Part 5 - A posteriori error estimation

5.1. Setting and model problem

Fixed elliptic setting for Part 5

Let

- $\Omega \subset \mathbb{R}^d$ be a bounded convex domain
- source term: $f \in L^2(\Omega)$,
- elliptic and Lipschitz-continuous diffusion coefficient: $\mathsf{k} \in C^{0,1}(\Omega, \mathbb{R}^{d\times d})$,

$$\mathsf{k}(x)\boldsymbol{\xi} \cdot \boldsymbol{\xi} \geq \mathsf{k}_0 |\boldsymbol{\xi}|^2 \qquad \text{for all } \boldsymbol{\xi} \in \mathbb{R}^d$$

Elliptic problem: find $u \in H_0^1(\Omega) \cap H^2(\Omega)$ such that

$$\int_\Omega k \nabla u \cdot \nabla v = \int_\Omega f\, v \qquad \text{for all } v \in H_0^1(\Omega).$$

Galerkin approximation: find $u_h \in V_{h,0}$ such that

$$\int_\Omega k \nabla u_h \cdot \nabla v_h = \int_\Omega f\, v_h \qquad \text{for all } v_h \in V_{h,0}.$$

Goal of a posteriori error estimation

Elliptic problem: find $u \in H_0^1(\Omega) \cap H^2(\Omega)$ such that

$$\int_\Omega k \nabla u \cdot \nabla v = \int_\Omega f\, v \qquad \text{for all } v \in H_0^1(\Omega).$$

Galerkin approximation: find $u_h \in V_{h,0}$ such that

$$\int_\Omega k \nabla u_h \cdot \nabla v_h = \int_\Omega f\, v_h \qquad \text{for all } v_h \in V_{h,0}.$$

We know:

$$\|u - u_h\|_{H^1(\Omega)} \leq C h \|f\|_{L^2(\Omega)}.$$

But, do we also know the size of the error a posteriori and can we also say on which elements (simplexes) is the error largest?

Goal of a posteriori error estimation

We know the a priori error estimate

$$\|u - u_h\|_{H^1(\Omega)} \leq Ch\|f\|_{L^2(\Omega)}.$$

Goal: a posteriori error estimate of the form

$$\|u - u_h\|_{H^1(\Omega)} \leq \sum_{K \in \mathcal{T}_h} R_K(u_h),$$

where $R_K(u_h)$ are computable error indicators, that measure the error on each simplex K.

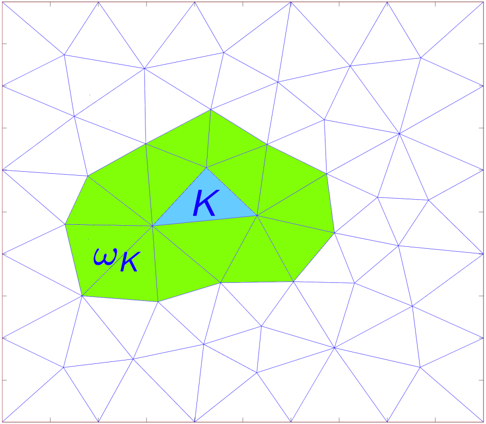
Part 5 - A posteriori error estimation

5.2. Error bounds

Element patch

For $K \in \mathcal{T}_h$, the element patch is

$$\omega_K := \bigcup \{ T \in \mathcal{T}_h | \, T \cap K \neq \emptyset \}.$$



Local quasi-interpolation estimates

The Clément quasi-interpolation operator

$$I_h : H_0^1(\Omega) \to V_{h,0}$$

is given by

$$I_h(v) := \sum_{z \in \mathcal{N}_{h,0}} \frac{(v, \phi_z)_{L^2(\Omega)}}{(1, \phi_z)_{L^2(\Omega)}} \phi_z .$$

For all $K \in \mathcal{T}_h$ and all $v \in H_0^1(\Omega)$ we have the local estimates

$$\|I_h(v) - v\|_{L^2(K)} \leq C h_K \|v\|_{H^1(\omega_K)}$$

and the trace estimate

$$\|I_h(v) - v\|_{L^2(\partial K)} \leq C h_K^{1/2} \|v\|_{H^1(\omega_K)} .$$

where $\omega_K := \bigcup \{T \in \mathcal{T}_h | \, T \cap K \neq \emptyset\}$ is the element patch and C is a constant independent of h_K and v.

