FEniCS Course

Lecture 3: Static nonlinear PDEs

FENICS PROJECT

Contributors
Marie E. Rognes

The Stokes equations

We consider the stationary Stokes equations: find the velocity u and the pressure p such that

$$\begin{aligned} -\operatorname{div}(\nu \nabla u - p\,\mathrm{I}) &= f \quad \text{in } \Omega \\ \operatorname{div} u &= 0 \quad \text{in } \Omega \end{aligned}$$

with boundary conditions

$$\begin{aligned} u &= 0 \quad \text{on } \partial\Omega_D \\ (\nu \nabla u - p\,\mathrm{I}) \cdot n &= p_0 \quad \text{on } \partial\Omega_N \end{aligned}$$

If viscosity ν varies with u (or p),

$$\nu = \nu(u)$$

this is a nonlinear system of partial differential equations.

The Stokes equations: variational formulation

Assume that $u \in V$ and $p \in Q$, then $w = (u, p) \in V \times Q = W$.
Let $f = 0$.
Multiply by test functions $(v, q) \in W$ and integrate first equation by parts

$$\int_\Omega \nu \nabla u \cdot \nabla v \, \mathrm{d}x - \int_\Omega p \operatorname{div} v \, \mathrm{d}x - \int_{\partial\Omega} (\nu \nabla u - p \, \mathrm{I}) \cdot n \cdot v \, \mathrm{d}s = 0$$

$$\int_\Omega \operatorname{div} u \, q \, \mathrm{d}x = 0$$

Adding the equations and incorporating the boundary conditions we obtain: find $(u, p) \in W = V_0 \times Q$ such that

$$\int_\Omega \nu \nabla u \cdot \nabla v \, \mathrm{d}x - \int_\Omega p \operatorname{div} v \, \mathrm{d}x - \int_\Omega \operatorname{div} u \, q \, \mathrm{d}x = \int_{\partial\Omega_N} p_0 \, v \cdot n \, \mathrm{d}s$$

for all $(v, q) \in W = V_0 \times Q$ where $V_0 = \{v \in V s.t. v|_{\partial\Omega_D} = 0\}$.

Canonical nonlinear variational problem

The following canonical notation is used in FEniCS for (possibly) nonlinear problems: find $w \in W$ such that

$$F(w; y) = 0$$

for all $y \in \hat{W}$.

Here, w is a function, and y is a test function, and so F is a *linear form*.

For the Stokes equations, we have $w = (u, p)$, $y = (v, q)$

$$F(w; y) = \int_\Omega \nu \nabla u \cdot \nabla v \,\mathrm{d}x - \int_\Omega p \,\mathrm{div}\, v \,\mathrm{d}x - \int_\Omega \mathrm{div}\, u \, q \,\mathrm{d}x - \int_{\partial\Omega_N} p_0 \, v \cdot n \,\mathrm{d}s$$

The Stokes equations introduce some new concepts

- Mixed function spaces
- Integration over boundaries
- Solving nonlinear problems (if nonlinear viscosity)
- (Reading a mesh from file)
- (Adjusting parameters)

Step by step: initializing a mesh from file

DOLFIN can read and write meshes from its own `.xml` or `.xml.gz` format

```
mesh = Mesh("dolfin-1.xml")
plot(mesh)
```

Conversion tools exist for other mesh formats

```
man dolfin-convert
```

We will need the normal on the mesh boundary facets:

```
n = FacetNormal(mesh)
```

Step by step: creating mixed function spaces

Mixed function spaces are created by taking the product of more basic spaces

```
V = VectorFunctionSpace(mesh, "CG", 2)
Q = FunctionSpace(mesh, "CG", 1)
W = V * Q
# W = MixedFunctionSpace([V, Q])
```

You can define functions on mixed spaces and split into components:

```
w = Function(W)
(u, p) = split(w)
```

... and arguments:

```
y = TestFunction(W)
(v, q) = split(y)
# (v, q) = TestFunctions(W)
```

Step by step: more about defining expressions

Again, the pressure boundary value can be defined using an expression:

```
p0 = Expression("1 - a*x[0]", degree=1, a=2)
```

When we specify the degree argument, this will be used as the polynomial degree when the expression is used in forms. Otherwise, the degree will be estimated heuristically.

All parameters (in this case `a`) must be specified at initialization, and can be modified later

FAQ: What is the difference between a `Function` and an `Expression`?

Function

... is described by expansion coefficients with reference to a `FunctionSpace` with a given basis: $u = \sum_i u_i \phi_i$

```
u = Function(V) # Defines the function space
u.vector()      # The coefficients
```

Expression

... given by an evaluation formula (more or less explicit)

```
f = Expression("...")

class Source(Expression):
    def eval(self, values, x)
        ...
```

An `Expression` can be projected or interpolated, or in some other way mapped, onto a `Function`, the converse is non-trivial.

Step by step: defining the viscosity

We may want to play with different viscosities.

