Mandatory Assignment 2, MEK 4250

Farnaz Rezvany

May 3, 2017

Exercise 7.1

There are 3 conditions for well-posedness of Stokes problem, using a mixed formulation.These are:

Boundness of a:

$$a(u_h, v_h) \leq C_1||u_h||_{V_h}||v_h||_{V_h}, \quad \forall \ u_h, v_h \epsilon V_h \qquad (1)$$

Boundness of b:

$$b(u_h, q_h) \leq C_2||u_h||_{V_h}||q_h||_{Q_h}, \quad \forall \ u_h \epsilon V_h, q_h \epsilon Q_h \qquad (2)$$

Coersivity of a:

$$a(u_h, u_h) \leq C_3||u_h||^2_{V_h}, \quad \forall \ u_h \epsilon V_h \qquad (3)$$

In this exercise we want to show that these 3 conditions are satisfied for $V_h = H^1_0$ and $Q_h = L^2$

We are going to use the Cauchy-Schwartz inequality:

$$| < u, v > | \leq ||u||||v||$$

and the Poincare inequality:

$$||u||_{L^2} \leq C||\nabla u||_{L^2} = C|u|_{H^1} \leq C||u||_{H_1}$$

For condition (1), we need to show that :

$$a(u_h, v_h) =\leq C_1||u_h||_{H^1}||v_h||_{H^1}$$

$$\begin{aligned} a(u_h, v_h) = \int \nabla u_h \colon \nabla v_h dx \leq \sqrt{\int (\nabla u \colon \nabla v)^2 dx} \\ = | < \nabla u_h, \nabla v_h > | \overset{\text{CS}}{\leq} ||\nabla u_h||_{L^2}||\nabla v_h||_{L_2} \\ = C_1|u|_{H^1}|v|_{H^1} \overset{\text{PC}}{\leq} C_1||u_h||_{H^1}||v_h||_{H^1} \end{aligned}$$

For condition (2), we need to show that :

$$b(u_h, q_h) \leq C_2||u_h||_{H^1}||q_h||_{L^2}$$

$$\begin{aligned} b(u_h, q_h) = \int q_h \nabla \cdot u_h dx \leq \sqrt{\int_\Omega (q \nabla \cdot u)^2 dx} \\ = | < q_h, \nabla \cdot u_h > | \overset{\text{CS}}{\leq} ||q_h||_{L^2}||\nabla \cdot u_h||_{L^2} \end{aligned}$$

Need to prove that

$$||\nabla \cdot u_h||_{L^2} \leq C||u_h||_{H^1} = ||u_h||_{L^2} + ||\nabla u||_{L^2}$$

In 2D we have:

$$||\nabla u||_{L^2}^2 = \int \left(\frac{\partial u_1}{\partial x}\right)^2 + \left(\frac{\partial u_2}{\partial x}\right)^2 + \left(\frac{\partial u_1}{\partial y}\right)^2 + \left(\frac{\partial u_2}{\partial y}\right)^2 dx \tag{3}$$

$$||\nabla \cdot u||_{L^2}^2 = \int \left(\frac{\partial u_1}{\partial x} + \frac{\partial u_2}{\partial y}\right)^2 dx = \int \left(\left(\frac{\partial u_1}{\partial x}\right)^2 + 2\frac{\partial u_1}{\partial x}\frac{\partial u_2}{\partial y} + \left(\frac{\partial u_2}{\partial y}\right)^2\right) dx \tag{4}$$

$$\int \left(\frac{\partial u_1}{\partial x} - \frac{\partial u_2}{\partial y}\right)^2 dx = \int \left(\left(\frac{\partial u_1}{\partial x}\right)^2 - 2\frac{\partial u_1}{\partial x}\frac{\partial u_2}{\partial y} + \left(\frac{\partial u_2}{\partial y}\right)^2\right) dx \tag{5}$$

$$(4) + (5) = 2\int \left(\left(\frac{\partial u_1}{\partial x}\right)^2 + \left(\frac{\partial u_2}{\partial y}\right)^2\right) dx \tag{6}$$

We can see that we have to add a positive term to $||\nabla \cdot u||_{L^2}^2$to get $||\nabla u||_{L^2}^2$, then $||\nabla \cdot u||_{L^2}^2$ must be smaller than $||\nabla u||_{L^2}^2$.

