The Finite Element Method

Computer Session B1

Overall Goal and Aim

This is the first of seven computer sessions for the course Partial Differential Equations and Finite Elements, 5MA032. These sessions are intended to merge the mathematical theory covered in class with the practical implementation of the numerical algorithms underlying the finite element method. Each computer session will therefore contain about a dozen exercises dealing with both analytical and numerical issues of finite elements. You are expected to solve these exercises, document your solutions, and present them to the instructor during the computer sessions. By working through the exercises you will not only learn the content of the course, but also gain some insight into the synthesis of physics, mathematics, and computer science that form the basis of modern simulation techniques. Your average workload should be one computer session per week.

The aim of this first studio session is to get you started with using Matlab's PDE Toolbox as a handy yet powerful aid for developing and understanding finite element software.

Instructions

✔ To pass this session you must present your solutions to all problems with this mark to the instructor.

✰ This mark indicates that the problem is also included in the course's problem demonstration sessions (gives bonus points).

Matlab's PDE-Toolbox

Let us acquaint ourselves with Matlab's PDE Toolbox by solving Poisson's equation

$$-\Delta u = f, \qquad \text{in } \Omega = [0,1]^2 \tag{1a}$$
$$n \cdot \nabla u = k(g-u), \quad \text{on } \partial\Omega \tag{1b}$$

where f is a given function on Ω, and $k > 0$ and g are given functions on the boundary $\partial\Omega$, which is assumed to be smooth with outward pointing unit normal n.

First, start Matlab by clicking on its desktop icon. Then, invoke the M-file editor by typing

```
edit
```

at the prompt.

The next step is to draw the geometry Ω. It is represented as a matrix `geom`.

```
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]';
```

Each row of `geom` describes one of the four sides of the unit square. Read the help for the build-in command `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] = initmesh(geom,'hmax',0.1);
```

The outputs `p`, `e`, and `t` are matrices describing the mesh. The point matrix `p` is of size $2 \times N$, where N is the number of nodes (i.e., triangle vertices). The rows of `p` contain the x- and y-coordinates of the node points. In the edge matrix `e`, the first and second rows contain indices of the starting and ending points of the edge segments making up the boundary of Ω. In the triangle matrix `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 `initmesh` is the meshsize h, that is, the maximal length of any triangle side.

Problem 1. Plot the mesh with `pdemesh` and with meshsize $h = 1$, 0.1, and 0.05.

Problem 2. The node coordinates of the mesh can be found by typing `x=p(1,:)` and `y=p(2,:)`. Use `x` and `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 `pdesurf(p,t,u)`. This is the linear interpolant $\pi_h u$ of u on the mesh.

Problem 3. The PDE-Toolbox also has a Graphical User Interface (GUI). To start the GUI, type the `pdetool` at the Matlab prompt. Try to create a mesh of the unit circle $C = \{(r, \theta) : 0 \leq r \leq 1, 0 \leq \theta \leq 2\pi\}$, where r and θ are the usual polar coordinates. Also, make a mesh of the rectangle $R = \{(x, y) : -2 < x < 3, -4 < y < 5\}$, and the domain $R \setminus C$, that is, a rectangle with a cut out circular hole.

Variational Formulation

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

$$a(u, v) = l(v), \quad \forall v \in H^1(\Omega) \tag{2}$$

where the bilinear form $a(\cdot, \cdot)$ and the linear form $l(\cdot)$ is defined by

$$a(u, v) = (\nabla u, \nabla v) + (ku, v)_{\partial\Omega} \tag{3}$$

$$l(v) = (f, v) + (kg, v)_{\partial\Omega} \tag{4}$$

Note that we use the following notation for integrals

$$(a, b) = \int_\Omega a \cdot b \, dx \qquad (a, b)_\Gamma = \int_\Gamma a \cdot b \, ds$$

where $\Gamma \subset \partial\Omega$ and $\cdot$ denotes the natural inner product for elements a and b.

Problem 4. ✫ Derive the variational equation (2). *Hint:* Multiply $-\Delta u = f$ by a smooth function v and integrate by parts over Ω. Substitute the boundary condition into any expressions involving the normal derivative $n \cdot \nabla u$.

Problem 5. ✫ Write down the variation formulation of $-\Delta u + cu = f$ and $-\nabla \cdot (a\nabla u) = f$, where $a > 0$ and $c \geq 0$ are given functions. Assume boundary conditions of the form $n \cdot \nabla u = k(g - u)$ and $n \cdot (a\nabla u) = k(g - u)$, respectively.

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 Ω with N nodes. A finite element approximation of the variational formulation (2) reads: find $u_h \in V_h$ such that

$$a(u_h, v) = l(v), \quad \forall v \in V_h \tag{5}$$

The finite element method (5) is equivalent to

$$a(u_h, \varphi_i) = l(\varphi_i), \quad i = 1, \dots, N \tag{6}$$

where φ_i is hat function i.

Further, expanding u_h viz.

$$u_h = \sum_{j=1}^{N} \xi_j \varphi_j \tag{7}$$

where ξ_j are N unknown coefficients, and substituting into (5), we obtain

$$\sum_{j=1}^{N} \xi_j a(\varphi_j, \varphi_i) = l(\varphi), \quad i = 1, \dots, N \tag{8}$$

which is a square $N \times N$ linear system for the ξ_j's. Introducing the notation

$$A_{ij} = (\nabla\varphi_j, \nabla\varphi_i) \tag{9}$$
$$R_{ij} = (k\varphi_j, \varphi_i)_{\partial\Omega} \tag{10}$$
$$b_i = (f, \varphi_i) \tag{11}$$
$$r_i = (kg, \varphi_i)_{\partial\Omega} \tag{12}$$

we can write (8) in matrix form as

$$(A+R)\xi = b + r \tag{13}$$

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. In the lecture notes we have shown how 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 limited support the integrals of A_{ij} 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 6. Draw one of the hat functions with `pdesurf` or `pdemesh`. Use that the node values of φ_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

$$\varphi_i = \frac{1}{2|K|}(a_i + b_i x + c_i y), \quad i = 1, 2, 3 \tag{14}$$

where $|K|$ is the area of K, and where

$$a_1 = x_2 y_3 - x_3 y_2, \quad b_1 = y_2 - y_3, \quad c_1 = x_3 - x_2, \tag{15}$$

$$a_2 = x_3 y_1 - x_1 y_3, \quad b_2 = y_3 - y_1, \quad c_2 = x_1 - x_3, \tag{16}$$

$$a_3 = x_1 y_2 - x_2 y_1, \quad b_3 = y_1 - y_2, \quad c_3 = x_2 - x_1. \tag{17}$$

These expressions are derived by requiring $\varphi_i(N_j) = \delta_{ij}$, $i, j = 1, 2, 3$, where δ_{ij} is 1 if $i = j$ and 0 otherwise. Note that the gradient of φ_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 3×3 element stiffness matrix A^K.

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

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 3×1 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 integrates the interpolant. Recall that the interpolant is given by

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

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

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

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

$$b^K_i = (f, \varphi_i)_K \approx \sum_{j=1}^{3} f_j (\varphi_j, \varphi_i)_K = \frac{|K|}{12} \sum_{j=1}^{3} (1 + \delta_{ij}) f_j, \quad i = 1,2,3 \tag{21}$$

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

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

and

$$r^E_i = (kg, \varphi_i)_E = kg\frac{|E|}{2}, \quad i = 1,2 \tag{23}$$

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 Ω. 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

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

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

