CS555: Numerical Methods for PDEs
Instructor: L. Olson
Due Tuesday, April 30, 2013

Instructions:

1. Your submitted work must be your own work. You may *discuss* the homework.
2. It is your responsibility to answer the question in detail, to convince the grader that you answered the problem, and to explain your solution.
3. Your submission should be as short as possible.
4. Items #1 and #2 are required; item #3 should be a guide.

To Submit:

- create a directory `sp13-cs555/`*`yournetid`*`/hw4`
- add your write-up `hw4.pdf` and supporting files `hw4*.py`
- commit your directory and files (svn details are on the web)

note: This homework has been tested with Numpy 1.7.0, Scipy 0.11.0, matplotlib 1.2.0, and Dolfin 1.0.0+.

1. Consider the function
$$f(x,y) = \frac{1}{((x-x_0)^2 + (y-y_0)^2)^\alpha}$$
for $(x_0, y_0) = (\frac{1}{3}, \frac{1}{3})$ and with $\alpha > 0$. What can we say about approximating this function if we consider the *approximation properties* of Lagrange basis functions (on a uniform mesh)? Use a *dolfin* script to investigate the behavior of $\|f - \Pi_h f\|_0$ w.r.t. h. You may use `test_conv.py` as a starting point.

2. Next consider solving the Poisson problem for the forcing function f defined in #1 with homogeneous Dirichlet boundary conditions. Construct a numerical test in *dolfin* to investigate the rate of convergence in the H^1-norm and the L^2-norm. To do this, use a very fine mesh as your "exact" solution. How is the rate impacted by the α in the description of f? Why? *note: notice that* `errornorm` *can be used for many different norms*

3. Recall that the (Dirichlet) Poisson problem is H^2-regular if the problem is convex or has a smooth boundary. Instead, consider here, an L-shaped domain (`lshape.xml`). To use this:

```
from dolfin import *
mesh = Mesh('lshape.xml')

plot(mesh, interactive=True)
```

Now, consider the exact solution

$$u = r^{2/3} \sin \frac{2}{3}\psi$$

for $\psi \in [0, \frac{3}{2}\pi]$.

The function can be implemented in *dolfin* (since `atan2` is defined on $[-\pi, \pi]$) as

```
import math
class uexact(Expression):
    def eval(self, values, x):
        psi = math.atan2(x[1], x[0])
        if psi < 0:
            psi += 2 * pi
        values[0] = (x[0]**2 + x[1]**2)**(1.0/3.0) * sin(2.0/3.0 * psi)
```

and then plotted with

```
u = uexact()
plot(u, mesh, interactive=True)
```

Verify that the solution is 1) the solution to the spherical Poisson problem and 2) that the solution is not in H^2. Also test the convergence of $\|u^* - u^h\|_0$ where u^* is the exact solution and u^h is a Lagrange Finite Element solution.

4. In the final problem, we wish to investigate *stability*. Consider the problem

$$-\nabla \cdot \nabla u = 2\pi^2 \sin(\pi x) \sin(\pi y)$$

which has the exact solution $u^* = \sin(\pi x) \sin(\pi y)$ on the unit square. Now consider a first-order form for this

$$q + \nabla u = 0 \tag{1}$$
$$\nabla \cdot q = f \tag{2}$$

Construct the weak form (no integration-by-parts is needed) and solve the problem using *dolfin*. In particular try two different spaces, vector Lagrange or Raviart-Thomás for the vector equation and constants for the scalar equation. That is

```
V = VectorFunctionSpace(mesh, 'CG', 1)
#V = FunctionSpace(mesh, 'RT', 1)
Q = FunctionSpace(mesh, 'DG', 0)
W = V * Q
(sigma, u) = TrialFunctions(W)
(tau, v) = TestFunctions(W)
```

Why do the spaces result is such different approximation results?