Poisson's Equation

Computer Lab 2

Poisson's Equation

Consider Poisson's equation: Solve $u(\boldsymbol{x})$ from

$$-\nabla \cdot (\kappa(\boldsymbol{x})\nabla u(\boldsymbol{x})) = f(\boldsymbol{x}), \quad \boldsymbol{x} \in \Omega, \tag{1}$$

$$\boldsymbol{n} \cdot \kappa(\boldsymbol{x})\nabla u(\boldsymbol{x}) = \gamma(g(\boldsymbol{x}) - u(\boldsymbol{x})), \quad \boldsymbol{x} \in \partial\Omega, \tag{2}$$

where $f(\boldsymbol{x})$ and $g(\boldsymbol{x})$ are given functions from $L_2(\Omega)$, $\kappa(\boldsymbol{x}) \in L_2(\Omega)$ is a positive function and γ is a positive parameter, and Ω is a polygonal domain with boundary $\partial\Omega$ and outward pointing unit normal $\boldsymbol{n}$. The boundary condition (2) is a so called Robin boundary condition, which is a combination of Dirichlet and Neumann boundary conditions.

Finite Element Approximation

The weak form of (1) reads: Find $u \in H^1$ such that

$$\int_\Omega \kappa \nabla u \cdot \nabla v \ \mathrm{d}\boldsymbol{x} + \int_{\partial\Omega} \gamma u v \ \mathrm{d}s = \int_\Omega f v \ \mathrm{d}\boldsymbol{x} + \int_{\partial\Omega} \gamma g v \ \mathrm{d}s, \tag{3}$$

for all $v \in H^1$, where H^1 is the Hilbert space $H^1 = \{v : \|v\| + \|\nabla v\| < \infty\}$.

Let $V_h \subset H^1$ be the space of all continuous piecewise linear functions on a partition $\mathcal{K} = \{K\}$ of Ω into triangles with diameter (longest edge) h_K.

The finite element approximation of (3) takes the form: Find $u_h \in V_h$ such that

$$\int_\Omega \kappa \nabla u_h \cdot \nabla v \ \mathrm{d}\boldsymbol{x} + \int_{\partial\Omega} \gamma u_h v \ \mathrm{d}s = \int_\Omega f v \ \mathrm{d}\boldsymbol{x} + \int_{\partial\Omega} \gamma g v \ \mathrm{d}s, \tag{4}$$

for all $v \in V_h$.

Problem 1. Derive the weak form (3).

How to obtain the finite element solution. Recall that the set of hat functions $\{\varphi_i\}_{i=1}^N$, where N is the number of nodes within the triangulation $\mathcal{K}$, is a basis for V_h.

The finite element approximation (4) is equivalent to

$$\int_\Omega \kappa \nabla u_h \cdot \nabla \varphi_i \ \mathrm{d}\boldsymbol{x} + \int_{\partial\Omega} \gamma u_h \varphi_i \ \mathrm{d}s = \int_\Omega f \varphi_i \ \mathrm{d}\boldsymbol{x} + \int_{\partial\Omega} \gamma g \varphi_i \ \mathrm{d}s, \tag{5}$$

for $i = 1, 2, \ldots, N$. This is due to the fact that if (5) holds for each basis function φ_i independently, then it also holds for any linear combination of them, and thus for any $v \in V_h$.

Since $u_h \in V_h$ it can be written as the sum

$$u_h = \sum_{j=1}^N x_j \varphi_j, \tag{6}$$

for come coefficients x_j to be determined.

Inserting (6) into (5) we end up with a $(N \times N)$ system of linear equations

$$\sum_{j=1}^N x_j \left(\int_\Omega \kappa \nabla \varphi_j \cdot \nabla \varphi_i \ \mathrm{d}\boldsymbol{x} + \int_{\partial\Omega} \gamma \varphi_j \varphi_i \ \mathrm{d}s \right) \tag{7}$$

$$= \int_\Omega f \varphi_i \ \mathrm{d}\boldsymbol{x} + \int_{\partial\Omega} \gamma g \varphi_i \ \mathrm{d}s, \qquad i = 1, 2, \ldots, N. \tag{8}$$

In matrix form we write this as

$$(A + R)x = b + r \tag{9}$$

where the entries of the left and right hand side matrices and vectors are given by

$$A_{ij} = \int_\Omega \kappa \nabla\varphi_i \cdot \nabla\varphi_j \; \mathrm{d}\boldsymbol{x}, \tag{10}$$

$$R_{ij} = \int_{\partial\Omega} \gamma\varphi_i\varphi_j \; \mathrm{d}s, \tag{11}$$

$$x = (x_1, x_2, \dots, x_N)^T, \tag{12}$$

$$b_j = \int_\Omega f\varphi_j \; \mathrm{d}\boldsymbol{x}, \tag{13}$$

$$r_j = \int_{\partial\Omega} \gamma g \varphi_j \; \mathrm{d}s. \tag{14}$$

All index runs over $i, j = 1, 2, \dots, N$.

Computer Implementation

Consider a single triangle K with nodes at its three corners $N_1 = (x_1, y_1)$, $N_2 = (x_2, y_2)$, and $N_3 = (x_3, y_3)$. To each node N_i, $i = 1, 2, 3$, there is a hat function φ_i associated, which takes on the value one at node N_i and zero at the other nodes. Each hat function is a linear function (i.e., a plane) on K. Hence

$$\varphi_i = a_i + b_i x + c_i y, \quad i = 1, 2, 3, \tag{15}$$

where the coefficients a_i, b_i, and c_i, are determined from the requirement

$$\varphi_i(N_j) = \begin{cases} 1, & i = j, \\ 0, & i \neq j. \end{cases} \tag{16}$$

Thus, for example, a_1, b_1 and c_1 are given by the equations $\varphi_1(x_1, y_1) = 1$, $\varphi_1(x_2, y_2) = 0$, and $\varphi_1(x_3, y_3) = 0$. In matrix form need to solve the following system to compute the coefficients of φ_1

$$\begin{pmatrix} 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \\ 1 & x_3 & y_3 \end{pmatrix} \begin{pmatrix} a_1 \\ b_1 \\ c_1 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}. \tag{17}$$

Note that the gradient of φ_i is just the constant vector $\nabla\varphi_i = (b_i, c_i)$.

