

UCB Math 228A, Fall 2013: Problem Set 6

Due November 26

1. Write a MATLAB function of the form

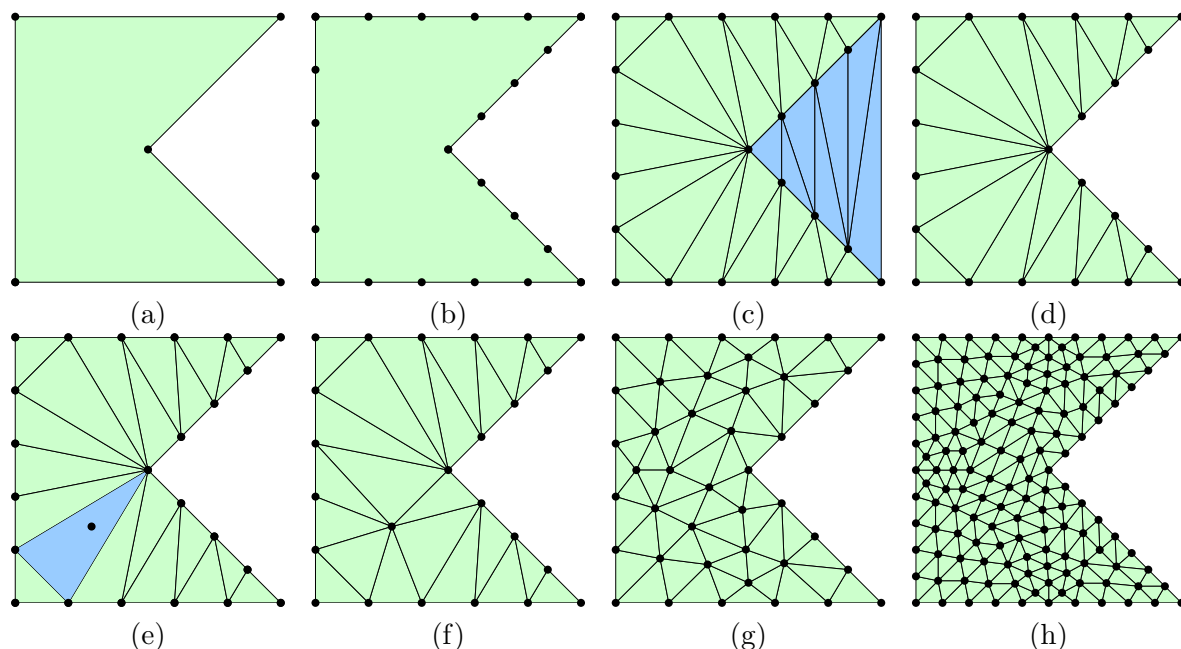
```
function [p,t,e]=pmesh(pv,hmax,nref)
```

which generates an unstructured triangular mesh of the polygon with vertices **pv**, with edge lengths approximately equal to $h_{\max}/2^{n_{\text{ref}}}$, using a simplified Delaunay refinement algorithm. The outputs are the node points **p** (N -by-2), the triangle indices **t** (T -by-3), and the indices of the boundary points **e**.

- The 2-column matrix **pv** contains the vertices x_i, y_i of the original polygon, with the last point equal to the first (a closed polygon).
- First, create node points along each polygon segment, such that all new segments have lengths $\leq h_{\max}$ (but as close to h_{\max} as possible). Make sure not to duplicate any nodes.
- Triangulate the domain using the `delaunayn` command.
- Remove the triangles outside the domain (see the `inpolygon` command).
- Find the triangle with largest area A . If $A > h_{\max}^2/2$, add the circumcenter of the triangle to the list of node points.
- Retriangulate and remove outside triangles (steps (c)-(d)).
- Repeat steps (e)-(f) until no triangle area $A > h_{\max}^2/2$.
- Refine the mesh uniformly n_{ref} times. In each refinement, add the center of each mesh edge to the list of node points, and retriangulate. Again, make sure not to duplicate any nodes, see e.g. the command `unique(p,'rows')`.