```
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)
  nodes = e(1:2,E); % node numbers of boundary edge E
  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 7. ☆ Derive explicit expressions for the hat functions φ_i, $i = 1, 2, 3$, on the reference triangle K with vertices at origo, $(1, 0)$, and $(0, 1)$.

Problem 8. ☆ Calculate the element stiffness matrix A^K and the element load vector b^K on the reference triangle assuming $f = 1$, $f = x$, and $f = 3 + 2x$.

Problem 9. ✔

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(abs(A-A')))`. 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 10. ☆ Consider the problem

$$\begin{aligned} -\Delta u &= 0, \quad \text{in } \Omega \\ u &= 0, \quad \text{on } \Gamma_D \\ n \cdot \nabla u &= g, \quad \text{on } \Gamma_N \end{aligned}$$

where g is a given function and Γ_D and Γ_N are two segments of the boundary associated with the Dirichlet and Neumann boundary conditions, and such that $\Gamma_D \cup \Gamma_N = \partial\Omega$ and $\Gamma_D \cap \Gamma_N = \emptyset$.

a) Make a variational formulation of this problem. *Hint:* The relevant space is $V = \{v : \|\nabla v\| < \infty,\ v|_{\Gamma_D} = 0\}$ with norm $\|v\|_V = \|\nabla v\|$.

b) Verify that the requirements for the Lax-Milgram lemma are fulfilled. *Hint:* You might find the Poincaré inequality $\|v\| \leq C\|\nabla v\|$ and the trace inequality $\|v\|_{\partial\Omega} \leq C(\|v\|^2 + \|\nabla v\|^2)^{\frac{1}{2}}$ useful.

c) Formulate a finite element approximation to the variational equation.

d) State and prove the Galerkin orthogonality property.

e) Derive the best approximation result $\|u - u_h\|_V \leq \|u - v\|_V$ for all $v \in V_h$.

f) Use interpolation theory to deduce the a priori estimate $\|u - u_h\|_V \leq Ch$.

Problem 11. ✔ Consider the model problem

$$\begin{aligned} -\Delta u &= 2x(1-x) + 2y(1-y), && \text{in } \Omega = [0,1]^2 \\ u &= 0, && \text{on } \partial\Omega \end{aligned}$$

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

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

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 $\|u_h\|_E^2 = (\nabla u_h, \nabla u_h)$ via the matrix-vector multiplication $\xi^T A \xi$ or the dot product $b^T \xi$. Compare the results with b).

Problem 12. ✔ 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?

Problem 13. ✔ The electric field $E \in \boldsymbol{R}^2$ due to a distribution of charge ρ is given by $\nabla \cdot E = \rho$. The work required to assemble the charge is given by a scalar potential function ϕ related to the electrical field by $E = -\nabla\phi$. Combining these results yields $-\Delta\phi = \rho$. Compute the electrical field within a cross-section of a long circular conducting wire with radius 1. Assume that the walls of the wire are perfectly conducting so that $\phi = 0$ on the boundary. Test with $\rho = 1$ and $\rho = \exp(-c((x-0.5)^2 + y^2)) - \exp(-c((x+0.5)^2 + y^2))$, with $c = 25$ say, which corresponds to a uniform distributed charge and two point charges of opposite sign, respectively. Once you have obtained ϕ use `pdegrad` to compute $E = -\nabla\phi$, and then visualize E with `pdeplot`.