

# Lab module 5: The Finite Element Method for Partial Differential Equations - goal-oriented adaptive error control

Johan Jansson (jjan@kth.se), Niyazi Cem Degirmenci, Johan Hoffman

March 2, 2017

## 0 Jupyter-FEniCS web PDE solver environment

The address of the web Jupyter-FEniCS cloud environment, described more in detail below, is provided via email with the ip of the cloud virtual machine and Jupyter login. To run a program in Jupyter-FEniCS, open the Python 2 notebook and select the Run command under the Cell menu.

You can find example Jupyter notebooks with implementations from the lab under the `src` directory in the zip archive of the lab module, which you can upload in the Jupyter interface.

**NB!:** The files in the web environment are **not** stored on disk, which means they will disappear if the system is rebooted. Do not forget to save your notebook regularly to your own computer, by using Download as > IPython Notebook (.ipynb) under File in your notebook.

You can also set up the environment at your own computer by using the command:

```
sudo docker run -t -i -p 80:8000 jjan/fenics-mooc:test
```

and using the login name and passwords listed on the terminal window by accessing localhost from a web browser.

## 1 Introduction

In this lab session you will use the FEniCS [2] framework for automated solution of partial differential equations (PDE) to formulate finite element methods (FEM) that solve PDE and investigate some of the concepts for ODE and PDE in the course.

Specifically you will investigate FEM for fluid flow modeled by the incompressible Navier-Stokes equations [4, 5] and transport and chemical reactions modeled by convection-reaction equations.

The goal of this session is to:

1. Become familiar and experiment with a posteriori error estimation and goal-oriented adaptive error control, here applied to the incompressible Navier-Stokes equations.

In this session we will work with the Python interface to FEniCS. We will use FEniCS version 1.6 which is installed in the Jupyter notebook (<http://jupyter.org>) Python web environment provided at the link at the top of the instructions. On the FEniCS home page [2] there is extensive documentation of the interface at both overview and detail level.

For reference material, please see the lecture notes from the DD2263 Methods in Scientific Computing course at KTH [3] and the book Computational Differential Equations [1].

## 2 Exercises

### 2.1 Goal-oriented adaptive error control

The setting of this exercise is adaptive error control in FEM, which is a methodology for satisfying a tolerance on the global discretization error measured in a quantity of interest (drag/lift on an object, displacement of a region, etc.) by determining how the cells in the mesh contribute to the global error and iteratively refining those cells which have the largest contribution (or generating a new mesh in some other way to satisfy the tolerance). This gives crucial *reliability* of the method for applications since a bound on the discretization error is known, and *efficiency* since a mesh that in some sense is optimal for a given tolerance can be constructed.

We will investigate the *do-nothing* [6] method for error control, a new method which is very simple in formulation and is automated in the sense that it doesn't require manual derivation for every PDE.

For a linear stationary boundary value PDE written as the weak residual  $r(u, v)$  with exact solution  $u$ :

$$r(u, v) = a(u, v) - L(v) = 0, \forall v \in V \quad (1)$$

the finite element method and continuous piecewise linear (cG(1)) FEM solution  $U$  belonging to the finite element space  $V_h \in V$  can be expressed as:

$$r(U, v) = a(U, v) - L(v) = 0, \forall v \in V_h \quad (2)$$

We can express the error  $e = u - U$  measured in a goal functional  $M(e)$  exactly using the weak residual  $r$  and an exact solution  $\phi$  to an adjoint problem formulated below by the *error representation*:

$$M(e) = a(e, \phi) = r(U, \phi) \quad (3)$$

the do-nothing error indicator for cell  $K$  in the mesh is simply:

$$\mathcal{E}_K = r(U, \Phi)_K \quad (4)$$

with the cG(1) adjoint solution  $\Phi$  defined by the following adjoint problem with the goal functional  $M$  as source:

$$a(w, \Phi) = M(w), \quad \forall w \in V_h \quad (5)$$

We will now apply the do-nothing adaptive method to the stationary incompressible Navier-Stokes equation.

We would like to solve the system of incompressible Navier-Stokes equations, which in weak form, with weak residual  $r$  can be stated as:

$$\begin{aligned} r(\hat{u}, \hat{v}) &= ((u \cdot \nabla)u + \nabla p, v) + (\nu \nabla u, \nabla v) + (\nabla \cdot u, q) = 0, \\ \hat{u} &\in [V_h]^2 \times Q_h, \quad \forall \hat{v} \in [V_h]^2 \times Q_h \\ \hat{u} &= (u, p) \quad (\text{Solution: velocity and pressure}) \\ \hat{v} &= (v, q) \quad (\text{Test function}) \end{aligned} \quad (6)$$

with  $\nu$  a diffusion parameter.