As usual the global stiffness matrix A is formed by summing local stiffness matrices A^K, which are found by performing the integration (10) successively over each triangle K.

We get the (3×3) local stiffness matrix

$$A_{ij}^K = \int_K \kappa \nabla\varphi_i \cdot \nabla\varphi_j \, \mathrm{d}\boldsymbol{x} = (b_i b_j + c_i c_j) \int_K \kappa \, \mathrm{d}\boldsymbol{x}, \quad i, j = 1, 2, 3. \tag{18}$$

The $3 \cdot 3 = 9$ entries of A^K are to be assembled (at the right locations) into A. In Matlab this can be done very elegantly using vectorized operations as shown below.

```
for K = 1:size(t,2); % loop over the triangles
  nodes = t(1:3,K);  % find triangle K's nodes
  AK = ...           % compute the (3 x 3) stiffness matrix AK
  A(nodes,nodes) = A(nodes,nodes)+AK; % add AK(i,j), i,j=1,2,3,
                                      % to A(nodes(i),nodes(j))
end
```

Problem 2. Compute A^K on the reference triangle with corners at origo, $(1, 0)$, and $(0, 1)$. Verify that one of the eigenvalues of A^K is zero. Can you explain why?

Similar to the stiffness matrix A, the load vector b is formed by summing local load vectors b^K, which are found by restricting the integration (13) to one triangle K at a time. Using one point quadrature we get the approximation

$$b_i^K = \int_K f\varphi_i \approx f(x_c, y_c)\, \mathrm{area}\,(K)/3, \quad i = 1, 2, 3, \tag{19}$$

where (x_c, y_c) is the centroid of K.

Finally, if two nodes of triangle K lies along the domain boundary $\partial\Omega$, then the edge between them will contribute to the line integrals (11) and (14) associated with the boundary conditions. The boundary mass matrix R is assembled by looping over all the triangle edges lying along the domain boundary $\partial\Omega$. These are the columns of the mesh data matrix `e`, c.f. the matlab command `initmesh`.

If the edge E lies between the boundary nodes N_1 and N_2, then we have the local boundary mass matrix

$$R_{ij}^E = \int_E \gamma\varphi_i\varphi_j \,\mathrm{d}s = \frac{\gamma}{6}(1+\delta_{ij})\,\mathrm{length}\,(E), \quad i,j = 1,2, \tag{20}$$

and the local boundary vector

$$r_i^E = \int_E \gamma g\varphi_i \,\mathrm{d}s \approx \frac{\gamma}{2}\, g(N_i)\,\mathrm{length}\,(E), \quad i = 1,2. \tag{21}$$

Note that these are the same as the one-dimensional local mass matrix and load vector.

assemble.m The following code is a template routine for performing the assembly of the system of equations (9). Besides completing the code the user must supply the routines, `f`, `gamma`, and `g`, defining the source term f, the boundary penalty parameter γ, and the boundary data g. For example if $f = \sin x \sin y$, then `f` would look like

```
function z=f(x,y)
z=sin(x).*sin(y)
```

```
function [A,R,b,r] = assemble(p,e,t)
N = size(p,2);
A = sparse(N,N);
R = sparse(N,N);
b = zeros(N,1);
r = zeros(N,1);
% assemble stiffness matrix A, and load vector b.
for K = 1:size(t,2);
  nodes = t(1:3,K);
  x = p(1,nodes);
  y = p(2,nodes);
```

```
  area_K = polyarea(x,y);
  AK = ???
  bK = ???
  A(nodes,nodes) = A(nodes,nodes) + AK;
  b(nodes) = b(nodes) + bK;
end
% assemble boundary mass matrix R, and the vector r.
for E = 1:size(e,2)
  nodes = e(1:2,E);
  x = p(1,nodes);
  y = p(2,nodes);
  length_E = sqrt((x(1)-x(2))^2+(y(1)-y(2))^2);
  R(nodes,nodes) = R(nodes,nodes) + gamma*length_E*[2 1;1 2]/6;
  r(nodes) = r(nodes) + gamma*length_E*g(x,y)'/2;
end
```

Problem 3.

a) Implement the finite element solver `MyPoissonSolver` described above. Test your code by solving a problem with known solution (e.g., $-\Delta u = 2\pi^2 \sin(\pi x)\sin(\pi y)$ in the unit square and with $u = 0$ on the boundary. The exact solution is $u = \sin(\pi x)\sin(\pi y)$).

b) Compute the energy norm $\|\nabla u_h\|^2_{L^2(\Omega)} = x^T A x$ for the problem in a), and compare with theory. Make a convergence plot of the energy norm error (i.e., $\|\nabla(u - u_h)\|^2_{L^2(\Omega)}$) versus the mesh size h. Plot the result using a log-log diagram. Use `polyfit` to compute the slope of the graph. It should be roughly 2. Why?

Problem 4. Solve $-\Delta u = 1$ on the unit square $\Omega = [0,1] \times [0,1]$ with the boundary conditions $u = \cos(2\pi y)$ on the line $x = 0$, and $n \cdot \nabla u = 0$ on the rest of the boundary.