MEK4250 MANDATORY EXERCISE 1

Soran Hussein Mohmmed / soranhm

April 2018

Equivalence of norms

To show that H^1 norm and semi-norm are equivalent. To prove equivalent, I need to show that it holds for $\leq$ and $\geq$.

$$|u|_1 \leq \|u\|_1 \quad \text{and} \quad |u|_1 \geq \|u\|_1$$

Few equations:

$$\begin{aligned}
|u|_1^2 &= \int_\Omega (\nabla u)^2 dx \\
\|u\|_1^2 &= \int_\Omega u^2 + (\nabla u)^2 dx \\
\text{Poincarê inequality:} \quad \|u\|_{L^2(\Omega)} &\leq C|u|_{H^1(\Omega}
\end{aligned}$$

Starting to prove $\leq$:

$$\begin{aligned}
&\|u\|_1 \leq |u|_1 : \\
&\|u\|_1^2 = \int_\Omega u^2 + (\nabla u)^2 dx =< u, u > + < \nabla u, \nabla u >= \|u\|_{L^2}^2 + \|\nabla u\|_{L^2}^2 \\
&\overset{Poincare}{\leq} \quad C\|\nabla u\|_{L^2}^2 + \|\nabla u\|_{L^2}^2 = (C+1)\|\nabla u\|_{L^2}^2 \\
&= \quad (C+1)|u|_1^2 \quad \Rightarrow \|u\|_1 \leq \sqrt{(C+1)}|u|_1 \quad \Rightarrow \underline{\underline{\|u\|_1 \leq |u|_1}}
\end{aligned}$$

Now proving $\geq$:

$$\begin{aligned}
&|u|_1 \leq \|u\|_1 : \\
&\|u\|_1 = \int_\Omega (\nabla u)^2 dx \leq \int_\Omega u^2 + (\nabla u)^2 dx = \|u\|_1 \quad \Rightarrow \\
&\underline{\underline{|u|_1 \leq \|u\|_1}}
\end{aligned}$$

Now we have proved that $\|u\|_1 \leq |u|_1$ and $\|u\|_1 \geq |u|_1$. Hencs we have $\|u\|_1 \sim |u|_1$

Stokes problem

Now we are going to study the Stokes problem:

$$\begin{aligned} -\Delta u + \nabla p =& f, \quad \text{in} \quad \Omega \qquad \text{Momentum equation} \\ \nabla \cdot u =& 0, \quad \text{in} \quad \Omega \qquad \text{Continuity equation} \\ u =& g_D, \quad \text{on} \quad \partial\Omega_D \\ \frac{\partial u}{\partial n} - pn =& h_N, \quad \text{on} \quad \partial\Omega_N \end{aligned}$$

- U is the fluid velocity and p is the pressure
- F is a given body force per unit volume
- g_D is a given boundary flow
- h_N is a given function for the natural boundary condition

Here f is body force, $\partial\Omega_D$ is the Dirichlet boundary, while $\partial\Omega_N$ is the Neumann boundary. Furthermore, g_D is the prescribed fluid velocity on the Dirichlet boundary, and h_N is the surface force or stress on the Neumann boundary

Exercise 7.1

The point of this Exercise is to show three conditions:

$$a(u_h, v_h) \leq C_1 \|u_h\|_{V_h} \|v_h\|_{V_h}, \quad \forall u_h, v_h \in V_h \tag{1}$$
$$b(u_h, q_h) \leq C_2 \|u_h\|_{V_h} \|q_h\|_{Q_h}, \quad \forall u_h \in V_h, q_h \in Q_h \tag{2}$$
$$a(u_h, u_h) \geq C_3 \|u_h\|^2_{V_h}, \quad \forall u_h \in Z_h \tag{3}$$

I need to show those conditions when $V_h = H^1_0$ and $Q_h = L^2$. u is a vector who looks like this $u = (u_1, u_2, u_3)$, same for v. Conditions (1) is straight forward, using

