Mandatory Exercise 2

Jørgen D. Tyvand

May 5, 2016

Exercise 1

7.1

We are to prove that the following conditions are satisfied for $V_h = H_0^1$ and $Q_h = L^2$

$$a(u_h, v_h) \leq C_1 |||u_h||_{V_h} |||v_h||_{V_h}, \qquad \forall\, u_h, v_h \in V_h \tag{7.14}$$

$$b(u_h, q_h) \leq C_2 |||u_h||_{V_h} |||q_h||_{Q_h}, \qquad \forall\, u_h \in V_h, q_h \in Q_h \tag{7.15}$$

$$a(u_h, u_h) \geq C_3 ||u_h||^2_{V_h}, \qquad \forall\, q_h \in Q_h \tag{7.16}$$

First we look at condition (7.14)

$$a(u_h, v_h) = \int_\Omega \nabla u_h : \nabla v_h \, dx = \langle \nabla u \,, \nabla v \rangle \leq |\langle \nabla u \,, \nabla v \rangle| \tag{1}$$

$$\leq ||\nabla u||_{L^2} \cdot ||\nabla v||_{L^2} \tag{2}$$

Next we look at condition (7.15)

$$b(u_h, q_h) = \int_\Omega \nabla \cdot u_h \, q \, dx \leq \sqrt{\int_\Omega (\nabla \cdot u_h \, q)^2 \, dx} \leq ||\nabla \cdot u_h \, q||_{L^2} \tag{3}$$

$$\leq ||q||_{L^2} ||\nabla \cdot u_h||_{L^2} \geq \frac{1}{2}(|u_h|^2_{H^1} + C||u_h||||^2_{L^2})$$

$$= C_3 ||u_h||^2_{H^1}$$

Finally we look at condition (7.16)

$$a(u_h, u_h) = \int_\Omega \nabla u_h : \nabla u_h \, dx = \int_\Omega (\nabla u_h)^2 \, dx = |u_h|^2_{H^1} \tag{4}$$

$$= \frac{1}{2}(|u_h|^2_{H^1} + |u_h|^2_{H^1}) \geq \frac{1}{2}(|u_h|^2_{H^1} + C||u_h||^2_{L^2})$$

$$= C_3 ||u_h||^2_{H^1}$$

7.6

We are to implement the Stokes problem using $u = (sin(\pi y), cos(\pi x)),$, $p = sin(2\pi x)$ and $f = -\Delta u - \nabla p$, and test whether the approximation

$$||u - u_h||_1 + ||p - p_h||_0 \leq C h^k ||u||_{k+1} + D k^{l+1} ||p||_{l+1}$$

where k and l are the polynomial degree of velocity and pressure.

