**ISC 5935 - Computational Tools for Finite Elements**

Homework #4

Assigned 24 September 2014, Due 01 October 2014

http://people.sc.fsu.edu/~jburkardt/classes/fem_2014/homework4.pdf

---

1. Get a copy of the python program **fem1d_model.py** from **http://people.sc.fsu.edu/~jburkardt/py_src/fem1d/fem1d_model.html** which solves the problem:

```
-u'' + u = x, 0 < x < 1
u(0) = 0, u(1) = 0
```

for which the exact solution is $u(x) = x - \sinh(x)/\sinh(1)$. Add code to compute the L2 norm and H1 seminorm of the error in the computed solution. Use your modified code to compute the L2 and H1 errors for meshes that use 2, 4, 8, 16, 32 and 64 elements. For each solution, compute the L2 norm and H1 seminorm of the error. Make a table of ratios of successive error norms, which begins as follows:

```
L2 ratios          H1 ratios
---------------    ---------------
L2( 2) / L2( 4)    H1( 2) / H1( 4)
L2( 4) / L2( 8)    H1( 4) / H1( 8)
L2( 8) / L2(16)    H1( 8) / H1(16)
L2(16) / L2(32)    H1(16) / H1(32)
L2(32) / L2(64)    H1(32) / H1(64)
```

We expect to see a definite pattern in these two columns. **Turn in** your table.

---

2. Assuming we can solve the problem

```
-u'' + u = x, 0 < x < 1
u(0) = 0, u(1) = 0
```

we might want to compute the "average" value of the solution over the interval, which is simply the integral of the solution divided by the length of the interval. Starting from a copy of **fem1d_model.py**, have the program estimate $\int_0^1 u^h(x)dx$. Compute and print this value for 2, 4, 8 elements. **Turn in** your table.

---

3. Now we want to solve the problem:

```
-u'' + u = x^3+3x^2-15x-6, 0 < x < 1
u(0) = 0, u'(1) = 0
```

This means we want to use a Neumann boundary condition on the right endpoint.

Starting from a copy of **fem1d_model.py**, modify the **exact_fn** function to be

```
value = x**3 + 3*x**2 - 9*x
```

and modify the **rhs_fn** function to be

```
value = x**3+3*x**2-15*x-6
```

Now change the boundary conditions. The Dirichlet condition on the left is still there, so we treat that the same way. The Neumman condition on the right is a *natural* condition, that is, it specifies a zero first derivative. Because this condition is "natural", we don't overwrite the last finite element equation by a boundary condition; instead, we do nothing there. So modify the program so that we **do not** overwrite the last row of the matrix and right hand side; cross your fingers and run the program.

Once the program has computed the answer, you can tell whether the derivative is zero at the right endpoint simply by looking at the coefficient vector. **Turn in** a table of the solution at the nodes, and explain why you are sure the function is "flat" at the right endpoint.