MATH 666: Finite Element Methods
Homework 2

Caleb Logemann

October 23, 2018

#1 Consider the 2D Poisson equation:

$$\begin{aligned} \textbf{PDE}: &\quad -\nabla\cdot\nabla u = f(x,y) \quad \text{in} \quad \Omega = [-1,1]\times[-1,1] \\ \textbf{BC}: &\quad u + \nabla u\cdot\hat{\mathbf{n}} = g \quad \text{on} \quad \partial\Omega \end{aligned}$$

(a) Recast this problem as a variational problem. Clearly state the test and trial function spaces.
In order to recast this as a variational problem, I will multiply by a test function and integrate over Ω.

$$\iint_\Omega -(\nabla\cdot\nabla u)v\,\mathrm{d}\mathbf{x} = \iint_\Omega fv\,\mathrm{d}\mathbf{x}$$

Integrating by parts gives

$$\iint_\Omega \nabla u\cdot\nabla v\,\mathrm{d}\mathbf{x} - \int_{\partial\Omega}(\nabla u\cdot\hat{\mathbf{n}})v\,\mathrm{d}s = \iint_\Omega fv\,\mathrm{d}\mathbf{x}$$

Using the boundary condition we see that $\nabla u\cdot\hat{\mathbf{n}} = g-u$ on $\partial\Omega$

$$\begin{aligned} \iint_\Omega \nabla u\cdot\nabla v\,\mathrm{d}\mathbf{x} - \int_{\partial\Omega}(g-u)v\,\mathrm{d}s &= \iint_\Omega fv\,\mathrm{d}\mathbf{x} \\ \iint_\Omega \nabla u\cdot\nabla v\,\mathrm{d}\mathbf{x} + \int_{\partial\Omega} uv\,\mathrm{d}s &= \iint_\Omega fv\,\mathrm{d}\mathbf{x} + \int_{\partial\Omega} gv\,\mathrm{d}s \end{aligned}$$

In order for this equation to be well defined u and v must be in the space $H^1(\Omega)$, that is they are square integrable on Ω and the norm of their gradients are square integrable on Ω. Note that being in $H^1(\Omega)$ implies $L^2(\partial\Omega)$, so the boundary integrals are also well defined.
So the bilinear form of this variational problem is to find $u\in H^1(\Omega)$ such that

$$B(u,v) = L(v)$$

for all $v\in H^1(\Omega)$, where

$$B(u,v) = \iint_\Omega \nabla u\cdot\nabla v\,\mathrm{d}\mathbf{x} + \int_{\partial\Omega} uv\,\mathrm{d}s$$

and

$$L(v) = \iint_\Omega fv\,\mathrm{d}\mathbf{x} + \int_{\partial\Omega} gv\,\mathrm{d}s.$$

(b) Show that the variational problem has a unique solution by showing that it meets all of the criteria of the Lax-Milgram Theorem.
The four conditions of the Lax-Milgram Theorem are symmetry, continuity of B, V-ellipticity, and continuity of L, and are shown below.

$$\begin{aligned} B(u,v) &= B(v,u) \\ |B(u,v)| &\le \gamma\|u\|_V\|v\|_V \\ B(v,v) &\ge \alpha\|v\|_V^2 \\ |L(v)| &\le \Gamma\|v\|_V \end{aligned}$$

I will show these four conditions in order. First, symmetry

$$\begin{aligned} B(u,v) &= \iint_\Omega \nabla u\cdot\nabla v\,\mathrm{d}\mathbf{x} + \int_{\partial\Omega} uv\,\mathrm{d}s \\ &= \iint_\Omega \nabla v\cdot\nabla u\,\mathrm{d}\mathbf{x} + \int_{\partial\Omega} vu\,\mathrm{d}s \\ &= B(v,u) \end{aligned}$$

Second, boundedness of B

$$\begin{aligned}|B(u,v)| &= \left|\iint_\Omega \nabla u \cdot \nabla v \,\mathrm{d}\mathbf{x} + \int_{\partial\Omega} uv \,\mathrm{d}s\right| \\ &\leq \left|\iint_\Omega \nabla u \cdot \nabla v \,\mathrm{d}\mathbf{x}\right| + \left|\int_{\partial\Omega} uv \,\mathrm{d}s\right|\end{aligned}$$

Using Cauchy-Schwarz on both integrals gives

$$\leq \|\|\nabla u\|\|_{L^2(\Omega)} \|\|\nabla v\|\|_{L^2(\Omega)} + \|u\|_{L^2(\partial\Omega)} \|v\|_{L^2(\partial\Omega)}$$

