University of Amsterdam

MSc Mathematics

Master Thesis

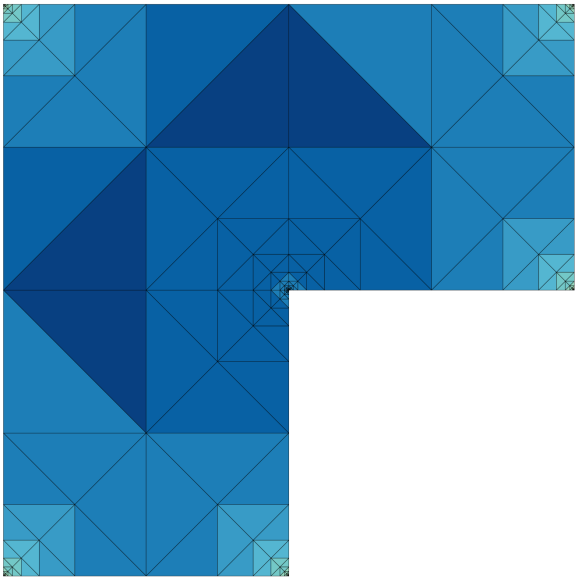
Two-dimensional *hp*-adaptive finite elements in theory and practice

---

Author:
Jan Westerdiep

Supervisor:
prof. dr. R.P. Stevenson

Examination date:
June 27th, 2017



Korteweg-de Vries Institute for
Mathematics

Abstract

Finite element methods find best approximations to the solution of an (elliptic) PDE by piecewise polynomials of some fixed degree w.r.t. a partition of the domain into, say, triangles. *Adaptive* finite elements produce a sequence of such approximations where each next approximation is constructed from the previous one by refining the partition there where the current error turns out to be large. This error is estimated locally by an *a posteriori error estimator*. Adaptive finite elements are the topic of an MSc. course taught at KdVI.

Adaptive *hp*-finite element methods go one step further, in the sense that not only the partition is adapted to the unknown solution, but also the polynomial degree on the individual triangles. They aim to achieve an *exponential* convergence rate in terms of the number of degrees of freedom, equal to the rate of a best sequence of *hp*-partitions for this (unknown) solution. Convergence theory of *hp*-finite elements is the center of current interest in the finite element community.

Besides the difficulties associated with any adaptive method handling locally refined partitions, an implementation requires dealing with polynomials of an (a priorily) unbounded degree whose bases moreover vary from triangle to triangle. The data structures involved in this algorithm requires one to develop specialized computational methods, which we will cover in depth.

Title: Two-dimensional *hp*-adaptive finite elements in theory and practice
Cover: A sample triangulation found by our implementation.
Deeper colors indicate higher local degree.

Author: Jan Westerdiep, jan.westerdiep@student.uva.nl, 10219242
Supervisor: prof. dr. R.P. Stevenson
Second Examiner: dr. J.H. Brandts

Examination date: June 27th, 2017

Korteweg-de Vries Institute for Mathematics
University of Amsterdam
Science Park 105-107, 1098 XG Amsterdam
`http://kdvi.uva.nl`

Contents

Introduction

The Finite Element Method (FEM) is a numerical technique for finding approximate solutions to (elliptic) partial differential equations, specifically boundary value problems. It uses a subdivision of a whole problem domain into simpler parts, called finite elements. Analogous to the idea that connecting many tiny straight lines can approximate a larger circle, FEM encompasses methods for connecting many simple element equations over many subdomains to approximate a more complex equation over a larger domain.

The classical finite element methods subdivide the problem domain into finite elements (say triangles), and use continuous elementwise polynomials (of fixed degree) to find a first approximation. In subsequent iterations, all elements are refined (e.g., for triangular elements, we can connect the edge midpoints to find four smaller triangles) and the process is repeated until the approximation is close enough to the exact solution, in that the norm of the difference of these two is satisfactorily small.

We can choose not to refine every finite element, but only the ones we predict will benefit most from refinement. This is called *h-adaptive* finite elements, as we adapt the method to the problem at hand by letting the diameter (denoted by h) of each element differ. The case of h-adaptivity has been studied thoroughly—we refer to the survey [37]—and an optimality proof was found in 2007 [43], showing that the method finds "the best possible solution" given a fixed number of refinements, or equivalently, degrees of freedom (DoFs).

We use elementwise contributions to solve a larger linear system that yields our desired solution. In classical and h-adaptive finite element methods, the number of DoFs per element (interpretable as the complexity of such a local contribution) is fixed. If we allow variation of the polynomial degree (denoted by p) per element in addition to h-adaptive refinement, we arrive at hp-adaptive finite elements. This creates more possibilities, as one can choose to either refine an element, increase its polynomial degree, or do nothing. The result is that better solutions can be expected given a fixed number of DoFs, but that the theory is also more difficult.

The case of hp-adaptivity started gaining momentum in the eighties with the works of Gui and Babuška [25, 26], but despite the interest, the field is much less developed than for the h-version. While (heuristics-based) hp-adaptive finite element methods have existed for quite some time, it was not until 2015 that Canuto *et al.* [15] proved the optimality of one such methods, at least in one and two dimensions. Their article builds upon the ideas of Binev [9], which we previously studied in [47].

The purpose of this work is to partly bridge the gap between theory and practice: Both theoretical insights and implementation-specific details will be considered. Currently, most hp-adaptive implementations used in the real world are built upon heuristics for the selection between h-refinement and p-enrichment, rather than a rigorous foundation. We will examine a slightly simplified version of the method by Canuto *et al.* in [15] to allow a focus on didactics rather than on bookkeeping.

In Chapter 1, we will study the classic h-adaptive case. In the h-AFEM algorithm, a sequence of triangulations is created, each a refinement of the previous. Under certain mild conditions, the error norms of the finite element solutions produced by h-AFEM decay *algebraically* in the number of DoFs:

$$\text{error norm} \ \sim \#\,\text{DoF}^{-s} \text{ for the best } s > 0.$$

The *hp*-adaptive *hp*-AFEM algorithm central to Chapter 2 creates a sequence of such finite element triangulations by alternating two routines—Reduce and NearBest. The first routine reduces the error norm of the approximate solution by constructing a refined triangulation with more degrees of freedom, and the second increases the efficiency of the current solution while sacrificing some accuracy. We can prove that, under mild conditions, the error norms of these solutions decay with exponential rate:

$$\text{error norm} \ \sim \exp\left(-\eta(\#\,\text{DoF})^{\tau}\right) \ \text{for some } \eta, \tau > 0.$$

Note the difference with the *h*-adaptive case: *hp*-adaptation is *much* better in terms of convergence speed.

This alternation between the two routines means that our *hp*-AFEM produces a *sawtooth graph* similar to Figure 0.0.1. A call to our Reduce routine brings the error down significantly but increases the total number of DoFs; a classic (*h*-adaptive) finite element method started from here would produce an error progression graph like the dotted line. The red line represents a call to NearBest and decreases the number of DoFs at the expense of (some) accuracy. Connecting all points corresponding to the solution after a call to NearBest yields the black dashed line, and shows the promised exponential convergence rate.

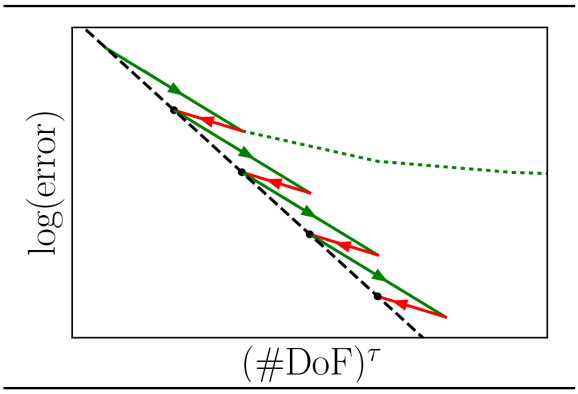


Figure 0.0.1.: Example of a sawtooth graph that our *hp*-AFEM algorithm could produce. The green lines correspond to a grid refinement though the Reduce routine; a dotted line shows typical (algebraic) error progression of an *h*-adaptive method. Red lines are calls to NearBest. The black dashed line shows the (exponential) convergence of solutions after each coarsening step.

After carefully studying the mathematical theory behind this *hp*-adaptive algorithm, we will take a more practical look on things. In Chapter 3, we will investigate the properties of a handful of bases for the degree-p polynomial space on each element. Of these, the *hierarchical basis*—a basis that allows for easy p-enrichment by writing the degree-$(p+1)$ basis as an adaptation of the basis on degree p—will be used to fully implement and experiment with *hp*-AFEM.

Chapter 4 will then focus on the *practical considerations* that our implementation hinges on. In it, we hope to cover every detail necessary for a successful implementation. We end the chapter with an overview of the C++ library that was written as part of this work.

In Chapter 5, we experiment with the implementation. We look at the behaviour of the algorithm in a few different scenarios, and conclude that it produces very high-quality triangulations which exhibit exponential decay. We end the thesis with an outlook into the future.

1. Theoretical foundation and h-adaptivity

This chapter will serve as a self-contained summary of the theory necessary to understand the rest of the thesis. We will start off with the standard *variational formulation* in §1.1, and provide the reader with a systematic way of constructing an approximate finite element solution to the variational problem in §1.2.

However, almost always, just finding a solution is not good enough. We want to know if this numerical solution is a *good* approximation to the true solution, and possibly find out how to make it *better*. We will explain why these concepts are both active fields of research by diving into error estimation (in §1.3 and §1.4) and grid refinement—or h-refinement—in §1.5.

We will finish our overview in §1.6 by touching on the classical h-AFEM algorithm—refining h adaptively on those elements where the error norm is indicated to be large—paving the way to our true objective: covering hp-AFEM.

1.1. Variational problems

Let us start out with the pure basics.

Definition 1.1.1. Define $\mathbb{N}_0 := \mathbb{N} \cup \{0\}$ to be the set of nonnegative integers. Then, for a multi-index $\boldsymbol{\alpha} \in \mathbb{N}_0^d$, define $|\boldsymbol{\alpha}| := \sum_k \alpha_k$. ◊

Definition 1.1.2. Given a real linear space V, a function $a : V \times V \to \mathbb{R}$ is *bilinear* iff both $v \mapsto a(v, w)$ and $w \mapsto a(v, w)$ are linear. It is *symmetric* when $a(v, w) = a(w, v)$ for all $v, w \in V$. ◊

Definition 1.1.3. A bilinear form $a(\cdot, \cdot)$ on a normed linear space $(V, \|\cdot\|)$ is *bounded* iff there is a $C \in \mathbb{R}$ with

$$|a(v, w)| \leq C\|v\|\|w\| \quad (v, w \in V)$$

and *coercive* when there is an $\alpha > 0$ with

$$a(v, v) \geq \alpha\|v\|^2 \quad (v \in V). \qquad \text{◊}$$

Proposition 1.1.4 ([13, (2.5.3)])**.** *Let $(V, \langle\cdot,\cdot\rangle)$ be a Hilbert space, and suppose $a(\cdot, \cdot)$ is a symmetric blinear form that is continuous and coercive on V. Then $(V, a(\cdot, \cdot))$ is a Hilbert space, and its induced norm $\|v\|_E := \sqrt{a(v, v)}$ is equivalent to the norm $\sqrt{\langle\cdot,\cdot\rangle}$. We call $\|\cdot\|_E$ the* energy norm*.*

Definition 1.1.5. Let $(V, \langle\cdot,\cdot\rangle)$ be a Hilbert space. If $a(\cdot, \cdot)$ is a bounded and coercive bilinear form on V, then the *variational problem* is stated as

$$\text{Given } F \in V', \text{ find } u \in V \text{ s.t. } a(u, v) = F(v) \quad (v \in V). \tag{1.1.6}$$

The *Ritz-Galerkin approximation problem* then is the following.

$$\begin{gathered}\text{Given a closed linear subspace } S \subset V,\\ \text{find } u_S \in S \text{ s.t. } a(u_S, v) = F(v) \quad (v \in S).\end{gathered} \tag{1.1.7}$$

When u and u_S solve these problems, a simple subtraction yields the *fundamental Galerkin orthogonality* relation between u and u_S:

$$a(u - u_S, v) = 0 \quad (v \in S). \qquad \Diamond$$

Three important questions arise:

1. Are there any interesting examples of variational problems? What is their relation to solving differential equations?
2. When are these variational problems solvable? Are the solutions unique?
3. What can we say about the approximation error $u - u_S$?

1.1.1. The model problem

We will begin by addressing the first question.

Let $\Omega \subset \mathbb{R}^2$ be a polygonal domain in the plane, and define $H^1(\Omega)$ to be the set of once weakly-differentiable functions on Ω with square-integrable derivatives. Then $H^1(\Omega)$ is an inner product (even Hilbert) space with

$$\langle v, w\rangle_{H^1(\Omega)} := \sum_{|\boldsymbol{\alpha}|\leq 1} \langle D^{\boldsymbol{\alpha}} v, D^{\boldsymbol{\alpha}} w\rangle_{L^2(\Omega)}, \quad \left(\boldsymbol{\alpha} \in \mathbb{N}_0^2\right)$$

and induced *Sobolev norm* $\|v\|_{H^1(\Omega)} := \langle v, v\rangle_{H^1(\Omega)}^{1/2}$. This norm gives rise to the $H^1(\Omega)$-*seminorm*

$$|v|_{H^1(\Omega)} := \left(\sum_{|\boldsymbol{\alpha}|=1} \|D^{\boldsymbol{\alpha}} v\|_{L^2(\Omega)}^2\right)^{1/2}.$$

We will study the closed subspace

$$H_0^1(\Omega) := \left\{v \in H^1(\Omega) : v|_{\partial\Omega} = 0\right\} \subset H^1(\Omega)$$

of functions vanishing on the boundary of the domain—this can be formalized by viewing these *traces* $v|_{\partial\Omega}$ as square-integrable functions on $\partial\Omega$ using the trace theorem, cf. [13, §1.6]. On this space, the $H^1(\Omega)$-seminorm becomes a norm—*the energy norm*—that we will denote by $\|\cdot\|_{H_0^1(\Omega)}$.

We equip the space $H_0^1(\Omega)$ with the inner product

$$\langle v, w\rangle_{H_0^1(\Omega)} := \sum_{|\boldsymbol{\alpha}|=1} \langle D^{\boldsymbol{\alpha}} v, D^{\boldsymbol{\alpha}} w\rangle_{L^2(\Omega)} = \int_\Omega \nabla v \cdot \nabla w, \quad \left(\boldsymbol{\alpha} \in \mathbb{N}_0^2\right).$$

Note that $\|\cdot\|_{H_0^1(\Omega)}$ is the induced norm with respect to this inner product.

On the space $H_0^1(\Omega)$, the *Poincaré inequality* [13, (5.3.5)] allows us to estimate the $L^2(\Omega)$-norm by its energy norm: There is a $C_{\mathsf{p}} = C_{\mathsf{p}}(\Omega) > 0$ with

$$\|v\|_{L^2(\Omega)} \leq C_{\mathsf{p}} \|v\|_{H_0^1(\Omega)} \quad \left(v \in H_0^1(\Omega)\right). \tag{1.1.8}$$

This equality ensures that on $H_0^1(\Omega)$, the $H^1(\Omega)$- and energy norm are *equivalent*, in that

$$\|v\|_{H_0^1(\Omega)} \leq \|v\|_{H^1(\Omega)} \leq \sqrt{1 + C_{\mathsf{p}}^2}\, \|v\|_{H_0^1(\Omega)}\,.$$

The mapping

$$a : H_0^1(\Omega) \times H_0^1(\Omega) : (v, w) \mapsto \langle v, w \rangle_{H_0^1(\Omega)}$$

is obviously bilinear. We will show its boundedness and coercivity.

Boundedness By Schwarz's inequality, we see that $a(\cdot, \cdot)$ is bounded with $C = 1$:

$$\begin{aligned} |a(v,w)| &\leq \int_\Omega \left| \sum_{|\boldsymbol{\alpha}|=1} (D^{\boldsymbol{\alpha}} v)(D^{\boldsymbol{\alpha}} w) \right| \\ &\leq \sum_{|\boldsymbol{\alpha}|=1} \|D^{\boldsymbol{\alpha}} v\|_{L^2(\Omega)} \|D^{\boldsymbol{\alpha}} w\|_{L^2(\Omega)} \\ &\leq \sqrt{\sum_{|\boldsymbol{\alpha}|=1} \|D^{\boldsymbol{\alpha}} v\|_{L^2(\Omega)}^2} \sqrt{\sum_{|\boldsymbol{\alpha}|=1} \|D^{\boldsymbol{\alpha}} w\|_{L^2(\Omega)}^2} \\ &= \|v\|_{H_0^1(\Omega)} \|w\|_{H_0^1(\Omega)}\,. \end{aligned}$$

Coercivity Its coercivity follows directly from the definition, with $\alpha = 1$:

$$\|v\|_{H_0^1(\Omega)}^2 = \langle v, v \rangle_{H_0^1(\Omega)} = a(v,v) \implies a(v,v) \geq \|v\|_{H_0^1(\Omega)}^2\,.$$

Define $f \in L^2(\Omega)$ to be a square-integrable *forcing function* over our domain. Look at the Poisson boundary value problem of finding $u \in H^2(\Omega)$—$H^1(\Omega)$ functions with square-integrable second weak derivatives—that satisfies

$$\begin{cases} -\triangle u = f & \text{on } \Omega, \\ u = 0 & \text{on } \partial\Omega \end{cases}\,. \tag{1.1.9}$$

If this function u exists, then the following must certainly also hold for all $v \in H_0^1(\Omega)$:

$$\int_\Omega -\triangle u(\mathbf{x}) v(\mathbf{x})\, \mathrm{d}\mathbf{x} = \int_\Omega f(\mathbf{x}) v(\mathbf{x})\, \mathrm{d}\mathbf{x}.$$

Now, v is weakly differentiable, so we can transfer a derivative to the other side [13, (5.1.6)] introducing a normal derivative term,

$$\int_\Omega -\triangle u(\mathbf{x}) v(\mathbf{x})\, \mathrm{d}\mathbf{x} = \int_\Omega \nabla u(\mathbf{x}) \cdot \nabla v(\mathbf{x})\, \mathrm{d}\mathbf{x} - \int_{\partial\Omega} \frac{\partial u}{\partial \mathbf{n}}(s) v(s)\, \mathrm{d}s.$$

The last term vanishes, as v itself vanishes on the boundary of our domain.