The following program checks if the approximation holds

```
from dolfin import *
import matplotlib.pyplot as plt
import numpy as np
import sys
set_log_active(False)

plt.figure(1)
plt.figure(2)
for i in [[4,3], [4,2], [3,2], [3,1]]:
        print('P%d-P%d' %(i[0],i[1]))
```

```
        hvals = []
        UE = []
        PE = []
        for N in [2, 4, 8, 16, 32, 64]:
                h = 1./N
                hvals.append(h)
                mesh = UnitSquareMesh(N,N)
                V = VectorFunctionSpace(mesh, 'Lagrange', i[0])
                Q = FunctionSpace(mesh, 'Lagrange', i[1])

                W = MixedFunctionSpace([V, Q])

                u, p = TrialFunctions(W)
                v, q = TestFunctions(W)

                f = Expression(("pow(pi,2)*sin(pi*x[1]) - 2*pi*cos(2*pi*x[0])",\
                                "pow(pi,2)*cos(pi*x[0])"))

                u_ex = Expression(("sin(pi*x[1])","cos(pi*x[0])"))

                p_ex = Expression("sin(2*pi*x[0])")

                bc_u = DirichletBC(W.sub(0), u_ex, "on_boundary")
                bc_p = DirichletBC(W.sub(1), p_ex, "on_boundary")
                bc = [bc_u, bc_p]

                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 == L, up_, bc)

                u_, p_ = up_.split(True)

                Uerr = errornorm(u_ex,u_, norm_type='h1', degree_rise=1)
                UE.append(Uerr)
                Perr = errornorm(p_ex, p_, norm_type='l2', degree_rise=1)
                PE.append(Perr)

        Ur = []
        Pr = []
        for j in range(1,len(UE)):
                Uconv = np.log(UE[j-1]/UE[j])/np.log(hvals[j-1]/hvals[j])
                Ur.append(Uconv)
                Pconv = np.log(PE[j-1]/PE[j])/np.log(hvals[j-1]/hvals[j])
                Pr.append(Pconv)
                print ('h = %f    r_U = %f    r_P = %f' % (hvals[j], Uconv, Pconv))
        plt.figure(1)
        label = 'P%d-P%d' %(i[0],i[1])
        plt.loglog(hvals,UE, label=label)
        plt.figure(2)
        label = 'P%d-P%d' %(i[0],i[1])
        plt.loglog(hvals,PE, label=label)
        print('\n')


plt.figure(1)
plt.legend(loc='upper left')
plt.savefig('Velocity_convergence_P%d-P%d.png' % (i[0],i[1]))

plt.figure(2)
plt.legend(loc='upper left')
plt.savefig('Pressure_convergence_P%d-P%d.png' % (i[0],i[1]))
```

7.7

Exercise 2

We are looking ate the topic of 'locking', and are considering the following problem on the unit square domain $\Omega = (0, 1)^2$:

$$-\mu\Delta\mathbf{u} - \lambda\nabla\nabla\cdot\mathbf{u} = f \text{ in } \Omega \tag{5}$$

$$\mathbf{u} = \mathbf{u_e} \text{ on } \partial\Omega \tag{6}$$

where $\mathbf{u_e} = \left(\frac{\partial\phi}{\partial y}, -\frac{\partial\phi}{\partial x}\right)$, $\phi = sin(\pi xy)$ and $\nabla\cdot\mathbf{u_e} = 0$.

a)

We are to derive and expression for f. We do this by looking at equation (1) using the exact solution $\mathbf{u_e}$. Seeing as $\Delta\cdot\mathbf{u_e} = 0$, equation(1) reduces to

$$-\mu\Delta\mathbf{u_e} = f \tag{7}$$

We look at the Laplacian term

$$\Delta\mathbf{u_e} = \nabla^2\mathbf{u_e} = (\nabla^2 u, \nabla^2 v) = \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}, \frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right) \tag{8}$$

where u and v are the components of $\mathbf{u_e}$. We get

$$\begin{aligned}\nabla^2\mathbf{u_e} = &\Bigg(\frac{\partial}{\partial x}[-\pi cos(\pi xy) - \pi^2 xysin(\pi xy)] + \frac{\partial}{\partial y}[-\pi^2 x^2 sin(\pi xy)], &(9)\\ &\frac{\partial}{\partial x}[\pi^2 y^2 sin(\pi xy)] + \frac{\partial}{\partial y}[-\pi cos(\pi xy) + \pi^2 xysin(\pi xy)]\Bigg)\\ = &\Bigg(- \pi^2 ysin(\pi xy) - \pi^2 ysin(\pi xy) - \pi^3 xy^2 cos(\pi xy) - \pi^3 x^3 cos(\pi xy), &(10)\\ &\pi^3 y^3 cos(\pi xy) + \pi^2 xsin(\pi xy) + \pi^2 xsin(\pi xy) + \pi^3 x^2 ycos(\pi xy)\Bigg)\\ = &\Bigg(- \pi^2[2ysin(\pi xy) + \pi x(x^2 + y^2)cos(\pi xy)], &(11)\\ &\pi^2[2xsin(\pi xy) + \pi y(x^2 + y^2)cos(\pi xy)]\Bigg)\end{aligned}$$

Inserting this into equation (3) gives us

$$\begin{aligned}f = \mu\Bigg(&\pi^2[2ysin(\pi xy) + \pi x(x^2 + y^2)cos(\pi xy)], &(12)\\ &-\pi^2[2xsin(\pi xy) + \pi y(x^2 + y^2)cos(\pi xy)]\Bigg)\end{aligned}$$

b)

Next we want to compute the numerical error for $\lambda = 1, 100, 10000$ at $h = 8, 16, 32, 64$ for polynomial order 1 and 2.

