The Finite Element Method

Computer Lab 1

Introduction

The aim of this first computer laboration is to get started with using Matlab’s PDE Toolbox for solving partial differential equations.

Matlab’s PDE-Toolbox

We consider the Poisson equation with Robin boundary conditions.

\[-\Delta u = f, \quad \text{in } \Omega = [0, 1]^2 \quad (1a)\]

\[n \cdot \nabla u = k(g - u), \quad \text{on } \partial \Omega \quad (1b)\]

where \( f \) is a given function on \( \Omega \), and \( k > 0 \) and \( g \) are given functions on the boundary \( \partial \Omega \).

Start Matlab and invoke the M-file editor by typing `edit` at the prompt.

The next step is to draw the geometry \( \Omega \). It is represented as a matrix `geom`.

```matlab
function geom = unitsquare()
geom=[2 0 1 0 0 1 0;
     2 1 1 0 1 1 0;
     2 1 0 1 1 1 0;
     2 0 0 1 0 1 0]';
```

1
Each row of \texttt{geom} describes one of the four sides of the unit square. Read the help for the build-in command \texttt{decsg} for a comprehensive explanation of the geometry format.

Now, create the triangular mesh that is used by the finite element method by typing

\[
[p,e,t] = \text{initmesh(geom,'hmax',0.1)};
\]

The outputs \texttt{p}, \texttt{e}, and \texttt{t} are matrices describing the mesh. The point matrix \texttt{p} is of size $2 \times N$, where \(N\) is the number of nodes (i.e. triangle vertices). The rows of \texttt{p} contain the \(x\)- and \(y\)-coordinates of the node points. In the edge matrix \texttt{e}, the first and second rows contain indices of the starting and ending points of the edge segments making up the boundary \(\partial \Omega\). In the triangle matrix \texttt{t}, the first three rows contain the vertex points making up the triangles, and the fourth row contains the subdomain number. The vertices of each triangle are ordered counter-clockwise. The third argument 0.1 to \texttt{initmesh} is the meshsize \(h\), that is, the maximal length of any triangle side.

**Problem 1.** Plot the mesh with \texttt{pdemesh} and with meshsize \(h = 1, 0.1, \) and 0.05.

**Problem 2.** The node coordinates of the mesh can be found by typing \texttt{x=p(1,:)} and \texttt{y=p(2,:)} . Use \texttt{x} and \texttt{y} to compute the nodal values of \(u(x,y) = xy\) (i.e., \(u=x.*y\)). Then plot the piecewise linear representation of \(u\) with \texttt{pdesurf(p,t,u)}. This is the linear interpolant \(\pi_h u\) of \(u\) on the mesh.

**Variational Formulation**

The variational formulation of Poisson’s equation (1) takes the form: find \(u \in H^1\) such that

\[
a(u,v) = l(v), \quad \forall v \in V_h
\]

(2)
where the bilinear form \(a(\cdot, \cdot)\) and the linear form \(l(\cdot)\) is defined by
\[
a(u, v) = (\nabla u, \nabla v) + (ku, v)_{\partial \Omega} \\
l(v) = (f, v) + (kg, v)_{\partial \Omega}
\] (3) (4)

**Problem 3.** Derive the variational equation (2). *Hint:* Multiply \(-\Delta u = f\) by a smooth function \(v\) and integrate by parts over \(\Omega\). Substitute the boundary condition into any expressions involving \(n \cdot \nabla u\).

**Finite Element Approximation**

Let \(\{\varphi_i\}_{i=1}^N\) be the standard basis of hat functions for the space of continuous piecewise linears \(V_h\) on a mesh of \(\Omega\) with \(N\) nodes. A finite element approximation of the variational formulation (2) reads: find \(u_h \in V_h\) such that
\[
a(u_h, v) = l(v), \quad \forall v \in V_h
\] (5)

The finite element method (5) is equivalent to
\[
a(u_h, \varphi_i) = l(\varphi_i), \quad i = 1, \ldots, N
\] (6)

where \(\varphi_i\) is hat function \(i\).

Further, expanding \(u_h\) viz.
\[
u_h = \sum_{j=1}^{N} \xi_j \varphi_j
\] (7)

where \(\xi_j\) are \(N\) unknown coefficients, and substituting into (5), we obtain
\[
\sum_{j=1}^{N} \xi_j a(\varphi_j, \varphi_i) = l(\varphi), \quad i = 1, \ldots, N
\] (8)
which is a square $N \times N$ linear system for the $\xi_j$’s. Introducing the notation

$$A_{ij} = (\nabla \varphi_j, \nabla \varphi_i) \quad (9)$$

$$R_{ij} = (k \varphi_j, \varphi_i)_{\partial \Omega} \quad (10)$$

$$b_i = (f, \varphi_i) \quad (11)$$

$$r_i = (k g, \varphi_i)_{\partial \Omega} \quad (12)$$

we can write (8) in matrix form as

$$(A + R)\xi = b + r \quad (13)$$

The $N \times N$ matrix $A$ (or, $A + R$) is called the stiffness matrix, and the $N \times 1$ vector $b + r$ the load vector.