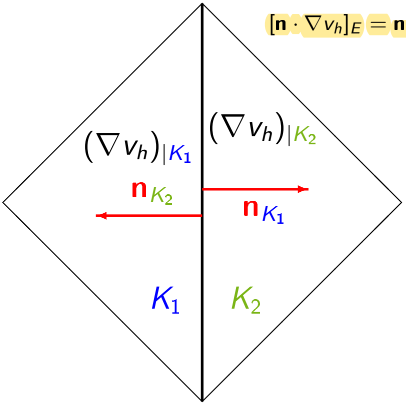
Jump in the normal flux of a function

The normal jump of $v_h \in V_{h,0}$ (piecewise linear) is:

$$[\mathbf{n} \cdot \nabla v_h]_{\partial K_1 \cap \partial K_2} := \mathbf{n}_{K_1} \cdot (\nabla v_h)_{|K_1} + \mathbf{n}_{|K_2} \cdot (\nabla v_h)_{|K_2}.$$

Observe that $\mathbf{n}_{K_1} = -\mathbf{n}_{K_2}$ on $\partial K_1 \cap \partial K_2$.

$$[\mathbf{n} \cdot \nabla v_h]_E = \mathbf{n}_{K_1} \cdot \left((\nabla v_h)_{|K_1} - (\nabla v_h)_{|K_2}\right)$$



A posteriori error estimate

With our assumptions for Part 5 and our usual assumptions on $V_{h,0}$, it holds

$$\|u - u_h\|^2_{H^1(\Omega)} \leq C \sum_{K \in \mathcal{T}_h} R_K(u_h)^2,$$

where C is a (computable) constant that is independent of h_K and

$$R_K(u_h)^2 := h_K^2 \|f + \nabla \cdot (\mathbf{k} \nabla u_h)\|^2_{L^2(K)} + \frac{1}{4} h_K \|[\mathbf{n} \cdot \mathbf{k} \nabla u_h]\|^2_{L^2(\partial K \setminus \partial \Omega)}.$$

The proof is given on the next slides.

A posteriori error estimate - Proof step 1

Let $e_h := u - u_h$. We get:

$$\alpha \|e_h\|^2_{H^1(\Omega)} \le \int_\Omega \mathbf{k}\nabla e_h \cdot \nabla e_h \overset{\text{Gal. Orth.}}{=} \int_\Omega \mathbf{k}\nabla e_h \cdot \nabla(e_h - I_h(e_h))$$

$$= \sum_{K\in\mathcal{T}_h} \int_K \mathbf{k}\nabla e_h \cdot \nabla(e_h - I_h(e_h))$$

$$\overset{\text{Green}}{=} \sum_{K\in\mathcal{T}_h} \int_K -\nabla \cdot \mathbf{k}\nabla e_h \, (e_h - I_h(e_h))$$

$$+ \sum_{K\in\mathcal{T}_h} \int_{\partial K \setminus \partial\Omega} (\mathbf{n}\cdot\mathbf{k}\nabla e_h) \, (e_h - I_h(e_h))$$

$$= \sum_{K\in\mathcal{T}_h} \int_K (f + \nabla \cdot \mathbf{k}\nabla u_h) \, (e_h - I_h(e_h))$$

$$+ \sum_{K\in\mathcal{T}_h} \int_{\partial K \setminus \partial\Omega} (\mathbf{n}\cdot\mathbf{k}\nabla e_h) \, (e_h - I_h(e_h)).$$

