

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/hw3`
- add your write-up `hw3.pdf` and supporting files `hw3*.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. In this homework you will implement the FEM using the steps from class. Consider the PDE

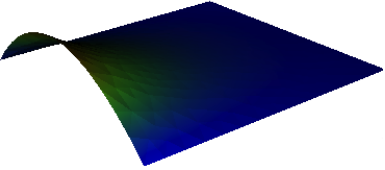
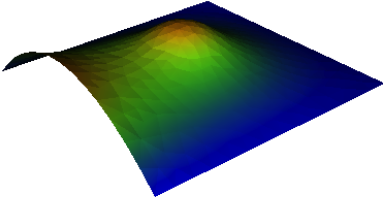
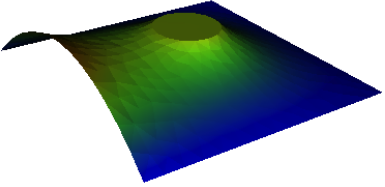
$$-\nabla(\kappa(x)\nabla u) = f \quad \text{on } \Omega = [-1, 1]^2 \quad (1)$$

$$u = g \quad \text{on } \partial\Omega \quad (2)$$

We use

$$g(x, y) = \begin{cases} 20(1 - y^2) & \text{for } x = -1 \\ 0 & \text{else} \end{cases} \quad (3)$$

and consider three tests:

test	$\kappa(x, y)$	$f(x, y)$	figure
(a)	0.1	$f(x, y) = 0$	
(b)	0.5	$\begin{cases} 25 & \text{for } \sqrt{x^2 + y^2} \leq 0.25 \\ 0 & \text{else} \end{cases}$	
(c)	$\begin{cases} 25 & \text{for } \sqrt{x^2 + y^2} \leq 0.25 \\ 0.1 & \text{else} \end{cases}$	$\begin{cases} 25 & \text{for } \sqrt{x^2 + y^2} \leq 0.25 \\ 0 & \text{else} \end{cases}$	

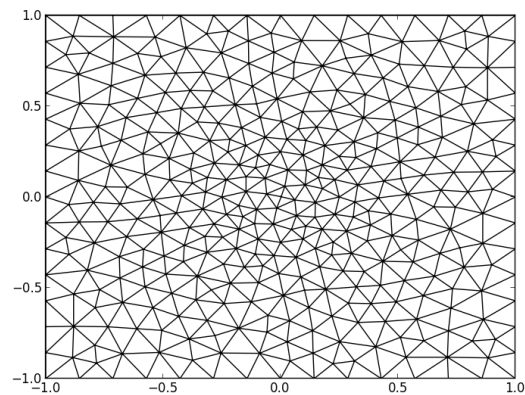
You will use `febasic_template.py` as a starting point. It should run as-is to yield a zero solution:

```
1 >>> run febasic_template.py
```

From this, you will implement the finite element with 12 steps:

Step 0 Read in a mesh. The mesh provided is an unstructured mesh of the square $[-1, 1]^2 \subset \mathbb{R}^2$. The mesh conforms to a disc of radius 0.25 centered at the origin as shown here:

```
1 import mesh2d
2 import matplotlib.pyplot as plt
3 V, E = mesh2d.xml('simple.xml')
4 plt.triplot(V[:,0], V[:,1], E)
5 plt.show()
```



note: this step is included in the template

For each element we need to do the following

Step 1 At this point, the list of vertices (\mathbf{V} , $\mathbf{nv} \times 2$) and elements (\mathbf{E} , $\mathbf{ne} \times 3$) are available. In this step, define the coordinates of the current element.

Step 2 Use the coordinates to define the Jacobian ($\mathbf{J} = J$), the inverse of the transposed Jacobian ($\mathbf{invJ} = J^{-T}$), and the determinant of the Jacobian ($\mathbf{detJ} = |J|$).

Step 3 Now define the gradient of the basis functions ($\mathbf{dbasis} = \nabla \lambda_r$) on the reference triangle S , for each basis function.

