FEniCS Course

Lecture 3: Static nonlinear PDEs

Contributors
Marie E. Rognes
Garth N. Wells

The Stokes equations

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

$$\begin{aligned} -\operatorname{div}(2\nu\epsilon(u) - p\,\mathrm{I}) &= f \quad \text{in } \Omega \\ \operatorname{div} u &= 0 \quad \text{in } \Omega \end{aligned}$$

where $\epsilon(u) = \frac{1}{2}\left(\operatorname{grad} u + (\operatorname{grad} u)^T\right)$ and with boundary conditions

$$\begin{aligned} u &= 0 \quad \text{on } \partial\Omega_D \\ -(2\nu\epsilon - p\,\mathrm{I}) \cdot n &= p_0\, n \quad \text{on } \partial\Omega_N \end{aligned}$$

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 $f = 0$.

Step 1

Multiply by test functions $(v, q) \in W$ and integrate first equation by parts

$$\int_\Omega 2\nu\epsilon(u) \cdot \operatorname{grad} v \,\mathrm{d}x - \int_\Omega p \operatorname{div} v \,\mathrm{d}x - \int_{\partial\Omega} (2\nu\epsilon(u) - p\,\mathrm{I}) \cdot n \cdot v \,\mathrm{d}s = 0$$

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

Step 2

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

$$\int_\Omega 2\nu\epsilon(u) \cdot \operatorname{grad} v \,\mathrm{d}x - \int_\Omega p \operatorname{div} v \,\mathrm{d}x - \int_\Omega \operatorname{div} u \, q \,\mathrm{d}x + \int_{\partial\Omega_N} p_0 \, v \cdot n \,\mathrm{d}s = 0$$

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

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 $y \in \hat{W}$.

Note

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

For the Stokes example

The functions are $w = (u, p)$, $y = (v, q)$ and the form F is

$$F(w; y) = \int_\Omega 2\nu\epsilon(u) \cdot \operatorname{grad} v \,\mathrm{d}x - \int_\Omega p \operatorname{div} v \,\mathrm{d}x - \int_\Omega \operatorname{div} u \, q \,\mathrm{d}x + \int_{\partial\Omega_N} p_0 \, v \cdot n \,\mathrm{d}s$$

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

Python code

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

Conversion tools exist for other mesh formats

Python code

```
$ man dolfin-convert
```

We will need the normal on the mesh boundary facets:

Python code

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

Step by step: creating mixed function spaces

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

Python code

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

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

Python code

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

... and arguments:

Python code

```
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:

Python code

```
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: $u = \sum_i u_i \phi_i$

Python code

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

Expression

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

Python code

```
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: $\nu = 0.1$

Python code

```
nu = 0.1
```

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

Python code

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

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

Python code

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

Python code

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

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

Python code

```
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 at the part where $x_0 = 0$.

Python code

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

Stokes: defining the variational form

Assume that we have

Python code

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

We can now specify the linear form F

Python code

```
epsilon = 2*sym(grad(u))
F = (nu*inner(epsilon, 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):

Python code

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

Or more verbosely

Python code

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

Extracting the subfunctions (as DOLFIN functions)

Python code

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

Step by step: adjusting parameters in DOLFIN

Adjusting global parameters

Python code

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

Adjusting local (and nested) parameters

Python code

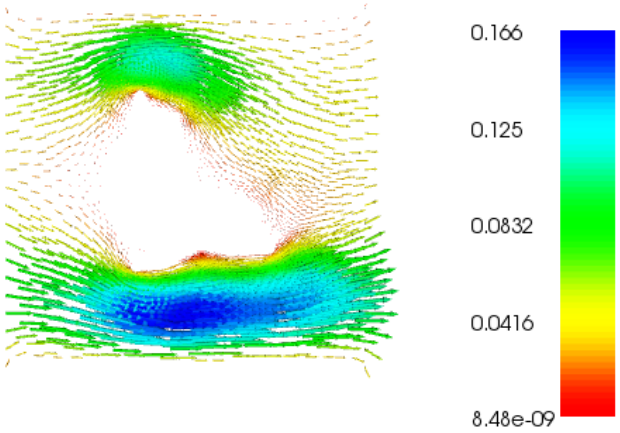
```
solver = NonlinearVariationalSolver(pde)
info(solver.parameters, True)
solver.parameters["symmetric"] = True
solver.parameters["newton_solver"]["maximum_iterations"]
    = 100
```

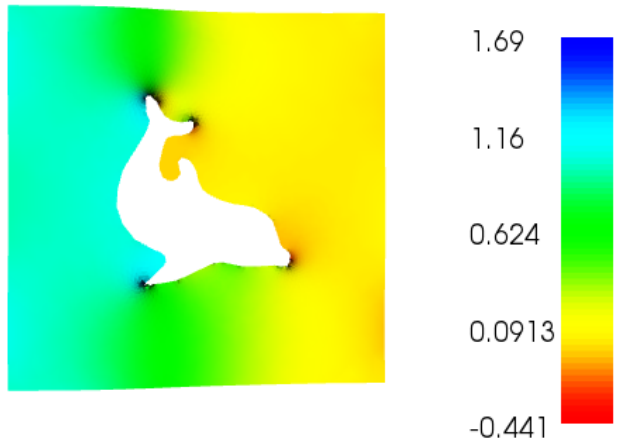
Stokes: a complete code example

Code

```
from fenics import *

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

# Define mesh and geometry
mesh = Mesh("dolfin-2.xml.gz")
n = FacetNormal(mesh)

# Define Taylor--Hood function space W
V = VectorFunctionSpace(mesh, "Lagrange" , 2)
Q = FunctionSpace(mesh , "Lagrange", 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("0.2*(1+pow(x[1],2))", degree=2)
p0 = Expression("1.0-x[0]", degree=1)
bcs = DirichletBC(W.sub(0), (0.0, 0.0),
                  "on_boundary && !(near(x[0], 0.0) || near(x[0], 1.0))")

# Define variational form
epsilon = sym(grad(u))
F = (2*nu*inner(epsilon, grad(v)) - div(u)*q - div(v)*p)*dx\
    + p0*dot(v,n)*ds

# Solve problem
solve(F == 0, w, bcs)

# Plot solutions
plot(u, title="Velocity", interactive=True)
plot(p, title="Pressure", interactive=True)
```

Stokes: running the example yields





Python code

```
$ python stokes_example.py
```

FEniCS programming challenge!

Solve the Stokes problem on Ω defined by the `dolfin-2.xml` mesh, defined by the following data

$$
\begin{aligned}
-\operatorname{div}(2\nu\epsilon(u) - p\,\mathrm{I}) &= 0 && \text{in } \Omega \\
\operatorname{div} u &= 0 && \text{in } \Omega \\
-(2\nu\epsilon(u) - p\,\mathrm{I}) \cdot n &= p_0\, n && \text{on } \partial\Omega_N = \{(x_0, x_1) |\, x_0 = 0 \text{ or } x_0 = 1\} \\
p_0 &= 1 - x_0 \\
u &= 0 && \text{on } \partial\Omega_D = \partial\Omega \backslash \partial\Omega_N
\end{aligned}
$$

Ex. 1 Consider a constant viscosity $\nu = 0.2$, compute and plot the solutions.

Ex. 2 Consider the nonlinear viscosity

$$\nu = \nu(u) = 0.5(\operatorname{grad} u \cdot \operatorname{grad} u)^{1/(2(n-1))}, n = 4$$

Compute and plot the solutions.

Hint: For Ex. 2, you may need to compute an approximation first in order to provide a suitable initial guess to the Newton solver