$$\begin{aligned}
||\nabla \cdot u||_{L^2}^2 &\leq \int \left(\frac{\partial u_1}{\partial x} + \frac{\partial u_2}{\partial y}\right)^2 + \left(\frac{\partial u_1}{\partial x} - \frac{\partial u_2}{\partial y}\right)^2 + 2\left(\frac{\partial u_2}{\partial x}\right)^2 + 2\left(\frac{\partial u_1}{\partial y}\right)^2 dx \\
&= 2\int \left(\frac{\partial u_1}{\partial x}\right)^2 + \left(\frac{\partial u_2}{\partial x}\right)^2 + \left(\frac{\partial u_1}{\partial y}\right)^2 + \left(\frac{\partial u_2}{\partial y}\right)^2 dx \\
&= C_2||\nabla u||_{L^2}^2
\end{aligned}$$

For condition (3), we need to show that:

$$a(u_h, u_h) \leq C_3||u_h||_{H^1}^2$$

$$\begin{aligned}
a(u_h, u_h) = \int_\Omega \nabla u_h \colon \nabla u_h \, dx &= \sqrt{\int_\Omega (\nabla u_h \colon \nabla u_h)^2 dx} \\
&= ||\nabla u \colon \nabla u||_{L^2} = ||\nabla u||_{L^2}^2
\end{aligned}$$

$$||u_h||_{H^1}^2 = ||u||_{L^2}^2 + ||\nabla u||_{L^2}^2 \leq (C||\nabla u||_{L^2})^2 + ||u||_{L^2}^2(C^2 + 1)$$

where $C_3 = \frac{1}{C^2+1}$

Exercise 7.6

Stokes problem:

$$\begin{aligned} -\Delta u - \nabla p = f \qquad & \text{in } \Omega = (0,1)^2 \\ \nabla \cdot u = 0 \qquad & \text{in } \Omega = (0,1)^2 \end{aligned}$$

$$\text{with } u_{exact} = (sin(\pi y), cos(\pi x)) \text{ and } p_{exact} = sin(2\pi x)$$

$$f = -\Delta u - \nabla p = (\pi^2 sin(\pi y) - 2\pi cos(2\pi x), \pi^2 cos(\pi x))$$

When *Brezzi* conditions are satisfied we can obtain order ptimal convergence rates that is :

$$||u - u_h||_{H^1} + ||p - p_h||_{L^2} \leq Ch^k ||u||_{k+1} + Dh^{l+1} ||p||_{l+1}$$

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

We expect the convergence rates to be as follows:

$$\textbf{P4-P3:} \quad ||u - u_h||_{H^1} + ||p - p_h||_{L_2} + \leq Ch^4(||u||_5 + ||p||_4)$$

and vi expect to have a convergence rate about **4**.

$$\textbf{P4-P2:} \quad ||u - u_h||_{H^1} + ||p - p_h||_{L_2} + \leq Ch^4 ||u||_5 + Dh^3 ||p||_4$$

Here we can see that the first term of the RHS of this inequality will go towards zero much faster than the second term and for that reason we expect a convergence rate about **3**.

$$\textbf{P3-P2:} \quad ||u - u_h||_{H^1} + ||p - p_h||_{L_2} + \leq Ch^3(||u||_4 + ||p||_3)$$

So we expect a convergence rate about **3**.

$$\textbf{P3-P1:} \quad ||u - u_h||_{H^1} + ||p - p_h||_{L_2} + \leq Ch^3 ||u||_4 + Dh^2 ||p||_2$$

Here We can see that, as h goes toward zero, the first term on the RHS of the inequality above will go towards zero much faster than the second term and we therefore we expect a convergence rate about **2**.

Table 1: Convergence rates

(a) Convergence rates for $P_4 - P_3$ (on the left) and $P_4 - P_2$ (on the right)

N	α_u	α_p
8	4.459785	4.033495
16	4.288271	4.016167
32	4.106070	4.004667
64	4.02773	3.993321