Cauchy–Schwarz inequality:

$$
\begin{aligned}
a(u,v) =< \nabla u, \nabla v > = \int_\Omega \nabla u : \nabla v dx = \int_\Omega \Big(& u_{1x}v_{1x} + u_{1y}v_{1y} + u_{1z}v_{1z} + u_{2x}v_{2x} \\
& + u_{2y}v_{2y} + u_{2z}v_{2z} + u_{3x}v_{3x} + u_{3y}v_{3y} + u_{3z}v_{3z} \Big) dx \\
& =< u_{1x}, v_{1x} > + < u_{1y}, v_{1y} > + < u_{1z}, v_{1z} > + ... + < u_{3z}, v_{3z} > \\
& \leq \|u_{1x}\| \|v_{1x}\| + \left\|u_{1y}\right\| \left\|v_{1y}\right\| + \|u_{1z}\| \|v_{1z}\| + ... + \|u_{3z}\| \|v_{3z}\| \\
& \Big(\|u_{1x}\| \leq \|\nabla u\| \quad \text{and} \quad \|v_{1x}\| \leq \|\nabla v\| \Big) ... \\
& \leq \|\nabla u\| \|\nabla v\| + \|\nabla u\| \|\nabla v\| + \|\nabla u\| \|\nabla v\| + ... + \|\nabla u\| \|\nabla v\| \\
& = 9 \|\nabla u\| \|\nabla v\| \\
a(u,v) & \underline{\underline{\leq 9 \|\nabla u\| \|\nabla v\|}}
\end{aligned}
$$

This means that $C_1 = 9$. The same applies for condition (2) with same u, aswell:

$$
\begin{aligned}
b(u,q) & =< \nabla \cdot u, q >= \int_\Omega q \nabla \cdot u dx \\
& = \int_\Omega \Big(u_{1x}q + u_{2y}q + u_{3z}q \Big) dx \\
& =< u_{1x}, q > + < u_{2y}, q > + < u_{3z}, q > \\
& \leq \|u_{1x}\| \|q\| + \left\|u_{2y}\right\| \|q\| + \left\|u_{3y}\right\| \|q\| = \|q\| \Big(\|u_{1x}\| + \left\|u_{2y}\right\| + \|u_{3z}\| \Big) \\
& \Big(\|u_{1x}\| \leq \|\nabla u\|, \quad \left\|u_{2y}\right\| \leq \|\nabla u\| \quad \text{and} \quad \|u_{3z}\| \leq \|\nabla u\| \Big) \\
& \leq \|q\| \Big(\|\nabla u\| + \|\nabla u\| + \|\nabla u\| \Big) = 3 \|q\| \|\nabla u\| \\
b(u,q) & \underline{\underline{\leq 3 \|q\| \|\nabla u\|}}
\end{aligned}
$$

This gives that $C_2 = 3$. The last condition is a bit more complicated, it is easier to show contition (3) if we solve it this way: $\leftarrow \geq \leftarrow$, and using H^1 norms:

$$
\begin{aligned}
C_4 \|u\|_1^2 & = C_4 \int_\Omega u^2 + (\nabla u)^2 dx = C_4 < u, u > + C_4 < \nabla u, \nabla u > \\
& \leq C_4 \|u\|^2 + C_4 \|\nabla u\|^2 \overset{Poincare}{\leq} (C_4 + C) \|\nabla u\|^2 + C_4 \|\nabla u\|^2 \\
& = (2C_4 + C) \|\nabla u\|^2 = (2C_4 + C) a(u,u) \\
a(u,u) & \underline{\underline{\geq \frac{C_4}{(2C_4 + C)} \|u\|_1^2}}
\end{aligned}
$$

Her we have that $C_3 = \frac{C_4}{(2C_4 + C)}$.

