MEK4250
Mandatory assignment 2

Krister Stræte Karlsen

April 26, 2016

The programs belonging to this assignment is available at
https://github.com/krikarls/MEK4250

1 Stokes equation

1.1 Analysis of the weak formulation (Exercise 7.1)

In this exercise we want to show some of the properties that must be satisfied in order to have a well posed Stokes problem using a mixed formulation: $u, v \in V = H^1$, $p, q \in Q = L_2 = H_0$.

Continuity of $a(u, v)$

In order to establish *continuity*, or *boundedness*, of $a(u, v)$ we must have that

$$\int \nabla u : \nabla v dx \leq C||u||_1||v||_1. \tag{1.1}$$

We square both sides

$$\left(\int \nabla u : \nabla v dx\right)^2 \leq (C||u||_1||v||_1)^2 \tag{1.2}$$

and claim that if (1.2) holds, so does (1.1). Next we apply *Cauchy-Schwartz'*:

$$\begin{aligned}
\left(\int \nabla u : \nabla v dx\right)^2 &\leq ||\nabla u||_0^2||\nabla v||_0^2 \\
&\leq 2||\nabla u||_0^2||\nabla v||_0^2 + 2||\nabla u||_0^2||v||_0^2 + 2||u||_0^2||\nabla v||_0^2 + 2||u||_0^2||v||_0^2 \\
&= 2\left(||u||_0^2 + ||\nabla u||_0^2\right)\left(||v||_0^2 + ||\nabla v||_0^2\right) \\
&= C||u||_1^2||v||_1^2
\end{aligned}$$

We are now happy and move on to show continuity of $b(u, q)$.

Continuity of $b(u, q)$

For continuity of $b(u, q)$ it is required that

$$\int q(\nabla \cdot u)dx \leq C||u||_1||q||_0.$$

Using *Cauchy-Schwartz'*

$$\int q(\nabla \cdot u)dx \leq ||q||_0||\nabla \cdot u||_0.$$

we realize that if

$$||\nabla \cdot u||_0 \leq C||u||_1.$$

holds we are on safe ground. Without any worries(since both sides are positive) both sides can be squared.

$$\begin{aligned} ||\nabla \cdot u||_0^2 &\leq ||u||_1^2 \\ &= (||u||_0 + ||\nabla u||_0) \\ &\leq \left((D|u|_1)^2 + ||\nabla u||_0^2\right) \\ &= \left((D||\nabla u||_0)^2 + ||\nabla u||_0^2\right) \\ &= (D^2 + 1)||\nabla u||_0^2 \end{aligned}$$

It is now sufficient to show that if $||\nabla \cdot u||_0^2 \leq D_2||\nabla u||_0^2$ holds we have boundedness of $b(u, q)$. We start by writing out both terms(here assuming 2D):

$$\begin{aligned} ||\nabla \cdot u||_0^2 &= \int \left(\frac{\partial u_1}{\partial x} + \frac{\partial u_2}{\partial y}\right)^2 dx \\ ||\nabla u||_0^2 &= \int \left(\frac{\partial u_1}{\partial x}\right)^2 + \left(\frac{\partial u_2}{\partial x}\right)^2 + \left(\frac{\partial u_1}{\partial y}\right)^2 + \left(\frac{\partial u_2}{\partial y}\right)^2 dx \end{aligned}$$

One can now see that by adding some carefully chosen positive terms to $||\nabla \cdot u||_0^2$ we obtain $||\nabla u||_0^2$, which ensures that $||\nabla \cdot u||_0^2$ must be smaller.

$$\begin{aligned} ||\nabla \cdot u||_0^2 &\leq \int \left(\frac{\partial u_1}{\partial x} + \frac{\partial u_2}{\partial y}\right)^2 + \left(\frac{\partial u_1}{\partial x} - \frac{\partial u_2}{\partial y}\right)^2 + 2\left(\frac{\partial u_2}{\partial x}\right)^2 + 2\left(\frac{\partial u_1}{\partial y}\right)^2 dx \\ &= 2\int \left(\frac{\partial u_1}{\partial x}\right)^2 + \left(\frac{\partial u_2}{\partial x}\right)^2 + \left(\frac{\partial u_1}{\partial y}\right)^2 + \left(\frac{\partial u_2}{\partial y}\right)^2 dx \\ &= D_2||\nabla u||_0^2 \end{aligned}$$