In the simplest case, it is just constant: $\nu = 0.1$

```
nu = 0.1
```

... or it can vary with the domain: $\nu = 1 + 100x_1$

```
nu = Expression("1 + 100*x[1]")
```

... or it can vary with the unknown: $\nu = (u \cdot u)^{1/2}$

```
y = Function(W)
(u, p) = split(y)
def viscosity(u):
    return inner(u, u)**(1./2)
nu = viscosity(u)
```

Step by step: defining a boundary condition on a subspace

Assume that we have a mixed function space:

```
V = VectorFunctionSpace(...)
Q = FunctionSpace(...)
W = V * Q
```

The subspaces of W can be retrieved using `sub`:

```
W0 = W.sub(0)
```

Note that `W0` is not completely the same as `V`

The following code defines a homogenous Dirichlet boundary condition on the first subspace

```
bc = DirichletBC(W.sub(0), (0.0, 0.0),
       "near(x[0], 0.0) || near(x[0], 1.0)")
```

Stokes: defining the variational form

Assume that we have

```
w = Function(W)
(u, p) = split(w)
(v, q) = TestFunctions(W)
p0 = ...; nu = ...; n = ...
```

We can now specify the linear form F

```
F = (nu*inner(grad(u), grad(v))
    - div(u)*q - div(v)*p)*dx
    - p0*dot(v, n)*ds
```

Note that `dx` denotes integration over cells, `ds` denotes integration over exterior (boundary) facets, `dS` denotes integration over interior facets.

FAQ: How to specify integration over only subdomains? See the *Poisson with multiple subdomains* demo.

Step by step: solving (nonlinear) variational problems

Once a variational problem has been defined, it may be solved by calling the `solve` function (as for linear problems):

```
solve(F == 0, w, bc)
```

Or more verbosely

```
dF = derivative(F, w)
pde = NonlinearVariationalProblem(F, w, bcs,dF)
solver = NonlinearVariationalSolver(pde)
solver.solve()
```

Extracting the subfunctions (as DOLFIN functions)

```
(u, p) = w.split(deepcopy=True)
```

Step by step: adjusting parameters in DOLFIN

Adjusting global parameters

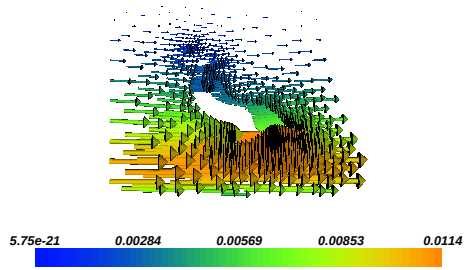
```
from dolfin import *
info(parameters, True)
parameters["form_compiler"]["cpp_optimize"] =
    True
#parameters["form_compiler"]["optimize"] = True
```

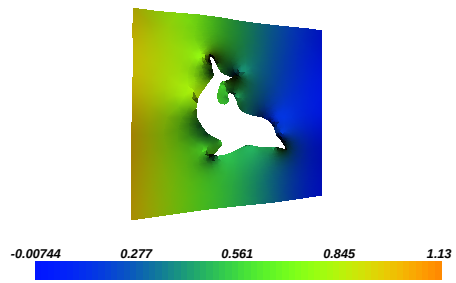
Adjusting local (and nested) parameters

```
solver = NonlinearVariationalSolver(pde)
info(solver.parameters, True)
solver.parameters["symmetric"] = True
solver.parameters["newton_solver"]["maximum_iterations"]
    = 100
```

Stokes: complete code (an example)

```
from dolfin import *

# Use -O2 optimization
parameters["form_compiler"]["cpp_optimize"] = True

# Define mesh and geometry
mesh = Mesh("dolfin-1.xml.gz")
n = FacetNormal(mesh)

# Define Taylor--Hood function space W
V = VectorFunctionSpace(mesh, "CG", 2)
Q = FunctionSpace(mesh, "CG", 1)
W = V * Q

# Define Function and TestFunction(s)
w = Function(W)
(u, p) = split(w)
(v, q) = TestFunctions(W)

# Define viscosity and bcs
nu = Expression("1 + 100*pow(x[1], 2)", degree=2)
p0 = Expression("1.0 - x[0]", degree=1)
bcs = DirichletBC(W.sub(0), (0.0, 0.0),
                  "near(x[1], 0.0) || near(x[1], 1.0)")

# Define form
F = (nu*inner(grad(u), grad(v))
     - div(u)*q - div(v)*p)*dx
    + p0*dot(v, n)*ds

solve(F == 0, w, bcs)
```





The FEniCS challenge!

Solve the Stokes equations

$$\begin{aligned} -\operatorname{div}(\nu \nabla u - p\,\mathrm{I}) &= f \quad \text{in } \Omega \\ \operatorname{div} u &= 0 \quad \text{in } \Omega \\ u &= 0 \quad \text{on } \partial\Omega_D \\ (\nu \nabla u - p\,\mathrm{I}) \cdot n &= p_0 \quad \text{on } \partial\Omega_N \end{aligned}$$

letting

- Ω be defined by the `dolfin-2.xml` mesh,
- $\Omega_D = \{x_1 = 0 \text{ or } x_1 = 1\}$,
- $p_0 = 1$ on $x_0 = 0$ and $p_0 = 0$ on $x_0 = 1$.
- $\nu = \nu(u) = 0.5(\nabla u \cdot \nabla u)^{1./(2(n-1))}$ with $n = 4$

Hint: You may need to compute an approximation first in order to provide a suitable initial guess to the Newton solver