A posteriori error estimate - Proof step 2

For $e_h := u - u_h$ we have seen

$$\alpha \|e_h\|^2_{H^1(\Omega)} \leq \sum_{K \in \mathcal{T}_h} \int_K (f + \nabla \cdot \mathrm{k} \nabla u_h) \, (e_h - I_h(e_h)) + \sum_{K \in \mathcal{T}_h} \int_{\partial K \setminus \partial \Omega} (\mathbf{n} \cdot \mathrm{k} \nabla e_h) \, (e_h - I_h(e_h)).$$

Here we note

$$\sum_{K \in \mathcal{T}_h} \int_{\partial K \setminus \partial \Omega} (\mathbf{n} \cdot \mathrm{k} \nabla u)(e_h - I_h(e_h)) = 0$$

since $[\mathbf{n} \cdot \mathrm{k} \nabla u] = 0$ because $u \in H^2(\Omega)$ which means that the normal trace is continuous and the jumps must be zero.

A posteriori error estimate - Proof step 3

We conclude

$$\alpha\|e_h\|^2_{H^1(\Omega)} \leq \underbrace{\sum_{K\in\mathcal{T}_h}\int_K (f+\nabla\cdot\mathbf{k}\nabla u_h)\,(e_h - I_h(e_h))}_{=:\mathrm{I}}$$
$$-\underbrace{\sum_{K\in\mathcal{T}_h}\int_{\partial K\setminus\partial\Omega}(\mathbf{n}\cdot\mathbf{k}\nabla u_h)\,(e_h - I_h(e_h))}_{=:\mathrm{II}}.$$

By Γ_h we denote the set of interior edges ($d=2$) / interior faces ($d=3$), i.e.

$$\Gamma_h := \{E = T\cap K|\, T, K\in\mathcal{T}_h,\;\; T\neq K$$
$$\text{and } E \text{ is set of Hausdorff dimension } d-1.\}$$

A posteriori error estimate - Proof step 4

We start with estimating II:

$$
\begin{aligned}
\mathrm{II} &= -\sum_{K\in\mathcal{T}_h}\int_{\partial K\setminus\partial\Omega} (\mathbf{n}\cdot\mathbf{k}\nabla u_h)\,(e_h - I_h(e_h)) \\
&= \sum_{E\in\Gamma_h}\int_E [\mathbf{n}\cdot\mathbf{k}\nabla u_h]\,(e_h - I_h(e_h)) \\
&= \frac{1}{2}\sum_{K\in\mathcal{T}_h}\int_{\partial K\setminus\partial\Omega} [\mathbf{n}\cdot\mathbf{k}\nabla u_h]\,(e_h - I_h(e_h)) \\
&\overset{\text{Cau.Schw.}}{\leq} \frac{1}{2}\sum_{K\in\mathcal{T}_h} \|[\mathbf{n}\cdot\mathbf{k}\nabla u_h]\|_{L^2(\partial K\setminus\partial\Omega)}\|e_h - I_h(e_h)\|_{L^2(\partial K\setminus\partial\Omega)} \\
&\overset{\text{Interpol.Est.}}{\leq} C\sum_{K\in\mathcal{T}_h}\frac{1}{2}\|[\mathbf{n}\cdot\mathbf{k}\nabla u_h]\|_{L^2(\partial K\setminus\partial\Omega)}h_K^{1/2}\|\nabla e_h\|_{L^2(\omega_K)} \\
&\overset{\text{Cau.Schw.}}{\leq} C\left(\sum_{K\in\mathcal{T}_h}\frac{h_K}{4}\|[\mathbf{n}\cdot\mathbf{k}\nabla u_h]\|^2_{L^2(\partial K\setminus\partial\Omega)}\right)^{1/2}\|e_h\|_{H^1(\Omega)}.
\end{aligned}
$$