Exercise 7.6

The last exercise is implementing the problem $u = (sin(\pi x), cos(\pi y))$, $p = sin(2\pi x)$ and $f = -\Delta u - \nabla p$ and testing whether the approximation is of the expected order for $P_4 - P_3$, $P_4 - P_2$, $P_3 - P_2$ and $P_3 - P_1$. Starting by solving f:

$$
\begin{aligned}
f =& - \Delta u - \nabla p : \\
\Delta u =& \nabla^2 u = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = \begin{pmatrix} 0 \\ -\pi^2 cos(\pi x) \end{pmatrix} + \begin{pmatrix} -\pi^2 sin(\pi y) \\ 0 \end{pmatrix} \\
=& \begin{pmatrix} -\pi^2 sin(\pi y) \\ -\pi^2 cos(\pi x) \end{pmatrix} \\
\nabla p =& \begin{pmatrix} \frac{\partial p}{\partial x} \\ \frac{\partial p}{\partial y} \end{pmatrix} = \begin{pmatrix} 2\pi cos(2\pi x) \\ 0 \end{pmatrix} \\
f =& - \begin{pmatrix} -\pi^2 sin(\pi y) \\ -\pi^2 cos(\pi x) \end{pmatrix} - \begin{pmatrix} 2\pi cos(2\pi x) \\ 0 \end{pmatrix} = \begin{pmatrix} \pi^2 sin(\pi y) - 2\pi cos(2\pi x) \\ \pi^2 cos(\pi x) \end{pmatrix}
\end{aligned}
$$

To implement this on FEniCS i need to solve it on weak form. Multiplying the momentum equation by a test function v and integrate by parts:

$$
\begin{aligned}
\int_\Omega \nabla u : \nabla v dx - \int_\Omega p \nabla \cdot v dx =& \int_\Omega f \cdot v dx + \int_{\partial\Omega_N} g_N \cdot v dx \\
\underbrace{< \nabla u, \nabla v >}_{\text{a(u,v)}} - \underbrace{< \nabla \cdot v, p >}_{\text{b(v,p)}} =& \underbrace{< f, v > + < g_N, v >_{\partial\Omega_N}}_{\text{L(v)}} \\
\underbrace{\pm < \nabla \cdot u, q >}_{\text{b(u,q)}} =& 0 \qquad \text{continutity equation}
\end{aligned}
$$

This sums up to: $A(u, p; v, q) := a(u, v) + b(v, p) + b(u, q) = L(v)$. Since the exercise don't say anything about the boundary, I use a simple Dirichlet boundary all around, then $< g_N, v >= 0$. The implementation of this is straight forward and is inserted down below, but the full code includes many more functions and is included in the delivery. The formula i use to calculate the abstract error estimate is:

$$
\begin{aligned}
\|u - u_h\|_1 + \|p - p_h\|_0 \leq& Ch^k \|u\|_{k+1} + Dh^{l+1} \|p\|_{l+1} \\
\|u - u_h\|_1 + \|p - p_h\|_0 \leq& Ch^k (\|u\|_{k+1} + \|p\|_k) \qquad \text{(When k = l + 1)}
\end{aligned}
$$

I calculated C when k = l + 1 and use the average from all the different mesh runs to find D when l is one less. To calculate D in $P_4 - P_2$ I need to use the average C from $P_4 - P_3$, same for D in $P_3 - P_1$ I need to use the average C from $P_3 - P_2$. I have calculated

