Modeling of Fluid Flow
Lab session 2: FEniCS introduction – Poisson equation

David Nolte[*], Cristóbal Bertoglio[†]

April 25, 2018

1 Lab exercises

Exercises for practicing programming in Python during the lab session.

Reading material:

- Chapters "Preliminaries", "Fundamentals" (only until (excluding) 'Deflection of a membrane') of the FEniCS tutorial: `https://fenicsproject.org/tutorial`
- Additionally the first 2 sections of the chapter "Subdomains and Boundary Conditions": `https://fenicsproject.org/pub/tutorial/sphinx1/._ftut1005.html` (more is handy of course)

1.1 Hello world

Solve the Poisson problem using the FEniCS library. Find $u \in V$, s.t.,

$$-\Delta u = f \qquad \text{in} \;\; \Omega \tag{1}$$
$$u = g \qquad \text{on} \;\; \partial\Omega, \tag{2}$$

where Ω is the unit square, $f = -10$, and the Dirichlet boundary condition $g = 1 + 2x^2 + 3y^2$ along the boundary of the domain, $\partial\Omega$.

Use piecewise linear basis functions (P_1).

- Plot the solution.

[*]d.j.nolte@rug.nl
[†]c.a.bertoglio@rug.nl

- Compute the L^2 error with respect to the exact solution, $u = 1 + 2x^2 + 3y^2$:

$$\|u - u_h\|_{L^2(\Omega)} = \sqrt{\int_\Omega (u - u_h)^2}$$

  where u_h is the numerical solution. Use the `assemble` function to compute the integral. NOTE: u can be a FEniCS `Expression`, it is not necessary to create an extra `Function`. Check the computed error with the FEniCS function `errornorm(u, u_h)`.

- How big is the error in comparison if you use basis functions of degree 2? Comment with your partner on how the result is consistent with Cea's lemma.

1.2 Dirichlet and Neumann boundary conditions

Consider the same problem with Dirichlet and Neumann boundary conditions. Now, the boundary of the domain is split in two parts, $\partial\Omega = \Gamma_D \cup \Gamma_N$. Γ_D is the left and right boundaries, $x = 0$ and $x = 1$, and Γ_N are the top and bottom boundaries, $y = 0$ and $y = 1$. The boundary conditions are[1]:

$$\begin{aligned} u = 1 + 2x^2 + 3y^2 \quad &\text{on} \quad \Gamma_D \\ \frac{\partial u}{\partial n} = 6y \quad &\text{on} \quad \Gamma_N. \end{aligned}$$

Verify that the solution is the same as in the previous exercise!

2 Assignments

Evaluation: discussions, appointments to be arranged.

2.1 Poisson: error analysis

Use the *method of manufactured solutions* to test the convergence of your FEM program. Given the analytical solution

$$u_0 = xy + \cos(2\pi x)\sin(2\pi y) \quad \text{with} \quad x, y \in [0, 1],$$

determine $f = -\Delta u_0$.

Then, solve the Dirichlet problem in the unit square,

$$\begin{aligned} -\Delta u = f \quad &\text{in} \quad \Omega \\ u = u_0 \quad &\text{on} \quad \partial\Omega. \end{aligned}$$

[1] Note that on the bottom boundary the normal vector is $\boldsymbol{n} = -\boldsymbol{e}_y$, so $\frac{\partial u}{\partial n} = -\frac{\partial u}{\partial y} = -6y$. On the other hand, $y = 0$. So as a "short cut" we use the formula below to combine the upper and lower boundaries in one expression, instead of treating both boundaries separately.

Check the convergence behavior by solving the problem for a sequence of meshes with $N = 2^k$, $k = 1, 2, ..., 10$ (N is the number of nodes per side, the argument in `UnitSquareMesh(N, N)`) and computing the L^2 error of each solution u_h w.r.t. the analytical solution, as was done in exercise 1. Use first order elements.

- Plot the error over N (or $h = 1/N$) in log–log-scale. The `matplotlib` command is `loglog(x, y)`. The additional options `basex=2, basey=2` switch to $\log_2$ for both axes.
- What can be said about the rate of convergence?
- How does the rate of convergence change if you use P_2 elements (i.e., a basis of second order polynomials)?