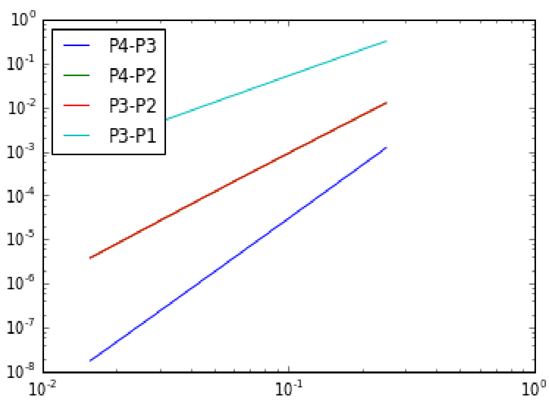
N	α_u	α_p
8	2.594438	2.854863
16	2.823205	2.882920
32	2.930570	2.949047
64	2.972018	2.979046

(b) Convergence rates for $P_3 - P_2$ (on the left) and $P_3 - P_1$ (on the right)

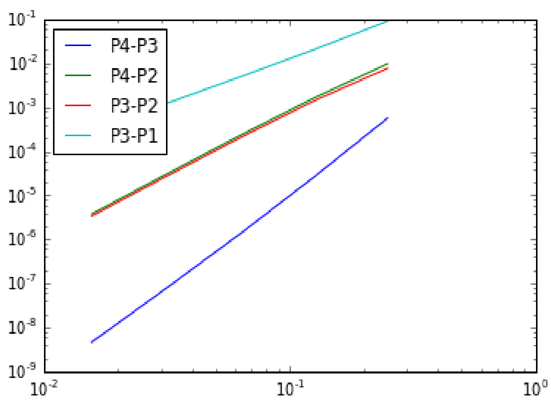
N	α_u	α_p
8	2.470843	2.871253
16	2.783089	2.886969
32	2.917043	2.950545
64	2.966947	2.979664

N	α_u	α_p
8	2.156116	1.970220
16	2.054004	1.993176
32	2.012088	1.998362
64	2.001497	1.999618

We can see from tabel:1 that the convergence rates go towards the expected value as N increases and we get a finer mesh. The code used to obtain these values is included in the appendix at the end of this report.



(a) H^1 error for u



(b) L^2 error for p

Figure 1: H^1 error for u and L^2 error for p

Linear elasticity

We are going to solve the following equation set:

$$
\begin{aligned}
-\mu\Delta u - \lambda\nabla\nabla\cdot u = f \qquad & in\ \ \Omega = (0,1)^2 \\
u = u_{exact} \qquad & on\ \ \partial\Omega
\end{aligned}
$$

Where $u_{exact} = \left(\frac{\partial\Phi}{\partial y}, -\frac{\partial\Phi}{\partial x}\right)$ and $\Phi = sin(\pi xy)$ and $\nabla\cdot u_{exact} = 0$

a) Expression for f

$$
\begin{aligned}
u_x &= \frac{\partial\Phi}{\partial y} = \pi x cos(\pi xy) \\
u_y &= -\frac{\partial\Phi}{\partial x} = -\pi y cos(\pi xy) \\
\Delta u &= \left(\underbrace{\frac{\partial^2 u_x}{\partial x^2}}_{(1)} + \underbrace{\frac{\partial^2 u_x}{\partial y^2}}_{(2)}, \underbrace{\frac{\partial^2 u_y}{\partial x^2}}_{(3)} + \underbrace{\frac{\partial^2 u_y}{\partial y^2}}_{(4)}\right)
\end{aligned}
$$

(1):

$$
\begin{aligned}
\frac{\partial}{\partial x}\left(\frac{\partial u_x}{\partial x}\right) &= \frac{\partial}{\partial x}(\pi cos(\pi xy) - \pi^2 xysin(\pi xy)) \\
&= -\pi^2 ysin(\pi xy) - \pi^2 ysin(\pi xy) - \pi^3 xy^2 cos(\pi xy) \\
&= -2\pi^2 ysin(\pi xy) - \pi^3 xy^2 cos(\pi xy)
\end{aligned}
$$

(2):

$$
\begin{aligned}
\frac{\partial}{\partial y}\left(\frac{\partial u_x}{\partial y}\right) &= \frac{\partial}{\partial y}(-\pi^2 x^2 sin(\pi xy)) \\
&= -\pi^3 x^3 cos(\pi xy)
\end{aligned}
$$

(3):

$$
\begin{aligned}
\frac{\partial}{\partial x}\left(\frac{\partial u_y}{\partial x}\right) &= \frac{\partial}{\partial x}(\pi^2 y^2 sin(\pi xy)) \\
&= \pi^3 y^3 cos(\pi xy)
\end{aligned}
$$