In light of the above, we define

$$a(u,v) := \int_\Omega \nabla u \cdot \nabla v\, \mathrm{d}\mathbf{x}, \quad F(v) := \int_\Omega f v\, \mathrm{d}\mathbf{x}. \tag{1.1.10}$$

The mapping F is obviously linear; its continuity follows from (1.1.8):

$$\|F\|_{H_0^1(\Omega)'} := \sup_{0\neq v\in H_0^1(\Omega)} \frac{|F(v)|}{\|v\|_{H_0^1(\Omega)}} \leq \sup_{0\neq v\in H_0^1(\Omega)} \frac{\|f\|_{L^2(\Omega)}\|v\|_{L^2(\Omega)}}{\|v\|_{H_0^1(\Omega)}}$$
$$\leq \sup_{0\neq v\in H_0^1(\Omega)} C_{\mathsf{p}} \frac{\|f\|_{L^2(\Omega)}\|v\|_{H_0^1(\Omega)}}{\|v\|_{H_0^1(\Omega)}} = C_{\mathsf{p}}\|f\|_{L^2(\Omega)}\,.$$

We now established that $a(\cdot,\cdot)$ and $F(\cdot)$ meet the conditions for the variational problem of (1.1.6). We call (1.1.6) together with (1.1.10) the *variational form* of the Poisson equation.

1.1.2. The Lax-Milgram theorem

The second question, on existence and uniqueness, is easily answered by the following theorem.

Theorem 1.1.11 (Lax-Milgram [13, §2.7])**.** *Given a Hilbert space $(V, \langle\cdot,\cdot\rangle)$, a bounded and coercive bilinear form $a(\cdot,\cdot)$, and a continuous linear functional $F \in V'$, there exists a unique $u = u_{a,F} \in V$ with*

$$a(u, v) = F(v) \quad (v \in V).$$

Moreover, the problem of finding u is well-posed in the sense that

$$F \mapsto u_{a,F} \text{ is bounded.}$$

Remark 1.1.12. The second assertion in the Lax-Milgram theorem shows that small perturbations in F lead to small perturbations in the solution u. This is an essential result: In a numerical setting, F is always slightly perturbed through floating-point representation. ◊

Corollary 1.1.13. *Let V, and $a(\cdot,\cdot)$ meet the conditions for a variational problem. Then* (1.1.6) *has a unique solution $u \in V$, and* (1.1.7) *has a unique solution $u_S \in S$.*

Proof. The space V is Hilbert. Applying Lax-Milgram yields the first result. Now, $S \subset V$ is a closed subspace of V so must be Hilbert as well; replacing V by S in Lax-Milgram then yields the second. □

1.1.3. Approximation errors

The answer to the third question, whether we can say anything valuable about the difference $u - u_S$, is more or less the central point to finite element analysis. Knowing that the (norm of) this difference is small, is essential to any convergence analysis. However, this is a hard problem: The proof of Lax-Milgram is not constructive, so even though we know that u and u_S exist, we have no idea what they look like. Even if we manage to construct u_S via something like a finite element method, it is entirely possible that no closed form of u exists so that still, bounding this error norm seems like an impossible task. We will dive into this issue in later sections. For now, we end this section with a few classic results.

Definition 1.1.14. The *error* $e := u - u_S$ is the difference between the real solution of problem (1.1.6) and its Galerkin approximation in (1.1.7). ◊

Lemma 1.1.15 ([13, (2.8.5)])**.** *Let $(V, \langle\cdot,\cdot\rangle)$ and the bilinear form $a(\cdot,\cdot)$ meet the conditions for a variational problem. If $a(\cdot,\cdot)$ is symmetric, then for the energy norm:*

$$\|e\|_E = \min_{v\in S} \|u - v\|_E .$$

In other words: If the bilinear form is symmetric, then u_S is the optimal approximation to u with respect to the energy norm.

Lemma 1.1.16. *Let $(V, \langle\cdot,\cdot\rangle)$ and $a(\cdot,\cdot)$ meet the conditions for a symmetric variational problem. For two closed subspaces $T \subset S \subset V$, with Galerkin solutions u_T resp. u_S, we have* Pythagoras's Theorem*:*

$$\|u - u_S\|_E^2 + \|u_T - u_S\|_E^2 = \|u - u_T\|_E^2 .$$

Proof. We know that $u - u_S \perp_E S$ by the Galerkin orthogonality, and that $u_T - u_S \in S$ by virtue of $T \subset S$. So Pythagoras must hold. □

1.2. Finite elements

Finite elements give us a systematic way of constructing a finite-dimensional space S and accompanying solution u_S. In this section, we will follow the general structure of [13, Ch. 3].

Ciarlet [16] defines a finite element as follows.

Definition 1.2.1. When

i) $K \subset \mathbb{R}^n$ is a bounded closed set with nonempty interior, and piecewise smooth boundary (the *element domain*);

ii) $\mathcal{P}$ is a finite-dimensional space of functions on K (the space of *shape functions*);

iii) $\mathcal{N} := \{N_1, \ldots, N_{\dim \mathcal{P}}\}$ is a basis for $\mathcal{P}'$ (the set of *degrees of freedom*),

then $(K, \mathcal{P}, \mathcal{N})$ is a *finite element.* ◊

It is often hard to determine when $\mathcal{N}$ is a basis for $\mathcal{P}'$; the following result usually helps.

Lemma 1.2.2 ([13, (3.1.4)])**.** *Let $\mathcal{P}$ be a d-dimensional vector space, and let $\mathcal{N} = \{N_i : 1 \le i \le d\} \subset \mathcal{P}'$ be a subset of its dual. Then the following two statements are equivalent.*

i) $\mathcal{N}$ is a basis for $\mathcal{P}'$;

ii) $\mathcal{N}$ determines *$\mathcal{P}$: If for a given $\psi \in \mathcal{P}$, all $N \in \mathcal{N}$ satisfy $N(\psi) = 0$, then $\psi = 0$.*

The following concepts will prove useful to our two-dimensional case.

Definition 1.2.3. A triangle is called *nondegenerate* when its volume is strictly positive. Equivalently, seeing a triangle as the convex hull of its three vertices, it is nondegenerate exactly when the vertices are not collinear.

In the following, we will often implicitly assume a triangle is nondegenerate. ◊

Definition 1.2.4 (Barycentric coordinates)**.** Given a (nondegenerate) triangle $K := \mathrm{hull}(\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3)$ being the convex hull of its three vertices, we can define the *barycentric coordinates* of a point $\mathbf{x} \in K$ as being the tuple $\boldsymbol{\lambda} \in \mathbb{R}^3$ for which

$$\lambda_1, \lambda_2, \lambda_3 \geq 0, \quad \lambda_1 + \lambda_2 + \lambda_3 = 1, \quad \text{and} \quad \mathbf{x} = \lambda_1 \mathbf{v}_1 + \lambda_2 \mathbf{v}_2 + \lambda_3 \mathbf{v}_3. \qquad \Diamond$$

Definition 1.2.5. We can sample these barycentric coordinates on the degree-p *lattice index set* $\mathcal{I}^p := \left\{\boldsymbol{\alpha} \in \mathbb{N}_0^3 : |\boldsymbol{\alpha}| = p\right\}$. For instance,

$$\mathcal{I}^1 = \left\{(1,0,0), (0,1,0), (0,0,1)\right\},$$
$$\mathcal{I}^2 = \left\{(2,0,0), (1,1,0), (0,1,0), (0,1,1), (0,0,1), (1,0,1)\right\}.$$

Doing so gives rise to the *lattice points* on K:

$$\mathcal{D}^p(K) := \left\{\mathbf{v}_{\boldsymbol{\alpha}} : \boldsymbol{\alpha} \in \mathcal{I}^p\right\}, \quad \mathbf{v}_{\boldsymbol{\alpha}} := \frac{\alpha_1}{p}\mathbf{v}_1 + \frac{\alpha_2}{p}\mathbf{v}_2 + \frac{\alpha_3}{p}\mathbf{v}_3.$$

See Figure 1.2.6. $\Diamond$

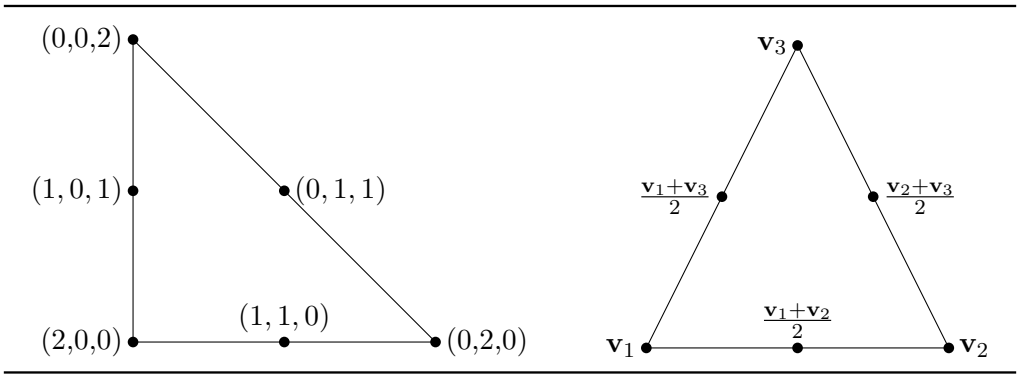


Figure 1.2.6.: Left: The lattice index set $\mathcal{I}^2$. Right: The lattice points $\mathcal{D}^2(K)$ of an arbitrary triangle $K := \mathrm{hull}(\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3)$.

Definition 1.2.7. On a nondegenerate triangle K, the space $\mathbb{P}_p(K)$ of polynomials on K of degree p is defined as

$$\mathbb{P}_p(K) := \mathrm{span}\left\{x^j y^k : 0 \leq j + k \leq p : (x, y) \in K\right\}. \qquad \Diamond$$

Lemma 1.2.8 ([13, Ex. 3.x.30])**.** *Let K be a (nondegenerate) triangle. The space $\mathbb{P}_p(K)$ is finite-dimensional and has dimension*

$$\dim \mathbb{P}_p(K) = \#\mathcal{I}^p = (p+1)(p+2)/2. \tag{1.2.9}$$

Lemma 1.2.10 ([13, (3.2.4)])**.** *Let K be a nondegenerate triangle. Then the set of point evaluations on $\mathcal{D}^p(K)$ determines $\mathbb{P}_p(K)$ in the sense of Lemma 1.2.2.*

With these quantities in hand, let us look at a classic example.

Example 1.2.11 (Lagrange element)**.** Let $K := \text{hull}(\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3)$ be a triangle in $\mathbb{R}^2$. Define $\mathcal{P} := \mathbb{P}_p(K)$ to be the space of degree-p polynomials on K. Let the set $\mathcal{N}$ consist of point evaluations on $\mathcal{D}^p(K)$—$\mathcal{N} := \left\{\psi \mapsto \psi(\mathbf{v}_{\boldsymbol{\lambda}}) : \mathbf{v}_{\boldsymbol{\lambda}} \in \mathcal{D}^p(K)\right\}$.

By Lemma 1.2.10, any degree-p polynomial is completely determined by its point evaluations on $\mathcal{D}^p(K)$. Hence, if a polynomial vanishes on *all* such points, it must be the zero function. Then, by Lemma 1.2.2, the tuple $(K, \mathcal{P}, \mathcal{N})$ is a finite element. We call this the (degree-p) *Lagrange element.* ◊

Definition 1.2.12. Let $(K, \mathcal{P}, \mathcal{N})$ be a finite element. The basis $\{\phi_1, \ldots, \phi_{\dim \mathcal{P}}\}$ of $\mathcal{P}$ dual to $\mathcal{N}$ (i.e., $N_i(\phi_j) = \delta_{ij}$) is called the *nodal basis* of $\mathcal{P}$. ◊

We now hold the definition of a finite element, and a concrete example of one. How do we link this to the construction of subspaces S of the Sobolev space V? For this, we need a couple more ingredients: triangulations and interpolants.

1.2.1. Triangulations

Definition 1.2.13 (Triangulation)**.** Consider some polygonal domain Ω. We define a *triangulation of* Ω as being a finite collection $\mathcal{K} := \{K\}$ of triangles with

i) $\cup_{K\in\mathcal{K}} K = \overline{\Omega}$ (the triangles cover Ω);

ii) $K^\circ \cap L^\circ = \varnothing$ when $K \neq L$ (no overlap);

Often, we also desire this triangulation to be *conforming*, in that

iii) For any two triangles $K \neq L$, $K \cap L$ is (a) empty, (b) a common vertex, or (c) a common edge. ◊

Especially the last condition is important: It tells us that there are no *hanging nodes*—a vertex in the middle of a triangle's edge—and that, essentially, the triangles "glue together nicely". See Figure 1.2.14 for a visualization.

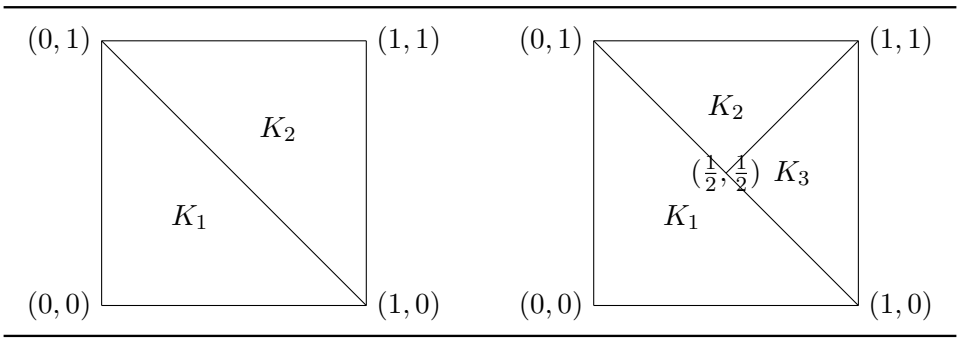


Figure 1.2.14.: Left: A triangulation of $\Omega := (0,1)^2$ with two triangles. Right: A non-conforming triangulation of Ω; the center has a hanging node.

Definition 1.2.15. Take a domain Ω with a conforming triangulation $\mathcal{K}$, and on each triangle, a finite element. The *finite element space* $V(\mathcal{K})$ associated with our boundary value problem from (1.1.9) is then defined as

$$V(\mathcal{K}) := \left\{ v \in H_0^1(\Omega) : v|_K \in \mathcal{P}(K) \quad (K \in \mathcal{K}) \right\}. \qquad \text{◊}$$

Theorem 1.2.16. *Let Ω be a domain with conforming triangulation $\mathcal{K}$. Equip each triangle with a finite element $(K, \mathcal{P}(K), \mathcal{N}(K))$. Then the finite element space satisfies*

$$V(\mathcal{K}) \subset C(\overline{\Omega}).$$

Proof. Choose $v \in V(\mathcal{K})$; then v is continuous on the interior of all triangles, so it suffices to show that v is continuous over edges. As $v|_{\partial\Omega} = 0$, edges on the domain boundary are a non-issue; we are left with the case when $e \not\subset \partial\Omega$.

Let $K_1, K_2 \in \mathcal{K}$ share an edge $e := K_1 \cap K_2$. Assume the function v is *not* continuous over e. We know that $v \in H^1_0(\Omega)$, hence in $H^1(K_1 \cup K_2)$, so it has weak derivatives $\partial^{\boldsymbol{\alpha}} v \in L^2(K_1 \cup K_2)$ for $\boldsymbol{\alpha} \in \mathbb{N}^2_0$ with $|\boldsymbol{\alpha}| = 1$, or in other words

$$\mathcal{L} : L^2(K_1 \cup K_2) \to \mathbb{R} : w \mapsto -\int\limits_{K_1 \cup K_2} \left(\partial^{\boldsymbol{\alpha}} v(\mathbf{x})\right) w(\mathbf{x}) \, \mathrm{d}\mathbf{x} \in L^2(K_1 \cup K_2)'.$$

This means that

$$\left|\mathcal{L}(w)\right| \leq C \|w\|_{L^2(K_1 \cup K_2)} \quad \left(w \in L^2(K_1 \cup K_2)\right). \tag{1.2.17}$$

Take an $\boldsymbol{\alpha} \in \mathbb{N}^2_0$ so that $\boldsymbol{\alpha} \cdot \mathbf{n}_{K_1} \neq 0$, with $\mathbf{n}_K$ the outward normal of K. Take a function $w \in C^\infty_0(K_1 \cup K_2) \subset L^2(K_1 \cup K_2)$. Then, by definition of the weak derivative, and using that $v|_{K_k} \in \mathcal{P}(K_k)$,

$$\begin{aligned}
\mathcal{L}(w) &= -\int\limits_{K_1 \cup K_2} \left(\partial^{\boldsymbol{\alpha}} v(\mathbf{x})\right) w(\mathbf{x}) \, \mathrm{d}\mathbf{x} := \int\limits_{K_1 \cup K_2} v(\mathbf{x}) \left(\partial^{\boldsymbol{\alpha}} w(\mathbf{x})\right) \mathrm{d}\mathbf{x} = \sum_{k=1}^{2} \int\limits_{K_k} v(\mathbf{x}) \left(\partial^{\boldsymbol{\alpha}} w(\mathbf{x})\right) \mathrm{d}\mathbf{x} \\
&= \sum_{k=1}^{2} \left[-\int\limits_{K_k} \left(\partial^{\boldsymbol{\alpha}} v(\mathbf{x})\right) w(\mathbf{x}) \, \mathrm{d}x + \int\limits_{\partial K_k} v(s) w(s) \boldsymbol{\alpha} \cdot \mathbf{n}_{K_k} \, \mathrm{d}s \right].
\end{aligned}$$

Now, w vanishes everywhere on ∂K_k except on e, so the above reduces to

$$\mathcal{L}(w) = -\sum_{k=1}^{2} \int\limits_{K_k} \left(\partial^{\boldsymbol{\alpha}} v(\mathbf{x})\right) w(\mathbf{x}) \, \mathrm{d}x + \int_e (v|_{K_1} - v|_{K_2})(s) w(s) \boldsymbol{\alpha} \cdot \mathbf{n}_{K_1} \, \mathrm{d}s.$$

To find (1.2.17), we must bound $\mathcal{L}(w)$. The first term can be bounded by

$$-\sum_{k=1}^{2} \int\limits_{K_k} \left(\partial^{\boldsymbol{\alpha}} v(\mathbf{x})\right) w(\mathbf{x}) \, \mathrm{d}x \leq \|\partial^{\boldsymbol{\alpha}} v\|_{L^2(K_1 \cup K_2)} \|w\|_{L^2(K_1 \cup K_2)}.$$

For the second term: One can find a sequence $(w_n)_n$ of test functions with unit $L^2(K_1 \cup K_2)$-norm such that the second term grows unbounded in n; this is due to the jump $v|_{K_1} - v|_{K_2} \neq 0$ and $\boldsymbol{\alpha} \cdot \mathbf{n}_{K_1} \neq 0$. The result is that (1.2.17) does not hold, which is a contradiction with the earlier assumption that v was not continuous over e. □

Example 1.2.18 (Linear Lagrange elements)**.** With Lagrange elements of degree $p = 1$, we can simplify a lot of the concepts we've seen thus far. Let Ω be some polygonal domain in $\mathbb{R}^2$, and let $\mathcal{K}$ be some conforming triangulation on this domain.

Locally on a triangle $K := \text{hull}(\mathbf{v}_1^K, \mathbf{v}_2^K, \mathbf{v}_3^K)$, we can define the nodal basis $\left\{\phi_1^K, \phi_2^K, \phi_3^K\right\}$ of linear functions with $\phi_i^K(\mathbf{v}_j) = \delta_{ij}$.

