FEniCS: Non-Linear Poisson Problem

From documented demonstration number 15 from Dolfin version 1.4.0

Ida Ang (Last edit: July 1, 2019)

1 Problem Definition

Nonlinear Poisson's equation for domain Ω and boundary $\partial\Omega = \Gamma_D \cup \Gamma_N$

$$\begin{aligned} -\nabla \cdot (q(u)\nabla u) &= f(x,y) \quad \text{in } \Omega \\ u &= 1 \quad \text{on } \Gamma_D \\ \nabla u \cdot n = \frac{\partial u}{\partial n} &= 0 \quad \text{on } \Gamma_N \end{aligned} \tag{1}$$

We can rewrite the Neumann boundary condition in indicial:

$$\nabla u \cdot n = \frac{\partial u_j}{\partial x_i}\big(\mathbf{e_j} \otimes \mathbf{e_i}\big) \cdot n_k \mathbf{e_k} = \frac{\partial u_j}{\partial x_i} n_i \mathbf{e_j} = g_j \mathbf{e_j}$$

Therefore the Neumann boundary Condition can be written as:

$$\frac{\partial u_j}{\partial x_i} n_i = 0 \tag{2}$$

2 Variational Weak Form

Write Eq. 1 part 1 in indicial notation where the gradient of a vector field is a 2nd order tensor:

$$\begin{aligned} -\nabla \cdot (q(u)\nabla u) &= f \\ -q\big[\nabla \cdot \frac{\partial u_j}{\partial x_i}\big(\mathbf{e_j} \otimes \mathbf{e_i}\big)\big] &= f_j \mathbf{e_j} \quad \text{where } \nabla \cdot \mathbf{A} = \frac{\partial A_{ik}}{\partial x_j}\big(\mathbf{e_i} \otimes \mathbf{e_k}\big)\mathbf{e_j} = \frac{\partial A_{ij}}{\partial x_j}\mathbf{e_i} \\ -q\frac{\partial^2 u_j}{\partial x_i \partial x_k}\big(\mathbf{e_j} \otimes \mathbf{e_i}\big)\mathbf{e_k} &= f_j \mathbf{e_j} \\ -q\frac{\partial^2 u_j}{\partial x_i \partial x_k}\mathbf{e_j}\delta_{ik} &= f_j \mathbf{e_j} \\ -q\frac{\partial^2 u_j}{\partial x_k^2}\mathbf{e_j} &= f_j \mathbf{e_j} \end{aligned}$$

Multiply by test function, v:

$$\begin{aligned} -q\frac{\partial^2 u_j}{\partial x_k^2}\mathbf{e_j} \cdot v_p \mathbf{e_p} &= f_j \mathbf{e_j} \cdot v_p \mathbf{e_p} \\ -q\frac{\partial^2 u_j}{\partial x_k^2} v_j &= f_j v_j \quad \text{Integrate over domain } \Omega \\ -q\int_\Omega \frac{\partial^2 u_j}{\partial x_k^2} v_j dx &= \int_\Omega f_j v_j dx \end{aligned} \tag{3}$$

Use Integration by Parts:

$$(fg)' = f'g + fg' \rightarrow f'g = (fg)' - fg' \rightarrow f''g = (f'g)' - f'g' \tag{4}$$

Where we can substitute $f' = \frac{\partial x_j}{\partial x_k}$ and $g = v$ into Eq. 4

$$\frac{\partial^2 u_j}{\partial x_k^2} v_j = \left(\frac{\partial u_j}{\partial x_k} v_j\right)_{,k} - \frac{\partial u_j}{\partial x_k}\frac{\partial v_j}{\partial x_k}$$