Since $\partial\Omega \subset \Omega$, this implies that $\|u\|_{L^2(\partial\Omega)} \leq \|u\|_{L^2(\Omega)}$, so

$$\leq \|\|\nabla u\|\|_{L^2(\Omega)} \|\|\nabla v\|\|_{L^2(\Omega)} + \|u\|_{L^2(\Omega)} \|v\|_{L^2(\Omega)}$$

Now since $\|u\|_{H^1(\Omega)}$ is greater than both $\|\|\nabla u\|\|_{L^2(\Omega)}$ and $\|u\|_{L^2(\Omega)}$

$$|B(u,v)| \leq 2\|u\|_{H^1(\Omega)} \|v\|_{H^1(\Omega)}$$

Third I will show V-ellipticity of B. There are two possible cases $v = 0$ on $\partial\Omega$ or $v \neq 0$ on $\partial\Omega$. If $v = 0$ on $\partial\Omega$, then

$$\begin{aligned}B(v,v) &= \iint_\Omega \|\nabla v\|^2 \,\mathrm{d}\mathbf{x} + \int_{\partial\Omega} v^2 \,\mathrm{d}s \\ &= \iint_\Omega \|\nabla v\|^2 \,\mathrm{d}\mathbf{x} \\ &= \frac{1}{2}\Big(\iint_\Omega \|\nabla v\|^2 \,\mathrm{d}\mathbf{x} + \iint_\Omega \|\nabla v\|^2 \,\mathrm{d}\mathbf{x}\Big) \\ &= \frac{1}{2}\Big(\|\|\nabla v\|\|^2_{L^2(\Omega)} + \|\|\nabla v\|\|^2_{L^2(\Omega)}\Big)\end{aligned}$$

Poincare's Inequality states that there exists a constant $C > 0$ such that $C\|v\|^2_{L^2(\Omega)} \leq \|\|\nabla v\|\|^2_{L^2(\Omega)}$, therefore

$$\begin{aligned}B(v,v) &\geq \frac{1}{2}\Big(\|\|\nabla v\|\|^2_{L^2(\Omega)} + C\|v\|^2_{L^2(\Omega)}\Big) \\ &\geq \frac{1}{2}\min\{1, C\}\Big(\|\|\nabla v\|\|^2_{L^2(\Omega)} + \|v\|^2_{L^2(\Omega)}\Big) \\ &= \frac{1}{2}\min\{1, C\}\|v\|_{H^1(\Omega)}\end{aligned}$$

If $v \neq 0$ on $\partial\Omega$, then we can make use of Friedrich's Inequality, which states that there exists constants $C_1 > 0$ and $C_2 > 0$ such that

$$\iint_\Omega v^2 \,\mathrm{d}\mathbf{x} \leq C_1 \iint_\Omega \|\nabla v\|^2 \,\mathrm{d}x + C_2 \int_{\partial\Omega} v^2 \,\mathrm{d}s$$

This is equivalent to

$$\frac{1}{C}\iint_\Omega v^2 \,\mathrm{d}\mathbf{x} \leq \iint_\Omega \|\nabla v\|^2 \,\mathrm{d}x + \int_{\partial\Omega} v^2 \,\mathrm{d}s$$

where $C = \max\{C_1, C_2\}$. Now,

$$\begin{aligned}B(v,v) &= \iint_\Omega \|\nabla v\|^2 \,\mathrm{d}\mathbf{x} + \int_{\partial\Omega} v^2 \,\mathrm{d}s \\ &\geq \frac{1}{2}\iint_\Omega \|\nabla v\|^2 \,\mathrm{d}\mathbf{x} + \frac{1}{2}\Big(\iint_\Omega \|\nabla v\|^2 \,\mathrm{d}\mathbf{x} + \int_{\partial\Omega} v^2 \,\mathrm{d}s\Big)\end{aligned}$$

Using Friedrich's Inequality

$$\begin{aligned}
&\geq \frac{1}{2}\iint_\Omega \|\nabla v\|^2 \,\mathrm{d}\mathbf{x} + \frac{1}{2C}\iint_\Omega v^2 \,\mathrm{d}\mathbf{x} \\
&\geq \min\left\{1, \frac{1}{C}\right\}\left(\iint_\Omega \|\nabla v\|^2 \,\mathrm{d}\mathbf{x} + \iint_\Omega v^2 \,\mathrm{d}\mathbf{x}\right) \\
&= \min\left\{1, \frac{1}{C}\right\}\|v\|_{H^1(\Omega)}
\end{aligned}$$