(4):

$$
\begin{aligned}
\frac{\partial}{\partial y}\left(\frac{\partial u_y}{\partial y}\right) &= \frac{\partial}{\partial y}(-\pi cos(\pi xy) + \pi^2 yxsin(\pi xy) \\
&= 2\pi^2 xsin(\pi xy) + \pi^3 yx^2 cos(\pi xy)
\end{aligned}
$$

$$
\begin{aligned}
\Rightarrow f &= -\mu\Delta u \\
&= \mu(2\pi^2 ysin(\pi xy) - \pi^3 xy^2 cos(\pi xy)) + \pi^3 x^3 cos(\pi xy)\vec{i} \\
&- \mu(2\pi^2 xsin(\pi xy) + \pi^3 yx^2 cos(\pi xy) + \pi^3 y^3 cos(\pi xy))\vec{j} \\
&= \mu(2\pi^2 ysin(\pi xy) + \pi^3 xcos(\pi xy)(x^2 + y^2))\vec{i} \\
&- \mu(2\pi^2 xsin(\pi xy) + \pi^3 ycos(\pi xy)(x^2 + y^2))\vec{j}
\end{aligned}
$$

c and d)

Computing the numerical error for polynomial orders 1 and 2, with different values for λ results in the following table for numerical error and convergence rate. The code used to obtain these values is included in the appendix at the end of this report.

Table 2: Numerical error and convergence rate for 1st order polynomials computed with a single function space

$\downarrow \lambda$ / N $\rightarrow$	8	16	32	64	Convergence rate
1	0.06033	0.01562	0.00394	0.00099	1.99664
10	0.10691	0.03281	0.00879	0.00224	1.97184
100	0.29792	0.16302	0.06037	0.01754	1.78325
1000	0.44459	0.45625	0.43295	0.35192	0.29896

Table 3: Numerical error and convergence rate for 2nd order polynomials computed with a single function space

$\downarrow \lambda$ / N $\rightarrow$	8	16	32	64	Convergence rate
1	0.00208	0.00025	3.12e-05	3.89e-06	3.00346
10	0.00352	0.00033	3.40e-05	3.98e-06	3.09229
100	0.01441	0.00149	0.00012	8.74e-06	3.77219
1000	0.02990	0.00718	0.00158	0.000272	2.53469

We can see from both table:2 and 3 that the convergence rate jumps alot and it's not what we expect, because of "locking". Also Finite element solutions vanish quickly to zero when λ gets very large. This problem is fixed by implementing a new function $p = \lambda \nabla \cdot u$ and then solving the system with a mixed function space which results in table:4 and 5, where the rate of convergence is more stable.

Table 4: Numerical error and convergence rate for 1st order polynomials computed with a mixed function space

$\downarrow \lambda$ / N $\rightarrow$	8	16	32	64	Convergence rate
1	0.00201	0.00025	3.11e-05	3.89e-06	3.00073
10	0.00261	0.00031	3.49494	4.06042	3.10557
100	0.00567	0.00077	9.39e-05	9.57e-06	3.29368
1000	0.01109	0.00249	0.00052	9.52e-05	2.47159

Table 5: Numerical error and convergence rate for 2nd order polynomials computed with a mixed function space

$\downarrow \lambda$ / N $\rightarrow$	8	16	32	64	Convergence rate
1	0.00199	0.00025	3.11e-05	3.89e-06	2.99945
10	0.00259	0.00030	3.49e-05	4.05e-06	3.10435
100	0.00564	0.00076	9.38e-05	9.57e-06	3.29317
1000	0.01106	0.00249	0.00052	9.52e-05	2.47139

Appendix: Codes and outputs

Exercise 7.6

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

def u_boundary(x):
    return x[0] < DOLFIN_EPS or x[1] > 1.0 - DOLFIN_EPS or x[1] < DOLFIN_EPS

def p_boundary(x):
    return x[0] > 1.0 - DOLFIN_EPS

for i in [[4,3], [4,2], [3,2], [3,1]]:

    h_values=[]
    uError=[]
    pError=[]

    for N in [4,8,16,32,64]:
        h=1./N
        h_values.append(h)
        mesh = UnitSquareMesh(N,N)


        V = VectorElement('Lagrange', mesh.ufl_cell(), i[0])
        Q = FiniteElement('Lagrange', mesh.ufl_cell(), i[1])

        W = FunctionSpace(mesh, MixedElement([V, Q]))