Look at the result of Theorem 1.2.16. We see that $V(\mathcal{K})$ contains the continuous functions that vanish on $\partial\Omega$ and are moreover piecewise linear with respect to $\mathcal{K}$. We collect all vertices on triangles in $\mathcal{K}$ in a set $\mathcal{V}$, i.e.,

$$\mathcal{V} := \bigcup_{K\in\mathcal{K}} \left\{\mathbf{v}_1^K, \mathbf{v}_2^K, \mathbf{v}_3^K : K = \text{hull}(\mathbf{v}_1^K, \mathbf{v}_2^K, \mathbf{v}_3^K)\right\},$$

and take the subset of all interior vertices $\mathcal{V}_{int} := \mathcal{V} \setminus \partial\Omega$. We can patch together (a subset of) the local basis functions into a global *hat function* $\phi_{\mathbf{v}}$ through

$$\phi_{\mathbf{v}} : \Omega \to \mathbb{R} : \mathbf{x} \mapsto \phi_{\mathbf{v}}(\mathbf{x})|_K = \begin{cases} \phi_i^K(\mathbf{x}) & \text{if } \mathbf{v} = \mathbf{v}_i^K \text{ for some } i \\ 0 & \text{else} \end{cases} \qquad (\mathbf{v} \in \mathcal{V}_{int}, K \in \mathcal{K}).$$

From this, we see that the global linear hat functions $\phi_{\mathbf{v}}$ are characterized by

$$\phi_{\mathbf{v}}(\mathbf{w}) = \delta_{\mathbf{vw}}, \quad (\mathbf{v} \in \mathcal{V}_{int}, \mathbf{w} \in \mathcal{V}),$$

that they span the entire space $V(\mathcal{K})$, and that they themselves are linearly independent, so they must be a basis for $V(\mathcal{K})$.

Let's look at an instructive example. Let Ω and $\mathcal{K}$ be characterized as in the left of Figure 1.2.19. The only two vertices not on the domain boundary are denoted $\mathbf{v}$ and $\mathbf{w}$, and make up the set $\mathcal{V}_{int}$. There are two global basis functions, and it is immediately obvious why $\phi_{\mathbf{v}}$ is called a hat function: It vanishes on all vertices except the one it is associated with. ◊

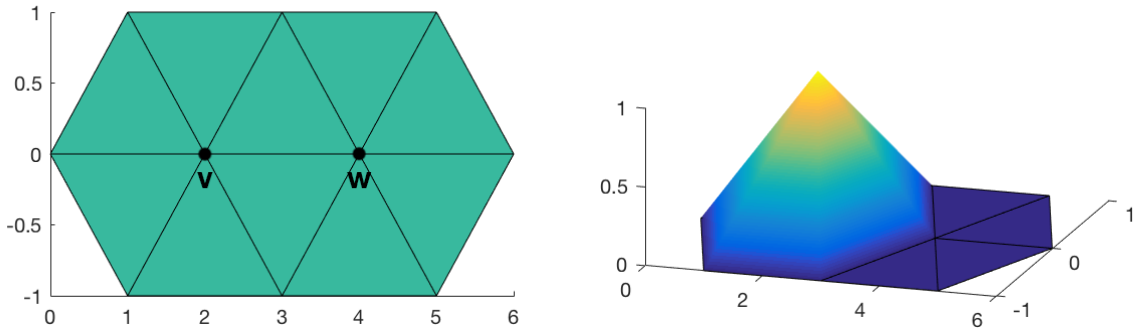


Figure 1.2.19.: Visual aid for Example 1.2.18. Left: domain Ω, with two interior vertices $\mathbf{v}$ and $\mathbf{w}$. Right: the global hat function $\phi_{\mathbf{v}}$.

We will now show that this systematic construction of $V(\mathcal{K})$ paves the way to an actual solution.

1.2.2. Constructing the Galerkin solution

We know by the Lax-Milgram theorem that the Ritz-Galerkin problem has *a unique* solution $u_{\mathcal{K}} \in V(\mathcal{K})$, and we will now look at actually constructing it.

Assume that one has a global basis $\Phi := \{\phi\}$ on $V(\mathcal{K})$. If we manage to find a function $u_{\mathcal{K}}$ for which

$$a(u_{\mathcal{K}}, \phi) = F(\phi) \quad (\phi \in \Phi)$$

holds, then by linearity, it must hold for *any* $v \in V(\mathcal{K})$ thereby solving the approximation problem. Viewing Φ as a vector of functions, we can formulate the equivalent problem of finding the vector $\mathbf{u} \in \mathbb{R}^{\#\Phi}$ that solves

$$a\left(\sum_j \mathbf{u}_j \phi_j, \phi_i\right) = F(\phi_i) \quad (1 \le i \le \#\Phi)\,.$$

Again by linearity of $a(\cdot, \phi_i)$, this boils down to finding $\mathbf{u}$ with

$$\sum_j \mathbf{u}_j a(\phi_j, \phi_i) = F(\phi_i) \quad (1 \le i \le \#\Phi)\,. \tag{1.2.20}$$

Definition 1.2.21. In (1.2.20), the quantities $a(\phi_j, \phi_i)$ and $F(\phi_i)$ can be collected in a *stiffness matrix* A resp. *load vector* $\boldsymbol{b}$, together forming the linear system

$$\boldsymbol{A}\mathbf{u} = \mathbf{b}, \quad \boldsymbol{A} := \Big[a(\phi_j, \phi_i)\Big]_{i,j=1}^{\#\Phi}, \quad \mathbf{b} := \Big[F(\phi_i)\Big]_{i=1}^{\#\Phi},$$

which can be solved using a suitable linear solver. ◊

1.3. A priori error estimation

Now that we know how to construct an approximation space and the accompanying finite element solution, it is time to shed some light on the estimation of $\|u - u_S\|_{H^1(\Omega)}$.

Some notation will prove useful.

Definition 1.3.1. For two scalar quantities A and B dependent on a variable—a real-valued function, for instance—we will write $A \lesssim B$ when there is a constant C independent of this variable such that $A \le CB$, and write $A \eqsim B$ when $A \lesssim B$ and $B \lesssim A$. ◊

Definition 1.3.2 (Affine equivalence)**.** Let $(\hat{K}, \hat{\mathcal{P}}, \hat{\mathcal{N}})$ be a finite element, and let $F(\hat{\mathbf{x}}) = A\hat{\mathbf{x}} + \mathbf{b}$ be an affine map with $\det A \neq 0$. Then $(\hat{K}, \hat{\mathcal{P}}, \hat{\mathcal{N}})$ and $(K, \mathcal{P}, \mathcal{N})$ are *affine equivalent* iff with $\phi := \hat{\phi} \circ F^{-1}$,

i) $F(\hat{K}) = K$;

ii) $\left\{\phi : \hat{\phi} \in \hat{\mathcal{P}}\right\} = \mathcal{P}$;

iii) $\left\{\hat{\phi} \mapsto N(\phi) : N \in \mathcal{N}\right\} = \hat{\mathcal{N}}$. ◊

Lemma 1.3.3 ([13, (3.4.10)])**.** *All Lagrange elements of given degree are affine equivalent.*

Definition 1.3.4. With $(K, \mathcal{P}, \mathcal{N})$ a finite element, we define

$$h_K := \inf\left\{\operatorname{diam}(S) : S \text{ ball containing } K\right\},$$
$$\rho_K := \sup\left\{\operatorname{diam}(S) : S \text{ ball contained in } K\right\}.$$

A family of finite elements $\left\{(K, \mathcal{P}, \mathcal{N})\right\}$ is called *uniformly shape regular* when

$$\sup_K h_K/\rho_K < \infty.$$ ◊

Such a uniformly shape regular family of elements has a bound on the "degeneracy" of the triangles it contains, so that triangles have similar shapes. This property can be used to arrive at a classical result bounding the local interpolation error.

Definition 1.3.5. Given a finite element $(K, \mathcal{P}, \mathcal{N})$, and $\Phi = \{\phi_i : 1 \leq i \leq d\}$ its nodal basis from Definition 1.2.12. If v is a function for which $\left\{N_i(v) : N_i \in \mathcal{N}\right\}$ are all defined, then, we can define the *local interpolant* as

$$\mathcal{I}_K(v) := \sum_{i=1}^{d} N_i(v)\phi_i. \qquad \Diamond$$

Theorem 1.3.6 ([42, Thm. 2.1])**.** *Let $(\hat{K}, \hat{\mathcal{P}}, \hat{\mathcal{N}})$ be an element in two dimensions (the* reference element*). Suppose that the degrees of freedom in $\hat{\mathcal{N}}$ don't involve any derivatives.*

Take a uniformly shape regular family of finite elements $\left\{(K, \mathcal{P}, \mathcal{N})\right\}$, each affine equivalent to the reference element.

If for $k \geq 1$ we have $\mathbb{P}_k(K) \subset \hat{\mathcal{P}}$, then it holds that

$$|v - \mathcal{I}_K v|_{H^m(K)} \lesssim h_K^{k+1-m} |v|_{H^{k+1}(K)}, \quad \left(0 \leq m \leq k+1, \quad v \in H^{k+1}(K)\right).$$

Corollary 1.3.7. *The degrees of freedom of the Lagrange element are simply point evaluations—there are no derivatives involved. With Lemma 1.3.3, we see that the above theorem holds for Lagrange finite elements.*

In fact, we can collect the local results as follows to arrive at a global bound.

Corollary 1.3.8. *Let Ω be a domain. Take a conforming triangulation $\mathcal{K}$ of elements affine equivalent to some reference element $(\hat{K}, \hat{\mathcal{P}}, \hat{\mathcal{N}})$. Then with the same conditions as in the previous theorem,*

$$\left(\sum_{K \in \mathcal{K}} h_K^{2(m-k-1)} \|v - I_K v\|_{H^m(K)}^2\right)^{1/2} \lesssim |v|_{H^{k+1}(\Omega)} \quad \left(v \in H^{k+1}(\Omega)\right).$$

1.3.1. Convergence of Galerkin solutions

Corollary 1.3.8 shows an important result. Say we are solving the Poisson problem, and assume its solution satisfies $u \in H^2(\Omega) \cap H_0^1(\Omega)$. For simplicity, assume linear Lagrange finite elements throughout.

Given a triangulation $\mathcal{K}$, we can define the global interpolant of u in a piecewise manner through

$$\mathcal{I}_{\mathcal{K}} u|_K := \mathcal{I}_K u.$$

This interpolant is well-defined and continuous on Ω, as it coincides with u on every vertex in the triangulation, hence it is a member of $H^1(\Omega)$. Now, through Corollary 1.3.8,

$$\|u - \mathcal{I}_{\mathcal{K}} u\|_{H^1(\Omega)} = \left(\sum_{K \in \mathcal{K}} \|u - I_K u\|_{H^1(K)}^2\right)^{1/2} \lesssim h|u|_{H^2(\Omega)}$$

where $h := \sup_{K \in \mathcal{K}} h_K$.

Moreover, by virtue of $u|_{\partial\Omega} = 0$, $\mathcal{I}_{\mathcal{K}}u$ vanishes on $\partial\Omega$ so it is even in $H_0^1(\Omega)$. Locally, $\mathcal{I}_{\mathcal{K}}u|_K \in \mathcal{P}(K)$, so $\mathcal{I}_{\mathcal{K}}u$ must be in $V(\mathcal{K})$. Stringing everything together yields the inequality

$$\|u - u_S\|_{H_0^1(\Omega)} = \min_{v\in V(\mathcal{K})} \|u - v\|_{H_0^1(\Omega)} \leq \|u - \mathcal{I}_{\mathcal{K}}u\|_{H_0^1(\Omega)} \leq \|u - \mathcal{I}_{\mathcal{K}}u\|_{H^1(\Omega)} \lesssim h|u|_{H^2(\Omega)}\,.$$

In words, when the solution is smooth, we can expect linear convergence of the finite element solution to the real solution in terms of the *mesh width* $h = h(\mathcal{K})$. In less smooth situations, we have the following similar but weaker result asserting convergence.

Theorem 1.3.9 ([16, Thm 3.2.3])**.** *Let Ω be a domain. Suppose we have a sequence of conforming triangulations $(\mathcal{K}_n)_n$ with $\lim_{n\to\infty} h(\mathcal{K}_n) = 0$, and that every element in these triangulations is affine equivalent to a Lagrange reference element $(\hat{K}, \hat{\mathcal{P}}, \hat{\mathcal{N}})$. Then the following relation holds:*

$$\lim_{n\to\infty} \|u - u_{\mathcal{K}_n}\|_{H^1(\Omega)} = 0.$$

This tells us that the solutions are known to converge, but it doesn't tell us anything about the rate. The error bounds of this section were all independent of the *actual* Galerkin solutions, measured in terms of h instead. We call this *a priori error estimation*, because we use a priori information on the exact solution u, such as $u \in H^2(\Omega)$. Often, we don't have this kind information. Luckily, using knowledge of the Galerkin solution locally on elements, we can make much more precise statements. This is called *a posteriori* error estimation which we will study in the next section.

Even when $u \notin H^2(\Omega)$, some smoothness can be expected: It is known (cf. [21, §6.4]) that in the Poisson equation, when $f \in L^2(\Omega)$, u shows *interior regularity* in that $u \in H^2(\Sigma)$ for every $\Sigma \subset \Omega$ with $\overline{\Sigma} \subset \Omega$. Therefore, u is smooth in the domain interior so it doesn't require the level of refinement its non-smooth parts need. In other words, it makes sense not to subdivide *every* triangle, but just those ones where the approximation error is large. This is problematic using the current tools, as they only give us information globally. We will develop tools for local error indication in the next section, and study adaptive grid refinement in §1.5.

1.4. A posteriori error estimation

A posteriori error estimation has spawned a thriving field of research. Let's focus on our model problem of solving the Poisson equation; see §1.1.1.

Definition 1.4.1. Given some conforming finite element triangulation $\mathcal{K}$, we can define a mapping

$$\eta : V(\mathcal{K}) \times \mathcal{K} \to \mathbb{R}_+ : (v, K) \mapsto \eta(v, K)$$

that gives some indication of the difference of the exact solution u and its approximation v in the *local energy norm* $|\cdot|_{H^1(K)}$. (In most cases, we select v either equal or closely related to $u_{\mathcal{K}}$, in which case $\eta(v, K)$ is related to the error norm $|e|_{H^1(K)}$.) We therefore call this mapping the *local error indicator*. For $\mathcal{M} \subset \mathcal{K}$, we can define

$$\eta(v, \mathcal{M})^2 := \sum_{K\in\mathcal{M}} \eta(v, K)^2.$$

It is often desirable for error indicators to be *efficient* in that there is a constant $C_{\mathsf{eff}} > 0$ with

$$\eta(u_{\mathcal{K}}, \mathcal{K})^2 \leq C_{\mathsf{eff}} \|e\|^2_{H_0^1(\Omega)}\,;$$

in other words, if η is large, then the error norm must be large as well. Conversely, it should be *reliable* in the sense that there is a $C_{\mathsf{rel}} > 0$ with

$$C_{\mathsf{rel}}\|e\|^2_{H^1_0(\Omega)} \leq \eta(u_{\mathcal{K}}, \mathcal{K})^2$$

so that when η is small, the true error norm is small as well. ◊

Remark 1.4.2. To shed some light on the difference in naming between error *indication* and *estimation*: Locally, these error indicators do not generally estimate the error norm (in the sense that $\eta(u_{\mathcal{K}}, K) \not\approx \|e\|_{H^1_0(K)}$); they merely give an *indication* of the local error norm. Globally, however, the quantity $\eta(u_{\mathcal{K}}, \mathcal{K})$ *does* estimate the global error norm $\|u - u_{\mathcal{K}}\|_{H^1_0(\Omega)}$ (modulo a—usually of higher order—data oscillation term we will discuss promptly). ◊

Definition 1.4.3. Assume a finite element method based on Lagrange elements of degree p. Take $\mathcal{M} \subset \mathcal{K}$, and fix $\mathbb{N}_0 \ni r \geq p - 2$. One then defines the *oscillation term* as

$$\operatorname{osc}(K)^2 = \operatorname{osc}_r(K)^2 := h_K^2\|f - P^r f\|^2_{L^2(K)}\,, \quad \operatorname{osc}(\mathcal{M})^2 := \sum_{K\in\mathcal{M}} \operatorname{osc}(K)^2$$

where P^r is the $L^2(K)$-projector onto $\mathbb{P}_r(K)$, and f is the forcing function present in the Poisson equation. The *total error* is then

$$\mathcal{E}(\mathcal{K}) := \sqrt{\|u - u_{\mathcal{K}}\|^2_{H^1_0(\Omega)} + \operatorname{osc}(\mathcal{K})^2}.$$

Often, efficiency and reliability is defined with respect to the total error instead of just error norm. This is because our finite element space is discrete and therefore cannot be expected to capture every detail of forcing functions in the (infinite-dimensional) space $L^2(\Omega)$. This is especially true for the highly oscillatory components, hence the name. In theory, this oscillation term can be of the same magnitude as the error norm; in practice however, it is often much smaller—cf. [42, Ex. 11.3]. ◊

1.4.1. Refinement error indicator

One can define a very simple error indicator with good practical results. It hinges on the notion of grid refinement, which we will study in more detail in §1.5. For now, assume that it is possible to subdivide every triangle in $\mathcal{K}$ into two smaller ones while maintaining uniform shape regularity. Recursively applying this rule yields a highly refined $\mathcal{K}^*$.

Relative to $\mathcal{K}$, $u_{\mathcal{K}^*}$ is (hoped to be) a good approximation to the real solution u by the convergence result of Theorem 1.3.9.

Definition 1.4.4. Given a conforming triangulation $\mathcal{K}$, define $\mathcal{K}^*$ by two recursive subdivisions of every triangle in $\mathcal{K}$. We can then estimate $|u - v|_{H^1(K)}$ by the *refinement error indicator*

$$\eta_{\mathsf{Ref}}(v, K) := |u_{\mathcal{K}^*} - v|_{H^1(K)}\,. \qquad ◊$$

We see by Lemma 1.1.16 that

$$\eta_{\mathsf{Ref}}(u_{\mathcal{K}}, \mathcal{K})^2 = \|u - u_{\mathcal{K}}\|^2_{H^1_0(\Omega)} - \|u - u_{\mathcal{K}^*}\|^2_{H^1_0(\Omega)} \leq \|u - u_{\mathcal{K}}\|^2_{H^1_0(\Omega)}$$

so this error estimator is efficient with $C_{\mathsf{eff}} = 1$. It is however not reliable; when $u_{\mathcal{K}} = u_{\mathcal{K}^*}$ with $u \neq u_{\mathcal{K}}$, then for each $C_{\mathsf{rel}} > 0$,

$$C_{\mathsf{rel}}\|e\|^2_{H^1_0(\Omega)} > \eta_{\mathsf{Ref}}(u_{\mathcal{K}}, \mathcal{K})^2 = 0.$$

Example 1.4.5 ([38, p. 62])**.** Let's look at our model Poisson problem from §1.1.1, on $\Omega := (0,1)^2$. Choose $f := 1$, so that there is no data oscillation. Take linear Lagrange elements on the three conforming triangulations $\mathcal{K}_0, \mathcal{K}_1, \mathcal{K}_2$ depicted in Figure 1.4.6. Each one is created from the previous by subdividing each triangle into two.