Finally, find the nodes **e** on the boundary using the `boundary_nodes` command. The following commands create the example in the figures. Also make sure that the function works with other inputs, that is, other polygons, h_{\max} , and n_{ref} .

```
pv=[0,0;1,0;.5,.5;1,1;0,1;0,0];
[p,t,e]=pmesh(pv,0.2,1);
tplot(p,t)
```



2. Implement a MATLAB function

```
function u=fempoi(p,t,e)
```

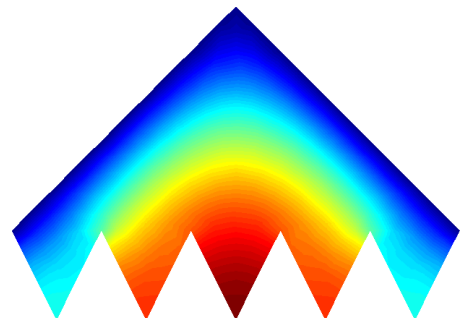
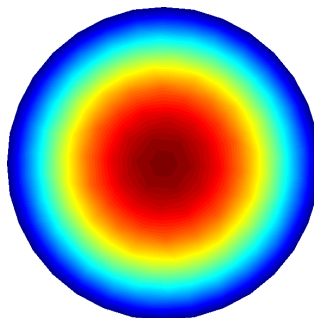
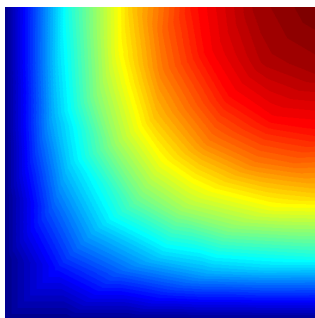
that solves Poisson's equation $-\nabla^2 u(x,y) = 1$ on the domain described by the unstructured triangular mesh \mathbf{p}, \mathbf{t} . The boundary conditions are homogeneous Neumann ($\mathbf{n} \cdot \nabla u = 0$) except for the nodes in the array \mathbf{e} which are homogeneous Dirichlet ($u = 0$).

Here are a few examples for testing the function.

```
% Square, Dirichlet left/bottom
pv=[0,0;1,0;1,1;0,1;0,0];
[p,t,e]=pmesh(pv,0.2,0);
e=e(p(e,1)==0 | p(e,2)==0);
u=fempoi(p,t,e);
tplot(p,t,u)

% Circle, all Dirichlet
n=32; phi=2*pi*(0:n)'/n;
pv=[cos(phi),sin(phi)];
[p,t,e]=pmesh(pv,2*pi/n,0);
u=fempoi(p,t,e);
tplot(p,t,u)

% Complex polygon geometry, mixed Dirichlet/Neumann
x=(0:.1:1)';
y=.1*cos(10*pi*x);
pv=[x,y; .5,.6; 0,.1];
[p,t,e]=pmesh(pv,0.05,0);
e=e(p(e,2)>=.6-abs(p(e,1)-.5));
u=fempoi(p,t,e);
tplot(p,t,u)
```



3. Implement a MATLAB function

```
function errors=poiconv(pv,hmax,nrefmax)
```

that solves the all-Dirichlet Poisson problem for the polygon `pv`, using the mesh parameters `hmax` and `nref=0,1,...,nrefmax`. Consider the solution on the finest mesh the exact solution, and compute the max-norm of the errors at the nodes for all the other solutions (note that this is easy given how the meshes were refined – the common nodes appear first in each mesh). The output `errors` is a vector of length `nrefmax` containing all the errors.

Test the function using the commands below:

```
hmax=0.15;
for pv={ [0,0;1,0;1,1;0,1;0,0], [0,0;1,0;.5,.5;1,1;0,1;0,0] }
    errors=poiconv(pv{1},hmax,3)
    loglog(hmax./2.^(0:2),errors)
    rate=log2(errors(end-1))-log2(errors(end))
end
```

Code Submission: E-mail all requested and supporting MATLAB files to David at anderson@math.berkeley.edu as a zip-file named `lastname_firstname_6.zip`, for example `anderson_david_6.zip`.