2.2 Diffusion–reaction and multiple boundary conditions

In the lecture, some properties were shown for the diffusion–reaction equation,

$$\begin{aligned} ru - \mu\Delta u &= f && \text{in } \Omega && (3)\\ u &= g_D && \text{on } \Gamma_1 \quad \text{(Dirichlet-BC)} && (4)\\ \frac{\partial u}{\partial n} &= g_N && \text{on } \Gamma_2 \quad \text{(Neumann-BC)} && (5)\\ u + \frac{\partial u}{\partial n} &= g_R && \text{on } \Gamma_3 \quad \text{(Robin-BC).} && (6) \end{aligned}$$

Consider again the same test case, where $\Omega = [0, 1] \times [0, 1]$ and $u = 1 + 2x^2 + 3y^2$. Set $r = 1$, $\mu = 1$, and as before, calculate the right hand side f.
Reusing the code you have written in exercise 2, implement the following 4 cases:

1. pure Dirichlet problem with BC (4), $g_D := u$ on all $\partial\Omega$
2. pure Neumann problem with BC (5): calculate (on paper) g_N and apply on all $\partial\Omega$
3. pure Robin problem with BC (6): calculate (on paper) g_R and apply on all $\partial\Omega$
4. Mixed Dirichlet, Neumann and Robin boundary conditions: Dirichlet on $x = 0$, Neumann on $x = 1$, Robin on $y = 0$ and $y = 1$.

Hint 1: you might consider writing a function of type

```python
def diffreac(N, bc_type='dirichlet'):
    ...
```

where `bc_type` selects the boundary condition and the function handles all these cases.
Hint 2: all versions should give the same result.
Hint 3: see `https://fenicsproject.org/pub/tutorial/sphinx1/._ftut1005.html#setting-multiple-dirichlet-neumann-and-robin-conditions`

2.3 Heat equation example

Apply what you have learned on an applied example! Suppose that we have a copper plate with a perforation. The copper plate is heated on the outside to a constant temperature. A cooling liquid flows through the perforation (cross section of a pipe, flow is perpendicular to the plate). The cooling effect can be modelled by a Robin boundary condition, without solving the flow inside the tube.

Consider the stationary heat equation (a Laplace equation),

$$k\Delta u = 0 \quad \text{in} \;\; \Omega \tag{7}$$

$$u = g_D \quad \text{on} \;\; \Gamma_1 \tag{8}$$

$$\frac{\partial u}{\partial n} = g_N \quad \text{on} \;\; \Gamma_2 \tag{9}$$

$$\frac{\partial u}{\partial n} = h(u_f - u) \quad \text{on} \;\; \Gamma_3. \tag{10}$$

The domain is the square $[-1,1] \times [-1,1]$ with a circular perforation in the center with radius $r = 0.25$, i.e., $\Omega = \{x, y \in [0,1] \mid x^2 + y^2 \geq r^2\}$. Such a mesh can be created with the `mshr` module that comes with FEniCS:

```
from fenics import *     # here we need Point
import mshr
domain = mshr.Rectangle(Point(-1, -1), Point(1, 1)) - \
    mshr.Circle(Point(0, 0), 0.25)
mesh = mshr.genereate_mesh(domain, N)   # i.e. N = 32
```

In Eq. (7), k is the thermal conductivity ($k = 401$ for copper).

The boundaries have to be defined as follows:

- Γ_1: left and right boundaries, constant temperatures $g_D = 1000$ K (Dirichlet)
- Γ_2: top and bottom boundaries, homogeneous Neumann BCs, modelling symmetry in normal direction or perfect isolation.
- Γ_3: boundary of the circle, Robin BC (10) where h is the convective heat transfer coefficient. Consider $h = 10$ for still air and $h = 10^4$ for moving water. u_f is the temperature of the liquid, use $u_f = 280$ K.

Plot the solution for both values of h. Using which liquid the overall temperature in the plate is reduced more? (Hint: you can compute the integral of a function u over the domain with `assemble(u*dx)`.)