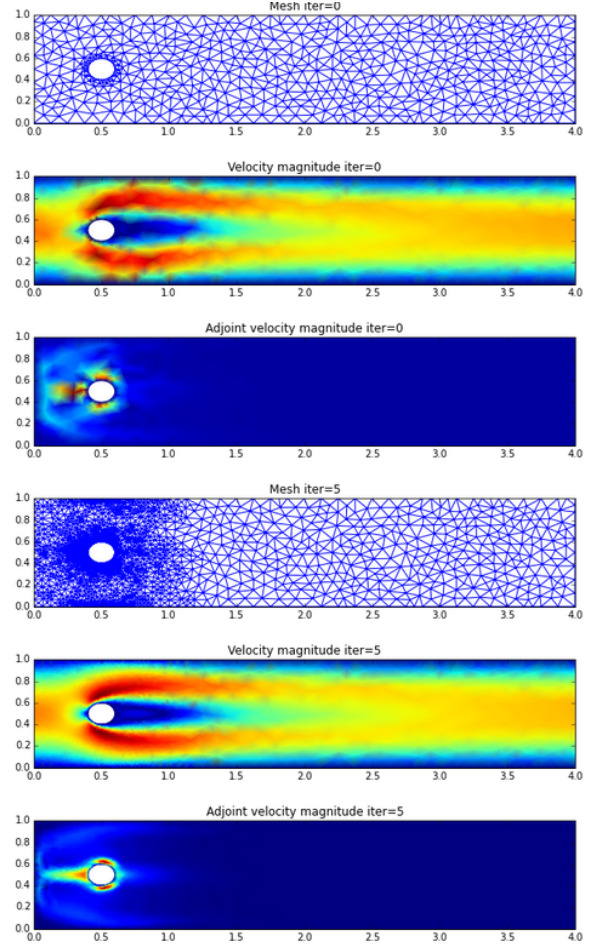


Figure 1: Plots of the mesh, primal velocity and adjoint velocity for iteration 1 and 6 in an adaptive refinement sequence for the goal: **M1** (drag).

In FEniCS notation this can be written:

```
r = (inner(grad(u)*u + grad(p),v) + nu*inner(grad(u), grad(v)) + div(u)*q)*dx
```

A Galerkin-Least Squares stabilized formulation can be written:

```
# Weak residual of stabilized FEM for Navier-Stokes eq.
r = ((inner(grad(u)*u + grad(p), v) + nu*inner(grad(u), grad(v)) + div(u)*q)*dx +
      gamma*(om*p*q + im*inner(u - uin, v) + nm*inner(u, v))*ds + # Weak boundary conditions
      d*(inner(grad(p) + grad(u)*u, grad(q) + grad(u)*v) + inner(div(u), div(v)))*dx) # Stabilization
```

The adjoint problem can be automatically generated and solved simply like so, here a linearized adjoint problem for the general case of a non-linear weak form:

```
# Generate the adjoint problem
a_star = adjoint(derivative(F, w))
L_star = M(mesh, v, q)

# Solve the adjoint problem
solve(a_star == L_star, phi)
```

and the error indicator **ei** can be expressed like so:

```
# Compute error indicators
z = TestFunction(Z)
(u, p) = w.split()
(phi_u, phi_p) = phi.split()
LR1 = r(W, w, z*phi, stab=False)
ei = Function(Z)
ei.vector[:] = assemble(LR1).array()
```

with  $Z$  the  $dG(0)$  finite element space of discontinuous piecewise constant functions.

The source code applying the do-nothing goal-oriented adaptive error control described above to stationary incompressible Navier-Stokes flow is available in the file:

**src/jupyter-fenics-adaptive-navierstokes01.ipynb** .

### 2.1.1 Standard exercise

Edit the file and try different data for the adaptivity:

1. Try different goal functionals by selecting which one to return in the **M()** function:

```
def M(mesh, u, p):
    n = FacetNormal(mesh)

    I = Identity(2)
    sigma = p*I - nu*epsilon(u)
    theta = Constant((1.0, 0.0))

    M1 = psimarker*p*n[0]*ds # Drag (only pressure)
    M2 = psimarker*p*n[1]*ds # Lift (only pressure)
    M3 = inner(psi, u)*dx # Mean of the velocity in a region
    M4 = psimarker*dot(dot(sigma, n), theta)*ds # Drag (full stress)
    M5 = u[0]*dx # Mean of the x-velocity in the whole domain

    return M1
```

What is the effect on the final mesh? For an advanced exercise, try defining your own goal functional.

2. Experiment with setting the refinement **adapt\_ratio** variable (**adapt\_ratio = 0.05** refines 5% of the cells in the mesh) for different goal functionals. Compare a low ratio (5% for example) with 100% (representing uniform refinement) with regard to efficiency.

3. Experiment with the viscosity `nu`. Try for example values of 0.1, 0.01 and 0.001.
4. Try different geometries.

## References

- [1] Kenneth Eriksson, Don Estep, Peter Hansbo, and Claes Johnson. *Computational Differential Equations*. Cambridge University Press New York, 1996.
- [2] FEniCS. Fenics project. <http://www.fenicsproject.org>, 2003.
- [3] Johan Hoffman. Lecture notes for dd2263 methods in scientific computing, 2017.
- [4] Johan Hoffman, Johan Jansson, Rodrigo Vilela de Abreu, Niyazi Cem Degirmenci, Niclas Jansson, Kaspar Müller, Murtazo Nazarov, and Jeannette Hiromi Spühler. Unicorn: Parallel adaptive finite element simulation of turbulent flow and fluid-structure interaction for deforming domains and complex geometry. *Computers and Fluids*, 2012.
- [5] Johan Hoffman and Claes Johnson. *Computational Turbulent Incompressible Flow: Applied Mathematics Body and Soul Vol 4*. Springer-Verlag Publishing, 2006.
- [6] Johan Jansson, Johan Hoffman, and Cem Degirmenci. Adaptive error control in finite element methods using the error representation as error indicator. Technical report, KTH, High Performance Computing and Visualization (HPCViz), 2013. QC 20131118.