Nečas Center for Mathematical Modeling

Numerical solution of continuum physics problems with FEniCS or how to use FEM and not vary (too much) about programming ...

Jaroslav Hron

Mathematical Institute, Charles University in Prague

CHARLES UNIVERSITY PRAGUE

faculty of mathematics and physics

✓ **lecture 1** introduction to FEniCS/python, FEM and how to solve laplace equation

✓ **lecture 2** boundary conditions, time discretization, convection-diffusion equation, (stabilization)

☞ **lecture 3** Stokes, incompressible Navier-Stokes equations

lecture 4 linear and non-linear elasticity

lecture 5 ALE-method, level-set method

Mesh generations in FEniCS

- internal commands for simple geometry:
  `IntervalMesh`, `RectangleMesh`, `BoxMesh`, etc. - see the docs
- FEniCS package MSHR - `https://bitbucket.org/benjamik/mshr` for mesh generation from CSG definitions:

```
from dolfin import *
import mshr

Center = Point(0.2, 0.2); Radius = 0.05; L=2.2; W=0.41

r0 = mshr.Rectangle(Point(0.0,0.0), Point(L,W))
c0 = mshr.Circle(Center,Radius,10)
geometry = r0 - c0
mesh=mshr.generate_mesh(geometry,50)
```

- GMSH + `dolfin-convert` [`http://geuz.org/gmsh`]
  1. use GMSH to create your domain and mesh, mark boundaries
  2. run: `dolfin-convert yourmesh.gmsh yourmesh.xml`
  3. then in the FEniCS script you can read the xml mesh:

```
mesh = Mesh("yourmesh.xml")
domain= MeshFunction("size_t", mesh, "yourmesh_physical_region.xml")
bndry= MeshFunction("size_t", mesh, "yourmesh_facet_region.xml")
```

Boundary marking in FEniCS

- using `SubDomain` objects

```
class LeftSD(SubDomain):
    def inside(self, x, on_boundary):
        return (on_boundary and near(x[0],0.0))

class RightSD(SubDomain):
    def inside(self, x, on_boundary):
        return (on_boundary and near(x[0],1.0))

Left=LeftSD(); Right=RightSD()
bparts = FacetFunction("size_t", mesh)
bparts.set_all(0)
Left.mark(bparts, 1); Right.mark(bparts, 2)
```

- using `AutoSubDomain` function

```
Left=AutoSubDomain(lambda x: near(x[0],0.0))
Right=AutoSubDomain(lambda x: near(x[1],0.0))
```

- using direct access to the mesh entities

```
bparts = FacetFunction("size_t", mesh)
bparts.set_all(0)
for f in facets(mesh):
    mp=f.midpoint()
    if near(mp[0],0.0) : bparts[f]=1      # Left
    if near(mp[0],1.0) : bparts[f]=2      # Right
```

Stokes equation - system of equations

Find velocity $\mathbf{v}$ and pressure p such that:

$$\begin{aligned} -\operatorname{div}(\nabla \mathbf{v}) + \nabla p &= \mathbf{f} && \text{in } \Omega \\ \operatorname{div} \mathbf{v} &= 0 && \text{in } \Omega \\ \mathbf{v} &= v_D && \text{on } \Gamma_D \\ [\nabla \mathbf{v}]\mathbf{n} &= \mathbf{g} && \text{on } \Gamma_N \end{aligned}$$

- weak form: find $(\mathbf{v}, p) \in V \times P$ such that

  $$\begin{aligned} a(\mathbf{v}, \bar{\mathbf{v}}) + b(p, \bar{\mathbf{v}}) &= L(\bar{\mathbf{v}}) \\ b(\bar{p}, \mathbf{v}) &= 0 \end{aligned}$$

  holds for all $(\bar{\mathbf{v}}, \bar{p}) \in V \times P$
- where

  $$a(\mathbf{v}, \mathbf{w}) = \int_\Omega \nabla \mathbf{v} \cdot \nabla \mathbf{w} \, \mathrm{d}x$$

  and

  $$b(p, \mathbf{v}) = \int_\Omega p \operatorname{div} \mathbf{v} \, \mathrm{d}x$$

