MATH 575 ASSIGNMENT 4

In this assignment, you will use the program FEniCS to numerically solve several boundary value problems for differential equations. To do this assignment, it will be helpful to read at least the first part of the FEniCS tutorial document, which can be found at

`https://fenicsproject.org/tutorial/`

and to install FEniCS on your computer. Instructions for installing FEniCS can be found at

`http://fenicsproject.org/download`

The simplest installation is on Ubuntu linux. However, it can also be installed on a Windows machine by first installing `Docker`. This is described on the installation page. More information about FEniCS in Docker can be found at

`http://fenics.readthedocs.io/projects/containers/en/latest/`

To run the programs provided for this assignment, you first need to copy them from

`http://www.math.rutgers.edu/~falk/math575/fenics-codes.html`

to a directory on your home computer. We will henceforth call this directory on your home computer `fenics-codes`.

Once you have installed `Docker`, (which in Windows will install an icon Docker Quickstart), select Docker Quickstart to bring up a Docker window. Type in this window

```
docker run -ti -v $(pwd):/home/fenics/shared quay.io/fenicsproject/stable
```

You should now be in the directory `/home/fenics`. Type `cd shared`. This will give you access to directories on your home machine. If you have stored your `.py` files in the directory `fenics-codes`, then type `cd fenics-codes`. You are now ready to run the codes referred to in the problems below.

Since you will be asked to hand in output from your FEniCS codes, to copy from a Docker/FEniCS window, first highlight the area to be copied, then move the mouse to the top bar, click the right button, and select edit. You can then paste this material by either selecting edit and paste or clicking on the right mouse button and selecting paste.

If you are unable to install FEniCS or are having problems using it, please come to my office for help.

1a. Use the code `bvp1-1d.py`, found at

`http://www.math.rutgers.edu/~falk/math575/fenics-codes.html`

to obtain the finite element solution using continuous, piecewise linear elements of the boundary value problem

$$-u'' + u = e - e^{-1} \qquad 0 < x < 1, \qquad u(0) = 0, \quad u(1) = 0.$$

Hand in the study of convergence in L^2, H^1, and the maximum error at the mesh points, produced by the program.

To run this program on a computer with FEniCS installed and the program `bvp1-1d.py` in your directory `fenics-codes`, type at the computer prompt:

```
python bvp1-1d.py
```

1b. Modify the code of Problem (1a) to produce analogous results for approximation of the boundary value problem

$$-u'' + u = (1 + 4\pi^2)x\cos(2\pi x) + 4\pi\sin(2\pi x), \quad 0 < x < 1, \qquad u(0) = 0,\ u'(1) = 1,$$

by continuous, piecewise linear elements. Hand in the study of convergence in L^2, H^1, and the maximum error at the mesh points, produced by the program. Also hand in a copy of your computer code. The solution of this problem is $u(x) = x\cos(2\pi x)$.

1c. Repeat Problem (1b) for continuous, piecewise cubic elements. Hand in the study of convergence in L^2, H^1, and the maximum error at the mesh points, produced by the program.

1d. Do these results correspond to what you expect from the theory developed in class?

2a. Use the code `poisson_convergence2.py`, found at

`http://www.math.rutgers.edu/~falk/math575/fenics-codes.html`

to obtain the finite element solution using continuous, piecewise linear elements of the boundary value problem

$$-\Delta u = f \quad \text{in } \Omega, \qquad u = 0 \quad \text{on } \partial\Omega,$$

where Ω is the unit square, and $f = [(x - 1/3)^2 + (y - 1/3)^2]^p$. For each of the choices $p = -.9$, $p = -.5$, and $p = -.1$, print out and hand in the study of convergence in L^2 and H^1 produced by the program. In this case the true solution is not known exactly, so we compute the errors by comparing to the finite element solution on a very high mesh with higher degree elements.

2b. Use the following information to give an explanation of why your numerical results do or do not confirm the theory developed in class.

(i) If $f \in L^2(\Omega)$, then $u \in H^2(\Omega)$.

(ii) For $p \geq 0$, $\int_\Omega f^2\,dx \leq 1$.

(iii) Let B be the circle of radius $1/3$ centered at $(1/3, 1/3)$ and $\Omega \setminus B$ be the set of points in Ω that are not in B. Then

$$\int_\Omega f^2\,dx = \int_{\Omega\setminus B} f^2\,dx + \int_B f^2\,dx.$$

For $p < 0$,

$$\int_{\Omega\setminus B} f^2\,dx \leq (9)^{2|p|}$$

$$\int_B f^2\,dx = \begin{cases} \frac{2\pi}{4p+2} r^{4p+2}\big|_{r=0}^{r=1/3}, & p \neq -1/2, \\ 2\pi \ln r\big|_{r=0}^{r=1/3} & p = -1/2 \end{cases}.$$

3. In this problem, we demonstrate the importance of adaptive mesh refinement for problems whose solutions are rapidly changing in some part of the domain.

3a. Use the code `adaptive-wedge.py` found at

`http://www.math.rutgers.edu/~falk/math575/fenics-codes.html`

to obtain the finite element solution using continuous, piecewise linear elements of the boundary value problem

$$-\Delta u = 1 \quad \text{in} \quad \Omega, \qquad u = 0 \quad \text{on} \quad \partial\Omega,$$

where Ω is the domain obtained by removing a wedge from the unit circle.

Hand in the mesh level, number of triangles, and number of vertices used in the final adaptive mesh. These numbers are produced by the program.

3b. Modify the computer program to use only uniform mesh refinement. Note that the command `mesh=refine(mesh)` with no additional arguments will do uniform refinement. The Mark step will no longer be needed. Hand in the same items as in 3a plus a copy of your computer code.