Therefore B is V-elliptic.
Lastly I will show that L is bounded,

$$\begin{aligned}
|L(v)| &= \left|\iint_\Omega fv \,\mathrm{d}\mathbf{x} + \int_{\partial\Omega} gv \,\mathrm{d}s\right| \\
&\leq \left|\iint_\Omega fv \,\mathrm{d}\mathbf{x}\right| + \left|\int_{\partial\Omega} gv \,\mathrm{d}s\right|
\end{aligned}$$

Using Cauchy-Schwarz on both integrals gives

$$\leq \|f\|_{L^2(\Omega)}\|v\|_{L^2(\Omega)} + \|g\|_{L^2(\partial\Omega)}\|v\|_{L^2(\partial\Omega)}$$

Now since $\|v\|_{H^1(\Omega)}$ is greater than both $\|v\|_{L^2(\Omega)}$ and $\|v\|_{L^2(\partial\Omega)}$

$$\begin{aligned}
&\leq \|f\|_{L^2(\Omega)}\|v\|_{H^1(\Omega)} + \|g\|_{L^2(\partial\Omega)}\|v\|_{H^1(\Omega)} \\
&\leq 2\max\left\{\|f\|_{L^2(\Omega)}, \|g\|_{L^2(\partial\Omega)}\right\}\|v\|_{H^1(\Omega)}
\end{aligned}$$

This shows that the variational problem satisfies all of the criteria for the Lax-Milgram theorem, therefore it has a unique solution.

(c) Develop a finite element method for this problem based on square elements. On each element, the solution on the four corners is interpolated with the following interpolant:

$$s(x, y) = \alpha_1 + \alpha_2 x + \alpha_3 y + \alpha_4 xy.$$

Write out the basis functions and explicitly write out the linear system that arises from your FEM(you can assume that the grid spacing in x and y are the same and uniform).
There will be a basis function for each node on the grid. Since we are implicitly enforcing the boundary conditions through the variational problem, all of the basis function can be the same. We don't need different basis functions for the boundary. Note that the basis functions corresponding with nodes on the boundary will have some support outside Ω which will be ignored in future integrals. I will let M denote the number of points in one direction, so there will be $N = M^2$ total nodes on the grid. I will use a row-wise numbering of these nodes, that is the first row of nodes will be numbered $1 \to M$, the second row $M + 1 \to 2M$, and so on. The top right node will be numbered M^2.
Now the basis function for node k can be expressed as

$$\phi_k[x, y] = \begin{cases}
1 - \frac{1}{h}(x - x_k) - \frac{1}{h}(y - y_k) + \frac{1}{h^2}(x - x_k)(y - y_k) & x_k \leq x \leq x_k + h \text{ and } y_k \leq y \leq y_k + h \\
1 + \frac{1}{h}(x - x_k) - \frac{1}{h}(y - y_k) - \frac{1}{h^2}(x - x_k)(y - y_k) & x_k - h \leq x \leq x_k \text{ and } y_k \leq y \leq y_k + h \\
1 + \frac{1}{h}(x - x_k) + \frac{1}{h}(y - y_k) + \frac{1}{h^2}(x - x_k)(y - y_k) & x_k - h \leq x \leq x_k \text{ and } y_k - h \leq y \leq y_k \\
1 - \frac{1}{h}(x - x_k) + \frac{1}{h}(y - y_k) - \frac{1}{h^2}(x - x_k)(y - y_k) & x_k \leq x \leq x_k + h \text{ and } y_k - h \leq y \leq y_k
\end{cases}$$

where x_k is the x coordinate of node k and y_k is the y coordinate, and h is the grid spacing. The derivatives of the basis functions are

$$\frac{\partial}{\partial x}(\phi_k[x,y]) = \begin{cases} -\frac{1}{h} + \frac{1}{h^2}(y-y_k) & x_k \le x \le x_k + h \text{ and } y_k \le y \le y_k + h \\ \frac{1}{h} - \frac{1}{h^2}(y-y_k) & x_k - h \le x \le x_k \text{ and } y_k \le y \le y_k + h \\ \frac{1}{h} + \frac{1}{h^2}(y-y_k) & x_k - h \le x \le x_k \text{ and } y_k - h \le y \le y_k \\ -\frac{1}{h} - \frac{1}{h^2}(y-y_k) & x_k \le x \le x_k + h \text{ and } y_k - h \le y \le y_k \end{cases}$$