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

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

        uExact = Expression(['sin(pi*x[1])','cos(pi*x[0])'],degree=i[1]+1)
        pExact = Expression('sin(2*pi*x[0])', degree=i[1]+1)

        BC1 = DirichletBC(W.sub(0), uExact, u_boundary)
        BC2 = DirichletBC(W.sub(1), pExact, p_boundary)
        bc=[BC1, BC2]

        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)

        uh, ph = up.split()

        u_Error = errornorm(uExact, uh, norm_type='h1', degree_rise=1)
        uError.append(u_Error)

        p_Error = errornorm(pExact, ph, norm_type='l2', degree_rise=1)
        pError.append(p_Error)


    u_Cr = []
    p_Cr = []

    for j in range (1, len(uError)):
        print '\n'

        uC = np.log(uError[j-1]/uError[j])/np.log(h_values[j-1]/h_values[j])
        u_Cr.append(uC)

        pC = np.log(pError[j-1]/pError[j])/np.log(h_values[j-1]/h_values[j])
        p_Cr.append(pC)


        print ('h = %f  \n  Conv_rate for u = %f  \n  Conv_rate for P = %f'

    plt.figure(1)
    plt.loglog(h_values,uError, label='P%d-P%d' % ((i[0],i[1])))

    plt.figure(2)
    plt.loglog(h_values,pError, label='P%d-P%d' % ((i[0],i[1])))


plt.figure(1)
plt.legend(loc='upper left')
plt.savefig('Velocity_convergence.png')

plt.figure(2)
plt.legend(loc='upper left')
plt.savefig('Pressure_convergence.png')

"""
h = 0.125000
  Conv_rate for u = 4.459785
  Conv_rate for P = 4.033495

h = 0.062500
  Conv_rate for u = 4.288271
  Conv_rate for P = 4.016167

h = 0.031250
```

```
  Conv_rate for u = 4.106074
  Conv_rate for P = 4.004682

h = 0.015625
  Conv_rate for u = 4.028000
  Conv_rate for P = 3.994613

h = 0.125000
  Conv_rate for u = 2.594438
  Conv_rate for P = 2.854863

h = 0.062500
  Conv_rate for u = 2.823205
  Conv_rate for P = 2.882920

h = 0.031250
  Conv_rate for u = 2.930570
  Conv_rate for P = 2.949047

h = 0.015625
  Conv_rate for u = 2.972018
  Conv_rate for P = 2.979046

h = 0.125000
  Conv_rate for u = 2.470843
  Conv_rate for P = 2.871253

h = 0.062500
  Conv_rate for u = 2.783089
  Conv_rate for P = 2.886969

h = 0.031250
  Conv_rate for u = 2.917043
  Conv_rate for P = 2.950545

h = 0.015625
  Conv_rate for u = 2.966947
  Conv_rate for P = 2.979664

h = 0.125000
  Conv_rate for u = 2.156116
  Conv_rate for P = 1.970220

h = 0.062500
  Conv_rate for u = 2.054004
  Conv_rate for P = 1.993176

h = 0.031250
  Conv_rate for u = 2.012088
  Conv_rate for P = 1.998362
```

```
h = 0.015625
  Conv_rate for u = 2.001497
  Conv_rate for P = 1.999618
"""
```

Exercise 3 - Linear Elasticity

```
#Oppgave 3b og 3c

from dolfin import *
import numpy as np
set_log_active(False)

mu = Constant(1.0)

class Boundaries(SubDomain):
    def inside(self, x, on_boundary):
        return on_boundary

def SingleFunctionSpace():
    print 'Single Function Space \n'

    for order in [1,2]:
        print("Order P%d Elements \n " % (order))
        lambda_values = [1, 10, 100, 10000]
        N = [8,16,32,64]

        for lamb in lambda_values:
            error = np.zeros(len(N))
            h = np.zeros(len(N))

            for n in range(len(N)):
                h[n] = 1./N[n]
                mesh = UnitSquareMesh(N[n],N[n])

                V = VectorFunctionSpace(mesh, 'CG', order)
                Q = VectorFunctionSpace(mesh, 'CG', order+1)