Coersivity of $a(u, q)$

Coersivity of $a(u, q)$ is established if the following inequality is fulfilled:

$$C||u||_1^2 \leq \int \nabla u : \nabla u dx.$$

We start by rewriting the left hand side and applying *Poincaré*

$$\begin{aligned} ||u||_1^2 &= ||u||_0^2 + ||\nabla u||_0^2 \\ &\leq (D||\nabla u||_0^2) + ||\nabla u||_0^2 \\ &= (D^2+1)||\nabla u||_0^2 \\ &= (D^2+1)\int \nabla u : \nabla u dx \end{aligned}$$

$C = 1/(D^2+1)$ and we are very happy.

1.2 Looking into the error estimate (Exercise 7.6)

To investigate the error estimate

$$||E_u||_1 + ||E_p||_0 \leq Ch^k||u||_{k+1} + Dh^{l+1}||p||_{l+1}$$

we define the *Stokes problem*:

$$-\Delta \mathbf{u} + \nabla p = \mathbf{f}, \quad \mathbf{u} = sin(\pi y)\mathbf{i} + cos(\pi x)\mathbf{j}.$$

Computing error and convergence rates for different combinations of polynomials we obtained the results below.

Table 1: Convergence rates for u and p respectively. Convergence rates for u are measured in H^1 while rates for p are measured in L_2.

$P_4 - P_3$		$P_4 - P_2$		$P_3 - P_2$		$P_3 - P_1$	
4.29515	4.07241	2.88045	2.91318	2.84922	2.91507	2.08926	2.17433
4.11510	4.02557	2.96138	2.97733	2.95217	2.97799	2.04644	2.08797
4.03899	4.00955	2.98766	2.99487	2.98475	2.99505	2.02451	2.04127

From figure 1 and table 1 we see that two of the polynomial-combinations, $P_4 - P_2$ and $P_3 - P_2$, yields the same results. Why is that? In search of answers we turn to the error estimate for the respective combinations of elements:

$$\begin{aligned} P_4 - P_2 : &\quad ||E_u||_1 + ||E_p||_0 \leq Ch^4||u||_5 + Dh^3||p||_3 \\ P_3 - P_2 : &\quad ||E_u||_1 + ||E_p||_0 \leq Ch^3\left(||u||_4 + ||p||_3\right) \end{aligned}$$

For the first error estimate, $P_4 - P_2$, the dominating term on the right hand side is the one related to the pressure, because the part related to the velocity tends to zero much faster as the mesh is refined. We are in a sense limited by the order of the polynomial we have for pressure and can obtain no better convergence than for $P_3 - P_2$.

The same reasoning goes for last two combinations of elements.

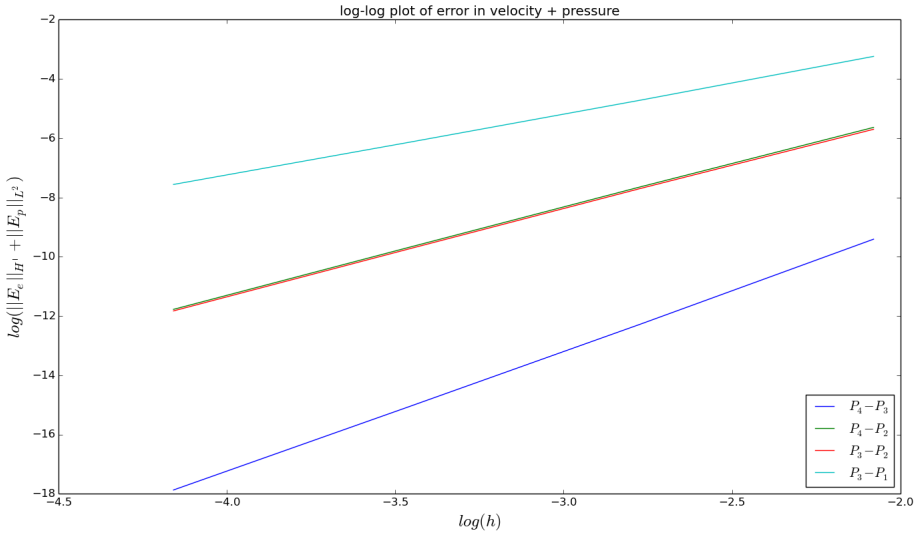

Figure 1: Log-log plot of error for different combinations of polynomials.