and

$$\frac{\partial}{\partial y}(\phi_k[x,y]) = \begin{cases} -\frac{1}{h} + \frac{1}{h^2}(x-x_k) & x_k \le x \le x_k + h \text{ and } y_k \le y \le y_k + h \\ -\frac{1}{h} - \frac{1}{h^2}(x-x_k) & x_k - h \le x \le x_k \text{ and } y_k \le y \le y_k + h \\ \frac{1}{h} + \frac{1}{h^2}(x-x_k) & x_k - h \le x \le x_k \text{ and } y_k - h \le y \le y_k \\ \frac{1}{h} - \frac{1}{h^2}(x-x_k) & x_k \le x \le x_k + h \text{ and } y_k - h \le y \le y_k \end{cases}$$

Now the gradient of ϕ is $\nabla\phi_k = \left[\frac{\partial \phi_k}{\partial x}, \frac{\partial \phi_k}{\partial y}\right]^T$.

The stiffness matrix of this problem is given by

$$A_{ij} = \iint_\Omega \nabla\phi_i \cdot \nabla\phi_j \, \mathrm{d}\mathbf{x} + \int_{\partial\Omega} \phi_i \phi_j \, \mathrm{d}s$$

I will separate this into two terms

$$A_{ij} = B_{ij} + C_{ij}$$

where

$$B_{ij} = \iint_\Omega \nabla\phi_i \cdot \nabla\phi_j \, \mathrm{d}\mathbf{x}$$
$$C_{ij} = \int_{\partial\Omega} \phi_i \phi_j \, \mathrm{d}s$$

First I will look at the structure of B, for a given node i, the entries of $B_{ij} \neq 0$ for $j = i-m-1, i-m, i-m+1, i-1, i, i+1, i+m-1, i+m, i+m+1$. For an interior node i, these integrals can be computed to be

$$B_{ii} = \iint_\Omega \|\nabla\phi_i\|^2 \, \mathrm{d}\mathbf{x} = \frac{8}{3}$$
$$B_{i(i-1)} = B_{i(i+1)} = B_{i(i+m)} = B_{i(i-m)}$$
$$B_{i(i-1)} = \iint_\Omega \nabla\phi_i \cdot \nabla\phi_{i-1} \, \mathrm{d}\mathbf{x} = -\frac{1}{3}$$
$$B_{i(i-m-1)} = B_{i(i-m+1)} = B_{i(i+m-1)} = B_{i(i+m+1)}$$
$$B_{i(i-m-1)} = \iint_\Omega \nabla\phi_i \cdot \nabla\phi_{i-m-1} \, \mathrm{d}\mathbf{x} = -\frac{1}{3}$$

Note that we are using symmetry and that the integral is split evenly among adjacent elements. For i in the lower left corner

$$B_{ii} = \iint_\Omega \|\nabla\phi_i\|^2 \, \mathrm{d}\mathbf{x} = \frac{2}{3}$$
$$B_{i(i+1)} = B_{i(i+m)}$$
$$B_{i(i+1)} = \iint_\Omega \nabla\phi_i \cdot \nabla\phi_{i-1} \, \mathrm{d}\mathbf{x} = -\frac{1}{6}$$
$$B_{i(i+m+1)} = \iint_\Omega \nabla\phi_i \cdot \nabla\phi_{i-m-1} \, \mathrm{d}\mathbf{x} = -\frac{1}{3}$$

These values can be applied to the other corners by symmetry For i on the lower boundary not on the corner

$$\begin{aligned} B_{ii} &= \iint_\Omega \|\nabla\phi_i\|^2 \,\mathrm{d}\mathbf{x} = \frac{4}{3} \\ B_{i(i-1)} &= B_{i(i+1)} \\ B_{i(i-1)} &= \iint_\Omega \nabla\phi_i \cdot \nabla\phi_{i-1} \,\mathrm{d}\mathbf{x} = -\frac{1}{6} \\ B_{i(i+m)} &= \iint_\Omega \nabla\phi_i \cdot \nabla\phi_{i+m} \,\mathrm{d}\mathbf{x} = -\frac{1}{3} \\ B_{i(i+m-1)} &= B_{i(i+m+1)} \\ B_{i(i+m-1)} &= \iint_\Omega \nabla\phi_i \cdot \nabla\phi_{i+m-1} \,\mathrm{d}\mathbf{x} = -\frac{1}{3} \end{aligned}$$