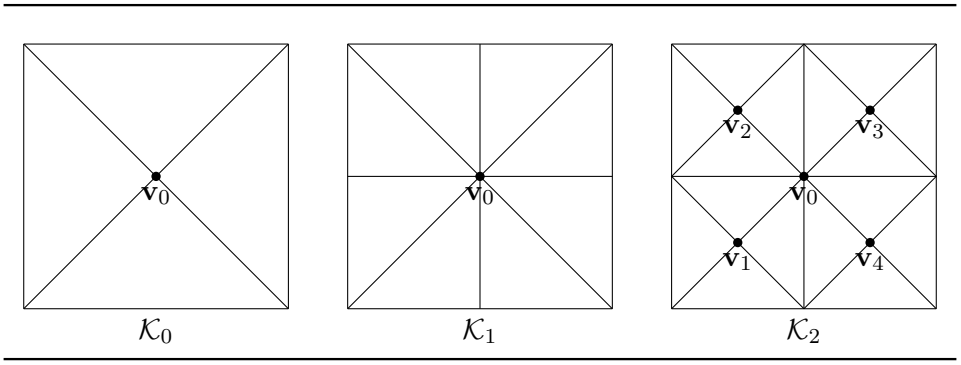


Figure 1.4.6.: Visual aid with Example 1.4.5.

Linear Lagrange elements induce a global basis of hat functions through the local nodal basis; cf. Example 1.2.18.

On $\mathcal{K}_0$ and $\mathcal{K}_1$, we see that every Lagrange node except for $\mathbf{v}_0$ is on $\partial\Omega$. Therefore the spaces $V(\mathcal{K}_0)$ and $V(\mathcal{K}_1)$ are both spanned by a single hat function ϕ_0, so their Galerkin solutions must coincide. In fact, the function

$$u_{\mathcal{K}_0} = \frac{1}{12}\phi_0 = u_{\mathcal{K}_1} \neq u$$

solves the Galerkin approximation problem on both $\mathcal{K}_0$ and $\mathcal{K}_1$. The refinement error indicator defined through a single uniform refinement is certainly not reliable.

It turns out, though, that even two refinements is not enough to get a positive error estimate. The space $V(\mathcal{K}_2)$ is spanned by five hat functions. But one can show that the function $u_{\mathcal{K}_2} := u_{\mathcal{K}_0}$ solves the Galerkin problem on $\mathcal{K}_2$ *as well*—the error estimate $\left\|u_{\mathcal{K}_0} - u_{\mathcal{K}_2}\right\|_{H^1_0(\Omega)}$ still vanishes after two uniform refinements. ♢

1.4.2. Residual error indicator

Any reliable error indicator should use information about the problem at hand. In this paragraph, we will sketch the derivation of the classical *residual-based error indicator*. The *residual* embedded in its name is defined by

$$R \in V' : R(v) = a(e, v) = a(u - u_{\mathcal{K}}, v) = F(v) - a(u_{\mathcal{K}}, v).$$

Note that this residual vanishes for $v_{\mathcal{K}} \in V(\mathcal{K})$, so take an arbitrary one. The first step is to decompose this equation into local contributions on each element:

$$\begin{aligned} R(v) &= R(v - v_{\mathcal{K}}) \\ &= \sum_{K \in \mathcal{K}} \left[\int_K f(\mathbf{x}) \left(v(\mathbf{x}) - v_{\mathcal{K}}(\mathbf{x}) \right) \mathrm{d}\mathbf{x} - \int_K \nabla u_{\mathcal{K}}(\mathbf{x}) \cdot \nabla \left(v(\mathbf{x}) - v_{\mathcal{K}}(\mathbf{x}) \right) \mathrm{d}\mathbf{x} \right]. \end{aligned}$$

We can then use partial integration to get rid of the gradient on $v - v_{\mathcal{K}}$, gaining an edge term:

$$R(v) = \sum_{K\in\mathcal{K}} \left[\int_K \big(f(\mathbf{x}) + \triangle u_{\mathcal{K}}(\mathbf{x})\big)\big(v(\mathbf{x}) - v_{\mathcal{K}}(\mathbf{x})\big)\,\mathrm{d}\mathbf{x} - \int_{\partial K} \frac{\partial u_{\mathcal{K}}}{\partial \mathbf{n}}(s)\big(v(s) - v_{\mathcal{K}}(s)\big)\,\mathrm{d}s \right]$$

with $\mathbf{n}$ the outward normal to ∂K. We know that both v and $v_{\mathcal{K}}$ vanish on all edges on $\partial\Omega$, so define $\mathcal{E}_{int}$ as the set of all interior edges. We can then collect edge terms by

$$R(v) = \sum_{K\in\mathcal{K}} \int_K \big(f(\mathbf{x}) + \triangle u_{\mathcal{K}}(\mathbf{x})\big)\big(v(\mathbf{x}) - v_{\mathcal{K}}(\mathbf{x})\big)\,\mathrm{d}\mathbf{x} - \sum_{e\in\mathcal{E}_{int}} \int_e [\![\nabla u_{\mathcal{K}}]\!]_e(s)\big(v(s) - v_{\mathcal{K}}(s)\big)\,\mathrm{d}s$$

where

$$[\![\nabla\phi]\!]_e(\mathbf{x}) := \lim_{\varepsilon\to 0}\big((\nabla\phi)(\mathbf{x} + \varepsilon\mathbf{n}_e) - (\nabla\phi)(\mathbf{x} - \varepsilon\mathbf{n}_e)\big)\cdot\mathbf{n}_e$$

is the jump operator, and $\mathbf{n}_e$ is *any* normal on e—either one of the two suffices.

Lastly, by Schwarz's inequality,

$$R(v) \le \sum_{K\in\mathcal{K}} \|f + \triangle u_{\mathcal{K}}\|_{L^2(K)}\|v - v_{\mathcal{K}}\|_{L^2(K)} + \sum_{e\in\mathcal{E}_{int}} \big\|[\![\nabla u_{\mathcal{K}}]\!]_e\big\|_{L^2(e)}\|v - v_{\mathcal{K}}\|_{L^2(e)}\,.$$

Select $v_{\mathcal{K}}$ to be the Scott-Zhang interpolant [40] of v. For this interpolant, we can bound the interpolation errors as follows, cf. [42, (9)]:

$$\|v - v_{\mathcal{K}}\|_{L^2(K)} \lesssim h_K\|v\|_{H^1_0(S(K,\mathcal{K}))}\,, \quad \|v - v_{\mathcal{K}}\|_{L^2(e)} \lesssim h_{K_e}^{1/2}\|v\|_{H^1_0(S(K_e,\mathcal{K}))} \tag{1.4.7}$$

where

$$S(K,\mathcal{K}) := \big\{K' \in \mathcal{K} : K \cap K' \neq \varnothing\big\}$$

is a patch around K, and K_e is some element adjacent to e. With this, the above bound reduces to

$$\begin{aligned} R(v) &\lesssim \sum_{K\in\mathcal{K}} h_K\|f + \triangle u_{\mathcal{K}}\|_{L^2(K)}\|v\|_{H^1_0(S(K,\mathcal{K}))} + \sum_{e\in\mathcal{E}_{int}} h_{K_e}^{1/2}\big\|[\![\nabla u_{\mathcal{K}}]\!]_e\big\|_{L^2(e)}\|v\|_{H^1_0(S(K_e,\mathcal{K}))} \\ &\le \sqrt{\sum_{K\in\mathcal{K}} h_K^2\|f + \triangle u_{\mathcal{K}}\|^2_{L^2(K)}}\sqrt{\sum_{K\in\mathcal{K}}\|v\|^2_{H^1_0(S(K,\mathcal{K}))}} \\ &+ \sqrt{\sum_{e\in\mathcal{E}_{int}} h_{K_e}\big\|[\![\nabla u_{\mathcal{K}}]\!]_e\big\|^2_{L^2(e)}}\sqrt{\sum_{e\in\mathcal{E}_{int}}\|v\|^2_{H^1_0(S(K_e,\mathcal{K}))}}. \end{aligned}$$

In turn, by uniform shape regularity, we find

$$\sum_{K\in\mathcal{K}}\|v\|^2_{H^1_0(S(K,\mathcal{K}))} \lesssim \|v\|^2_{H^1_0(\Omega)}\,, \quad \sum_{e\in\mathcal{E}_{int}}\|v\|^2_{H^1_0(S(K_e,\mathcal{K}))} \lesssim \|v\|^2_{H^1_0(\Omega)}$$

yielding

$$R(v) \lesssim \|v\|_{H^1_0(\Omega)}\left(\sum_{K\in\mathcal{K}} h_K^2\|f + \triangle u_{\mathcal{K}}\|^2_{L^2(K)} + \sum_{e\in\mathcal{E}_{int}} h_{K_e}\big\|[\![\nabla u_{\mathcal{K}}]\!]_e\big\|^2_{L^2(e)}\right)^{1/2}.$$

Noting that

$$\|u-u_{\mathcal{K}}\|_{H_0^1(\Omega)} \leq \sup_{0\neq v\in V} \frac{a(u-u_{\mathcal{K}},v)}{\|v\|_{H_0^1(\Omega)}} = \sup_{0\neq v\in V} \frac{R(v)}{\|v\|_{H_0^1(\Omega)}},$$

we then find a bound on the error norm as follows:

$$\|u-u_{\mathcal{K}}\|^2_{H_0^1(\Omega)} \lesssim \sum_{K\in\mathcal{K}} h_K^2 \|f+\triangle u_{\mathcal{K}}\|^2_{L^2(K)} + \sum_{e\in\mathcal{E}_{int}} h_{K_e} \left\|[\![\nabla u_{\mathcal{K}}]\!]_e\right\|^2_{L^2(e)}.$$

Collecting edge terms over each triangle gives rise to the following definition.

Definition 1.4.8 (Residual-based error indicator)**.** Given a conforming finite element triangulation $\mathcal{K}$ of Lagrange elements with fixed degree p, define the *residual-based error indicator* as

$$\eta_{\mathsf{Res}}(v,\mathcal{K})^2 := h_K^2\|f+\triangle u_{\mathcal{K}}\|^2_{L^2(K)} + \frac{h_K}{2}\left\|[\![\nabla u_{\mathcal{K}}]\!]_e\right\|^2_{L^2(\partial K\setminus\partial\Omega)}. \qquad \lozenge$$

Theorem 1.4.9. *The residual-based error indicator is both efficient and reliable.*

Proof. In deriving the error indicator, we also implicitly proved its reliability. For a proof of its efficiency, we refer to [42, Thm. 7.2]. □

1.4.3. Melenk-Wohlmuth error indicator

We broaden our view slightly and allow the Lagrange degree p to vary from element to element. By looking closely at the constants hidden in the $\lesssim$-type estimations above—some, such as (1.4.7), depend on p locally—Melenk and Wohlmuth [33] derived an error indicator akin to the residual-based error indicator, applied to this *hp*-case.

Definition 1.4.10. Given a conforming finite element triangulation $\mathcal{K}$ of Lagrange elements with local degree p_K, define the *Melenk-Wohlmuth error indicator* as

$$\eta_{\mathsf{MW}}(v,K)^2 := \frac{h_K^2}{p_K^2}\left\|P^{(p_K-1)}f+\triangle u_{\mathcal{K}}\right\|^2_{L^2(K)} + \sum_{e\subset\partial K\setminus\partial\Omega}\frac{|e|}{2p_e}\left\|[\![\nabla u_{\mathcal{K}}]\!]_e\right\|^2_{L^2(e)}, \quad (K\in\mathcal{K}) \tag{1.4.11}$$

where P^r is the $L^2(K)$-projector on $\mathbb{P}_r(K)$, $|e|$ is the length of the edge, and $p_e := \max\{p_{K_1}, p_{K_2}\}$ with K_1 and K_2 the two elements adjacent to e. $\lozenge$

Their results rely on the following assumption.

Assumption 1.4.12. The polynomial degrees of neighbouring elements satisfy

$$\exists \nu \geq 1 \text{ s.t. } \frac{p_K+1}{p_{K'}+1} \leq \nu. \qquad \lozenge$$

Under this assumption, Melenk and Wohlmuth provide us with very specific efficiency and reliability bounds, summarized in the following theorem.

Theorem 1.4.13 ([33, Thm. 3.6])**.** *Take $\varepsilon>0$. Then there exist $C_{\mathsf{rel}}, C_{\mathsf{eff}}(\varepsilon)>0$ independent of h and $\{p_K\}$ such that*

$$C_{\mathsf{rel}}\|e\|^2_{H_0^1(\Omega)} \leq \sum_{K\in\mathcal{K}} \eta_{\mathsf{MW}}(u_{\mathcal{K}},K)^2 + \frac{1}{p_K^2}\operatorname{osc}^2_{p_K-1}(K),$$

$$\eta_{\mathsf{MW}}(u_{\mathcal{K}},K)^2 \leq C_{\mathsf{eff}}(\varepsilon)p_K^{1+2\varepsilon}\left(p_K\|e\|^2_{H_0^1(S(K,\mathcal{K}))} + \frac{1}{p_K^{2-2\varepsilon}}\operatorname{osc}^2_{p_K-1}(S(K,\mathcal{K}))\right) \quad (K\in\mathcal{K}).$$

Corollary 1.4.14. *Forget for a second about the data oscillation (by, for instance, assuming $f = 1$). Then the second global efficiency bound in Theorem 1.4.13 reads as*

$$\eta_{\mathsf{MW}}(u_{\mathcal{K}}, \mathcal{K})^2 \lesssim C_{\mathsf{eff}}(\varepsilon)\|p_{\mathcal{K}}\|_\infty^{2+2\varepsilon}\|e\|^2_{H^1_0(\Omega)}, \quad \|p_{\mathcal{K}}\|_\infty := \max_{K\in\mathcal{K}} p_K.$$

In other words, the efficiency is (essentially linearly) dependent on the maximal polynomial degree.

Their numerical results suggest that this efficiency degradation is not seen in practice; however, it is mentioned in [11] that p-robustness*—that the reliability and efficiency bounds do not degrade as p increases—can not be expected of this estimator, and that the efficiency is truly dependent on p.*

1.4.4. Equilibrated flux indicator

The final error indicator we will mention in this paragraph is the *Equilibrated flux indicator*. It was proven to be p-robust in [11]. Its formulation is much more complex than the previous indicators so we will refrain from diving into it; the estimator was studied in great detail by van Venetië [46].

1.5. Grid refinement

In this section, we will focus on the two-dimensional case. Recall from Theorem 1.3.9 that given a sequence of triangulations with mesh width going to zero, we guaranteed convergence of their finite element solutions to the real solution. This section studies the refinement of triangulations as to produce such a sequence.

We will study the refinement of a triangulation by subdividing selected triangles into smaller ones. Subsequent error estimates benefit from certain properties, such as:

1. being able to refine a triangulation locally;
2. ensuring conformity when refining a conforming triangulation;
3. ensuring uniform shape regularity among all triangles in these triangulations.

Definition 1.5.1. Two triangles K' and K'' are *similar* when there is an orthogonal matrix $Q \in \mathbb{R}^{2\times 2}$, a vector $\mathbf{b} \in \mathbb{R}^2$, and a scalar $\lambda > 0$, so that

$$F(\mathbf{x}) := \lambda Q\mathbf{x} + \mathbf{b} \text{ with } F(K') = K''. \qquad \lozenge$$

Example 1.5.2. Given some conforming triangulation $\mathcal{K}$ with mesh width h, we can create a refined triangulation $\mathcal{K}^*$ with mesh width $h/2$ by connecting the midpoints of the edges of all triangles. Indeed: this will create four children triangles $\{K^i\}$ per $K \in \mathcal{K}$, each similar to its parent, with $\lambda = \frac{1}{2}$. Then each K^i must have $h_{K^i} = h_K/2$, and $\rho_{K^i} = \rho_K/2$ so that $\mathcal{K}^*$ is still uniformly shape regular, with mesh width $h/2$.

This method of connecting midpoints in a pairwise manner is easy, but it does introduce hanging nodes when refining locally, thereby hampering (p-robust) a posteriori error estimation. $\lozenge$

Of course, finding such refinements is not a goal in itself: It serves as a way to drive the approximation error down. Given some insight about the local errors, we can also focus our energy by adaptively refining only those triangles on which the error is large. This is not trivial: In the previous example, subdividing just one triangle into its four children introduces hanging nodes. This in itself is no disaster, but efficiency and reliability bounds are dependent on the (maximum) number of hanging nodes per edge, so p-robust error indicators require an upper bound on this number. We restrict ourselves to the simplest case by requiring that no hanging nodes appear anywhere; this is in line with our earlier definition of a conforming triangulation: a triangulation *with no hanging nodes*.

We will now provide a method that allows meeting all three criteria above.

Definition 1.5.3. Given a triangle $K := \text{hull}(\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3)$, we can select one of its vertices as the *newest vertex* $v(K)$. The edge opposite this newest vertex is called its *refinement edge* $e(K)$.

Newest vertex bisection of a triangle K works by connecting $v(K)$ to the middle point $\mathbf{p}$ of $e(K)$, resulting in two triangles K^1 and K^2. For both child triangles, we select $v(K^1) := \mathbf{p} =: v(K^2)$ as their newest vertex.

See Figure 1.5.4 for an illustration. ◊

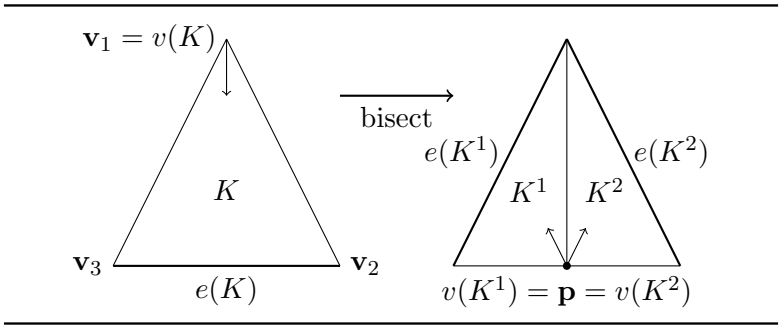


Figure 1.5.4.: Newest vertex bisection. Left: A triangle K with newest vertex $v(K)$ (signified by the arrow) and bisection edge $e(K)$. Right: The two child triangles created after bisecting K; the newly created vertex $\mathbf{p}$ is the newest vertex of both children.

Definition 1.5.5. Let $\mathcal{K}_0$ be an initial conforming triangulation of Ω. Define $\mathbb{K}$ to be the collection of all (not necessarily conforming) triangulations of Ω generated from $\mathcal{K}_0$ by means of newest vertex bisection. The set $\mathfrak{K} := \cup_{\mathcal{K}\in\mathbb{K}}\mathcal{K}$ of all triangles in $\mathbb{K}$ can then be interpreted as a multiple-rooted infinite binary tree, each root being a triangle in $\mathcal{K}_0$.