1.3 Approximation of the shear stress (Exercise 7.7)

In this section we will have a look at the order of approximation for the *wall shear stress*; that is, for a wall parallel to the $x - axis$

$$\tau = \mu \frac{du}{dy}.$$

We study this by computing the wall shear stress analytically, then numerically and last compute the error.

Table 2: Computed convergence rates r_n using a L^2-norm for different combinations of polynomials. Increase in n refers to refined mesh.

	$P_4 - P_3$	$P_4 - P_2$	$P_3 - P_2$	$P_3 - P_1$
r_1	4.136169	2.950468	3.025279	2.285654
r_2	4.048773	2.997513	3.080835	2.148821
r_3	4.021954	3.006015	3.065985	2.054565

By looking at the convergence rates(table 2) and the log-log plot of the error(figure 2) one can observe that error behaves like a H^1-norm, even though we used a L^2-norm to compute the error. That should come as no surprise to the shrewd reader, as the shear stress involves the derivative of the velocity, as does the H^1- norm. In other words; the results for the convergence rates here are of the same order as in the previous section(table 1) where we measured the error in velocity in H^1.

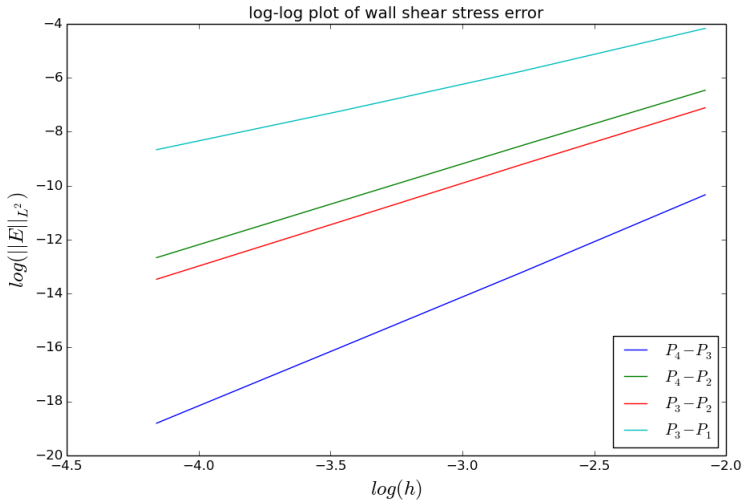

Figure 2: Log-log plot of the wall shear stress error(L^2) for different combinations of polynomials.

2 Linear elasticity

To study a phenomenon in numerical linear elasticity called *locking* we look a the following problem:

$$-\mu\Delta\mathbf{u} - \lambda\nabla\nabla\cdot\mathbf{u} = \mathbf{f} \quad in \quad \Omega = (0,1)^2, \tag{2.1}$$

$$\mathbf{u} = \left(\frac{\partial\phi}{\partial u}, -\frac{\partial\phi}{\partial x}\right) \quad on \quad \partial\Omega, \tag{2.2}$$

where $\phi = sin(\pi xy)$.

We start deriving and expression for the external forces $\mathbf{f}$.

Since $\nabla\cdot\mathbf{u} = 0$ we have that $\mathbf{f} = -\mathbf{u}\Delta u$. This can be computed using `sympy` or by hand calculations, like they did in the old days. We find that:

$$\mathbf{f} = \pi^2(\pi x(x^2+y^2)cos(\pi xy) + 2ysin(\pi xy))\mathbf{i}$$
$$-\pi^2(\pi y(x^2+y^2)cos(\pi xy) + 2xsin(\pi xy))\mathbf{j}.$$

Problem (2.1)-(2.2) will now be solved using finite element methods in FEniCS. Three different approaches for the $\lambda\nabla\nabla\cdot\mathbf{u}-$term will be tested and discussed. Here listed from stupid to clever:

1) Keep the $\lambda\nabla\nabla\cdot\mathbf{u}$-term as it is.
2) Integrate $\lambda\nabla\nabla\cdot\mathbf{u}$ by parts.
3) Define $p = \lambda(\nabla\cdot\mathbf{u})$ such that $\lambda\nabla\nabla\cdot\mathbf{u} = \nabla p$ and use a mixed finite element formulation.