These values can be translated to the other boundaries as well. Therefore B has the block tridiagonal form

$$B = \begin{bmatrix} \frac{1}{2}D & E & & & \\ E & D & E & & \\ & \ddots & \ddots & \ddots & \\ & & E & D & E \\ & & & E & \frac{1}{2}D \end{bmatrix}$$

where

$$D = \frac{1}{6}\begin{bmatrix} 8 & -2 & & & \\ -2 & 16 & -2 & & \\ & \ddots & \ddots & \ddots & \\ & & -2 & 16 & -2 \\ & & & -2 & 8 \end{bmatrix} \qquad E = \frac{1}{6}\begin{bmatrix} -1 & -2 & & & \\ -2 & -2 & -2 & & \\ & \ddots & \ddots & \ddots & \\ & & -2 & -2 & -2 \\ & & & -2 & -1 \end{bmatrix}$$

Now consider C_{ij}, for a given i on the boundary only $j = i-m, i-1, i, i+1, i+m$ can be nonzero. By symmetry there are only two possible values, either $j = i$ and the integral is evaluated over two elements or they only overlap on one element. Consider i on the lower boundary then

$$\begin{aligned} C_{ii} &= \int_{\partial\Omega} \phi_i\phi_i \,\mathrm{d}s = \frac{2h}{3} \\ C_{i(i-1)} &= C_{i(i+1)} \\ C_{i(i-1)} &= \int_{\partial\Omega} \phi_i\phi_{i-1} \,\mathrm{d}s = \frac{h}{6} \end{aligned}$$

This can be extended to all other boundary nodes by symmetry, the corner node also behave the same. Therefore C has the following form

$$C = \begin{bmatrix} F_1 & G & & & \\ G & F_2 & G & & \\ & \ddots & \ddots & \ddots & \\ & & G & F_2 & G \\ & & & G & F_1 \end{bmatrix}$$

where

$$F_1 = \frac{h}{6}\begin{bmatrix} 4 & 1 & & & \\ 1 & 4 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & 4 & 1 \\ & & & 1 & 4 \end{bmatrix} \quad F_2 = \frac{h}{6}\begin{bmatrix} 4 & & & \\ & 0 & & \\ & & \ddots & \\ & & & 4 \end{bmatrix} \quad G = \frac{h}{6}\begin{bmatrix} 1 & & & \\ & 0 & & \\ & & \ddots & \\ & & & 1 \end{bmatrix}$$

Now the final stiffness matrix A is just the sum of B and C.
The right hand side vector is given by

$$L_i = \iint_\Omega f\phi_i \, \mathrm{d}\mathbf{x} + \int_{\partial\Omega} g\phi_i \, \mathrm{d}s.$$

This can be computed using quadrature rules.

(d) The following code generates the stiffness matrix and right hand side vector given in part (c) and solves the system.

```
function [u, A] = h02_01(a, b, N, f, g)
    % this includes points at both ends
    h = (b-a)/(N-1);

    % the y position of row i is y(i)
    % this is indexed starting at 1
    x = @(j) a + (j-1)*h;
    y = @(i) a + (i-1)*h;

    % here k goes from 1 to N^2, and i and j go from 1 to N
    % point k = point (i, j)
    % this is row-wise odering, first row is 1 - N, second row N+1 - 2N, ...
    % if you are at the kth overall point and want to find the row use iFun(k), to
      ↪ find the colum use jFun(k)
    % if you have the row and column and want to know what point you are at use
      ↪ kFun(i, j)
    kNodeFun = @(i, j) j + (i-1)*N;
    iNodeFun = @(k) floor((k-1)/N) + 1;
    jNodeFun = @(k) mod(k-1,N)+1;

    e = ones(N, 1);
    % D matrix gradients on omega
    % D includes i, i-1, and i+1 terms
    % matrix D for middle rows
    D = spdiags([-1/3*e, 8/3*e, -1/3*e], -1:1, N, N);
    % left and right boundary points
    D(1, 1) = 4/3;
    D(end, end) = 4/3;

    % E matrix gradients on omega
    % E matrix includes i+m, i-m, i+m+1, i+m-1, i-m-1, and i-m+1
    E = spdiags([-1/3*e, -1/3*e, -1/3*e], -1:1, N, N);
    % left and right boundary points
    E(1, 1) = -1/6;
    E(end, end) = -1/6;

    % F matrix functions on boundaries
    % F includes i-1, i, i+1
    F1 = spdiags([h/6*e, 4*h/6*e, h/6*e], -1:1, N, N);
    F2 = sparse(N, N);
```

```
    F2(1,1) = 4*h/6;
    F2(end,end) = 4*h/6;

    G = sparse(N, N);
    G(1, 1) = h/6;
    G(end, end) = h/6;

    I = speye(N, N);
    I2 = I;
    I2(1,1) = 1/2;
    I2(end,end) = 1/2;
    I3 = I;
    I3(1, 1) = 0;
    I3(end, end) = 0;
    I4 = sparse(N, N);
    I4(1, 1) = 1;
    I4(end, end) = 1;
    UpperDiagonal = sparse(1:N-1, 2:N, 1, N, N);
    LowerDiagonal = sparse(2:N, 1:N-1, 1, N, N);
    diags = LowerDiagonal + UpperDiagonal;

    B = kron(I2, D) + kron(diags, E);
    C = kron(I4, F1) + kron(I3, F2) + kron(diags, G);
    A = B + C;

    b = zeros(N^2, 1);
    % loop over nodes
    for k = 1:N^2
        i = iNodeFun(k);
        j = jNodeFun(k);
        quadf = h^2*f(x(j), y(i));
        if (k == 1 || k == N || k==N^2 || k==N^2-N) % corner point
            b(k) = quadf/4 + h*g(x(j), y(i));
        elseif (i == 1 || j == 1 || i == N || j == N) % boundary
            b(k) = quadf/2 + h*g(x(j), y(i));
        else % interior
            b(k) = quadf;
        end
    end
    u = A\b;
end
```

