Programming project in TMA4220, part 2A: *Adaptive mesh refinement for Poisson's equation*

October 2, 2015

1 Introduction

In this project you will program an adaptive mesh refinement algorithm for Poisson's equation

$$\begin{cases} -\Delta u = f & \text{in } \Omega \\ u = g & \text{on } \partial\Omega \end{cases} \tag{1}$$

for given functions $f \in L^2(\Omega)$ and $g \in L^2(\Omega)$. Recall from Exercise Set 2 that if g is "nice enough" then (1) can be transformed to the homogeneous Dirichlet problem

$$\begin{cases} -\Delta u = f & \text{in } \Omega \\ u = 0 & \text{on } \partial\Omega. \end{cases} \tag{2}$$

If we approximate (2) by piecewise linear finite elements, i.e. $V_h = X_h^1$, then we have shown the a priori error estimate

$$\|u - u_h\|_{H^1(\Omega)} \leqslant C \left(\sum_{K \in \mathcal{T}_h} E_K^2 \right)^{1/2}, \qquad E_K := h_K |u|_{H^2(K)} \tag{3}$$

(both in one and two space dimensions) whenever $u \in H^2(\Omega)$. The residual-based a posteriori error estimate says that

$$\|u - u_h\|_{H^1(\Omega)} \leqslant C \left(\sum_{K \in \mathcal{T}_h} \sigma_K^2 \right)^{1/2}, \qquad \sigma_K := C_1 h_K \|f + \Delta u_h\|_{L^2(K)} + C_2 \sqrt{h_k} \left\| \left[\frac{\partial u_h}{\partial n} \right] \right\|_{L^2(\partial K)}, \tag{4}$$

where C_1 and C_2 depend only on the minimum angle appearing in any $K \in \mathcal{T}_h$, and $C = \sqrt{n}$, where n is the maximal number of neighboring triangles any triangle $K \in \mathcal{T}_h$ can have (which is small if the mesh is not too irregular).

2 Project description

NOTE: This is a *suggestion* for a project description. You will be graded based on what you do and how you do it, NOT on how many of the points below you have completed. However, at a *minimum* you need to present a working a priori or a posteriori error adaptive code in either 1D, 2D or 3D, using either a refinement or a remeshing strategy. You also need to compare your results with "quasi-optimal" meshes in some simple examples (see below).

1. To start with, consider the one-dimensional version of (1),

$$\begin{cases} -\frac{d^2}{dx^2} u = f & \text{in } \Omega := (a, b) \\ u(a) = g_a,\ u(b) = g_b \end{cases} \tag{5}$$

for a given function $f \in L^2(\Omega)$ and numbers $g_a, g_b \in \mathbb{R}$.

(a) Write a program that approximates (5) using piecewise linear finite elements. Implement either an a priori or a posteriori error indicator. Write a program that takes as input an error tolerance $\varepsilon > 0$, and by successive refinement of an initially uniform mesh, finds a mesh $\mathcal{T}_h$ and a corresponding finite element solution u_h with error at most ε:

$$\|u - u_h\|_{H^1(\Omega)} < \varepsilon.$$

Note: *If you want to do a posteriori error adaptation then you need to derive a one-dimensional version of* (4)*, which should be quite straightforward. The term* $\left\| \left[\frac{\partial u_h}{\partial n} \right] \right\|_{L^2(\partial K)}$ *will in one dimension look like* $\left| \left[\frac{du_h}{dx} \right]_{N_i} \right|$*, where* N_i *is a node in the mesh with neighboring intervals* K_i*,* K_{i+1}*, and* $\left[\frac{du_h}{dx} \right]_{N_i} = \frac{d(u_h|_{K_{i+1}})}{dx}(N_i) - \frac{d(u_h|_{K_i})}{dx}(N_i)$*, the jump in the derivative at* N_i*.*

(b) Test your program, for instance on the following data:

- $\Omega = (0, 2\pi)$, $f(x) = \sin(x)$, $g_a = g_b = 0$.
- $\Omega = (0, 2)$, $f(x) = -60x^2$, $g_a = 0$, $g_b = 80$.

Note that the exact solution is easily computed in these simple cases.

(c) If $E_K \equiv$ const. $\forall\, K$ then the error produced in individual elements will be roughly equal for all elements. In simple cases such as those above, it is possible to find u explicitly, and hence compute $|u|_{H^2(K)}$ on an arbitrary element K. For one or two of the above test cases, find a $\mathcal{T}_h^*$ such that $E_{K^*} \equiv$ const., and compare the mesh $\mathcal{T}_h$ computed by your adaptive mesh refinement algorithm with this "quasi-optimal" mesh $\mathcal{T}_h^*$. Which mesh is more efficient? You can estimate efficiency e.g. as $M\|u - u_h\|_{H^1(\Omega)}$ (where M is the number of elements K in the mesh) or $\frac{\|u-u_h\|_{H^1(\Omega)}}{h}$.

2. Consider now the two-dimensional problem (1) with $\Omega \subset \mathbb{R}^2$. Write an adaptive mesh refinement algorithm for this problem. Test your code on an L-shaped domain, a "half-eaten cake" domain $\Omega = \{(r, \theta) : 0 < r < R,\ 0 < \theta < \omega\}$ for some $\omega > \pi$, or some other non-convex domain. Find an exact solution of (1) on this domain, compute a "quasi-optimal" mesh $\mathcal{T}_h^*$ where $E_{K^*} \approx$ const., and compare this with what your program computes.

Note: *You can read more about this in Brenner and Scott, Section 5.5. The book is freely available from SpringerLink:* `http://link.springer.com/book/10.1007/978-0-387-75934-0`*.*

3 Further ideas

- Implement regularization techniques for your mesh to ensure that the refinement process does not create a very irregular mesh. (These include diagonal swapping of edges and node displacement; see Chapter 6.5 in Quarteroni.) Be careful that the regularization does not destroy the error adaptivity of the mesh.
- Do *remeshing* instead of *refinement*. To this end, you need to utilize the information gained from your candidate solution to define a spacing function $H(x)$, and implement a triangulation procedure that uses the spacing function to create a new mesh.