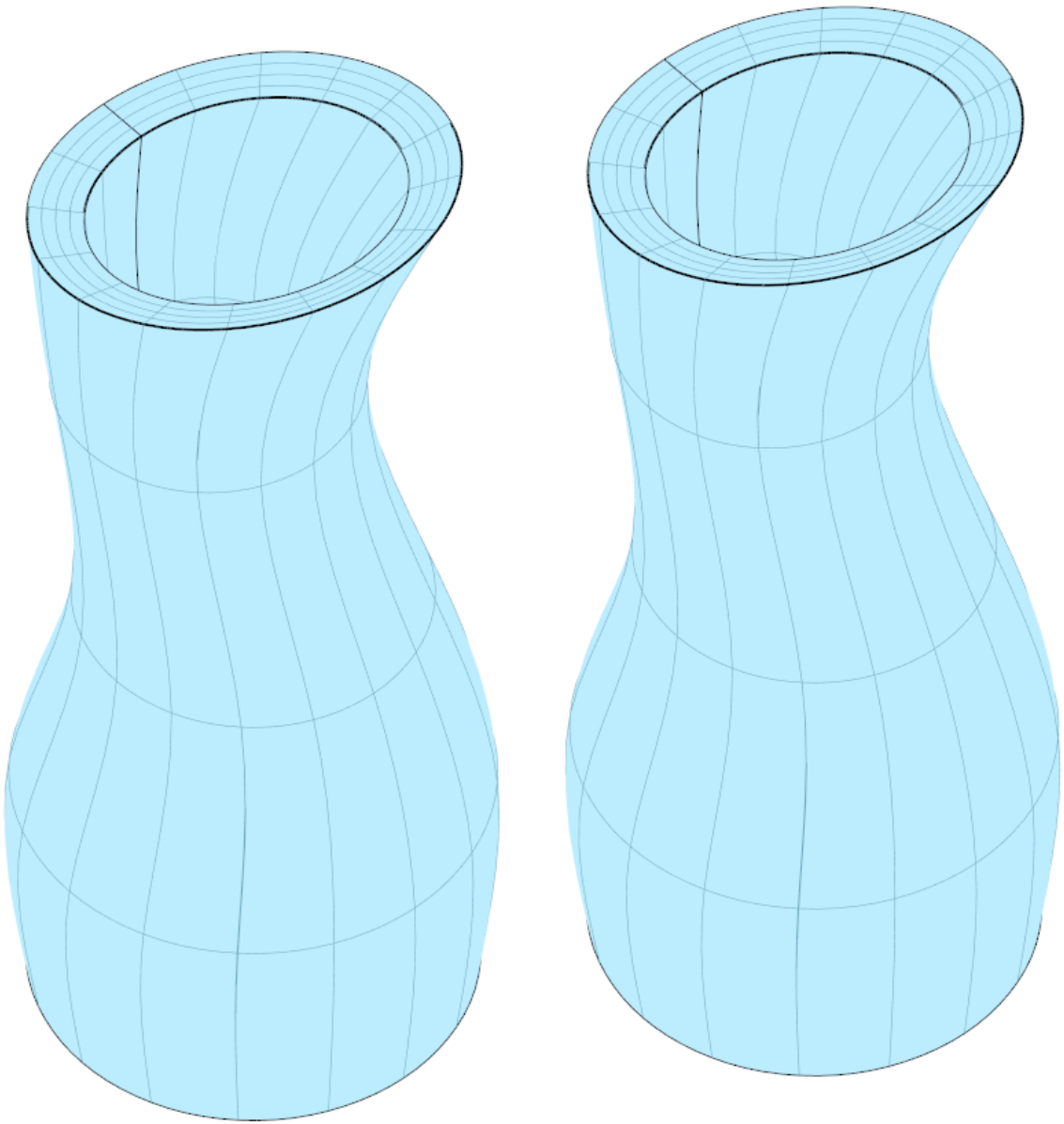
Geometry modelling and preprocessing

- Problem with CAD tools: Water-tight models and refinement control.
- Splipy: Geometry and simulation model setup.
- Allows building models bottom-up.
- Implemented as a python module.
- Intended workflow: Every step is scripted.
- Same script, several output formats (comparison to other tools).
- Available at SintefMath/Splipy.

Geometry modelling and preprocessing

```
import splipy.curve_factory as cf
import splipy.surface_factory as sf
import splipy.volume_factory as vf
from splipy.io.g2 import G2

discs = []

line = cf.line((3,0,0), (4,0,0))
discs.append(sf.revolve(line))

line = cf.line((3,0,5), (4.3,0,5))
discs.append(sf.revolve(line))

line = cf.line((2.1,0,10), (3.2,0,10))
discs.append(sf.revolve(line))

line = cf.line((3,0,15), (4,0,15))
discs.append(sf.revolve(line - (1.2,0,0)) + (1.2,0,0))

line = cf.line((3,0,0), (4,0,0))
discs.append(sf.revolve(line, axis=(0.25,0,1)) + (0,0,20))

for patch in discs:
    patch.refine(3)

vol = vf.loft(*discs) + (7,7,-2)
```

Geometry modelling and preprocessing



Example - NREL5MW meshing script

- Basic usage:

```
nrel.py paramfile=examples/simple3d.yaml
```

- Overriding parameters:

```
nrel.py paramfile=examples/simple3d.yaml
                n_length=40
```

- Output for OpenFOAM:

```
nrel.py paramfile=examples/simple3d.yaml
                format=OpenFOAM
```