$\|u\|_{k+1}$ and $\|p\|_{l+1}$ analytically with the formula: $< u, v >_{H^k} = \sum_{i=0}^{k} < D^i u, D^i v >_{L^2}$:

$$
\begin{aligned}
\|u\|_5 &= \Big(< u, u > + < \Delta u, \Delta u > + < \nabla u, \nabla u > \\
&\quad + < D^3 u, D^3 u > + < D^4 u, D^4 u > + < D^5 u, D^5 u > \Big)^{\frac{1}{2}} \\
\Delta u &= (\pi cos(\pi y), -\pi sin(\pi x) \\
\nabla u &= \Delta^2 u = (-\pi^2 sin(\pi y), -\pi^2 cos(\pi x)) \\
D^3 u &= (-\pi^3 cos(\pi y), \pi^3 sin(\pi x)) \\
D^4 u &= (\pi^4 sin(\pi y), \pi^4 cos(\pi x)) \\
D^5 u &= (\pi^5 cos(\pi y), -\pi^5 sin(\pi x)) \\
\|p\|_4 &= \Big(< p, p > + < \Delta p, \Delta p > + < \nabla p, \nabla p > \\
&\quad + < D^3 p, D^3 p > + < D^4 p, D^4 p > \Big)^{\frac{1}{2}} \quad \text{(Straight forward caluclation)}
\end{aligned}
$$

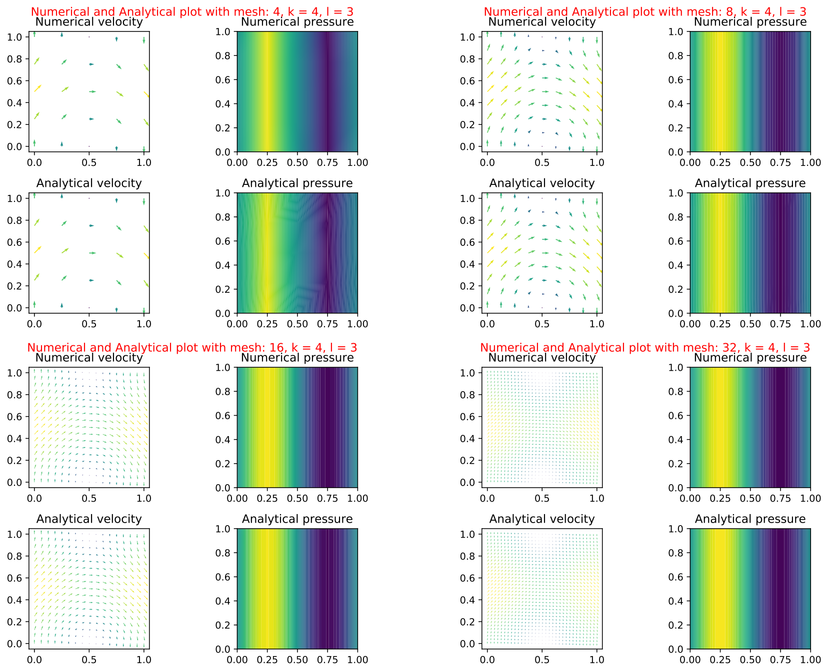
Listing 1: Calculating error:

```
    if (error):
        """ Calculating error estimate """
        #LHS
        u_uh = sqrt(assemble(inner(U_analytical - U,U_analytical - U)*dx))
        p_ph = sqrt(assemble(inner(P_analytical - P,P_analytical - P)*dx))
        u_uh_p_ph = u_uh + p_ph
        #RHS
        #Analytic: ... (Look at the file:"oblig.py")
        h_k = (1./m)**k
        h_l = (1./m)**(l+1)
        C = u_uh_p_ph/(h_k*u_k5*p_l4)
        return h_k,h_l,u_uh_p_ph,C
```