A set $\mathcal{T} \subset \mathfrak{K}$ is a *subtree* when

i) $\mathcal{T}$ has a finite number of nodes;

ii) $\mathcal{T}$ contains all roots of $\mathfrak{K}$;

iii) when $K \in \mathcal{T}$ is not a root of $\mathfrak{K}$, its parent and sibling are both in $\mathcal{T}$ as well.

The nodes of $\mathcal{T}$ with no children are called *leaves*, which we will denote by $\mathcal{L}(\mathcal{T})$.

With this definition, there is a one-to-one mapping between triangulations $\mathcal{K} \in \mathbb{K}$ and the leaves of subtrees $\mathcal{T}_\mathcal{K}$ of $\mathfrak{K}$. See Figure 1.5.6. ◊

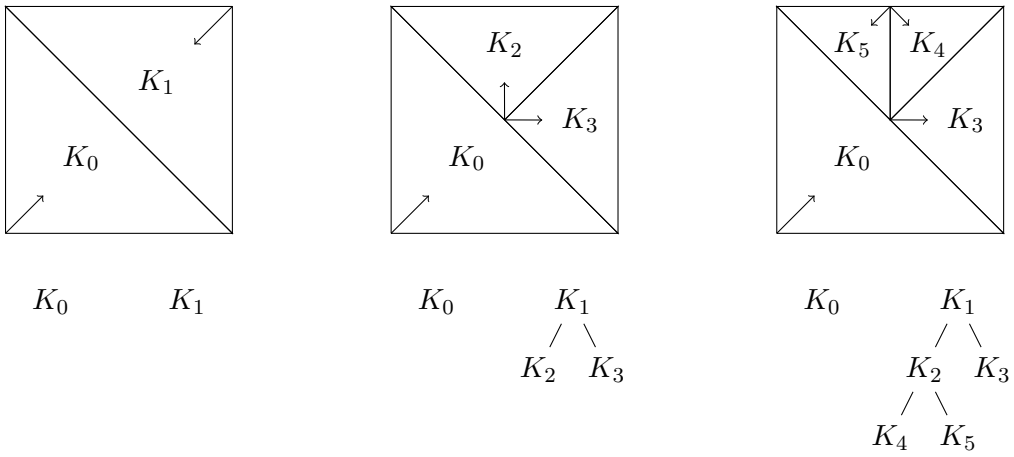


Figure 1.5.6.: Three triangulations of the domain $\Omega := (0,1)^2$. Top: The triangulations $\mathcal{K}$ seen as elements of $\mathbb{K}$. Bottom: the triangulations seen as the leaves of subtrees $\mathcal{T}_{\mathcal{K}}$.

Theorem 1.5.7 ([34])**.** *All triangles $K \in \mathfrak{K}$ are uniformly shape regular.*

Proof (sketch). By virtue of the newest vertex bisection, any descendant of $K \in \mathcal{K}_0$ inside $\mathfrak{K}$ is in one of four equivalence classes ($K \sim L$ when K and L are similar). There is a finite number of triangles in $\mathcal{K}_0$ hence a finite number of equivalence classes in total, so $\mathfrak{K}$ is uniformly shape regular. □

Definition 1.5.8. For $\mathcal{K}, \mathcal{K}' \in \mathbb{K}$, we will say that $\mathcal{K}' \geq \mathcal{K}$ when $\mathcal{K}'$ is a *refinement* of $\mathcal{K}$, i.e., $\mathcal{K} \subset \mathcal{K}'$ when viewed as subtrees of $\mathfrak{K}$. ♢

Definition 1.5.9. Note again that the triangulations $\mathcal{K} \in \mathbb{K}$ are not necessarily conforming; take as an example the middle triangulation of Figure 1.5.6.

We can define the subset $\mathbb{K}_c \subset \mathbb{K}$ of *conforming* triangulations. We will see that both sets $\mathbb{K}$ and $\mathbb{K}_c$ play a big role in the next chapters; for now, we will focus mainly on conforming triangulations $\mathcal{K} \in \mathbb{K}_c$. ♢

Lemma 1.5.10 ([44, Thm. 4.3])**.** *If $\mathcal{K} \in \mathbb{K}_c$ is a conforming triangulation, then recursively performing two refinements on every triangle—two* uniform refinements*—yields a triangulation $\mathcal{K}' \geq \mathcal{K}$ that is conforming. In other words, $\mathcal{K}' \in \mathbb{K}_c$ as well.*

Definition 1.5.11. A conforming triangulation $\mathcal{K} \in \mathbb{K}_c$ is said to satisfy the *matching condition* when for each triangle $K \in \mathcal{K}$, its refinement edge is either on the boundary of the domain, or its neighbour along this edge *also* has its refinement edge there. See Figure 1.5.14(a) and (b) for two examples of a matching and a non-matching triangulation. ♢

Theorem 1.5.12 ([10, Lem. 2.1])**.** *On any conforming triangulation, one can select the newest vertices in such a way that the matching condition is satisfied.*

Remark 1.5.13 ([10, p. 229])**.** Even though existence of such a selection is ensured, the proof is not constructive and no efficient algorithm is known. The usual solution is to perform two uniform refinements—yielding a conforming triangulation; cf. Lemma 1.5.10—and relabeling

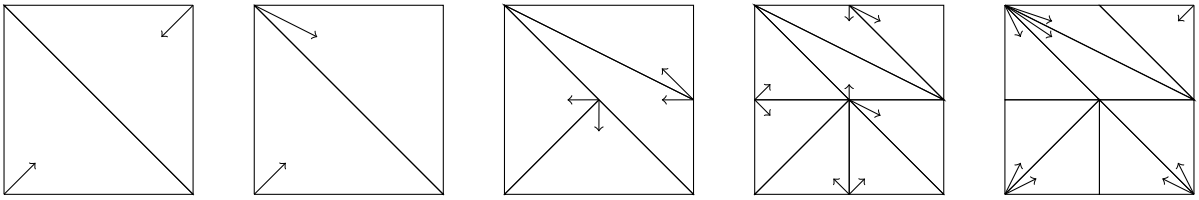


Figure 1.5.14.: Understanding the matching condition. Arrows signify newest vertices. (a) a matching triangulation; (b) a non-matching triangulation $\mathcal{K}_1$; (c) bisecting each triangle in $\mathcal{K}_1$ to get $\mathcal{K}_2$; (d) bisecting each triangle in $\mathcal{K}_2$ to get $\mathcal{K}_3$; (e) relabeling newest vertices in $\mathcal{K}_3$ to get a matching triangulation $\mathcal{K}$.

the vertices in accordance with Figure 1.5.14 by creating pairs of elements that share a refinement edge. ♢

Assumption 1.5.15. From this point on, we will assume any initial triangulation to be matching. ♢

We will end this section with two important algorithms; the first will play a central role in the analysis of *h*-AFEM, while the second is invaluable for *hp*-AFEM.

Theorem 1.5.16 ([43],[10, Lem. 2.5])**.** *Assume $\mathcal{K}_0 \in \mathbb{K}_c$ is matching (see above assumption).*

For each conforming $\mathcal{K} \geq \mathcal{K}_0$, Algorithm 1.5.17(a) terminates and yields the smallest conforming refinement of $\mathcal{K}$ in which a given $K \in \mathcal{K}$ is bisected.

If $\mathbb{K}_c \not\ni \mathcal{K} \geq \mathcal{K}_0$ is not *conforming, but still found from $\mathcal{K}_0$ by means of newest vertex bisection, Algorithm 1.5.17(b) can be used to find the smallest conforming refinement*

$$\mathcal{C}(\mathcal{K}) := \arg\min \left\{ \#\tilde{\mathcal{K}} : \mathbb{K}_c \ni \tilde{\mathcal{K}} \geq \mathcal{K} \right\} \in \mathbb{K}_c$$

of $\mathcal{K}$. It holds that $\#\mathcal{C}(\mathcal{K}) \lesssim \#\mathcal{K}$.

1.6. *h*-AFEM

We can use this notion of local refinements to formulate an adaptive finite element method when using Lagrange elements of fixed degree. We utilize the local a posteriori error indicators $\eta(u_{\mathcal{K}}, K)$ to mark *those* elements with large indicated error and refine those into a new conforming triangulation, hoping for a large reduction of the total error. It can be summarized in the following loop:

$$\texttt{SOLVE} \to \texttt{ESTIMATE} \to \texttt{MARK} \to \texttt{REFINE}. \tag{1.6.1}$$

In more detail:

SOLVE We construct a Galerkin solution u_k from the triangulation $\mathcal{K}_k$ using the technique developed in §1.2.2.

```
function Refine(𝒦 conforming, K ∈ 𝒦)
    if e(K) ⊂ ∂Ω then
        bisect K, and update 𝒦;
        return 𝒦.

    K′ is the neighbour of K along e(K);
    if e(K′) = e(K) then
        bisect K and K′, and update 𝒦;
    else
        𝒦 := Refine(𝒦, K′);
        K̃ := child of K′ s.t. e(K̃) = e(K);
        bisect K and K̃, and update 𝒦;
    return 𝒦.
```

```
function MakeConform(𝒦)
    M := {};
    for all K ∈ 𝒦 do
        if K has hanging vertex then
            put K into M;
    while M ≠ ∅ do
        extract some K from M;
        bisect K and update 𝒦;
        for both children K′ of K do
            if K′ has hanging vertex then
                put K′ into M;
        for all neighbours K′ of K do
            if K′ has hanging vertex then
                put K′ into M;
    return 𝒦.
```

Algorithm 1.5.17: The Refine and MakeConform procedures for finding the smallest conforming refinement of a conforming resp. non-conforming triangulation.

ESTIMATE We then estimate the local error this Galerkin solution makes with the true solution. In estimating the local error, it is useful to use an efficient and reliable error indicator. For h-AFEM, the residual-based error indicator of §1.4.2 is often used.

MARK Selecting a minimal set $\mathcal{M}_k \subset \mathcal{K}_k$ for which $\eta(u_k, \mathcal{M}_k) \geq \theta\eta(u_k, \mathcal{K}_k)$ can simply be achieved by selecting elements from large to small indicators. Marking a set that captures a fraction θ of the error is called *bulk chasing*; marking a minimal set is called a *Dörfler marking*.

REFINE We define the next triangulation by refining all marked elements. By Theorem 1.5.16, the Refine-step produces the smallest conforming refinement in which all $K \in \mathcal{M}_k$ are bisected.

The algorithm

The loop above yields h-AFEM in Algorithm 1.6.2. The algorithm iteratively computes (conforming) triangulations $\mathcal{K}_k$, and Galerkin solutions $u_{\mathcal{K}_k}$. One can prove that under certain mild conditions, the solutions produced by h-AFEM show an optimal convergence rate to the real solution—we will look into this shortly.

h-AFEM converges with the best possible rate

We will end this section with a classical result by Stevenson [43].

Definition 1.6.3. Let $\Omega \subset \mathbb{R}^2$ be a domain. Take $\mathcal{K}_0$ an initial (conforming) triangulation.

```
1: procedure h-AFEM(𝒦_0, θ ∈ (0,1], ε > 0)
2:     for all k ∈ ℕ_0 do
3:         // SOLVE
4:         solve u_k ∈ V(𝒦_k) from (1.1.7);
5:         // ESTIMATE
6:         compute {η(u_k, K) : K ∈ 𝒦_k};
7:         if η(u_k, 𝒦_k) < ε then
8:             return
9:         // MARK
10:        select a smallest ℳ_k ⊂ 𝒦_k with
11:          η(u_k, ℳ_k) ≥ θη(u_k, 𝒦_k);
12:        // REFINE
13:        while ℳ_k ∩ 𝒦_k ≠ ∅ do
14:            Take K ∈ ℳ_k ∩ 𝒦_k;
15:            𝒦_k := Refine(𝒦_k, K);
16:
17:        𝒦_{k+1} := 𝒦_k.
```

Algorithm 1.6.2: The h-adaptive finite element method as described in [43].

For $s > 0$, define the approximation class

$$\mathcal{A}_s := \left\{ u \in H^1_0(\Omega) : \triangle u \in L^2(\Omega), \quad |u|_{\mathcal{A}_s} < \infty \right\},$$
$$|u|_{\mathcal{A}_s} := \sup_{N \in \mathbb{N}} (N+1)^s \min_{\{\mathcal{K} \in \mathbb{K}_c : \#\mathcal{K} - \#\mathcal{K}_0 \leq N\}} \mathcal{E}(\mathcal{K})$$

where $\mathcal{E}(\mathcal{K})$ was the total error on $\mathcal{K}$. ♢

Remark 1.6.4. When $u \in \mathcal{A}_s$, the *best* conforming triangulation $\mathcal{K}_N$ with $\#\mathcal{K}_0 + N$ triangles has a total error $\mathcal{E}(\mathcal{K}_N) \leq (\#\mathcal{K}_N + 1)^{-s} |u|_{\mathcal{A}_s}$. In other words, when $u \in \mathcal{A}_s$, there is a sequence of conforming triangulations $(\mathcal{K}_N)_N$ for which the total error decays with rate s. ♢

Definition 1.6.5. An a posteriori error indicator η provides *discrete reliability* when given two conforming triangulations—one a refinement of the other—we can bound the norm of the difference between the two discrete solutions by the indicators on the refined elements.

More precisely, select $\mathcal{K} \in \mathbb{K}_c$, and take some $\mathbb{K}_c \ni \mathcal{K}' \geq \mathcal{K}$. Then $\mathcal{K} \setminus \mathcal{K}'$ is the set of elements that were refined going from $\mathcal{K}$ to $\mathcal{K}'$. An error indicator shows discrete reliability when there is a constant C_{dr} such that

$$\|u_{\mathcal{K}'} - u_{\mathcal{K}}\|^2_{H^1_0(\Omega)} \leq C_{\mathsf{dr}} \eta(u_{\mathcal{K}}, \mathcal{K} \setminus \mathcal{K}')^2 \quad (\mathcal{K}, \mathcal{K}' \in \mathbb{K}_c \text{ with } \mathcal{K}' \geq \mathcal{K}). \tag{1.6.6}$$

♢

Remark 1.6.7. Discrete reliability is a strong property: We are able to bound the norm of two functions whose difference is nonzero globally using a few pieces of local information. It is no surprise that, in the process of proving discrete reliability, the proof of regular "continuous" reliability usually follows (with the same constant). ♢

Theorem 1.6.8 ([46])**.** *The above discrete reliability bound is satisfied for both the Melenk-Wohlmuth and the equilibrated flux estimators described in §1.4.*

Theorem 1.6.9 ([42, Thm. 11.7]). *Let η be an a posteriori error indicator with efficiency constant C_{eff}, and discrete reliability constant C_{dr}. Ensure that the marking parameter θ satisfies $\theta^2 < (C_{\mathsf{eff}}(C_{\mathsf{dr}}+1))^{-1}$. If $u \in \mathcal{A}_s$ for some $s > 0$, then for the sequence of triangulations $(\mathcal{K}_k)_k$ produced by h-AFEM,*

$$\#\mathcal{K}_k - \#\mathcal{K}_0 \lesssim |u|_{\mathcal{A}_s}^{1/s}\, \mathcal{E}(\mathcal{K}_k)^{-1/s}.$$

This statement contains a lot of information. Taking the parameter θ small enough, Theorem 1.6.9 ensures that if for a *best* sequence of triangulations $(\mathcal{K}_N)_N$ the total error decays algebraically with rate s, then this error decay also holds for the sequence $(\mathcal{K}_k)_k$ produced by h-AFEM. In other words, h-AFEM converges with the best possible rate allowed by the polynomial degree.

Remark 1.6.10. It is even more remarkable that while the best sequence $(\mathcal{K}_N)_N$ is not (necessarily) nested, the sequence $(\mathcal{K}_k)_k$ produced by h-AFEM *is*, and it is still optimal! ♢

2. Theory of hp-adaptivity

In the previous chapter, we derived a framework for the theory of finite elements, and we discussed its main brainchild: the h-adaptive finite element method. Moreover, we found that under certain conditions, this algorithm converges with optimal (algebraic) rate, in that

$$\text{approximation error norm} \sim N^{-s} \quad \text{for the best possible} \quad s > 0,$$

where N equals the size of the triangulation. This N is (proportional to) the size of the linear system one solves to find the Galerkin solution.

As it turns out, extending this framework allows us to devise algorithms with even better convergence behaviour. Analogous to the previous chapter, we will focus our efforts on solving the variational formulation of the Poisson problem on a polygonal domain in two dimensions.

Moreover, to focus on the central elements of the theory, we will omit any data oscillation by assuming the forcing function f is piecewise smooth (e.g., polynomial) on the domain. It must be noted that the theory also applies when the forcing function is arbitrary in $L^2(\Omega)$.

An *hp-adaptive finite element method* generalizes classical finite elements in the way elements can be refined: Instead of just allowing h-refinement by subdivision, we also allow p-enrichment by expanding the local shape space. This allows for faster convergence rates than with pure h-refinement.

In this chapter, we will describe the hp-adaptive FEM of Canuto *et al.* [15] from 2015, driven by efforts from Binev [9]. With this new algorithm, it is possible to find exponential convergence:

$$\text{approximation error norm} \sim e^{-\eta N^{\tau}} \quad \text{for some} \quad \eta, \tau > 0,$$

where N denotes the size of the system. We will see later in this thesis that this exponential decay is actually seen in practice.

Before the result of Stevenson [43] proving optimality of the strictly refining algorithm h-AFEM, provably optimal h-adaptive finite element methods required the incidental use of a *coarsening routine* in which the current triangulation is made less refined. Noting that each (conforming) triangulation $\mathcal{K}$ induces a finite element search space $V(\mathcal{K})$, a coarsening strategy essentially throws away nearly redundant degrees of freedom to arrive at a subspace of $V(\mathcal{K})$ with favourable properties—*near-best*, for some definition of best. We will not dive into these h-adaptive coarsening strategies, but rather use them as a stepping stone for hp-adaptivity.

The current state of provable hp-adaptive FEMs is comparable to what the landscape for h-adaptivity looked like before Stevenson [43]: The hp-AFEM method proposed in [15] is provably optimal, but it requires intermediate "coarsening" steps between refinements. Coarsening should be understood in the more general *near-best* sense: The induced finite element space of such a "coarsened" triangulation has a dimensionality $\leq \dim V(\mathcal{K})$. It is not necessarily a subspace of $V(\mathcal{K})$, but it does throw away 'near'-redundancies, thereby making the triangulation more efficient. A triangulation with a single triangle of degree 800 can, for example, be "coarsened" to a near-best one with two triangles, each carrying degree 1.

