Mandatory Assignment 1 MEK 4250

Farnaz Rezvany Hesary

March 6, 2017

Exercise 1

Considering the Poisson equation on the domain $\Omega = (0,1)^2$:

$$\begin{aligned} -\Delta u &= f \quad \text{in } \Omega \\ u &= 0 \quad \text{on } x = 0 \text{ and; } x = 1 \\ \frac{\partial u}{\partial n} &= 0 \quad \text{on } y = 0 \text{ and } y = 1 \end{aligned}$$

We assume that:

$$u = sin(k\pi x)cos(k\pi y)$$

and compute $f = -\Delta u$.

$$\begin{aligned} f &= -\frac{\partial}{\partial x}\left(\frac{\partial u}{\partial x}\right) - \frac{\partial}{\partial y}\left(\frac{\partial u}{\partial y}\right) \\ &= -(-(k\pi)^2 cos(k\pi y)sin(k\pi x) - (k\pi)^2 sin(k\pi x)cos(k\pi y)) \\ &= 2(k\pi)^2 u \end{aligned}$$

a)

We will compute the H^p norm of u analytically. The general expression for the H^p norm is:

$$\|H\|^p = \left(\sum_{i \leqslant p} \int_\Omega \left|\nabla^i u\right|^2 d\mathbf{x}\right)^{\frac{1}{2}}$$

Which can also be written as follows:

$$\|u\|_H^p = \left(\int_\Omega \left((u)^2 + (\frac{\partial \mathbf{u}}{\partial \mathbf{x}})^2 + (\frac{\partial^2 \mathbf{u}}{\partial \mathbf{x}^2})^2 + (\frac{\partial^3 \mathbf{u}}{\partial \mathbf{x}^3})^2 + ... + (\frac{\partial^p \mathbf{u}}{\partial \mathbf{x}^p})^2\right) dx\right)^{\frac{1}{2}}$$

$$\int_0^1 \int_0^1 (\mathbf{u})^2 \, dxdy = \int_0^1 \int_0^1 (\sin^2(k\pi x)\cos^2(k\pi x)) \, dxdy = \frac{1}{4}$$

$$\int_0^1 \int_0^1 (\nabla u)^2 \, dy \, dx \int_0^1 \int_0^1 \Big((k\pi \cos(k\pi x)\cos(k\pi y)) - (k\pi \sin(k\pi x)\sin(k\pi y))\Big)^2 \, dx \, dy = \frac{1}{2}(\pi k)^2$$

$$\int_0^1 \int_0^1 (\nabla^n u)^2 \, dy \, dx = \frac{1}{2}(\pi^2 k^2)^n$$

The H^p norm of $\mathbf{u}$ is then:

$$\|\mathbf{u}\|_{H^p} = \left(\frac{1}{4} + \frac{1}{2}(\pi k)^2 + ... + \frac{1}{2}((\pi k)^2)^n\right)^{\frac{1}{2}} = \left(\frac{1}{4}\sum_{i \leqslant p}(2\pi k)^i\right)^{\frac{1}{2}}$$

b)

The following code computes the L_2 and H^1 errors for $\frac{1}{h} = 8, 16, 32, 64$ when using first and second order Lagrangian elements when $k = 1, 10$.

```
1
2  from dolfin import * #FEniCS
3  from math import *
4
5  for i in [1,2]:
6      for N in[8,16,32,64]: # h=(b-a)/N=1/N
7          mesh = UnitSquareMesh(N,N)
8          V = FunctionSpace(mesh, "Lagrange", i)
9          V2 = FunctionSpace(mesh, "Lagrange", i+2)
10
11         u = TrialFunction(V)
12         v = TestFunction(V)
13
14         bcs = [DirichletBC(V, Constant(0), 'x[0] < DOLFIN_EPS'), DirichletBC(V,
       Constant(0), 'x[0] > 1- DOLFIN_EPS')]
15
16         for k in [1,10]:
17             f = Expression('2*pi*pi*k*k*sin(pi*k*x[0])*cos(pi*k*x[1])', k=k, degree
       =i)
18             uh = Function(V)
19
20             a=inner(grad(u),grad(v))*dx
21             L=f*v*dx
22
23             solve(a == L, uh, bcs)
24
25             uexp= Expression('sin(pi*k*x[0])*cos(pi*k*x[1])', k=k, degree=1)
26             u_Exact= interpolate(uexp, V2)
27
28             L2_norm = errornorm(u_Exact, uh, 'l2', degree_rise=3)
29             print 'i=%g ,1/h=%g ,k=%g' % (i, N, k)
30             print 'L2 = %g' % (L2_norm)
31
32             H1_norm = errornorm(u_Exact, uh, 'H1', degree_rise=3)
33             print 'H1 = %g' % (H1_norm)
34             print '\n'
```

The output from the code above is listed below.