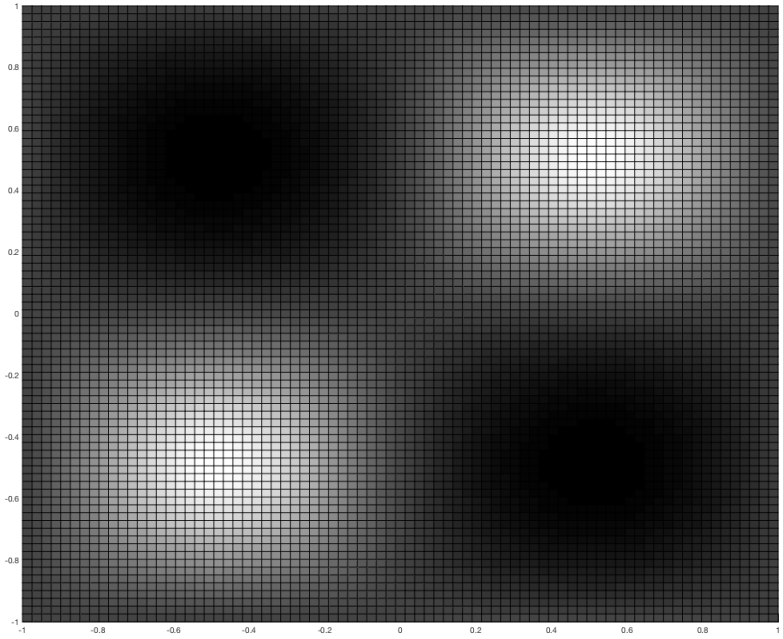
The following scipt uses the previous function to solve the system and evaluate the errors

```
%% Problem 1(d)
f = @(x, y) pi^2*exp(sin(pi*x).*sin(pi*y))*(2*sin(pi*x)*sin(pi*y) + 2*cos(pi*x).^2*
    ↪ cos(pi*y).^2 - cos(pi*x).^2 - cos(pi*y).^2);
g = @(x, y) (1 + pi*sin(pi*y))*(x == -1 && abs(y) < 1) + (1 - pi*sin(pi*y))*(x==1
    ↪ && abs(y) < 1) + (1 + pi*sin(pi*x))*(y == -1) + (1 - pi*sin(pi*x))*(y == 1);
uExact = @(x, y) exp(sin(pi*x).*sin(pi*y));
uExactX = @(x, y) (cos(pi*x).*sin(pi*y)).*exp(sin(pi*x).*sin(pi*y));
uExactY = @(x, y) (sin(pi*x).*cos(pi*y)).*exp(sin(pi*x).*sin(pi*y));
a = -1;
b = 1;

hArray = [];
EnergyErrorArray = [];
L2ErrorArray = [];
LInfErrorArray = [];
for N = [10, 20, 40, 80]
```

```
    [u, A] = h02_01(a, b, N, f, g);
    uMat = reshape(u, N, N);
    h = (b - a)/(N-1);
    hArray = [hArray, h];
    x = a:h:b;
    [X, Y] = meshgrid(x, x);
    surf(X, Y, uMat);
    view(2);
    U = uExact(X, Y);
    UGradX = uExactX(X, Y);
    UGradY = uExactY(X, Y);
    UExact = reshape(U,N^2,1);
    %surf(X, Y, U);
    %view(2);
    e = UExact - u;
    eMat = U - uMat;
    uMatX = [(uMat(2,:) - uMat(1,:))/h; (uMat(3:end, :) - uMat(1:end-2,:))/(2*h); (
        ↪ uMat(end,:) - uMat(end-1,:))/h];
    uMatY = [(uMat(:,2) - uMat(:,1))/h, (uMat(:,3:end) - uMat(:,1:end-2))/(2*h), (
        ↪ uMat(:,end) - uMat(:,end-1))/h];
    eMatX = uMatX - UGradX;
    eMatY = uMatY - UGradY;
    EnergyError = norm(h*sqrt(eMatX.^2 + eMatY.^2), 'fro');
    EnergyErrorArray = [EnergyErrorArray, EnergyError];
    L2Error = norm(h*e);
    L2ErrorArray = [L2ErrorArray, L2Error];
    LInfError = max(abs(e));
    LInfErrorArray = [LInfErrorArray, LInfError];
end
EnergyOrder = log(EnergyErrorArray(1:end-1)./EnergyErrorArray(2:end))./log(hArray(1
    ↪ :end-1)./hArray(2:end));
L2Order = log(L2ErrorArray(2:end)./L2ErrorArray(1:end-1))./log(hArray(2:end)./
    ↪ hArray(1:end-1));
LInfOrder = log(LInfErrorArray(2:end)./LInfErrorArray(1:end-1))./log(hArray(2:end)
    ↪ ./hArray(1:end-1));
disp(EnergyOrder);
disp(L2Order);
disp(LInfOrder);
```