Of course, such a near-best triangulation might very well increase the error norm (a little) in return for the gain in efficiency, with efficiency understood in that the number of degrees of freedom is decreased significantly. Making one or a couple of refinements (a Reduce step) drives the error norm back down at the cost of efficiency, opening the door for another round of

near-best approximation. This juggling of two routines is the main characteristic of *hp*-AFEM, which can be expressed in the following loop:

$$\underbrace{\mathsf{Refine} \to \cdots \to \mathsf{Refine}}_{\mathsf{Reduce}} \to \mathsf{NearBest}.$$

This chapter will first extend our definitions to suit *hp*-adaptivity in §2.1. In §2.2, we will apply the near-best tree generation theory of appendix A to our finite element context by building near-best triangulations. Section 2.3 will be devoted to an in-depth formulation of the Reduce step. With this, all the puzzle pieces are in place to formulate *hp*-AFEM in §2.4, and prove its convergence and optimality.

Near-best approximations

We will develop a near-best approximation with the following abstract interpretation. Let $(V,\|\cdot\|)$ be a normed space. Assume we are looking for some unknown $u \in V$, and that any approximation $\tilde{u}$ of u carries a *complexity* $\mathcal{C}(\tilde{u})$—number of degrees of freedom. Naturally, the lower this complexity is, the more desirable and *efficient* the approximation is.

Suppose our current approximation of u is $\tilde{u}$. Say we know it to be within some tolerance ε from u, in that

$$\|u - \tilde{u}\| \leq \varepsilon.$$

Given $\delta > \varepsilon$, our "coarsening" strategy computes an approximation m of the *known* $\tilde{u}$ satisfying

$$\|m - \tilde{u}\| \leq \delta \quad \text{and} \quad \mathcal{C}(m) \leq B \inf_{\{v \,:\, \|\tilde{u}-v\| \leq \delta\}} \mathcal{C}(v) \tag{2.0.1}$$

for some constant $B > 1$. In other words, m is (modulo B) the most efficient approximation of $\tilde{u}$ within tolerance δ. Then, with

$$z := \operatorname*{arg\,inf}_{\{v \,:\, \|u-v\| \leq \delta-\varepsilon\}} \|u - v\|,$$

we see that

$$\|z - \tilde{u}\| \leq \|z - u\| + \|u - \tilde{u}\| < \delta - \varepsilon + \varepsilon = \delta \implies B\mathcal{C}(z) \geq \mathcal{C}(m),$$

hence m is (modulo B) *more efficient* than any approximation of u within tolerance $\delta - \varepsilon$. In other words, m has *near-best* complexity.

Usually, we will pick δ in terms of ε. For instance, take $\delta = 2\varepsilon$. Given an approximation $\tilde{u}$ of u with $\|u - \tilde{u}\| \leq \varepsilon$, our routine finds an approximation m of $\tilde{u}$ for which

$$\|m - \tilde{u}\| \leq \varepsilon, \quad \|m - u\| \leq 3\varepsilon, \quad \mathcal{C}(m) \leq B\mathcal{C}(z) \quad \forall z \text{ with } \|z - u\| \leq \varepsilon.$$

So in going from $\tilde{u}$ to w, we sacrifice error norm (the upper bound 3ε for $\|m - u\|$ exceeds the one—being ε—for $\|\tilde{u} - u\|$) but gain near-best complexity.

2.1. A framework for *hp*-adaptivity

In *h*-finite elements, it sufficed to restrict ourselves to triangulations that were *conforming*, cf. Def. 1.2.13. The framework we will build for *hp*-finite elements however relies on intermediate triangulations of the domain that are nonconforming.

Definition 2.1.1. Recall the multiple-rooted infinite binary tree $\mathfrak{K}$ defined in definition 1.5.5. We will define an *hp-element* as being a tuple $D := (K_D, d_D) \in \mathfrak{K} \times \mathbb{N}$, and call d_D its *local complexity*. The associated finite element $(K_D, \mathcal{P}(D), \mathcal{N}(D))$ will possess a shape space $\mathcal{P}(D)$ of dimension $\eqsim d_D$—usually polynomials of certain degree. Its set of local degrees of freedom $\mathcal{N}(D)$ are unimportant to this chapter; example *hp*-elements will be studied in Chapter 3. ◊

An *hp*-element can have any local complexity $d_D \in \mathbb{N}$. However, in our two-dimensional application, not every $d_D \in \mathbb{N}$ corresponds with the dimensionality of a full polynomial space. It is much easier to work with full polynomial spaces only, so we will overcome this discrepancy through the following construction.

Definition 2.1.2. Recall that the dimensionality of the space of degree-p polynomials equals

$$\dim \mathbb{P}_p(K) = \#\mathcal{I}^p = \binom{p+2}{2}$$

where $\#\mathcal{I}^p$ is the pth triangle number defined in (1.2.9). For any $d \in \mathbb{N}$, the largest p such that $\#\mathcal{I}^p \leq d$ is defined as

$$p(d) := \arg\max \{p \in \mathbb{N}_0 : \#\mathcal{I}^p \leq d\};$$

the dimension of the space spanned by polynomials of this degree is then

$$\lfloor d \rfloor_\triangle := \#\mathcal{I}^{p(d)}, \quad (d \in \mathbb{N}).$$

The mapping $\lfloor d \rfloor_\triangle$ essentially floors d onto the set of triangle numbers.

The first few values of d, $p(d)$, and $\lfloor d \rfloor_\triangle$ are given in Table 2.1.3. ◊

Quantity	Values														
d	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$p(d)$	0	0	1	1	1	2	2	2	2	3	3	3	3	3	4
$\lfloor d \rfloor_\triangle$	1	1	3	3	3	6	6	6	6	10	10	10	10	10	15

Table 2.1.3.: The first few values of $d, p(d)$, and $\lfloor d \rfloor_\triangle$.

In our application, the *hp*-element D is a finite element on K_D equipped with the polynomials of degree $p_D := p(d_D)$, having dimension $\lfloor d_D \rfloor_\triangle \eqsim d_D$.

Definition 2.1.4. We will call a collection $\mathcal{D}$ of *hp*-elements an *hp-triangulation* when the set of its element domains $\mathcal{K}(\mathcal{D}) := \{K_D : D \in \mathcal{D}\} \in \mathbb{K}$ is a triangulation of Ω, and a *conforming hp-triangulation* when $\mathcal{K}(\mathcal{D}) \in \mathbb{K}_c$ is conforming. The set of all *hp*-triangulations is denoted $\mathbb{D}$, and the set of all conforming ones $\mathbb{D}_c$. ◊

Definition 2.1.5. We define the *broken hp-approximation space* as

$$V_b(\mathcal{D}) := \big\{v : \Omega \to \mathbb{R} : v|_{K_D} \in \mathcal{P}(D) \quad (D \in \mathcal{D})\big\} \quad (\mathcal{D} \in \mathbb{D})$$

and the *complexity* of an *hp*-triangulation as

$$\#\mathcal{D} := \sum_{D\in\mathcal{D}} d_D. \tag{2.1.6}$$

This complexity is $\eqsim$ the dimensionality of $V_b(\mathcal{D})$. ◊

Remark 2.1.7. We call $V_b(\mathcal{D})$ *broken* because its functions need not be continuous between two elements. If $\mathcal{D}$ is a conforming *hp*-triangulation of elements, then the space

$$V(\mathcal{D}) := V_b(\mathcal{D}) \cap H^1_0(\Omega) = \Big\{f \in H^1_0(\Omega) : f|_{K_D} \in \mathbb{P}_{p_D}(K_D), \quad f|_{\partial\Omega} = 0\Big\}$$

coincides with the finite element space defined in Definition 1.2.15, by virtue of Theorem 1.2.16. ◊

Definition 2.1.8. Analogous to the *h*-case, we say that $\tilde{\mathcal{D}} \geq \mathcal{D}$—$\tilde{\mathcal{D}}$ *refines* $\mathcal{D}$—when $V_b(\tilde{\mathcal{D}}) \supset V_b(\mathcal{D})$, or equivalently, the following two conditions hold:

$$\begin{cases} \mathcal{K}(\tilde{\mathcal{D}}) \geq \mathcal{K}(\mathcal{D}), \\ d_{\tilde{D}} \geq d_D \text{ for all } D \in \mathcal{D}, \tilde{D} \in \tilde{\mathcal{D}} \text{ with } K_{\tilde{D}} \subset K_D. \end{cases}$$ ◊

2.2. Near-best approximations

In appendix A, we give an overview of the theory of near-best *hp*-subtree generation developed by Binev [9]. For completeness, we repeat its main result here.

Take any abstract *error mapping* $e : \mathfrak{K} \times \mathbb{N} \to \mathbb{R} : D = (K_D, d_D) \mapsto e_D$ that is decreasing under *h*-refinement and *p*-enrichment. Then, define the *global hp-error* as the sum of local errors

$$E_{\mathcal{D}} := \sum_{D\in\mathcal{D}} e_D.$$

The *best N-term hp-adaptive approximation error* σ_N^{hp} is defined as

$$\sigma_N^{hp} := \inf_{\{\mathcal{D} \text{ } hp\text{-triangulation} \,:\, \#\mathcal{D}\leq N\}} E_{\mathcal{D}}.$$

Theorem 2.2.1 (See also Thm. A.3.12)**.** *The routine* Near-best *hp-subtree of Algorithm A.3.9 produces hp-triangulations $\mathcal{D}_N$ of complexity $\#\mathcal{D}_N = N$ that provide near-best hp-approximation in the sense that*

$$E_{\mathcal{D}_N} \leq \frac{2N-1}{N-n+1}\sigma_n^{hp}$$

for any $n \leq N$. One can derive that the hp-error of $\mathcal{D}_N$ is better than four times the error on the best *hp-triangulation with half the complexity.*

In this section, we will apply Algorithm A.3.9 in a finite element setting. The error mapping will, in our case, depend on the Galerkin solution $u_{\mathcal{D}}$ on some refined *hp*-triangulation $\mathcal{D}$.

Given some *hp*-element $D := (K_D, d_D)$, it measures the squared distance between $u_{\mathcal{D}}|_{K_D}$ and its best approximation from $\mathbb{P}_{p_D}(K_D)$. To ensure that this best approximation is unique, we measure this distance in a special norm; see the following construction.

Definition 2.2.2. For a Banach space $(V,\|\cdot\|_V)$, and a closed subspace $W \subset V$, we can define a similarity

$$v \sim w \iff v - w \in W \quad (v, w \in V).$$

Let $[v]_W$ be the equivalence class of v. With this, we can define the *quotient space* V/W as

$$V/W := \left\{ [v]_W : v \in V \right\}.$$

This is a linear space, even Banach, with norm

$$\left\| [v]_W \right\|_{V/W} := \min_{w \in W} \|v - w\|_V . \qquad \Diamond$$

Lemma 2.2.3. *Consider the case $V := H^1(K_D)$, and $W := \mathbb{P}_0(K_D)$. Then*

$$\left\| [v]_W \right\|_{V/W} \eqsim |v|_V \quad (v \in V).$$

In other words, $|\cdot|_V$ is a norm on V/W, equivalent to the quotient norm.

Proof. On the one hand, with $w \in W$ the minimizer of $\|v - w\|_V$, we see that

$$|v|_V = |v - w|_V \leq \|v - w\|_V = \left\| [v]_W \right\|_{V/W} .$$

By Friedrichs' inquality (cf. [13, (4.3.14)]),

$$\|v - \overline{v}\|_{L^2(K_D)} \lesssim |v|_V \quad \text{where} \quad \overline{v} := \int_{K_D} v,$$

so that on the other hand, we have

$$\begin{aligned} \left\| [v]_W \right\|_{V/W} &= \min_{w \in W} \sqrt{|v - w|_V^2 + \|v - w\|_{L^2(K_D)}^2} \\ &= \min_{w \in W} \sqrt{|v|_V^2 + \|v - w\|_{L^2(K_D)}^2} \\ &\leq \sqrt{|v|_V^2 + \|v - \overline{v}\|_{L^2(K_D)}^2} \\ &\lesssim |v|_V . \end{aligned}$$

□

We are now ready to define the error functional.

Definition 2.2.4. Consider the quotient space $H^1(K_D)/\mathbb{P}_0(K_D)$ of equivalence classes of $H^1(K_D)$-functions that differ by a constant. In light of the previous lemma, define the functional

$$e_D : H^1(K_D) \to \mathbb{R} : v \mapsto |v - P^{p_D} v|_{H^1(K_D)}^2 \tag{2.2.5}$$

where P^p is the orthogonal projector onto the subspace $\mathbb{P}_p(K_D)/\mathbb{P}_0(K_D)$. For fixed v, the mapping $D \mapsto e_D(v)$ is an error mapping in the earlier definition.

The quantity $e_D(v)$ then measures the squared best approximation error of v from $\mathbb{P}_{p_D}(K_D)$ in quotient norm. $\Diamond$

Remark 2.2.6. The notion of an *error functional* is completely unrelated to the *error estimators* discussed in §1.4; the latter plays an important role in finite element analysis as a whole, whereas the former is only useful inside the current near-best setting. $\Diamond$

Definition 2.2.7. See Definition A.1.3. The global *hp*-error functional induced by our error functional is defined on $H_0^1(\Omega)$ and reads

$$E_{\mathcal{D}} : H_0^1(\Omega) \to \mathbb{R} : v \mapsto \sum_{D \in \mathcal{D}} e_D(v)$$

which measures the squared distance between v and the space $V_b(\mathcal{D})$. We will often refer to $E_{\mathcal{D}}(v)$ as the *(squared) broken error* with respect to v. ◊

Note that $E_{\mathcal{D}}(v)$ implicitly defines a function $m(v, \mathcal{D}) \in V_b(\mathcal{D})$ being the piecewise polynomial that minimizes (2.2.5) locally on each element. This function is, in general, not continuous. We will look at explicitly constructing m in Chapter 4.

In light of (2.0.1) in the introduction of this chapter, we want to find, given a known $v \in H_0^1(\Omega)$ and tolerance $\varepsilon > 0$, a near-best triangulation $\mathcal{D}_{\mathsf{NB}}$ with $E_{\mathcal{D}_{\mathsf{NB}}}(v) \leq \varepsilon^2$.

Remark 2.2.8. Let us look at a short motivation for the coice of error functional. If the minimizer $m(v, \mathcal{D}_{\mathsf{NB}})$ *was* continuous (and vanished on $\partial\Omega$), then $E_{\mathcal{D}_{\mathsf{NB}}}(v)$ would equal $\left\|v - m(v, \mathcal{D}_{\mathsf{NB}})\right\|^2_{H_0^1(\Omega)}$ which is exactly the case described in the introduction. ◊

This near-best triangulation $\mathcal{D}_{\mathsf{NB}}$ is provided by Algorithm 2.2.12 below. It requires the following result.

Proposition 2.2.9. *For each fixed v, the error mapping $e_D(v)$ is decreasing under both p-enrichment and h-refinement.*

Proof. h-**refinement**: Let D^1, D^2 be children of D with $d_{D^1} = d_{D^2} = d_D$. We see that

$$\mathbb{P}_{p_D}(K_D) \subset \prod_{k=1}^{2} \mathbb{P}_{p_{D^k}}(K_{D^k}) \implies e_{D^1}(v) + e_{D^2}(v) \leq e_D(v).$$

p-**enrichment**: Let D' be so that $K_{D'} = K_D$ and $d_{D'} \geq d_D$. Then

$$\mathbb{P}_{p_D}(K_D) \subset \mathbb{P}_{p_{D'}}(K_D) \implies e_{D'}(v) \leq e_D(v). \qquad \square$$

To make the near-best subtree generation algorithm useful in *hp*-adaptive finite element context, we need to address a few issues.

1. Near-best *hp*-subtree of Algorithm A.3.9 produces *hp*-triangulations with a specified *complexity*, whereas our desire is to get the *hp-error* below a certain tolerance;
2. Moreover, the algorithm produces *hp*-triangulations that are in general *nonconforming*, whereas we desire a *conforming hp*-triangulation to find a Galerkin solution;
3. Lastly, it drives the broken *hp*-error $E_{\mathcal{D}}$ down, whereas in finite element context, a result in terms of the $H_0^1(\Omega)$-seminorm is more natural.

We will handle each point separately.

2.2.1. Minimizing global *hp*-error and *hp*-NearBest

The following result offers a solution to the first point above.

Corollary 2.2.10 ([15, Cor. 3.1])**.** *In light of the near-best result of Theorem 2.2.1, we know that for each $\varepsilon > 0$, there is a smallest $N \in \mathbb{N}$ such that the hp-triangulation $\mathcal{D}_{\mathsf{NB}} := \mathcal{D}_N$ produced by* near-best hp-subtree *of Algorithm A.3.9 satisfies*

$$E_{\mathcal{D}_{\mathsf{NB}}}^{1/2} \leq \varepsilon.$$

Moreover, with $B > 1$, there is a $b = b(B) \in (0,1)$ such that

$$\#\mathcal{D}_{\mathsf{NB}} \leq B\#\tilde{\mathcal{D}} \quad \forall \tilde{\mathcal{D}} \text{ with } E_{\tilde{\mathcal{D}}}^{1/2} \leq b\varepsilon. \tag{2.2.11}$$

In words, the broken error is less than the prescribed tolerance ε, and its complexity is bounded by B times the complexity of any hp-triangulation that realizes a tolerance $b\varepsilon$.

Proof. The first property is satisfied by definition of N. We will show that the second condition of (2.2.11) holds. Define

$$b := \sqrt{\frac{1}{2}\left(1 - \frac{1}{B}\right)}.$$

If $N = 1$, then *any* $\tilde{\mathcal{D}}$ must have $\#\mathcal{D}_{\mathsf{NB}} = 1 \leq B\#\tilde{\mathcal{D}}$ so it holds trivially. Assume $N > 1$; we will proceed by contradiction.

Suppose there is a $\tilde{\mathcal{D}} \in \mathbb{D}$ with $E_{\tilde{\mathcal{D}}}(v) \leq b^2\varepsilon^2$, but $N = \#\mathcal{D} > B\#\tilde{\mathcal{D}}$. Then, take $n := \#\tilde{\mathcal{D}}$. Note that $N > Bn$, so

$$1 - \frac{1}{B} < \frac{N-n}{N}.$$

Moreover, $n \leq N - 1$ even, so that for the *hp*-triangulation $\mathcal{D}_{N-1}$ in the previous iteration,

$$\begin{aligned} E_{\mathcal{D}_{N-1}} &\leq \frac{2(N-1)}{N-1-n+1}\sigma_n^{hp} \leq \frac{2(N-1)}{N-n}E_{\tilde{\mathcal{D}}} \\ &\leq \frac{2(N-1)}{N-n}\frac{1}{2}\left(1-\frac{1}{B}\right)\varepsilon^2 = \frac{N-1}{N-n}\left(1-\frac{1}{B}\right)\varepsilon^2 \\ &< \frac{N-1}{N-n}\frac{N-n}{N}\varepsilon^2 = \frac{N-1}{N}\varepsilon^2 < \varepsilon^2. \end{aligned}$$

In other words, the first property is even satisfied for $\mathcal{D}_{N-1}$. But we assumed N was the first iteration for which it was satisfied; therefore, any $\tilde{\mathcal{D}} \in \mathbb{D}$ with $E_{\tilde{\mathcal{D}}}(v) \leq b^2\varepsilon^2$ must have $\#\mathcal{D}_{\mathsf{NB}} \leq B\#\tilde{\mathcal{D}}$. □

Algorithm 2.2.12 (*hp*-NearBest)**.** In light of the above corollary, define *hp*-NearBest as the algorithm that produces the near-best *hp*-triangulation $\mathcal{D}_{\mathsf{NB}}$ given $v \in H_0^1(\Omega)$ and $\varepsilon > 0$. ♢

2.2.2. Smallest conforming refinement

We prefer to construct our Galerkin approximations on conforming triangulations, but *hp*-NearBest offers us a triangulation that is not necessarily conforming. In light of this, let us extend the construction of the smallest conforming refinement to *hp*-context.