Stokes equation - system of equations

- weak form: find $(\mathbf{v}, p) \in V \times P$ such that

$$\begin{aligned} a(\mathbf{v}, \bar{\mathbf{v}}) + b(p, \bar{\mathbf{v}}) &= L(\bar{\mathbf{v}}) \\ b(\bar{p}, \mathbf{v}) &= 0 \end{aligned}$$

  holds for all $(\bar{\mathbf{v}}, \bar{p}) \in V \times P$
- Choice of mixed FE spaces $V_h \times P_h$ - needs to satisfy LBB-condition (Taylor-Hood, Crouzeix-Raviart, etc. - see the book p. 387)
- FEniCS code for mixed spaces:

```
# Define function spaces (Taylor-Hood)
V = VectorFunctionSpace(mesh, "CG", 2)
P = FunctionSpace(mesh, "CG", 1)

W = MixedFunctionSpace([V, P])

w = Function(W)
(v, p) = split(w)

(_v, _p) = TestFunctions(W)
```

Stokes equation - system of equations

- weak form: find $(\mathbf{v}, p) \in V \times P$ such that
$$
\begin{aligned}
a(\mathbf{v}, \bar{\mathbf{v}}) + b(p, \bar{\mathbf{v}}) &= L(\bar{\mathbf{v}}) \\
b(\bar{p}, \mathbf{v}) &= 0
\end{aligned}
$$
holds for all $(\bar{\mathbf{v}}, \bar{p}) \in V \times P$
- FEniCS form definitions:

```
w = Function(W)        # v.s. (v, p) = Functions(W)
(v, p) = split(w)      #symbolic split
(v, p) = w.split()     #symbolic split
(v, p) = w.split(True) #split and create new objects

(_v, _p) = TestFunctions(W)

# Define variational form for Stokes
def a(u,v): return inner(grad(u),grad(v))*dx
def b(p,v): return p*div(v)*dx
def L(v): return inner(Constant((0.0,0.0)),v)*dx

ST = a(v,_v) + b(p,_v) + b(_p,v) - L(_v)

J = derivative(ST, w)
problem=NonlinearVariationalProblem(ST, w, bcs, J)
solver=NonlinearVariationalSolver(problem)

solver.solve()
```

Task no.3 - Navier-Stokes benchmark

1. extend the `stokes_ex.py` to the Navier-Stokes equation:

$$
\begin{aligned}
\varrho\left(\frac{\partial \mathbf{v}}{\partial t} + [\nabla \mathbf{v}]\mathbf{v}\right) - \operatorname{div}(\boldsymbol{\sigma}) &= \varrho \mathbf{f} && \text{in } \Omega \\
\operatorname{div} \mathbf{v} &= 0 && \text{in } \Omega \\
\mathbf{v} &= v_D && \text{on } \Gamma_D \\
[\nabla \mathbf{v}]\mathbf{n} &= \mathbf{0} && \text{on } \Gamma_N
\end{aligned}
$$

   where $\boldsymbol{\sigma} = -p\mathbf{I} + 2\mu\mathbf{D}$ is the stress tensor and $D = \frac{1}{2}(\nabla \mathbf{v} + \nabla \mathbf{v}^T)$.

2. Compute the stationary version of the NS benchmark 'Flow around cylinder' and compare the drag and lift
   see: `http://www.featflow.de/en/benchmarks/cfdbenchmarking/flow/dfg_benchmark1_re20.html`

3. Try to do the nonstationary version
   see: `http://www.featflow.de/en/benchmarks/cfdbenchmarking/flow/dfg_benchmark3_re100.html`

- How to log in to the cluster ....

```
ssh -Y -C IName@10.4.8.14
```

- Copy files or folders from/to a cluster - to be run on your local machine
  - copy local folder to your home directory on the cluster

    ```
    scp -r folder IName@10.4.8.14:~/
    ```

  - copy a folder from cluster to your local current directory

    ```
    scp -r IName@10.4.8.14:~/path/folder ./
    ```

- Lectures and example downloadable at `http://www.karlin.mff.cuni.cz/~hron/warsaw_2014/`