Step 4 Next construct the gradient of the basis function on the element ($\mathbf{dphi} = \nabla \phi_r = J^{-T} \lambda_R$) for each basis function.

Step 5 In this step, calculate the bilinear form on the element:

$$\begin{aligned} \mathbf{Aelem} &= \int_E \kappa(x) \nabla \phi_r \cdot \nabla \phi_s dx \\ &= \int_E \kappa(x) (J^{-T} \nabla \lambda_r)^T (J^{-T} \nabla \lambda_s) dx \\ &= (J^{-T} \nabla \lambda_r)^T (J^{-T} \nabla \lambda_s) |J| \int_E \kappa(T(\alpha)) d\alpha \end{aligned}$$

Step 6 ...and the linear functional (right-hand side) on each element:

$$\begin{aligned} \mathbf{belem} &= \int_E f(x) \phi_r dx \\ &= |J| \int_S f(T(\alpha)) \lambda_r(\alpha) d\alpha \end{aligned}$$

note: in steps 5 and 6, you will have to compute both $\int_S \lambda_r d\alpha$ and $\int_S 1 d\alpha$.

Step 7 Now that the element contributions to the matrix and the right-hand side are computed, they need to be added to the global matrix (\mathbf{A}).

note: this step is included in the template

Step 8 The matrices were constructed in COO format, meaning that duplicates were formed. Convert to CSR to contract the duplicate entries. Then convert back to COO for easier manipulation.

note: this step is included in the template

Step 9 Boundary conditions are applied in this step. You should define a function u_0 that is equal to g on the $x = 0$ boundary and zero elsewhere. To do this, first create two boolean vectors: one that is true for each vertex on the $x = 0$ boundary (\mathbf{gflag}) and the other that is true for each vertex on the whole boundary (\mathbf{Dflag}). Then set u_0 appropriately.

Next, we need to set

$$b \leftarrow b - Au_0,$$

which is just a matrix-vector product. Finally, we need to set:

$$A_{ij} = 0$$

$$A_{ii} = 1$$

$$b_i = 0,$$

for each boundary node i . The best strategy is to loop over all of the non-zero entries in A by looking at `A.row`, `A.col`, and `A.data` in COO format. Check each row/column index for presence on the boundary and set the values accordingly.

Step 10 The matrix problem $Au = b$ is now solved and the solution is formed: $u \leftarrow u + u_0$.

note: this step is included in the template

Step 11 The solution is plotted.

note: this step is included in the template

2. Perform the same test in Dolfin. The Poisson example from the Dolfin tutorial:

<http://fenicsproject.org/documentation/tutorial/fundamentals.html#implementation-1> is a good starting point. In addition, the following definitions for f and κ may be useful:

```
1 if example == 'a':
2     f = Constant(0.0)
3     kappa = Constant(0.1)
4
5 if example == 'b':
6     f = conditional( le( sqrt(x[0]**2 + x[1]**2), 0.25), 25.0, 00.0)
7     kappa = Constant(0.1)
8
9 if example == 'c':
10    f = conditional( le( sqrt(x[0]**2 + x[1]**2), 0.25), 25.0, 0.0)
11    kappa = conditional( le( sqrt(x[0]**2 + x[1]**2), 0.25), 25.0, 0.1)
```

A template is also provided (`febasic_dolfin_template.py`) and can be run with

```
1 >>> run febasic_dolfin_template.py
```

Then the function f can be plotted with

```
1 plot(project(f,FunctionSpace(mesh,'DG',0)))
```

The steps to follow are similar, but much shorter:

Step 0 Read in the mesh and define the problem conditions.

note: this step is included in the template

Step 1 Define boundary conditions.

Step 2 Define test and trial functions.

Step 3 Define the bilinear form and linear functional.

Step 4 Solve the problem and plot.

What do you need to turn in? Something that convinces me that 1) you went through the steps to code the FEM and 2) that you have a working code.

1. Working code in a single script (aside from the functions that I provided). Plots of each example.
2. Working code in a single script (aside from the functions that I provided). Plots of each example.