Quadratic elements for pdetool

documentation by Janne Viljas

May 11, 1999

0.0.1 A Non-mathematician's Apology This document is meant primarily for displaying some of the relevant formulas in a neat form. No mathematical rigor is sought for in the presentation. Upon the unlikely occurrence of some, it should be considered accidental. No exhaustive, or even near-to-complete coverage of all the subtleties of my beautiful creation is intended. –JKV

---

1 Equation and boundary conditions

We consider here elliptic equations of the rather general form

$$-\nabla \cdot (c\nabla u) + au = f \quad \text{in } \Omega \in \mathbb{R}^2, \tag{1}$$

where Ω is some two-dimendional region, possibly multiplty connected, and c, a and f are functions $c, a, f : \ \Omega \longrightarrow \mathbb{R}$. The function c could also be 2×2 matrix valued, as in pdetool. Here it will be considered as such, but the implementation can only handle scalar c, at least for now.

The boundary $\partial\Omega$ of Ω is split essentially into to two parts, Γ_1 and Γ_2 ($\Gamma_1 \cup \Gamma_2 = \partial\Omega$) such that on Γ_1 we impose Dirichlet boundary conditions and on Γ_2 generalized Neumann boundary conditions:

$$\begin{aligned} u &= r \quad \text{on } \Gamma_1 \\ \vec{n} \cdot (c\nabla u) + qu &= g \quad \text{on } \Gamma_2. \end{aligned} \tag{2}$$

Here r, q and g are functions $r : \ \Gamma_1 \longrightarrow \mathbb{R}$ and $q, g : \ \Gamma_2 \longrightarrow \mathbb{R}$.

2 FEM

We now apply the ideas of the finite element method for the variational form of this problem in the usual way. (For simplicity, we will assume that

$\partial\Omega$ was polygonal — otherwise a polygonal approximation will have to be assumed from here on.) To be a bit more specific, we use a trial function space V_h consisting of piecewise continuous functions which are polynomials on the elements (here triangles) and equal to g on the nodes on Γ_1. Now let N_p be the set of indices of all lattice nodes P_i and N_{Γ_n} the set of indices corresponding to points on the boundary Γ_n, $n = 1, 2$. Then we seek the FEM soloution in V_h in the form

$$u_h = \sum_{j \in N_p \setminus N_{\Gamma_1}} U_j \phi_j + \sum_{j \in N_{\Gamma_1}} V_j \phi_j, \tag{3}$$

where the ϕ_j:s are the global basis functions such that $\phi_j(P_i) = \delta_{ij}$. The U_j:s are the unknowns to be determined and the V_j:s are given by the boundary condition.

The test function space W_h is like the trial space, but now with the requirement that the functions be equal to 0 on Γ_1. We thus "test" only with global basis functions ϕ_i where $i \in N_p \setminus N_{\Gamma_1}$ and end up with the set of equations

$$\begin{aligned} &\sum_{j \in N_p \setminus N_{\Gamma_1}} (K_{i,j} + M_{i,j}) U_j + \sum_{j \in N_{\Gamma_2}} Q_{i,j} U_j \\ &\qquad = F_i + G_i - \sum_{j \in N_{\Gamma_1}} (K_{i,j} + M_{i,j} + Q_{i,j}) V_j, \qquad i \in N_p \setminus N_{\Gamma_1}, \end{aligned} \tag{4}$$

where we have denoted

$$\begin{aligned} K_{i,j} &= \int_\Omega (c \nabla \phi_j) \cdot \nabla \phi_i \; dxdy \\ M_{i,j} &= \int_\Omega a \phi_i \phi_j \; dxdy \\ Q_{i,j} &= \int_{\Gamma_2} q \phi_i \phi_j \; ds \\ F_i &= \int_\Omega f \phi_i \; dxdy \\ G_i &= \int_{\Gamma_2} g \phi_i \; ds. \end{aligned} \tag{5}$$

In practise, when we code the program, we do not have to worry about the Dirichlet boundary conditions until at the very end: To simplify notation and help thinking, we can consider the V_j:s unknowns as well and absorb the last term on the right of (4) to the terms on the left. Then we can write the system easily in matrix form

$$(K + M + Q)U = F + G. \tag{6}$$

The matrices K and M are called the stiffness and mass matrices, respectively, and F the load vector. (More simply we could call the whole sum $K + M + Q$ the stiffness matrix and $F + G$ the load vector.) U is the vector for the values of the solution at all node points. Only *after* K, M, Q, F and G have been obtained in this way, we *separately eliminate* the equations corresponding to the Dirichlet boundary points from the system. This is done, for example, by replacing the equations corresponding to Dirichlet nodes by assignments given by the boundary condition and possibly by "moving the V_j-terms" to the right hand side. After this the system is solved.