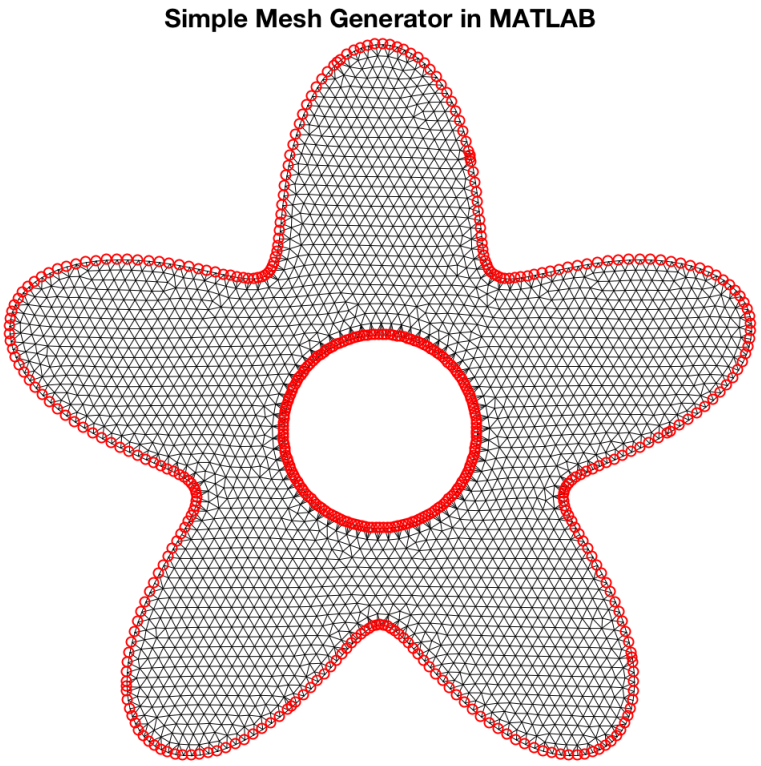
The script produces the following example plot.



The following table shows the order of convergence, but I was unable to properly compute the energy error.

M	h	Energy Error	Energy Order	L2 Error	L2 Order
10	0.2	-	-	0.19	-
20	0.1	-	-	0.0357	2.283
40	0.05	-	-	0.007877	2.102
80	0.025	-	-	0.001867	2.039

#2 Using the code posted online I was able to generate the following mesh for the given level set function.



#3 The following code implements the cg1 method for any given triangulation in 2D.