A posteriori error estimate - Proof step 5

It only remains to estimate I:

$$
\begin{aligned}
\mathrm{I} \quad &= \quad \sum_{K\in\mathcal{T}_h}\int_K (f+\nabla\cdot\mathbf{k}\nabla u_h)\,(e_h - I_h(e_h)) \\
&\overset{\text{Interpol.Est.}}{\leq} \quad C\sum_{K\in\mathcal{T}_h} h_K \|f+\nabla\cdot\mathbf{k}\nabla u_h\|_{L^2(K)}\|\nabla e_h\|_{L^2(\omega_K)} \\
&\overset{\text{Cau.Schw.}}{\leq} \quad C\left(\sum_{K\in\mathcal{T}_h} h_K^2\|f+\nabla\cdot\mathbf{k}\nabla u_h\|^2_{L^2(K)}\right)^{\frac{1}{2}}\|e_h\|_{H^1(\Omega)}.
\end{aligned}
$$

Combining the estimates for I and II yields

$$
\|e_h\|^2_{H^1(\Omega)} \leq C\left(\sum_{K\in\mathcal{T}_h} h_K^2\|f+\nabla\cdot(\mathbf{k}\nabla u_h)\|^2_{L^2(K)} + \frac{1}{4}h_K\|[\mathbf{n}\cdot\mathbf{k}\nabla u_h]\|^2_{L^2(\partial K\setminus\partial\Omega)}\right)^{1/2}\|e_h\|_{H^1(\Omega)}.
$$

Reminder: A posteriori error estimate

With our usual assumptions, it holds

$$\|u - u_h\|_{H^1(\Omega)} \leq C \left(\sum_{K \in \mathcal{T}_h} h_K^2 \|f + \nabla \cdot (\mathbf{k} \nabla u_h)\|_{L^2(K)}^2 + \frac{1}{4} h_K \|[\mathbf{n} \cdot \mathbf{k} \nabla u_h]\|_{L^2(\partial K \setminus \partial\Omega)}^2 \right)^{1/2},$$

where C is a (computable) constant.

- The term $\|f + \nabla \cdot (\mathbf{k} \nabla u_h)\|_{L^2(K)}^2$ describes the residual on each element
- and $\|[\mathbf{n} \cdot \mathbf{k} \nabla u_h]\|_{L^2(\partial K \setminus \partial\Omega)}^2$ the size of jump-terms.

Part 5 - A posteriori error estimation

5.3. Adaptive mesh refinements

Reminder: A posteriori error estimate

We have seen:

$$\|u - u_h\|^2_{H^1(\Omega)} \leq C \left(\sum_{K \in \mathcal{T}_h} R_K(u_h)^2 \right)^{1/2},$$

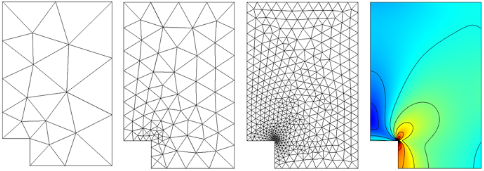
where

$$R_K(u_h)^2 := h_K^2 \| f + \nabla \cdot (\mathbf{k} \nabla u_h) \|^2_{L^2(K)} + \frac{1}{4} h_K \| [\mathbf{n} \cdot \mathbf{k} \nabla u_h] \|^2_{L^2(\partial K \setminus \partial \Omega)}.$$

If the error is large, we do not want to refine the whole mesh $\mathcal{T}_h$, because that can be expensive.

Instead, we only want to refine the mesh locally where $R_K(u_h)$ is large.

Adaptive mesh refinements



$$\|u - u_h\|^2_{H^1(\Omega)} \leq C \left(\sum_{K \in \mathcal{T}_h} R_K(u_h)^2 \right)^{1/2}.$$

We want to refine the mesh where $R_K(u_h)$ is large.