**Computer Implementation**

In order to obtain $u_h$ we must compute the stiffness matrix $A$ and the load vector $b$. This is done by breaking the integrals (9) and (11) into sums over the elements, and then assemble $A$ and $b$ from elemental contributions. Since the hat functions have very local support the integrals of $A_{ii}$ and $b_i$ only need to be computed on the triangles that contain node $N_i$. Also $A_{ij}$ is zero unless $N_i$ and $N_j$ are nodes on the same triangle.

**Problem 4.** Draw one of the hat functions with pdesurf or pdemesh. Use that the node values of $\varphi_i$ are zero at all nodes except at node $i$, where it is unity.

**Element Matrices, and Loads.** Let us consider just a single element $K$ with nodes $N_i = (x_i, y_i), i = 1, 2, 3$. Each of these three nodes correspond to a non-zero hat function, given by

$$\varphi_i = \frac{1}{2|K|} (a_i + b_ix + c_iy), \quad i = 1, 2, 3 \quad (14)$$
where $|K|$ is the area of $K$, and where
\begin{align*}
a_1 &= x_2 y_3 - x_3 y_2, & b_1 &= y_2 - y_3, & c_1 &= x_3 - x_2, \\
a_2 &= x_3 y_1 - x_1 y_3, & b_2 &= y_3 - y_1, & c_2 &= x_1 - x_3, \\
a_3 &= x_1 y_2 - x_2 y_1, & b_3 &= y_1 - y_2, & c_3 &= x_2 - x_1.
\end{align*}
(15)
(16)
(17)

These expressions are derived by requiring $\varphi_i(N_j) = \delta_{ij}$, $i, j = 1, 2, 3$, where $\delta_{ij}$ is 1 if $i = j$ and 0 otherwise. Note that the gradient of $\varphi_i$ is just the constant vector $\nabla \varphi_i = \frac{1}{2}[b_i, c_i]/|K|$.

Because there are only three non-zero hat functions on element $K$ we get a total of 9 integral contributions to the stiffness matrix $A$ from $K$. These are the entries of the $3 \times 3$ element stiffness matrix $A^K$.

$$A^K_{ij} = (\nabla \varphi_i, \nabla \varphi_j)_K = \frac{1}{4|K|} (b_i b_j + c_i c_j), \quad i, j = 1, 2, 3$$
(18)

These entries are to be added into $A$ via the local-to-global mapping of the node numbers. For example, if $N_1$ and $N_2$ have node number 4 and 7, then $A^K_{12}$ should be added to $A_{47}$.

The entries of the $3 \times 1$ element load vector $b^K$ are usually hard to compute exactly since $f$ might be tricky to integrate. It is customary to approximate $f$ with its linear interpolant $\pi^K_h f$ on $K$ and then integrate the interpolant. Recall that the interpolant is given by

$$\pi^K_h f = \sum_{j=1}^3 f_j \varphi_j$$
(19)

where $f_j = f(N_j)$ is the value of $f$ at node $N_j$.

Further, there is a formula for integrating products of three hat functions over a general triangle $K$

$$\int_K \varphi_1^m \varphi_2^n \varphi_3^p \, dx \, dy = \frac{m! n! p!}{(2 + m + n + p)!} 2|K|, \quad m, n, p \geq 0$$
(20)

Thus, using (20) we get the element load vector

$$b^K_i = (f, \varphi_i)_K \approx \frac{|K|}{12} \sum_{j=1}^3 (1 + \delta_{ij}) f_j, \quad i = 1, 2, 3$$
(21)
Finally, if two nodes of $K$ lie along the domain boundary $\partial \Omega$, then the edge between them contributes to the line integrals (10) and (12) associated with the boundary conditions. In particular, if edge $E$ lies between the boundary nodes $N_1$ and $N_2$, then we have

$$R_{ij}^E = (k \varphi_i, \varphi_j)_E = k \frac{|E|}{6} (1 + \delta_{ij}), \quad i, j = 1, 2$$  \hspace{1cm} (22)

and

$$r_i^E = (kg, \varphi_i)_E = kg \frac{|E|}{2}, \quad i = 1, 2$$ \hspace{1cm} (23)

where we have assumed that $k$ and $g$ are constant on $E$.