Table 1: The L_2 and H^1 errors 1. order Lagrangian elements.

L_2 N	k =1	k = 10	H^1 N	k = 1	k = 10
8	0.0327753	0.667057	8	0.436592	26.4815
16	0.0084627	0.365538	16	0.218166	17.5464
32	0.0021332	0.178190	32	0.109055	10.6024
64	0.0005344	0.054904	64	0.054524	5.43986

Table 2: The L_2 and H^1 errors 2. order Lagrangian elements.

L_2 N	k =1	k = 10	H^1 N	k = 1	k = 10
8	0.00057	0.435611	8	0.0331846	19.1245
16	6.93E-05	0.089599	16	0.0083894	6.92035
32	8.61E-06	0.010206	32	0.0021055	1.97796
64	1.08E-06	0.001140	64	0.0005272	0.51842

As expected we can see for $k = 1$ we get small error norms and that the error decreases with a finer mesh. When the gradients are not steep we don't get large values for the H_1 norm.
For $k = 10$ the frequency of the oscillations are higher and the gradients are steep we get larger errors. The H^1 values get bigger.

c)

Considering a general error estimate:
$\|u - u_h\|_q \leqslant Ch^\alpha$
We want to estimate C and α and also check whether the error estimate is valid. We can solve for C and α using the least squares method. We start by rewriting the expression for the error estimate by taking the logarithm.

$$\log(\|u - u_h\|_1) = \log(C_\alpha) + \alpha \log(h) \qquad H1$$
$$\log(\|u - u_h\|_0) = \log(C_\beta) + \beta \log(h) \qquad L_2$$

which resembles the line equation:

$$Y = b + Ax$$
$$\|u - u_h\|_q = \log(C_\alpha) + \alpha \log(h)$$

The code that computes C_α and α for H^1 error norm and C_β and β for the L_2 error norm and also checks the validity of the results is attached. The code also contains a printout at the end.

Table 3: Convergence rate for L_2 for 1. order Lagrangian elements.

(a) k=1

L_2 N	β	C_β
8	-	-
16	1.98341*	1.90395*
32	1.97076	1.9818
64	1.98037	2.0308

(b) k=10

L_2 N	β	C_β
8	-	-
16	0.867788*	4.05372*
32	0.952196	4.92664
64	1.18451	8.89149

Table 4: Convergence rate for L_2 for 2. order Lagrangian elements.

(a) k=1

L_2 N	β	C_β
8	-	-
16	3.03703	0.314739
32	3.0232*	0.304842*
64	3.0154*	0.298858*

(b) k=10

L_2 N	β	C_β
8	-	-
16	2.28148*	50.0587*
32	2.70779	134.045
64	2.88679	211.265

Table 5: Convergence rate for H^1 for 1. order Lagrangian elements.

(a) k=1

H^1 N	α	C_α
8	-	-
16	1.00086*	3.49898*
32	1.00061*	3.49701*
64	1.00044	3.49542

(b) k=10

H^1 N	α	C_α
8	-	-
16	0.593803	91.0332
32	0.660297	106.151
64	0.757682	135.96

Table 6: Convergence rate for H^1 for 2. order Lagrangian elements.

(a) k=1

H^1 N	α	C_α
8	-	-
16	1.98387*	2.05376*
32	1.98912	2.07884
64	1.99227	2.09552

(b) k=10

H^1 N	α	C_α
8	-	-
16	1.46651	403.626
32	1.63667	598.034
64	1.74223	782.07

We can see from the tables above that the convergence rate for the L_2 norm goes towards 2 for Lagrangian polynomials of order 1, and towards 3 for polynomials of order 2., but for the H^1 norm the convergence rate goes towards 1 for 1. order polynomials and towards 2 for 2. order polynomials.

Exercise 2

In this exercise we have the diffusion equation, on the domain $\Omega = (0,1)^2$:

$$\begin{aligned} -\mu\Delta u + u_x &= 0 \quad \text{in } \Omega \\ u &= 0 \quad \text{on } x = 0 \\ u &= 1 \quad \text{on } x = 1 \\ \frac{\partial u}{\partial n} &= 0 \quad \text{for } y = 0 \text{ and } y = 1 \end{aligned}$$

a)

We assume that the solution is a product of two separate functions:

$$u(x,y) = X(x)Y(y)$$

Inserting this into the problem equation, we get

$$-\mu(X''(x)Y(y) + X(x)Y''(y)) + X'(x)Y(y) = 0$$

Dividing by $X(x)Y(y)$ we get:

$$-\mu\Big(\frac{X''(x)}{X(x)} + \frac{Y''(y)}{Y(y)}\Big) + \frac{X'(x)}{X(x)} = 0$$