Main difficulties are:

- no hanging nodes,
- shape regular triangulations (e.g. avoid very small angles in the triangles).

Adaptive mesh refinements

Main difficulties:

- no hanging nodes,
- shape regular triangulations (e.g. avoid very small angles in the triangles).

There are several algorithms that have these properties, including

- ∗ Rivara refinement (largest edge),
- ∗ regular refinement.

A combination is used in Matlab.

Adaptive mesh refinements

In Matlab a regular mesh refinement can be obtained as follows. Here, `g`, `p`, `e` and `t` denote the usual geometry, point, edge and triangulation matrices.

First we define a column vector that contains the indices of the triangles (from `t`) that shall be refined. For example:

```
t_refine = [ 1 ; 5 ; 6 ; ... ];
```

Now the mesh can be refined with:

```
[p,e,t] = refinemesh(g,p,e,t,t_refine,'regular');
```

Matlab example code for mesh refinement

```
% create 2D mesh with a corner
% -----------------------------------------
% geometry matrix
 g = [ 2 2 2 2 2 2 2 2 ;
       0 1 2 2 1 1 0 0 ; 1 2 2 1 1 0 0 0 ;
       0 0 0 1 1 2 2 1 ; 0 0 1 1 2 2 1 0 ;
       0 0 0 0 0 0 0 0 ; 1 1 1 1 1 1 1 1 ];

% initialize point, edge and triangle matrix:
[p,e,t] = initmesh( g , 'hmax' , 1.0);
% -----------------------------------------

% index of triangles to refine
triangles_to_refine = [ 5 ; 6 ];
% refine the mesh so that the triangles 5 and 6 are refined:
[p,e,t] = refinemesh(g,p,e,t,triangles_to_refine,'regular');

pdemesh(p,e,t); % plots the mesh
```

Adaptive mesh refinements

Messages:

- Matlab can refine the meshes automatically in a "good" way.
- All we have to do is find triangles with a large error and mark these triangles for a mesh refinement.
- How does an algorithm look like that finds and marks elements for refinement?

Adaptive mesh refinements

Algorithm:

1. Construct initial mesh $\mathcal{T}_h$.
2. Solve finite element problem for u_h.
3. Compute local indicators $R_K(u_h)^2$.
4. Compute maximum $m := \max_{K \in \mathcal{T}_h} R_K(u_h)^2$.
5. Mark elements with error over $\gamma \cdot m$, where $0 < \gamma < 1$ is a fixed parameter.
6. Refine elements and get new mesh $\mathcal{T}_h$.
7. Return to step 2) (stop if N becomes too large or when error $\sum_{K \in \mathcal{T}_h} R_K(u_h)^2$ is small enough)

Matlab also provides build-in methods for directly solving PDEs with adaptive finite elements. Example:

```
% we want to solve  " - Laplacian u = 1 "  on an L-shaped domain
% with zero Dirichlet boundary condition

% create geometry matrix for a 2D mesh with a corner:
g = [ 2 2 2 2 2 2 2 2 ;  0 1 2 2 1 1 0 0 ; 1 2 2 1 1 0 0 0 ;
       0 0 0 1 1 2 2 1 ; 0 0 1 1 2 2 1 0 ;
       0 0 0 0 0 0 0 0 ; 1 1 1 1 1 1 1 1 ];

% initialize point, edge and triangle matrix:
[p,e,t] = initmesh( g , 'hmax' , 1.0);

% solve "- div (k grad u) + c u = f" with
f = 1;
k = 1;
c = 0;  % lower order term
zeroDirichlet = allzerobc( g );  % zero Dirichlet conditions

% automatic adaptive mesh refinement including solving
[u,p,e,t] = adaptmesh( g , zeroDirichlet , k , c , f , 'maxt', 500 );
% 'maxt' 5000 = maximum number of new triangles = 500

pdesurf(p,t,u)  % plot solution
pdemesh(p,e,t);  % plots the mesh
```