3 Local K, M, Q, F and G, Quadratic elements

I assume here that the quadratic element is already familiar to the reader. I'll say something about it, but won't draw any images :-) I will also assume known what is meant by the concepts of shape functions, local stiffness and mass matrices etc.

The quadratic element triangle has six degrees of freedom, and we number the positions in the local element with which they are associated so that the corners are given numbers from 1 to 3 and the middle points of sides opposite to these from 4 to 6, in the same order (this is the *local numbering*). In practise, the assembly of the global K, M, Q, F and G is done by summing the contributions from the local K, M, Q, F and G of each element. To get these local matrices and vectors we only have to calculate the expressions (4) for the local shape functions, with numbering as mentioned above.

Since functions c, a f, q and g are arbitrary, we cannot calculate the integrals exactly. For quadratic elements the error of FEM is $O(h^3)$. Therefore, the error produced by the quadratures that will be used, should be no more – nor much less – than $O(h^3)$ either. To this end, we use the midpoint rule for area integrations

$$\int_T h(x, y)\ dxdy \approx \frac{\text{area}(T)}{3} \sum_{i=4}^{6} h(P_i) \tag{7}$$

and the Simpson rule for line integrals

$$\int_I h(s)\ ds \approx \frac{\text{length}(I)}{6} (h(P_a) + 4h(P_c) + h(P_b)), \tag{8}$$

where a & b and c are the indices of the end and middle points of the curve I. The midpoint rule is accurate for second order polynomials (two variables) and is thus exactly what we need. The Simpson rule is accurate even for third order polynomials (one variable), and is thus superoptimal, but seems to be the most reasonable choise.

The good thing about these choises is that the we only need the integrand values at points where the local shape functions are centered. This renders almost everything nice and diagonal. The results for the local K, M, Q, F and G, which are 6×6 matrices or 6×1 vectors, are as follows. The stiffness matrix takes the form

$$K_{i,j} \approx \frac{\text{area}(T)}{3} \sum_{k=4}^{6} (\nabla\phi_i(P_k) \cdot (c(P_k)\nabla\phi_j(P_k))), \tag{9}$$

where the gradients at the points P_k are calculated in the accompanying *Maple* worksheet. The mass matrix looks like

$$M_{i,j} \approx \frac{\text{area}(T)}{3} a(P_i)\delta_{ij} \quad \text{for } i,j = 4,5,6 \text{ and zero otherwise} \tag{10}$$

the load vector becomes

$$F_i \approx \frac{\text{area}(T)}{3} f(P_i) \quad \text{for } i = 4,5,6 \text{ and zero otherwise} \tag{11}$$

and finally the boundary contributions

$$Q_{i,j} \approx \begin{cases} (1/6)\|P_a - P_b\| q(P_i)\delta_{ij} & \text{for } i,j = a,b \\ (2/3)\|P_a - P_b\| q(P_i)\delta ij & \text{for } i,j = c \end{cases} \tag{12}$$

and

$$G_{i,j} \approx \begin{cases} (1/6)\|P_a - P_b\| g(P_i) & \text{for } i = a,b \\ (2/3)\|P_a - P_b\| g(P_i) & \text{for } i = c \end{cases}, \tag{13}$$

where $\|\cdot\|$ is the normal Euclidean norm.

4 About the implementation

I will not put too much time in trying to describe the implementation itself. The M-files involved are short and should be self-explanatory.

If it wasn't clear from the beginning, I should say now that the purpose of this project was to expand the capabilities of *Matlab*'s PDE toolbox to the realm of quadratic elements. The toolbox itself is still needed in order to take advantage from these additions. The expansion consists of essentially four functions. Their descriptions are given below:

- qmesh, This function is written by Heikki Laitinen. It is so good as-is that there was no point in rewriting it. It transforms the mesh data structures p, e and t created by pdetool to a form suitable for quadratic elements.

- qassema, forms the global K, M and F without worrying about Dirichlet boundary conditions.

- qassemb, forms the global Q and G plus a vector R, which contains the values of the solution at the Dirichlet boundary nodes.

- qassempde, This is the main function, which uses the two previous functions and then eliminates the equations corresponding to Dirichlet boundary conditions from the linear system. It gives out the final coefficient matrix and the rhs vector — a system which can be solved to get the final solution of the problem.

So, to use the quadratic elements one should always first define the equations, boundary conditions and the geometry with the pdetool GUI, for example, and then run qmesh to transform the mesh. Then run qassempde and produce the solution. The function I have made to plot the solution is quite archaic, but it works; it is called qplotsol.

I also completed the linear element functions lassema, lassemb and lassempde, much of whose structure was already given on the course WWW-page. I shall include them with the "quadratic functions", but will not comment on them further. The quadratic functions were constructed to follow same principles as they do.