



# 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

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

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

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

If viscosity  $\nu$  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

$$\int_{\Omega} 2\nu\epsilon(u) \cdot \text{grad } v \, dx - \int_{\Omega} p \, \text{div } v \, dx - \int_{\partial\Omega} (2\nu\epsilon(u) - pI) \cdot n \cdot v \, ds = 0$$
$$\int_{\Omega} \text{div } u \, q \, dx = 0$$

## Step 2

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

$$\int_{\Omega} 2\nu\epsilon(u) \cdot \text{grad } v \, dx - \int_{\Omega} p \, \text{div } v \, dx - \int_{\Omega} \text{div } u \, q \, dx + \int_{\partial\Omega_N} p_0 v \cdot n \, ds = 0$$

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

$$\begin{aligned} F(w; y) = & \int_{\Omega} 2\nu \epsilon(u) \cdot \text{grad } v \, dx - \int_{\Omega} p \, \text{div } v \, dx - \int_{\Omega} \text{div } u \, q \, dx \\ & + \int_{\partial\Omega_N} p_0 v \cdot n \, ds \end{aligned}$$

# 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.gz")  
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 elements are created by taking the product of more basic elements

```
# Define Taylor--Hood function space W
P2 = VectorElement("Lagrange", triangle, 2)
P1 = FiniteElement("Lagrange", triangle, 1)
TH = P2 * P1
#TH = MixedElement([P2, P1])
W = FunctionSpace(mesh, TH)
```

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("...", degree=...)

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

During assemble an Expression is interpolated into a polynomial space of the given degree on each cell.

## 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]", degree=1)
```

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

```
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:

```
W = FunctionSpace(mesh, 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 at the part where  $x_0 = 0$ .

```
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

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

We can now specify the linear form  $F$

```
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  $\mathbf{dx}$  denotes integration over cells,  $\mathbf{ds}$  denotes integration over exterior (boundary) facets,  $\mathbf{dS}$  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, bc, 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 fenics 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: a complete code example

Code

```
from __future__ import print_function
from fenics import *

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

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

# Define Taylor--Hood function space W
P2 = VectorElement("Lagrange", triangle, 2)
P1 = FiniteElement("Lagrange", triangle, 1)
TH = MixedElement([P2, P1])
W = FunctionSpace(mesh, TH)

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

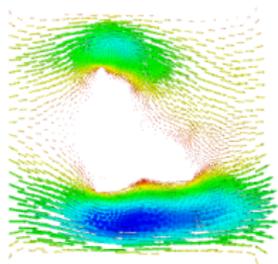
# Define viscosity and bcs
nu = Expression("0.2*(1+pow(x[1],2))", degree=2)
p0 = (1.0 - x[0]) # or Expression("1.0-x[0]", degree=1)
bcs = DirichletBC(W.sub(0), (0.0, 0.0),
                  "on_boundary && !(near(x[0], 0.0) || near(x[0], 1.0))")

# 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")
plot(p, title="Pressure")
# interactive()
```

# Stokes: running the example yields



0.166

0.125

0.0832

0.0416

8.48e-09



Help



1.69

1.16

0.624

0.0913

-0.441



Help

```
$ python stokes_example.py
```

## *FEniCS programming challenge!*

Solve the Stokes problem on  $\Omega$  defined by the `dolfin-1.xml.gz` mesh, defined by the following data

$$-\operatorname{div}(2\nu\epsilon(u) - p\mathbf{I}) = 0 \quad \text{in } \Omega$$

$$\operatorname{div} u = 0 \quad \text{in } \Omega$$

$$-(2\nu\epsilon(u) - p\mathbf{I}) \cdot n = p_0 n \quad \text{on } \partial\Omega_N = \{(x_0, x_1) \mid x_0 = 0 \text{ or } x_0 = 1\}$$

$$p_0 = 1 - x_0$$

$$u = 0 \quad \text{on } \partial\Omega_D = \partial\Omega \setminus \partial\Omega_N$$

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

**Ex. 2** Consider the nonlinear viscosity

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

Plot the solutions and estimate the **average velocity** in the  $x[0]$ -direction.

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