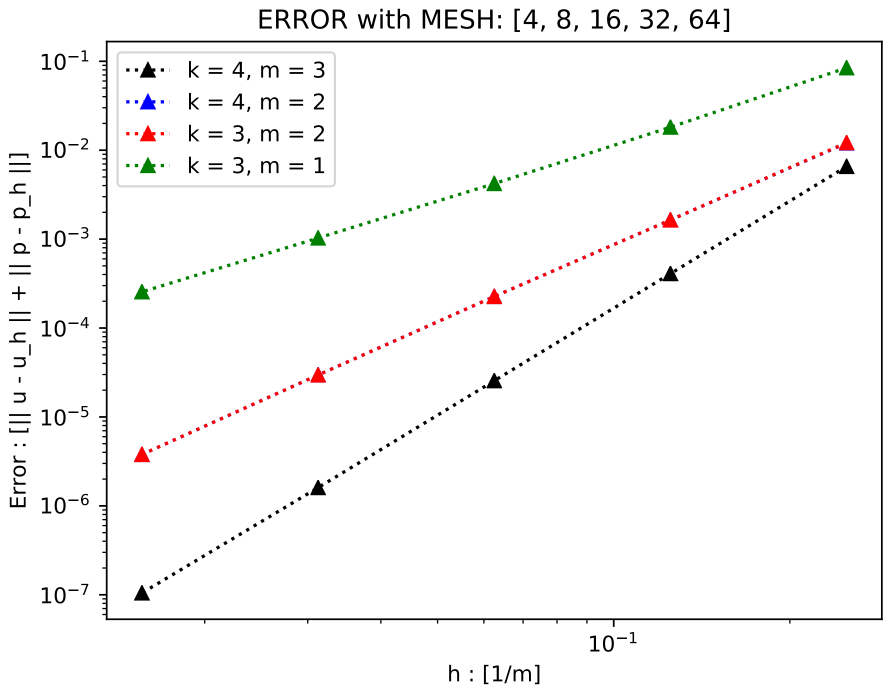
As we see in the example run below, that C and D getting smaller and smaller as Mesh increase. A run of the program with all the order and mesh = [4, 8, 16 ,32, 64] and plot for m = [4, 8, 16 ,32], k = 4 and l = 3. To look at the plots better, you could run savefigure, which saves the figure for Paraview:

```
soran@soran > python3 oblig.py
Enter a multiple MESH values, seperate with space:4 8 16 32 64
Choose: plot, error or/and savefigure:,separated by space:error
Printing out the error for all the given m,k and l values ...

+--------------------------P_4 - P_3--------------------------+
|  || u - u_h || + || p - p_h || =< C*h^4*(||u||_5 + ||p||_4)  |
|     Mesh =   4 : 6.56e-03   =< C*5.7636 --> C >= 4.51e-06    |
|     Mesh =   8 : 4.09e-04   =< C*1.4252 --> C >= 9.41e-07    |
|     Mesh =  16 : 2.55e-05   =< C*0.3827 --> C >= 2.09e-07    |
|     Mesh =  32 : 1.60e-06   =< C*0.3165 --> C >= 1.57e-08    |
|     Mesh =  64 : 1.05e-07   =< C*0.0055 --> C >= 5.88e-08    |
```

```
+--------------------------------------------------------------+

+---------------------------------P_4 - P_2---------------------------------+
|       || u - u_h || + || p - p_h || =< C*h^4*||u||_5 + D*h^3*||p||_3       |
|  Mesh =   4 : 1.20e-02   =< 1.15e-06*1.2610 + D*2.8275 --> D >= 5.36e-03   |
|  Mesh =   8 : 1.64e-03   =< 1.15e-06*0.0788 + D*0.3500 --> D >= 3.69e-04   |
|  Mesh =  16 : 2.26e-04   =< 1.15e-06*0.0049 + D*0.0497 --> D >= 2.23e-05   |
|  Mesh =  32 : 2.96e-05   =< 1.15e-06*0.0003 + D*0.0145 --> D >= 6.07e-07   |
|  Mesh =  64 : 3.78e-06   =< 1.15e-06*0.0000 + D*0.0014 --> D >= 3.54e-08   |
+---------------------------------------------------------------------------+

+-------------------------P_3 - P_2-------------------------+
|  || u - u_h || + || p - p_h || =< C*h^3*(||u||_4 + ||p||_3)  |
|     Mesh =   4 : 1.21e-02   =< C*7.2439 --> C >= 2.10e-05    |
|     Mesh =   8 : 1.64e-03   =< C*1.3830 --> C >= 1.35e-05    |
|     Mesh =  16 : 2.26e-04   =< C*1.3668 --> C >= 1.64e-06    |
|     Mesh =  32 : 2.96e-05   =< C*0.2016 --> C >= 1.45e-06    |
|     Mesh =  64 : 3.78e-06   =< C*0.0266 --> C >= 1.40e-06    |
+--------------------------------------------------------------+

+---------------------------------P_3 - P_1---------------------------------+
|       || u - u_h || + || p - p_h || =< C*h^3*||u||_4 + D*h^2*||p||_2       |
|  Mesh =   4 : 8.44e-02   =< 7.79e-06*1.6055 + D*79.519 --> D >= 1.70e-03   |
|  Mesh =   8 : 1.81e-02   =< 7.79e-06*0.2007 + D*28.033 --> D >= 1.30e-04   |
|  Mesh =  16 : 4.21e-03   =< 7.79e-06*0.0251 + D*23.770 --> D >= 4.44e-06   |
|  Mesh =  32 : 1.03e-03   =< 7.79e-06*0.0031 + D*8.611  --> D >= 3.71e-07   |
|  Mesh =  64 : 2.55e-04   =< 7.79e-06*0.0004 + D*7.921  --> D >= 1.22e-08   |
+---------------------------------------------------------------------------+
```