The following code computes and outputs the numerical error

```python
from dolfin import *
import sys
set_log_active(False)


lvals = [1., 100., 10000.]
nvals = [8,16,32,64]

for i in [0,1]:
        print("P%d Elements\n" % (i+1))
        for l in lvals:
                for N in nvals:

                        mesh = UnitSquareMesh(N,N)

                        h = 1.0/N

                        V = VectorFunctionSpace(mesh, 'Lagrange', i+1)
                        V_1 = VectorFunctionSpace(mesh, 'Lagrange', i+1)

                        u = TrialFunction(V)
                        v = TestFunction(V)


                        u_ex = interpolate(Expression(("pi*x[0]*cos(pi*x[0]*x[1])",\
                                        "-pi*x[1]*cos(pi*x[0]*x[1])")),V_1)

                        bc = DirichletBC(V, u_ex, "on_boundary")
                        bcs = [bc]

                        mu = 1.

                        f = interpolate(Expression(("mu * pow(pi,2) \
                                                * (2 * x[1] * sin(pi*x[0]*x[1])\
                                                + pi * x[0] *(pow(x[0],2) + \
                                                pow(x[1],2))*cos(pi*x[0]*x[1]))"
                                                , "-mu * pow(pi,2) *\
                                                (2 * x[0] * sin(pi*x[0]*x[1]) + pi * x[1] \
                                                *(pow(x[0],2) + pow(x[1],2)) *\
                                                cos(pi*x[0]*x[1]))"),    mu=mu),V_1)


                        a = mu*inner(grad(u),grad(v))*dx + l*inner(div(u),div(v))*dx
                        L =      dot(f,v)*dx
                        u_ = Function(V)

                        solve(a == L, u_, bcs)

                        err = errornorm(u_, u_ex, norm_type='l2', degree_rise=1)
                        print ('h = %f    lambda = %-5d   error = %e' % (h, l, err))
                print
        print
```

The output from the program is listed below

```
P1  Elements

h = 0.125000    lambda = 1       error = 3.665490e-02
h = 0.062500    lambda = 1       error = 9.893346e-03
h = 0.031250    lambda = 1       error = 2.523885e-03
h = 0.015625    lambda = 1       error = 6.342241e-04

h = 0.125000    lambda = 100     error = 2.909305e-01
h = 0.062500    lambda = 100     error = 1.624590e-01
h = 0.031250    lambda = 100     error = 6.039903e-02
h = 0.015625    lambda = 100     error = 1.756415e-02

h = 0.125000    lambda = 10000   error = 4.379122e-01
h = 0.062500    lambda = 10000   error = 4.551903e-01
h = 0.031250    lambda = 10000   error = 4.327456e-01
h = 0.015625    lambda = 10000   error = 3.518862e-01


P2  Elements

h = 0.125000    lambda = 1       error = 6.622818e-04
h = 0.062500    lambda = 1       error = 4.404308e-05
h = 0.031250    lambda = 1       error = 2.810815e-06
h = 0.015625    lambda = 1       error = 1.769528e-07

h = 0.125000    lambda = 100     error = 1.425217e-02
h = 0.062500    lambda = 100     error = 1.477785e-03
h = 0.031250    lambda = 100     error = 1.154063e-04
h = 0.015625    lambda = 100     error = 7.836387e-06

h = 0.125000    lambda = 10000   error = 2.981601e-02
h = 0.062500    lambda = 10000   error = 7.169928e-03
h = 0.031250    lambda = 10000   error = 1.576881e-03
h = 0.015625    lambda = 10000   error = 2.721675e-04
```