Definition 2.2.13. For any *hp*-triangulation $\mathcal{D} \in \mathbb{D}$, let

$$\mathcal{C}(\mathcal{D}) := \arg\min \left\{ \#\tilde{\mathcal{D}} : \mathbb{D}_c \ni \tilde{\mathcal{D}} \geq \mathcal{D} \right\} \in \mathbb{D}_c$$

be its smallest conforming refinement. This $\mathcal{C}(\mathcal{D})$ can be found by applying MakeConform from Algorithm 1.5.17 to the *h*-triangulation $\mathcal{K}(\mathcal{D})$, and at any moment an *hp*-element D is subdivided, endowing both its children with local complexity d_D. ◊

In Theorem 1.5.16, we saw that the smallest conforming refinement $\mathcal{C}(\mathcal{K}) \in \mathbb{K}_c$ of an *h*-triangulation $\mathcal{K} \in \mathbb{K}$ satisfies $\#\mathcal{C}(\mathcal{K}) \lesssim \#\mathcal{K}$. In our *hp*-case, though, no such bound exists; see the following example.

Example 2.2.14. Take the domain $\Omega := (0,1)^2$, and consider the sequence of nonconforming *hp*-triangulations $(\mathcal{D}_N)_N$ depicted in the left of Figure 2.2.15. We assign to the element $D_N \in \mathcal{D}_N$ in the bottom-left corner a local complexity $d_{D_N} := N$, and equip each $D \in \mathcal{D}_N \setminus \{D_N\}$—dotted in the figure—with unit local complexity. Each $\mathcal{D}_{N+1}$ is created from $\mathcal{D}_N$ by two recursive bisections of the dotted triangle in the bottom-right corner. The total number of elements in $\mathcal{D}_N$ with unit complexity is $2N$, so $\#\mathcal{D}_N := \sum_D d_D = 3N$.

The smallest conforming refinement $\mathcal{C}(\mathcal{D}_N)$ of $\mathcal{D}_N$ is created by applying $2N-1$ recursive bisections to the non-dotted triangle in the bottom-right corner; see the right of Figure 2.2.15. In this process, $2N$ elements appear in place of D_N, all of complexity N. Therefore,

$$\mathcal{C}(\mathcal{D}_N) = 2N + 2N^2 \implies \frac{\#\mathcal{C}(\mathcal{D}_N)}{\#\mathcal{D}_N} = \frac{2(N+1)}{3}$$

which tends to infinity for $N \to \infty$. We conclude that, in general,

$$\#\mathcal{C}(\mathcal{D}) \not\lesssim \#\mathcal{D} \quad (\mathcal{D} \in \mathbb{D}).$$

This is one of the main open problems of the current algorithm: We cannot control the increase in complexity going from a nonconforming output triangulation emanating from *hp*-NearBest to its smallest conforming refinement used in the next finite element step. ◊

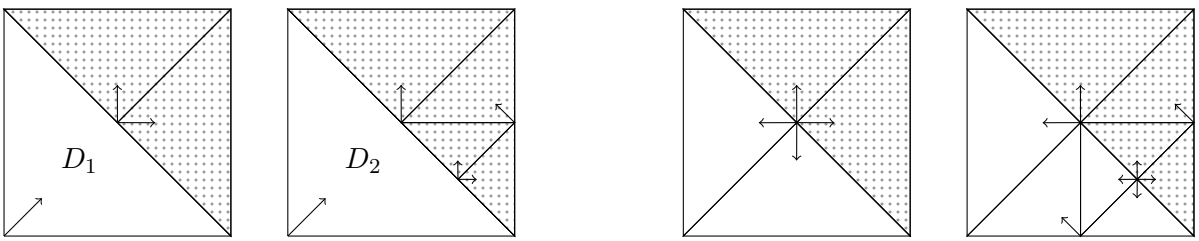


Figure 2.2.15.: Left: First two *hp*-triangulations of $\Omega := (0,1)$ in the sequence $(\mathcal{D}_N)_N$ of Example 2.2.14. The element D_N is of complexity N; the dotted area contains elements of local complexity 1. Right: Smallest conforming refinements of the two *hp*-triangulations on the left.

2.2.3. Relating the broken norm and energy norm

The routine *hp*-NearBest is near-best at reducing the broken error on some triangulation that is not necessarily conforming. However, in the finite element context, we desire a result in terms of the $H^1_0(\Omega)$-*seminorm* with respect to a *conforming* triangulation. We end this section with a remarkable result that relates the two.

Theorem 2.2.16 ([15, Thm. 5.1])**.** *Given is $v \in H^1_0(\Omega)$, and some hp-triangulation $\mathcal{D} \in \mathbb{D}$ with smallest conforming refinement $\mathcal{C}(\mathcal{D})$. For $\mathcal{D} \in \mathbb{D}$, define $\|p_{\mathcal{D}}\|_\infty := \max_{D\in\mathcal{D}} p_D$. Then for the global hp-error induced by the error functional* (2.2.5)*, the following holds:*

$$\inf_{w\in V(\mathcal{C}(\mathcal{D}))} \|v-w\|_{H^1_0(\Omega)} \le C_{\mathsf{B}}(\mathcal{D}) E_{\mathcal{D}}(v)^{1/2} \quad (v \in H^1_0(\Omega)) \tag{2.2.17}$$

with

$$C_{\mathsf{B}}(\mathcal{D}) \eqsim (1+\log\|p_{\mathcal{D}}\|_\infty)^{3/2}.$$

Remark 2.2.18. Ideally, we would have liked the constant $C_{\mathsf{B}}(\mathcal{D})$ to be independent of $\mathcal{D}$ in the sense that

$$C_{\mathsf{B}} := \sup_{\mathcal{D}\in\mathbb{D}} C_{\mathsf{B}}(\mathcal{D}) < \infty.$$

Regrettably, no such bound has been found yet, but we are quite close. The above remarkable result shows that $C_{\mathsf{B}}(\mathcal{D})$ increases *just logarithmically* with the largest polynomial degree present in $\mathcal{D}$. ◊

2.3. The routine Reduce

In the introduction of this chapter, we already formulated the properties of an *error reduction routine*. More formally, we will derive a routine with the following description.

Reduce takes a conforming *hp*-triangulation $\mathcal{D} \in \mathbb{D}_c$, and a reduction factor ρ. It produces a conforming triangulation $\mathbb{D}_c \ni \overline{\mathcal{D}} \ge \mathcal{D}$ with

$$\left\|u - u_{\overline{\mathcal{D}}}\right\|_{H^1_0(\Omega)} \le \rho \|u - u_{\mathcal{D}}\|_{H^1_0(\Omega)}\,. \tag{2.3.1}$$

We will construct Reduce using the a posteriori error estimators described in §1.4. We make the dependency of local indicators on the triangulation $\mathcal{D} \in \mathbb{D}_c$ explicit by writing $\eta_{\mathcal{D}}(v, \mathcal{M})$ for $\mathcal{M} \subset \mathcal{D}$. Recall that our model problem has no data oscillation, so any oscillation term in the error estimator vanishes.

Reduce is akin to *h*-AFEM in structure, in that we iterate

$$\texttt{SOLVE} \to \texttt{ESTIMATE} \to \texttt{MARK} \to \texttt{REFINE}$$

until (2.3.1) is satisfied. See Algorithm 2.3.2.

2.3.1. Sufficient properties of an error estimator

To control the number of iterations $M(\rho)$ required for this reduction factor, we require a few properties of our error estimator.

Require: $\theta \in (0,1]$

function Reduce($\rho \in (0,1]$, $\mathcal{D} \in \mathbb{D}_c$)
  compute $M := M(\rho)$ as in Thm. 2.3.8;
  $\mathcal{D}_0 := \mathcal{D}$;
  // SOLVE
  compute $u_{\mathcal{D}_0}$;
  for $m = 1, \dots, M$ **do**
    // ESTIMATE
    compute $\{\eta_{\mathcal{D}_{m-1}}(u_{\mathcal{D}_{m-1}}, D) : D \in \mathcal{D}_{m-1}\}$;
    // MARK
    mark a smallest $\mathcal{M}_{m-1} \subset \mathcal{D}_{m-1}$ with
      $\eta_{\mathcal{D}_{m-1}}(u_{\mathcal{D}_{m-1}}, \mathcal{M}_{\mathcal{D}_{m-1}}) \geq \theta \eta_{\mathcal{D}_{m-1}}(u_{\mathcal{D}_{m-1}}, \mathcal{D}_{m-1})$;
    // REFINE
    $\mathcal{D}_m := \tilde{D}(\mathcal{M}_{m-1})$;
    // SOLVE
    compute $u_{\mathcal{D}_m}$;
  return $\overline{\mathcal{D}} := \mathcal{D}_M$.

Algorithm 2.3.2: The Reduce routine used in *hp*-AFEM.

Definition 2.3.3. An error estimator η is *stable* in its first argument when there is a constant $C_{\mathsf{stab}}(\mathcal{D})$ for $\mathcal{D} \in \mathbb{D}_c$ with

$$\left|\eta_{\mathcal{D}}(v, \mathcal{D}) - \eta_{\mathcal{D}}(w, \mathcal{D})\right| \leq C_{\mathsf{stab}}(\mathcal{D}) \|v - w\|_{H^1_0(\Omega)}, \quad (v, w \in V(\mathcal{D})).$$

In words, it means that small perturbations in the input lead to small perturbations in the output. ◊

Definition 2.3.4. An error estimator satisfies *error reduction upon refinement* when there is a $\gamma < 1$ such that for any conforming *hp*-triangulation $\mathcal{D} \in \mathbb{D}_c$, and for any set of marked elements $\mathcal{M} \subset \mathcal{D}$, the following holds.

There is an *error-reducing refinement* $\mathcal{D}_{\mathsf{ER}} := \mathcal{D}_{\mathsf{ER}}(\mathcal{D}, \mathcal{M}) \in \mathbb{D}_c$ of $\mathcal{D}$ with $\#\mathcal{D}_{\mathsf{ER}} \lesssim \#\mathcal{D}$ such that

$$\eta^2_{\mathcal{D}_{\mathsf{ER}}}(u_{\mathcal{D}}, \mathcal{M}_{\mathsf{ER}}) \leq \gamma \eta^2_{\mathcal{D}}(u_{\mathcal{D}}, \mathcal{M}), \quad \eta^2_{\mathcal{D}_{\mathsf{ER}}}(u_{\mathcal{D}}, \mathcal{D}_{\mathsf{ER}} \setminus \mathcal{M}_{\mathsf{ER}}) \leq \eta^2_{\mathcal{D}}(u_{\mathcal{D}}, \mathcal{D} \setminus \mathcal{M}), \tag{2.3.5}$$

where

$$\mathcal{M}_{\mathsf{ER}} := \left\{\tilde{D} \in \mathcal{D}_{\mathsf{ER}} : \exists D \in \mathcal{M} \text{ s.t. } K_{\tilde{D}} \subset K_D\right\}$$

is the set of elements that were refined in going from $\mathcal{D}$ to $\mathcal{D}_{\mathsf{ER}}$. ◊

This definition contains a lot of information. In words, it compares the error indications on two triangulations, one a specific refinement of the other. On a set of marked elements, we require the indicated error to be strictly reduced (by a factor γ), and on the complement of this set, we don't want the indicated error to increase. Note that all indicated errors are with respect to the Galerkin solution of the *coarser* triangulation.

Definition 2.3.6. Select an error estimator η that is efficient and satisfies *error reduction upon refinement*. With θ from Reduce, and γ from (2.3.5), define $\tilde{\gamma} := (1 - \theta) + \gamma\theta$. For $\mathcal{D} \in \mathbb{D}_c$,

and with $C_{\mathsf{eff}}(\mathcal{D})$ the efficiency constant of the error estimator η, we define the (squared) *total error* as

$$\mathcal{E}_{\mathcal{D}}^2(u_{\mathcal{D}}) := \|u - u_{\mathcal{D}}\|_{H_0^1(\Omega)}^2 + \frac{1-\sqrt{\bar{\gamma}}}{C_{\mathsf{eff}}(\mathcal{D})}\eta_{\mathcal{D}}^2(u_{\mathcal{D}}, \mathcal{D}).$$

This total error is equivalent to the approximation error norm, in that

$$\|u - u_{\mathcal{D}}\|_{H_0^1(\Omega)}^2 \leq \mathcal{E}_{\mathcal{D}}^2(u_{\mathcal{D}}) \leq 2\|u - u_{\mathcal{D}}\|_{H_0^1(\Omega)}^2 . \qquad \lozenge$$

Lemma 2.3.7 ([15, Prop. 2.2])**.** *Assume that η is reliable, efficient, stable, and satisfies* error reduction upon refinement*. Then iterands produced inside* Reduce *show contraction for the total error, in that*

$$\mathcal{E}_{\mathcal{D}_m}(u_{\mathcal{D}_m}) \leq \kappa \mathcal{E}_{\mathcal{D}_{m-1}}(u_{\mathcal{D}_{m-1}}), \quad \kappa = \kappa(\mathcal{D}_m, \mathcal{D}_{m-1}) := \sqrt{1 - \frac{(1-\sqrt{\bar{\gamma}})^2}{2C_{\mathsf{eff}}(\mathcal{D}_m)C_{\mathsf{rel}}(\mathcal{D}_{m-1})}}.$$

Theorem 2.3.8 ([15, Prop. 2.2])**.** *Assume that η is reliable, efficient, stable, and satisfies* error reduction upon refinement*. If the efficiency resp. reliability constants of η are uniformly bounded in that*

$$C_{\mathsf{eff}} := \sup_{\mathcal{D}\in\mathbb{D}_c} C_{\mathsf{eff}}(\mathcal{D}) < \infty, \quad C_{\mathsf{rel}} := \sup_{\mathcal{D}\in\mathbb{D}_c} C_{\mathsf{rel}}(\mathcal{D}) < \infty, \tag{2.3.9}$$

then the number $M = M(\rho)$ of iterations required to satisfy (2.3.1) *inside Algorithm 2.3.2 satisfies $M(\rho) = \mathcal{O}(\log \rho^{-1})$, and is independent of $\mathcal{D}$. Moreover, for the output triangulation, $\#\overline{\mathcal{D}} \lesssim \#\mathcal{D}$.*

Proof. In light of the previous lemma, when (2.3.9) holds, we can bound $\kappa(\mathcal{D}_m, \mathcal{D}_{m-1})$ by the independent quantity

$$\kappa(\mathcal{D}_m, \mathcal{D}_{m-1}) \leq \kappa_0 := \sqrt{1 - \frac{(1-\sqrt{\bar{\gamma}})^2}{2C_{\mathsf{eff}}C_{\mathsf{rel}}}}.$$

Given M—the number of iterations in Reduce—the lemma yields

$$\begin{aligned}
\big\|u - u_{\overline{\mathcal{D}}}\big\|_{H_0^1(\Omega)} &= \big\|u - u_{\mathcal{D}_M}\big\|_{H_0^1(\Omega)} \eqsim \mathcal{E}_{\mathcal{D}_M}(u_{\mathcal{D}_M}) \\
&\leq \kappa_0^M \mathcal{E}_{\mathcal{D}_0}(u_{\mathcal{D}_0}) \eqsim \kappa_0^M \big\|u - u_{\mathcal{D}_0}\big\|_{H_0^1(\Omega)} \\
&= \kappa_0^M \|u - u_{\mathcal{D}}\|_{H_0^1(\Omega)} .
\end{aligned}$$

In order to satisfy the reduction property, we want $\kappa_0^M \leq \rho$. This means that

$$\begin{aligned}
\kappa_0^{-M} \geq \rho^{-1} &\implies -M \leq \log_{\kappa_0}(\rho^{-1}) = \frac{\log \rho^{-1}}{\log \kappa_0} \\
&\implies M \geq \frac{\log \rho^{-1}}{-\log \kappa_0}.
\end{aligned}$$

We choose the smallest such M, resulting in

$$M = \left\lceil \frac{\log \rho^{-1}}{-\log \kappa_0} \right\rceil = \mathcal{O}(\log \rho^{-1}).$$

This M depends on ρ and κ_0 only, and is hence independent of $\mathcal{D}$. Lastly, by *error reduction upon refinement*, we see that

$$\#\overline{\mathcal{D}} = \#\mathcal{D}_M \lesssim \#\mathcal{D}_{M-1} \lesssim \cdots \lesssim \#\mathcal{D}_0 = \#\mathcal{D}. \qquad \square$$

We continue this section looking at how the Melenk-Wohlmuth error estimator fares against these properties, and finish it looking at a novel idea using equilibrated fluxes.

2.3.2. Properties of the Melenk-Wohlmuth error estimator

We already saw in Theorem 1.4.13 that the Melenk-Wohlmuth error estimator is, under the comparability assumption of Assumption 1.4.12, both reliable and efficient. In terms of local complexity, this assumption can be restated as

$$\exists \nu > 1 \text{ s.t. } \frac{d_D}{d_{D'}} \leq \nu \quad (D, D' \in \mathcal{D}, \quad K_D \cap K_{D'} \neq \varnothing), \tag{2.3.10}$$

meaning that complexities of adjacent elements should be comparable.

Theorem 2.3.11 ([15, Cor. 5.1])**.** *Under Assumption 1.4.12, the Melenk-Wohlmuth a posteriori error estimator is stable, with* $C_{\mathsf{stab}}(\mathcal{D}) = C_{\mathsf{eff}}(\mathcal{D})^{1/2}$.

Consider (2.3.10). Realizing this assumption is not trivial. For instance, the sequence $(\mathcal{C}(\mathcal{D}_N))_N$ found in Example 2.2.14 violates this assumption already. We can mend this issue by tightening the definition of *conforming* a little.

Fixing some $\nu \geq 1$, let $\tilde{\mathbb{D}}_c$ denote the set of conforming *hp*-triangulations that satisfy the comparability assumption. For each $\mathcal{D} \in \mathbb{D}_c$, there is a $\tilde{\mathcal{D}} := \tilde{\mathcal{D}}(\mathcal{D}) \in \tilde{\mathbb{D}}_c$ with $\mathcal{K}(\tilde{\mathcal{D}}) = \mathcal{K}(\mathcal{D})$ and $\tilde{D} \geq \mathcal{D}$, found by increasing local complexities until the quotient is bounded from above by ν.