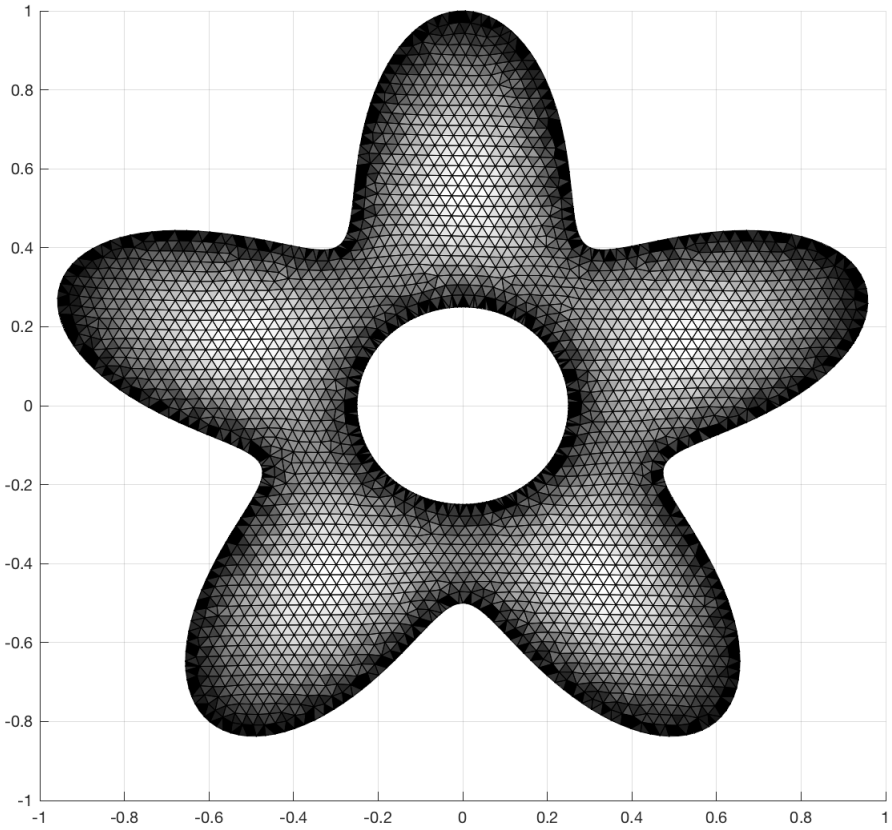
```
function [u] = cg1_2D(nodes, elements, NIN, vandermondeMatrix, gradPhiFun, bilinearForm
    ↪ , f)
    % Uses homogenous dirichlet boundary conditions
    % TODO: Allow for other boundary conditions
    A = sparse(NIN, NIN);
    b = zeros(NIN, 1);
    [numElements, nodesPerElement] = size(elements);
    [numNodes, ~] = size(nodes);
    % loop over elements
    for i = 1:numElements
        elementNodes = elements(i, :);
        x = nodes(elementNodes,:);
        a = polygonArea(x);
        M = vandermondeMatrix(x);
        v = eye(nodesPerElement);
        phi = (M\v)';
        grad_phi = gradPhiFun(phi);
        for j = 1:nodesPerElement
            for k = 1:nodesPerElement
                % only update for interior
                % would be different for other bc
                if (elementNodes(j) <= NIN && elementNodes(k) <= NIN)
                    A(elementNodes(j), elementNodes(k)) = ...
                        A(elementNodes(j), elementNodes(k)) + ...
```

```
                        bilinearForm(x, phi(j, :), phi(k,:), ...
                        @(x, y) grad_phi(j, x, y), @(x, y) grad_phi(k, x, y), a);
                end
            end
            % form b
            % only interior - force zero on boundary
            if (elementNodes(j) <= NIN)
                b(elementNodes(j)) = b(elementNodes(j)) + 1/nodesPerElement*a*f(x(j,1),
                    ↪  x(j,2));
            end
        end
    end

    u = A\b;
    u = [u; zeros(numNodes - NIN, 1)];
end
```

#4 The following script uses the previous function to solve the poisson problem on the mesh found in problem 2.

```
%% Problem 4
h = 0.03;
[p, t, NIN] = sample_mesh(h);
vandermondeMatrix = @(x) [ones(3, 1), x(:,1), x(:,2)];
gradPhiFun = @(phi) (@(j, x, y) [phi(j, 2); phi(j, 3)]);
bilinearForm = @(x, phi1, phi2, gphi1, gphi2, a) a*gphi1(x(1,1), x(1,2))'*gphi2(x(1,1),
    ↪  x(1,2));
f = @(x, y) 1;
u = cg1_2D(p, t, NIN, vandermondeMatrix, gradPhiFun, bilinearForm, f);
trisurf(t, p(:,1), p(:,2), u)
view(2);
```


1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-1
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
1