**My2DPoissonSolver.m** The following code is a complete finite element solver for Poisson’s equation. Input is a geometry matrix `geom` describing the computational domain $\Omega$. The user must supply the subroutines $f$, $k$, and $g$ defining the source term $f$, the boundary penalty parameter $k$, and the boundary data $g$, respectively. For example, if $f = \sin(x) \sin(y)$, then $f$ takes the form

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

Note the point-wise operator `.*`. The main routine looks like:

```matlab
function My2DPoissonSolver(geom)
[p,e,t] = initmesh(geom,'hmax',0.1);
[A,R,b,r] = assemble(p,e,t);
U = (A+R)\(b+r);
pdesurf(p,t,U)
```

The actual assembly of the involved matrices are done with the subroutine `assemble`: 
function [A,R,b,r] = assemble(p,e,t)
N = size(p,2); % number of nodes
A = sparse(N,N); 
R = sparse(N,N);
b = zeros(N,1);
r = zeros(N,1);
for K = 1:size(t,2);
    % node numbers for triangle K
    nodes = t(1:3,K);
    % node coordinates
    x = p(1,nodes); y = p(2,nodes);
    % triangle area
    area = polyarea(x,y);
    % hat function gradients (N.B. factor 2*area)
    b_ = [y(2)-y(3); y(3)-y(1); y(1)-y(2)}/2/area;
    c_ = [x(3)-x(2); x(1)-x(3); x(2)-x(1)]/2/area;
    % element stiffness matrix
    AK = (b_*b_’+c_*c_’)*area;
    % element load vector
    bK = [2 1 1; 1 2 1; 1 1 2]*area/12*feval('f',x,y)’;
    % add element contributions to A and b
    A(nodes,nodes) = A(nodes,nodes) + AK;
    b(nodes) = b(nodes) + bK;
end
for E = 1:size(e,2)
    % node numbers of boundary edge E
    nodes = e(1:2,E);
    % node coordinates
    x = p(1,nodes); y = p(2,nodes);
    ds = sqrt((x(1)-x(2))^2+(y(1)-y(2))^2);
    k_ = feval('k’,mean(x),mean(y));
    g_ = feval('g’,mean(x),mean(y));
    R(nodes,nodes) = R(nodes,nodes) + k_*[2 1; 1 2]*ds/6;
    r(nodes) = r(nodes) + k_*g_*[1; 1]*ds/2;
end

These matrices can also be assembled by the build-in functions assema and assemb. See the help for these commands for a description of their input and output.
Problem 5. Calculate the element stiffness matrix $A^K$ and the element load vector $b^K$ on the reference triangle, with corners $(0, 0)$, $(0, 1)$, and $(1, 0)$, assuming $f = 1$, $f = x$, and $f = 3 + 2x$.

Problem 6.

a) Implement the finite element solver `My2DPoissonSolver` described above for Poisson's equation. Test your code by solving the problem $-\Delta u = 2\pi^2 \sin(\pi x) \sin(\pi y)$ on the unit square with $u = 0$ on the boundary. The analytic solution is $u = \sin(\pi x) \sin(\pi y)$. Hint: You must approximate the boundary condition $u = 0$ with $n \cdot \nabla u = k(g - u)$ by choosing $k$ and $g$ appropriately.

b) Use the command `spy` to look at the nonzero entries of the stiffness matrix $A$.

c) Verify that $A$ is symmetric by computing $\max(\max(\text{abs}(A - A^T)))$. Also verify that $A$ is positive (semi) definite by computing the eigenvalues of $A$. What happens when you add $R$ to $A$? Hint: See the help for `eig` or `eigs`.

Problem 7. Consider the model problem

$$-\Delta u = 2x(1 - x) + 2y(1 - y), \quad \text{in } \Omega = [0, 1]^2$$

$$u = 0, \quad \text{on } \partial \Omega$$

a) Verify that the analytic solution is $u = x(1 - x)y(1 - y)$.

b) Compute the energy norm $\|u\|^2_E = (\nabla u, \nabla u)$.

c) Use `My2DPoissonSolver` to compute a sequence of finite element solutions $u_h$ on 10 meshes with meshsize ranging from $h = 0.125$ to 0.03. For each solution $u_h$ compute the energy norm $\|u_h\|^2_E = (\nabla u_h, \nabla u_h)$ via the matrix-vector multiplication $\xi^T A\xi$ or the dot product $b^T \xi$. Compare the results with b).

Problem 8. Extend your solver to handle also the equation $-\Delta u + \beta \cdot \nabla u = 1$. Run the code with $\beta = [1, 0]$, $[0, 5]$, and $[10, 10]$. What role does the the term $\beta \cdot \nabla u$ seem to play?