Change signs and substitute into Eq. 3

$$\begin{aligned} -q\int_\Omega \left(\frac{\partial u_j}{\partial x_k} v_j\right)_{,k} dx + q\int_\Omega \frac{\partial u_j}{\partial x_k}\frac{\partial v_j}{\partial x_k} dx &= \int_\Omega f_j v_j dx \quad \text{Divergence theorem on first term} \\ -q\int_\Omega \frac{\partial u_j}{\partial x_k} n_k v_j dx + q\int_\Omega \frac{\partial u_j}{\partial x_k}\frac{\partial v_j}{\partial x_k} dx &= \int_\Omega f_j v_j dx \quad \text{Neumann BC are 0 (Eq. 2)} \\ q\int_\Omega \frac{\partial u_j}{\partial x_k}\frac{\partial v_j}{\partial x_k} dx &= \int_\Omega f_j v_j dx \end{aligned} \tag{5}$$

Writing this form in direct notation, we have the following variational (weak) form:

$$\int_\Omega q(u)\nabla u \cdot \nabla v dx = \int_\Omega f v dx \tag{6}$$

3 Nonlinear Variational Form

The canonical form of the nonlinear variational form:

$$F(u; v) = 0 \qquad (\forall v \in \hat{V}) \tag{7}$$

F: $V \times \hat{V}$ is a semi-linear form. The argument u is linear.

Therefore, we can seek a solution for u $\in$ V that satisfies eq. 7

$$F(u; v) = \int_\Omega q(u)\nabla u \cdot \nabla v dx - \int_\Omega f v dx = \text{residual} = 0 \tag{8}$$

In the implementation we want to reduce this residual value to a value close to machine precision.

4 FEniCS Implementation

Some edits were made to the code based on the FEniCS book
Domain is a unit square:

$$\Omega = [0, 1] \times [0, 1]$$

This can be implemented with the built-in function UnitSquareMesh:

```
mesh = UnitSquareMesh(32, 32)
```

Define the function space using 'CG', for Continuous Galerkin, as a synonym for 'Lagrange'. Degree 1 is for the standard linear Lagrange element, which is a triangle with nodes at the three vertices (or in other words, continuous piecewise linear polynomials).

```
V = FunctionSpace(mesh, "CG", 1)
```

Neumann boundary (Bottom, top, and left side of square):

$$\Gamma_N = (x, 0) \cup (x, 1) \cup (0, y) \subset \partial\Omega$$

Dirichlet boundary (right side of square):

$$\Gamma_D = (1, y) \subset \partial\Omega$$

Define a class which isolates the Dirichlet boundary:

```
class DirichletBoundary(SubDomain):
   def inside(self, x, on_boundary):
      return abs(x[0] - 1.0) < DOLFIN_EPS and on_boundary
```

Recall that the Dirichlet BC is set to 1 which can be defined by a constant

```
u_0 = Constant(1.0)
bc = DirichletBC(V, u_0, DirichletBoundary())
```

Note that the Neumann BC is expressed in the weak form of the nonlinear Poisson problem (Natural). Dirichlet BCs (Essential) have to be defined explicitly outside of the weak form

We can define the unknown, u, and the test function.

```
u = Function(V)
v = TestFunction(V)
```

Note

- Note how u is not defined as a trial function but simply as a function (unknown)
- By omitting the trial function `TrialFunction`, FEniCs assumes the problem is nonlinear

Input functions are defined below:

$$f(x, y) = x \sin(y) \tag{9}$$

$$q(u) = 1 + u^2 \tag{10}$$

Write the defined input function f, Eq. 9, and q, Eq. 10, in C++ syntax

```
f = Expression("x[0]*sin(x[1])", degree=2)
q = 1 + u**2
```

We can write the semilinear form in one line

```
F = q*inner(grad(u), grad(v))*dx - f*v*dx
```

Solve this by specifying the Newton solver within solver parameters:

```
solve(F == 0, u, bc,
     solver_parameters="newton_solver":"relative_tolerance":1e-6)
```

The Newton procedure has converged when the residual r_n at iteration n is less than the absolute tolerance or the relative residual $\frac{r_n}{r_0}$ is less than the relative tolerance 1×10^{-6}.