Approach #1

We think no further, and ignore what we have learned, and implement and solve problem (2.1)-(2.2) in FEniCS as:

```
a = inner(grad(u),grad(v))*dx-lmbda*inner(grad(div(u)),v)*dx
L = inner(f,v)*dx
```

The results obtained by this approach are very poor and can be viewed in table 3 and 4.

Table 3: L2-errors from using approach #1 and second order polynomials.

$\lambda\backslash$N	8	16	32	64
1	0.014961	0.003950	0.001002	0.000251
10	0.725727	0.691354	0.026575	0.006363
100	3.750028	0.606093	0.650818	2.416408
1000	2.088700	0.823344	1.165615	0.514192
10000	1.426623	1.681094	3.957783	1.143778

Table 4: Convergence-rates using approach #1 and second order polynomials.

$\lambda \backslash$N	r_1	r_2	r_3
1	1.921346	1.979257	1.994746
10	0.070003	4.701297	2.062245
100	2.629290	-0.102715	-1.892538
1000	1.343039	-0.501525	1.180714
10000	-0.236797	-1.235292	1.790886

So what is happening?

We can definitely see that when λ gets bigger the numerical solution is bad. Only for $\lambda = 1$ are the results reasonable. The convergence rates are in this case stable, but lower than expected. If we plot and compare the solutions we see that the deformations/displacements computed numerically are less then they should be. One might even say that the deformation seems to be *"locked"*, see figure 3.

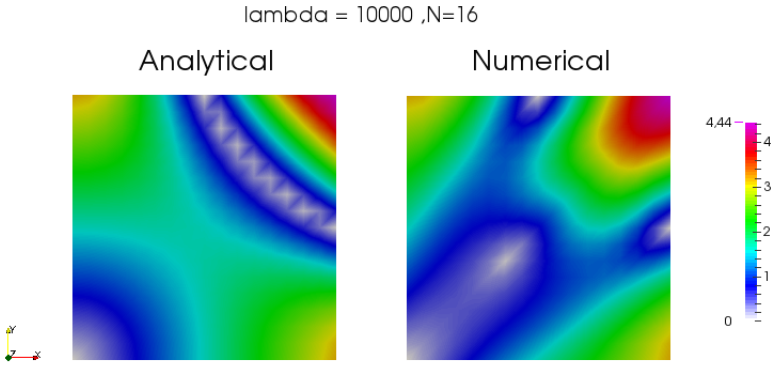

Figure 3: Comparing analytical and numerical solution for approach #1.

This is because we use elements that approximate the divergence poorly and when that term of the equation starts to dominate things fall apart. This phenomenon is called *locking*.

Approach #2

Now, let's integrate the bad term by parts and see what happens. The variational formulation now reads:

```
a = inner(grad(u),grad(v))*dx+lmbda*inner(div(u),div(v))*dx
L = inner(f,v)*dx
```

Table 5: L2-errors from using approach #2 and second order polynomials.

$\lambda \backslash$N	8	16	32	64
1	2.080473e-03	2.52145e-04	3.12446e-05	3.89691e-06
10	3.510135e-03	3.24355e-04	3.40039e-05	3.98917e-06
100	1.439727e-02	1.49814e-03	1.19452e-04	8.74370e-06
1000	2.699475e-02	5.14354e-03	6.89956e-04	6.37015e-05
10000	2.989324e-02	7.17495e-03	1.57723e-03	2.72196e-04