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

                u_ex = Expression(["pi*x[0]*cos(pi*x[0]*x[1])", "-pi*x[1]*co
                f = Expression(["-mu * (pi * pi * (-2 * x[1] *sin(pi * x[0]
                                "-mu * (pi * pi * (2 * x[0] * sin(pi * x[0]

                u_ex = interpolate(u_ex, Q)
                u_exact1 = project(u_ex, V)


                boundaries = Boundaries()
                bc = DirichletBC(V, u_ex, boundaries)
```

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

            solve(a==L, u_, bc)
            error[n] = errornorm(u_, u_ex)
            if n == 0:
                print 'Lambda=',lamb,', n=',N[n], 'error=', error[n]
            else:
                convergence_rate = np.log(abs(error[n]/error[n-1]))/np.l
                print 'Lambda=',lamb,', n=',N[n],', error=',error[n],',
        print
    print


def MixedFunctionSpace():
    print 'Mixed Function Space \n'
    for order in [1,2]:
        print("Order P%d Elements \n " % (order))
        lambda_values = [1, 10, 100, 10000]
        N = [8,16,32,64]
        for lamb in lambda_values:
            error = np.zeros(len(N))
            h = np.zeros(len(N))
            for n in range(len(N)):
                h[n] = 1./N[n]
                mesh = UnitSquareMesh(N[n],N[n])

                V = VectorElement('Lagrange', mesh.ufl_cell(), 2)
                Q = FiniteElement('Lagrange', mesh.ufl_cell(), 2)

                W = FunctionSpace(mesh, MixedElement([V, Q]))


                up = TrialFunction(W)
                vq = TestFunction(W)

                u, p = split(up)
                v, q = split(vq)

                u_ex = Expression(["pi*x[0]*cos(pi*x[0]*x[1])", "-pi*x[1]*co
                f = Expression(["-mu * (pi * pi * (-2 * x[1] *sin(pi * x[0]
                                "-mu * (pi * pi * (2 * x[0] * sin(pi * x[0]


                boundaries = Boundaries()
                bc = DirichletBC(W.sub(0), u_ex, boundaries)

                a1 = mu*inner(grad(u), grad(v))*dx + p*div(v)*dx
                a2 = p*q*dx - lamb*div(u)*q*dx
```

```
            A = a1+a2
            L = dot(f,v)*dx
            up_ = Function(W)

            solve(A==L, up_, bc)
            u_ , p_ = up_.split()
            error[n] = errornorm(u_ex, u_, degree_rise = 2)
            if n == 0:
                print 'Lambda=',lamb,', n=',N[n], 'error=', error[n]
            else:
                convergence_rate = np.log(abs(error[n]/error[n-1]))/np.l
                print 'Lambda=',lamb,', n=',N[n],', error=',error[n],',

        print
    print

SingleFunctionSpace()
MixedFunctionSpace()

"""
Single Function Space

Order P1 Elements

Lambda= 1 , n= 8 error= 0.0603332539231
Lambda= 1 , n= 16 , error= 0.0156150963357 , convergence rate: 1.95001192267
Lambda= 1 , n= 32 , error= 0.00393966554654 , convergence rate: 1.9867964059
Lambda= 1 , n= 64 , error= 0.000987211055652 , convergence rate: 1.996642702

Lambda= 10 , n= 8 error= 0.106913800826
Lambda= 10 , n= 16 , error= 0.0328182291491 , convergence rate: 1.7038787943
Lambda= 10 , n= 32 , error= 0.00879465678597 , convergence rate: 1.899798211
Lambda= 10 , n= 64 , error= 0.00224200637662 , convergence rate: 1.971836895

Lambda= 100 , n= 8 error= 0.297919662028
Lambda= 100 , n= 16 , error= 0.163016065551 , convergence rate: 0.8699091894
Lambda= 100 , n= 32 , error= 0.0603678831797 , convergence rate: 1.433161033
Lambda= 100 , n= 64 , error= 0.0175386032648 , convergence rate: 1.783247354

Lambda= 10000 , n= 8 error= 0.444598626404
Lambda= 10000 , n= 16 , error= 0.456252800413 , convergence rate: -0.0373299