In the loglog plot down belwo we can se that the bigest choise of k and l gives least error. We can see that the error is linear and $P_4 - P_2$, $P_3 - P_2$ are very similar:



The full program is very long and includes most of the calculation of C, D, $\|u\|_{k+1}$, $\|p\|_{l+1}$, the error and much more. My pc does not have enough memory to run the program with big mesh and the most of the calculations should have been calculated by Sympy.

Listing 2: Most important part of the code:

```
from fenics import *
import numpy as np

def oblig(m,k,l, plo = False, savef = False, error = False):
    """ Function to calculate u and v , and plot them """
    def u_boundary(x, on_boundary):
        return on_boundary #x[0] < DOLFIN_EPS or x[0] > 1.0 - DOLFIN_EPS

    mesh = UnitSquareMesh(m,m)

    # Define function spaces
    V = VectorElement("Lagrange", mesh.ufl_cell(), k)
    Q = FiniteElement("Lagrange", mesh.ufl_cell(), l)
    TH = V * Q
    W = FunctionSpace(mesh, TH)

    u, p = TrialFunctions(W)
    v, q = TestFunctions(W)
    # Calculated f
```

```
    f = Expression(("pi*pi*sin(pi*x[1])-2*pi*cos(2*pi*x[0])","pi*pi*cos(pi*x[0])"),pi = np.pi,degree =2)

    #given u and p
    u_analytical = Expression(("sin(pi*x[1])", "cos(pi*x[0])"),pi = np.pi, degree =k+1)
    p_analytical = Expression(("sin(2*pi*x[0])"),pi = np.pi,degree =l+1)

    bc_u = DirichletBC(W.sub(0), u_analytical, u_boundary)
    bc = [bc_u]

    # A(u,p;v,q) := a(u,v) + b(v,p) + b(u,q) = L(v)
    a = inner(grad(u), grad(v))*dx + div(u)*q*dx + div(v)*p*dx
    L = inner(f, v)*dx
    UP = Function(W)
    A, b = assemble_system(a, L, bc)

    solve(A, UP.vector(), b, "lu")
    U, P = UP.split()

    P_average = assemble(P*dx)
    #problem with Element method, so used Space method
    V2 = VectorFunctionSpace(mesh, "Lagrange", k+1)
    Q2 = FunctionSpace(mesh, "Lagrange", l+1)
    U_analytical = project(u_analytical, V2)
    P_analytical = project(p_analytical + P_average, Q2)

    return
```