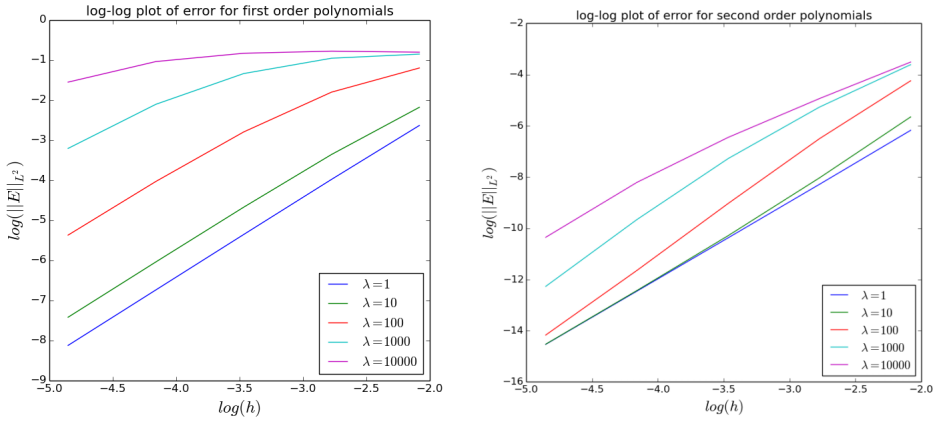

Figure 4: Log-log plot of error for first and second order polynomials for approach #2.

Table 6: Convergence-rates using approach #2 and second order polynomials.

$\lambda \backslash$N	r_1	r_2	r_3
1	3.044586	3.01258	3.00320
10	3.435881	3.25380	3.09154
100	3.264555	3.64867	3.77204
1000	2.391844	2.89819	3.43711
10000	2.058779	2.18558	2.53468

The solution has now improved, but that are still some problems. For $\lambda = 1$ everything works fine, the error decreases, the convergence rates are stable and of the expected order. For $\lambda \geq 10$ we can see fluctuations in the convergence rate, even though the L_2-errors are fairly low.

Next; let's try to be really clever!

Approach #3: Stabilization using a mixed finite element formulation

We define $p = \lambda(\nabla \cdot \mathbf{u})$ such that $\lambda \nabla \nabla \cdot \mathbf{u} = \nabla p$. Problem (2.1)-(2.2) can then be formulated

$$-\mu \Delta \mathbf{u} - \nabla p = \mathbf{f},$$
$$(\nabla \cdot \mathbf{u}) - \frac{1}{\lambda} p, = 0$$

with the weak formulation

$$\int_\Omega \nabla \mathbf{u} : \nabla \mathbf{v} dx + \int_\Omega p(\nabla \cdot \mathbf{v}) dx = \int_\Omega \mathbf{f} \cdot \mathbf{v} dx \quad \forall \mathbf{v} \in V,$$
$$\int_\Omega (\nabla \cdot \mathbf{u}) q dx - \frac{1}{\lambda} \int_\Omega p q dx = 0 \quad \forall q \in Q.$$

We implement this in FEniCS and look at the results with a happy face.

```
 a = inner(grad(u),grad(v))*dx + div(v)*p*dx
a2 = q*div(u)*dx - (1.0/lmbda)*p*q*dx
L = inner(f,v)*dx
```

Table 7: L2-errors using a mixed FEM formulation $P_2 - P_1$.

$\lambda \backslash$N	8	16	32	64
1	1.987451e-03	2.48875e-04	3.11388e-05	3.89358e-06
10	1.968511e-03	2.48207e-04	3.11171e-05	3.89289e-06
100	1.971773e-03	2.48317e-04	3.11211e-05	3.89304e-06
1000	1.973119e-03	2.48367e-04	3.11230e-05	3.89311e-06
10000	1.973277e-03	2.48373e-04	3.11233e-05	3.89312e-06

Table 8: Convergence-rates using a mixed FEM formulation $P_2 - P_1$.

$\lambda \backslash$N	r_1	r_2	r_3
1	2.997423	2.99864	2.99955
10	2.987488	2.99577	2.99879
100	2.989240	2.99621	2.99892
1000	2.989934	2.99642	2.99899
10000	2.990015	2.99644	2.99899

Our results are now very satisfactory. The numerical solution is now unaffected by the size of λ. We have successfully avoided *locking*.