Lambda= 10000 , n= 32 , error= 0.43295442431 , convergence rate: 0.075618248
Lambda= 10000 , n= 64 , error= 0.351920976562 , convergence rate: 0.29896365


Order P2 Elements

Lambda= 1 , n= 8 error= 0.00208791582394
Lambda= 1 , n= 16 , error= 0.000252387932594 , convergence rate: 3.048348712
Lambda= 1 , n= 32 , error= 3.12521270152e-05 , convergence rate: 3.013616644
Lambda= 1 , n= 64 , error= 3.89714578313e-06 , convergence rate: 3.003464577
```

```
Lambda= 10 , n= 8 error= 0.00352404710379
Lambda= 10 , n= 16 , error= 0.000325009730888 , convergence rate: 3.43867838
Lambda= 10 , n= 32 , error= 3.40286842489e-05 , convergence rate: 3.25565963
Lambda= 10 , n= 64 , error= 3.9899878309e-06 , convergence rate: 3.092295117

Lambda= 100 , n= 8 error= 0.0144103718292
Lambda= 100 , n= 16 , error= 0.00149902721882 , convergence rate: 3.26500907
Lambda= 100 , n= 32 , error= 0.000119506039712 , convergence rate: 3.6488711
Lambda= 100 , n= 64 , error= 8.74676399034e-06 , convergence rate: 3.7721903

Lambda= 10000 , n= 8 error= 0.0299024684882
Lambda= 10000 , n= 16 , error= 0.00717555036588 , convergence rate: 2.059103
Lambda= 10000 , n= 32 , error= 0.00157727077303 , convergence rate: 2.185659
Lambda= 10000 , n= 64 , error= 0.000272199980745 , convergence rate: 2.53469


Mixed Function Space

Order P1 Elements

Lambda= 1 , n= 8 error= 0.00201057640528
Lambda= 1 , n= 16 , error= 0.000249842636743 , convergence rate: 3.008517556
Lambda= 1 , n= 32 , error= 3.11733423691e-05 , convergence rate: 3.002634947
Lambda= 1 , n= 64 , error= 3.89470176164e-06 , convergence rate: 3.000728084

Lambda= 10 , n= 8 error= 0.00261221960374
Lambda= 10 , n= 16 , error= 0.000309057190166 , convergence rate: 3.07933045
Lambda= 10 , n= 32 , error= 3.49494493402e-05 , convergence rate: 3.14453219
Lambda= 10 , n= 64 , error= 4.06042883908e-06 , convergence rate: 3.10556571

Lambda= 100 , n= 8 error= 0.00566781588876
Lambda= 100 , n= 16 , error= 0.000768254615166 , convergence rate: 2.8831364
Lambda= 100 , n= 32 , error= 9.3903670675e-05 , convergence rate: 3.03233106
Lambda= 100 , n= 64 , error= 9.57603280557e-06 , convergence rate: 3.2936815

Lambda= 10000 , n= 8 error= 0.0110940407961
Lambda= 10000 , n= 16 , error= 0.00249981642951 , convergence rate: 2.149890
Lambda= 10000 , n= 32 , error= 0.000528373445491 , convergence rate: 2.24219
Lambda= 10000 , n= 64 , error= 9.52611983498e-05 , convergence rate: 2.47159


Order P2 Elements

Lambda= 1 , n= 8 error= 0.00199274200148
Lambda= 1 , n= 16 , error= 0.000249187816109 , convergence rate: 2.999449502
Lambda= 1 , n= 32 , error= 3.11516312137e-05 , convergence rate: 2.999853915
Lambda= 1 , n= 64 , error= 3.89401076316e-06 , convergence rate: 2.999978933

Lambda= 10 , n= 8 error= 0.00259158062311
Lambda= 10 , n= 16 , error= 0.000308075110647 , convergence rate: 3.07247823
```

```
Lambda= 10 , n= 32 , error= 3.49063922614e-05 , convergence rate: 3.14171897
Lambda= 10 , n= 64 , error= 4.05881993345e-06 , convergence rate: 3.10435901

Lambda= 100 , n= 8 error= 0.00564122679926
Lambda= 100 , n= 16 , error= 0.000766854504904 , convergence rate: 2.8789841
Lambda= 100 , n= 32 , error= 9.38309367105e-05 , convergence rate: 3.0308173
Lambda= 100 , n= 64 , error= 9.57201580172e-06 , convergence rate: 3.2931689

Lambda= 10000 , n= 8 error= 0.0110645195381
Lambda= 10000 , n= 16 , error= 0.00249793882124 , convergence rate: 2.147130
Lambda= 10000 , n= 32 , error= 0.000528259351901 , convergence rate: 2.24141
Lambda= 10000 , n= 64 , error= 9.52545727685e-05 , convergence rate: 2.47138
"""
```