Then we can rewrite the equation:

$$\frac{X''(x)}{X(x)} - \frac{1}{\mu}\frac{X'(x)}{X(x)} = \frac{Y''(y)}{Y(y)} = \lambda^2$$

We see that we can get two expressions that are only dependent on x or y, so these can be said to be equal to a common constant λ. Rewriting the equation we get an equation set:

$$Y''(y) - \lambda^2 Y(y) = 0 \tag{1}$$

$$X''(x) - \frac{1}{\mu}X'(x) - \lambda^2 X(x) = 0 \tag{2}$$

The solution for $Y(y)$ is:

$$\begin{aligned} Y(y) &= A\cos(\lambda y) + B\sin(\lambda y) \\ Y'(y) &= -A\lambda\sin(\lambda y) + B\lambda\cos(\lambda y) \end{aligned}$$

Using Neumann boundary conditions we get:

$$\begin{aligned} Y'(y=0) &= -A\lambda\sin(0) + B\lambda\cos(0) = B\lambda = 0 \\ Y'(y=1) &= -A\lambda\sin(\lambda) + B\lambda\cos(\lambda) = 0 \end{aligned}$$

We assume $\lambda = 0$

We can see that both constants become zero if $\lambda \neq 0$, so we assume that $\lambda = 0$ which gives:

$$Y(y) = A\cos(0) = A$$

For convenience we choose $A = 1$

The second equation then becomes:

$$\frac{X''(x)}{X(x)} - \frac{1}{\mu}\frac{X'(x)}{X(x)} = 0$$
$$X''(x) - \frac{1}{\mu}X'(x) = 0$$

which has the solution:

$$X(x) = b_1 + b_2 e^{\frac{x}{\mu}}$$

Using the boundary conditions with $Y(y) = A = 1$ we get:

$$u(x, y) = \frac{e^{\frac{x}{\mu}} - 1}{e^{\frac{a}{\mu}} - 1}$$

Task b)

Table 7: The L_2 error for different values of μ

(a) 1. order polynomials

N/μ	8	16	32	64
1	1.402E-03	3.508E-04	8.770E-05	2.193E-05
0.1	2.375E-02	6.177E-03	1.561E-03	3.915E-04
0.01	2.379E-01	1.039E-01	3.819E-02	1.126E-02

(b) 2. order polynomials

N/μ	8	16	32	64
1	1.151E-05	1.448E-06	1.817E-07	2.277E-08
0.1	2.245E-03	3.038E-04	3.884E-05	4.888E-06
0.01	8.513E-02	3.039E-02	7.598E-03	1.326E-03

Table 8: The H^1 error for different values of μ

(a) 1. order polynomials

N/μ	8	16	32	64
1	3.752E-02	1.877E-02	9.383E-03	4.692E-03
0.1	7.671E-01	3.981E-01	2.010E-01	1.008E-01
0.01	7.238E+00	6.684E+00	5.007E+00	2.969E+00

(b) 2. order polynomials

N/μ	8	16	32	64
1	5.970E-04	1.504E-04	3.772E-05	9.448E-06
0.1	1.183E-01	3.164E-02	8.066E-03	2.028E-03
0.01	5.140E+00	3.604E+00	1.705E+00	5.661E-01

We can see that the approximations get poorer when μ gets small which will give steep gradients.

Task c)

The code to compute convergence rates for different values for μ that also checks the validity of the computed norm is attached. The code also contains a nice printout.

We can see from the output that the convergence rate decreases for smaller values of μ which was expected since smaller μ give higher Errors.

Task d)

The main difference is the test function. If V is the Galerkin basis function, the SUPG basis function is $w = v + \beta u_x$, where β is a constant that we choose to set to $\frac{1}{2}$.Inserted into the weak form for our problem, we get

$$\mu \int_\Omega \nabla u \nabla v dn + \int_\Omega u_x v dn = 0$$

The new weak form of the equation is:

$$\mu \int_\Omega \nabla u \nabla w dn + \int_\Omega u_x v dn = \mu \int_\Omega \nabla u \nabla v dn + \int_\Omega u_x v dn \beta \mu \int_\Omega \nabla u \nabla u_x dn + \beta \int_\Omega u_x^2 dn$$

The code with a nice printout for the SUPG approximation is also attached.
We can see that for μ down to 0.01 we get good approximations and the convergence rate goes towards 1.5, but the approximation gets poorer when $\mu < 0.1$. In comparison with the results in part b) and c) we can see that the SUPG method give better approximations when μ gets small.