In light of the above, using the Melenk-Wohlmuth error estimator requires us to work with $\tilde{\mathbb{D}}_c$ instead of $\mathbb{D}_c$. Defining the mapping $\tilde{\mathcal{C}} := \tilde{\mathcal{D}} \circ \mathcal{C}$, all instances of $\mathbb{D}_c$ should be replaced by by $\tilde{\mathbb{D}}_c$, and $\mathcal{C}$ by $\tilde{\mathcal{C}}$. Obviously, $\tilde{\mathcal{D}}(\mathcal{D})$ can be constructed such that $\left\| p_{\tilde{\mathcal{D}}(\mathcal{D})} \right\|_\infty = \|p_{\mathcal{D}}\|_\infty$ so that the result from Theorem 2.2.16 still holds.

Unfortunately, there is no uniform bound on $\frac{\#\tilde{\mathcal{D}}(\mathcal{D})}{\#\mathcal{D}}$ across all $\mathcal{D} \in \mathbb{D}_c$.

Example 2.3.12. Let $\nu \geq 1$ be arbitrary. Consider the sequence of *hp*-triangulations $(\mathcal{D}_N)_N$ where each domain Ω_N is a regular $(N+1)$-polygon, and $\mathcal{D}_N$ is the set of $N+1$ triangles that triangulate Ω_N. Each element is endowed with local complexity 1, except for one single element D_N carrying complexity N. Note that every element has nonempty intersection with D_N.

See Figure 2.3.13. We find $\tilde{\mathcal{D}}(\mathcal{D}_N)$ by raising each unit local complexity to $\lceil N/\nu \rceil$—this makes the local complexities comparable. Then $\#\mathcal{D}_N = 2N$, and $\#\tilde{\mathcal{D}}(\mathcal{D}_N) = N + N \cdot \lceil N/\nu \rceil$. Therefore, their quotient

$$\frac{\#\tilde{\mathcal{D}}(\mathcal{D}_N)}{\#\mathcal{D}_N} = \frac{N(1 + \lceil N/\nu \rceil)}{2N} = \frac{1}{2}\left(1 + \lceil N/\nu \rceil\right)$$

tends to infinity for $N \to \infty$.

This is another open problem. On top of not being able to control the complexity of a smallest conforming refinement (see Example 2.2.14), we cannot control the complexity of the smallest *comparable* refinement $\tilde{\mathcal{D}}(\mathcal{D})$ uniformly over $\mathbb{D}_c$. ◊

Definition 2.3.14. For the Melenk-Wohlmuth error estimator, we can define a suitable error-reducing refinement $\mathcal{D}_{\mathsf{ER}}$ given $\mathcal{M} \subset \mathcal{D}$ as follows.

Perform two recursive bisections on each $D \in \mathcal{M}$, denoting the resulting *hp*-triangulation with $\overline{\mathcal{D}}(\mathcal{M})$, then take its smallest conforming refinement $\mathcal{C}(\overline{\mathcal{D}}(\mathcal{M})) =: \mathcal{D}_{\mathsf{ER}}$. See Figure 2.3.15 for an example. ◊

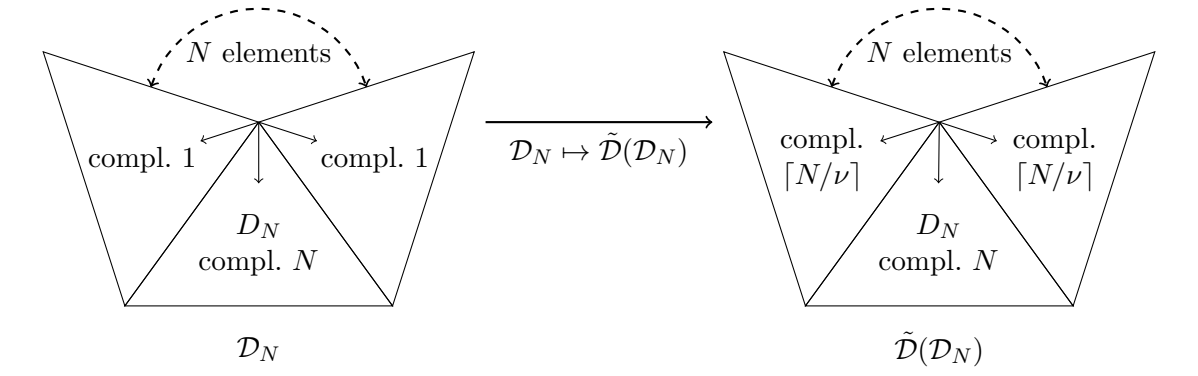


Figure 2.3.13.: Visual aid for Example 2.3.12. Two triangulations of the regular $(N+1)$-polygon into $N+1$ triangles. Left: Triangulation $\mathcal{D}_N$ with N elements of unit complexity, and one element D_N having local complexity N. Right: the smallest refinement of $\mathcal{D}_N$ that meets the requirement in (2.3.10).

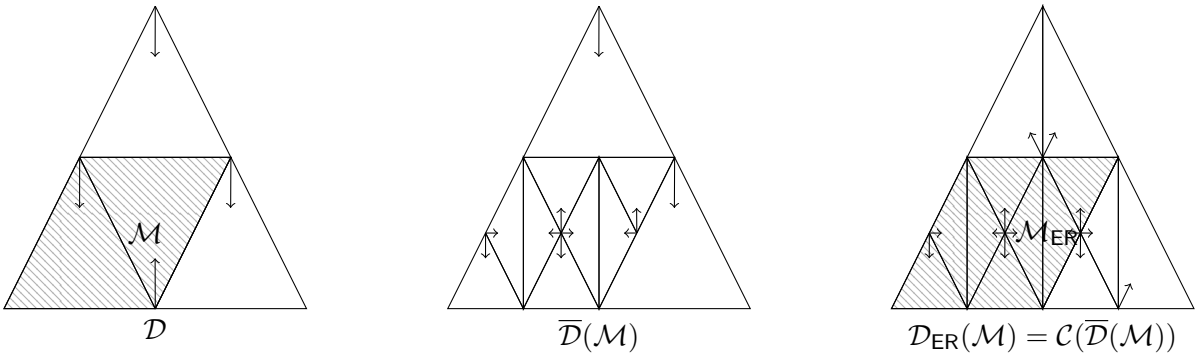


Figure 2.3.15.: Example of the procedure described in Definition 2.3.14.

Proposition 2.3.16 ([15, Prop. 5.3])**.** *The Melenk-Wohlmuth error estimator satisfies* error reduction upon refinement *with* $\gamma = \frac{1}{2}$.

Proof. The second condition of (2.3.5) holds trivially. We will prove the first condition. Recall that the MW-error estimator is defined as

$$\eta_{\mathsf{MW}}(v,D)^2 := \frac{h_D^2}{p_D^2}\left\| P^{(p_D-1)} f + \triangle u_{\mathcal{D}} \right\|^2_{L^2(K_D)} + \sum_{e \subset \partial K_D \setminus \partial\Omega} \frac{|e|}{2p_e} \left\| [\![\nabla u_{\mathcal{K}}]\!]_e \right\|^2_{L^2(e)}, \quad (D \in \mathcal{D}).$$

Let's look at the definition of $\mathcal{M}_{\mathsf{ER}}$. Each $\tilde{D} \in \mathcal{M}_{\mathsf{ER}}$ is created from a $D = D(\tilde{D}) \in \mathcal{M}$, and satisfies $h_{\tilde{D}} = h_D/2$. For the polynomial degrees, the substitution of $\mathcal{C}$ by $\tilde{\mathcal{C}}$ (cf. Example 2.3.12) implies that $p_{\tilde{D}} \geq p_D$.

Each edge $\tilde{e}$ on $\mathcal{M}_{\mathsf{ER}}$ falls in one of two classes: It is either internal to *or* on the boundary of some element $D \in \mathcal{M}$. In the former case, its contribution to the local error indication must vanish—$u_{\mathcal{D}}$ is a polynomial on D. In the latter case, $\tilde{e}$ is exactly one half of some edge $e = e(\tilde{e}) \subset D$; therefore, $h_{\tilde{e}} = h_e/2$ and $p_{\tilde{e}} \geq p_e$.

Noting that $\|v\|^2_{L^2(A\cup B)} = \|v\|^2_{L^2(A)} + \|v\|^2_{L^2(B)}$, we see that

$$\begin{aligned}
\eta^2_{\mathcal{D}_{\mathsf{ER}}}(u_{\mathcal{D}}, \mathcal{M}_{\mathsf{ER}}) &= \sum_{\tilde{D}\in\mathcal{M}_{\mathsf{ER}}} \frac{h^2_{\tilde{D}}}{p^2_{\tilde{D}}} \left\| P^{(p_{\tilde{D}}-1)} f + \triangle u_{\mathcal{D}} \right\|^2_{L^2(K_{\tilde{D}})} + \sum_{\tilde{e}} \frac{h_{\tilde{e}}}{2p_{\tilde{e}}} \left\| [\![\nabla u_{\mathcal{D}}]\!]_e \right\|^2_{L^2(\tilde{e})} \\
&\leq \frac{1}{4} \sum_{\tilde{D}\in\mathcal{M}_{\mathsf{ER}}} \frac{h^2_{D}}{p^2_{D}} \left\| P^{(p_{D}-1)} f + \triangle u_{\mathcal{D}} \right\|^2_{L^2(K_{\tilde{D}})} + \frac{1}{2} \sum_{\tilde{e}} \frac{h_{e}}{2p_{e}} \left\| [\![\nabla u_{\mathcal{D}}]\!]_e \right\|^2_{L^2(\tilde{e})} \\
&= \frac{1}{4} \sum_{D\in\mathcal{M}} \frac{h^2_{D}}{p^2_{D}} \left\| P^{(p_{D}-1)} f + \triangle u_{\mathcal{D}} \right\|^2_{L^2(K_{D})} + \frac{1}{2} \sum_{e} \frac{h_{e}}{2p_{e}} \left\| [\![\nabla u_{\mathcal{D}}]\!]_e \right\|^2_{L^2(e)} \\
&\leq \frac{1}{2} \eta^2_{\mathcal{D}}(u_{\mathcal{D}}, \mathcal{M}),
\end{aligned}$$

so the first part of (2.3.5) holds true with $\gamma = \frac{1}{2}$. □

It must be noted that the Melenk-Wohlmuth error estimator does **not** satisfy the conditions of Theorem 2.3.8 which shows boundedness of M inside Reduce. In Corollary 1.4.14, we saw that the efficiency constant satisfies $C_{\mathsf{eff}}(\mathcal{D}, \varepsilon) \eqsim \|p_{\mathcal{D}}\|^{2+2\varepsilon}_\infty$ for any $\varepsilon > 0$, which of course has infinite supremum over $\mathbb{D}_c$. As a result, we cannot bound the number of iterations independently of $\mathcal{D}$, but we can find something quite close.

Lemma 2.3.7 tells us that the total error is reduced by a factor κ in each iteration of Reduce. This κ depends on the efficiency constant $C_{\mathsf{eff}}(\mathcal{D}, \varepsilon)$, but it is constant within a single call to Reduce. Viewing this κ as a function of C_{eff}, we see through a Taylor expansion that

$$\frac{1}{-\log(\kappa)} = C_{\mathsf{eff}}(\mathcal{D}, \varepsilon) + \text{h.o.t.} \eqsim \|p_{\mathcal{D}}\|^{2+2\varepsilon}_\infty .$$

Looking at the proof of Theorem 2.3.8, we conclude that for the number M of iterations required, it holds that

$$M \eqsim \frac{\log(\rho^{-1})}{-\log(\kappa)} \eqsim \log(\rho^{-1}) \|p_{\mathcal{D}}\|^{2+2\varepsilon}_\infty .$$

The following example shows that this result is unsatisfactory; the computational cost *can* grow exponentially in the total number of degrees of freedom.

Example 2.3.17. Take a triangular domain K, and define the *hp*-triangulation $\mathcal{D}_N$ on this domain through $\mathcal{D}_N := \{(K, N)\}$. For simplicity, choose the Dörfler marking parameter θ to be 1. Then $\|p_{\mathcal{D}_N}\|_\infty \simeq \sqrt{N}$, so that

$$M \simeq \log(\rho^{-1}) N^{1+\varepsilon}.$$

The triangulation $\overline{\mathcal{D}}_N$ that Reduce outputs for input $\mathcal{D}_N$ is found by M uniform bisections of all triangles, yielding a total of 2^M triangles in $\overline{\mathcal{D}}_N$, each carrying a local complexity of N. Therefore,

$$\#\overline{\mathcal{D}}_N = N 2^M \simeq N 2^{\log(\rho^{-1}) N^{1+\varepsilon}},$$

which is obviously exponential in N. The result is that when solving for the Galerkin solution on $\overline{D}_N$, our direct solver, requiring in the order of $M^{1.5}$ flops with M the size of the global stiffness matrix, then requires a number of flops that grows exponentially in N. ◊

2.3.3. Equilibrated fluxes error estimator

Most recently, a result by Canuto *et al.* [14] improved on the results presented above. Instead of error reduction through adaptive h-refinement based on the Melenk-Wohlmuth estimator, they propose reducing the error using the p-robust equilibrated fluxes error estimator through uniform p-refinement.

Starting out with a Galerkin solution u_p of uniform degree p, they investigate the lowest value $q(p)$ necessary to achieve the reduction property

$$\left\|u - u_{p+q(p)}\right\|_{H^1_0(\Omega)} \leq \rho \left\|u - u_p\right\|_{H^1_0(\Omega)}$$

where ρ is the desired reduction factor, and $u_{p+q(p)}$ is the Galerkin solution of uniform degree $p + q(p)$. Using Galerkin orthogonality, one can restate this problem, now hoping to find *p-robust saturation* in that

$$\sup_{p\in\mathbb{N}} \frac{\left\|u - u_p\right\|_{H^1_0(\Omega)}}{\left\|u_p - u_{p+q(p)}\right\|_{H^1_0(\Omega)}} < \infty.$$

Their computational results suggest that $q(p) := \lceil \lambda p \rceil$ for some $\lambda > 0$ yields *p-robust saturation*. Contrastingly, no p-robust saturation is observed for $q(p) = q$ when q is independent of p, although for $q(p) = 4$, the quotient above grows very slowly in p; cf. [14, Tbl. 10].

Remark 2.3.18. In terms of computational cost, this method of p-enrichment is much more desirable than the h-refinement scheme above. Noting that u_p is found by solving a system of size $\simeq p^2 \simeq N^4$, we see that the computational cost of solving for $u_{p+\lceil \lambda p \rceil}$ is polynomial in N. This is in contrast with the earlier result of possibly exponential growth (see Example 2.3.17). ♢

2.4. *hp*-AFEM

With the algorithms *hp*-NearBest and Reduce in place, we can formulate the *hp*-adaptive finite element algorithm. Recall that the defining property of these two algorithms is the following.

hp-**NearBest** takes as input $\varepsilon > 0$ and $v \in H^1_0(\Omega)$, and produces a triangulation $\mathcal{D}_{\mathsf{NB}} \in \mathbb{D}$ that is a near-best approximation of v in the following exact sense:

$$\begin{cases} E_{\mathcal{D}_{\mathsf{NB}}}(v)^{1/2} \leq \varepsilon, \\ \exists 0 \leq b \leq 1 \leq B \text{ s.t. } \#\mathcal{D}_{\mathsf{NB}} \leq B\#\tilde{\mathcal{D}} \;\; \forall \tilde{D} \in \mathbb{D} \text{ with } E_{\tilde{\mathcal{D}}}(v)^{1/2} \leq b\varepsilon. \end{cases}$$

Reduce takes an *hp*-triangulation $\mathcal{D} \in \mathbb{D}_c$, and a reduction factor ρ. It produces a triangulation $\mathbb{D}_c \ni \overline{\mathcal{D}} \geq \mathcal{D}$ with

$$\left\|u - u_{\overline{\mathcal{D}}}\right\|_{H^1_0(\Omega)} \leq \rho \|u - u_{\mathcal{D}}\|_{H^1_0(\Omega)}\,.$$

Inside *hp*-AFEM, the input triangulation of *hp*-NearBest will be the current finite element approximation. The input to Reduce will be the smallest conforming refinement of the output of *hp*-NearBest. The *hp*-AFEM algorithm can now be formulated—see Algorithm 2.4.1.

We will now look at some of the properties of *hp*-AFEM.

Require: $\mu \in (0,1)$, $\omega \in \left(\frac{C_{\mathsf{L}}}{b}, \infty\right)$

procedure *hp*-AFEM($u_0 \in H^1_0(\Omega)$, $\varepsilon > 0$)
  find $\varepsilon_0 > 0$ with $\|u - u_0\|_{H^1_0(\Omega)} \leq \varepsilon_0$.
  for all $k \in \mathbb{N}$ **do**
    $\mathcal{D}^*_k := hp\text{-}\mathsf{NearBest}(u_{k-1}, \omega\varepsilon_{k-1})$;
    $\mathcal{D}_k := \mathsf{Reduce}(\frac{\mu}{1+C_{\mathsf{B}}(\mathcal{D}^*_k)\omega}, \mathcal{C}(\mathcal{D}^*_k))$;
    $u_k := u_{\mathcal{D}_k}$;
    $\varepsilon_k := \mu\varepsilon_{k-1}$;
    if $\varepsilon_k < \varepsilon$ **then**
      return

Algorithm 2.4.1: The *hp*-adaptive finite element algorithm *hp*-AFEM.

Definition 2.4.2. The global error functional $E_{\mathcal{D}}$ is *Lipschitz continuous* when

$$\exists C_{\mathsf{L}} > 0 \text{ s.t. } \left|E_{\mathcal{D}}(w)^{1/2} - E_{\mathcal{D}}(v)^{1/2}\right| \leq C_{\mathsf{L}}\|w - v\|_{H^1_0(\Omega)} \quad (\mathcal{D} \in \mathbb{D}, \quad v, w \in H^1_0(\Omega)). \tag{2.4.3}$$

♢

Lemma 2.4.4. *The global error functional induced by* (2.2.5) *is Lipschitz continuous with* $C_{\mathsf{L}} = 1$.

Proof. First, we will show that

$$\left|e_D(v)^{1/2} - e_D(w)^{1/2}\right| \leq |v - w|_{H^1(K_D)} \quad \left(D \in \mathcal{D}, \quad v, w \in H^1_0(\Omega)\right).$$

Recall that

$$e_D(v) := |v - P^{p_D}v|^2_{H^1(K_D)},$$

where P^p is the orthogonal projector onto $\mathbb{P}_p(K_D)/\mathbb{P}_0(K_D)$. By Lemma 2.2.3, $|\cdot|_{H^1(K_D)}$ is a norm on the space $H^1(K_D)/\mathbb{P}_0(K_D)$, so that the reverse triangle inequality holds. Combining this with the fact that $I - P^{p_D}$ is also an orthogonal projector, we get

$$\begin{aligned}
\left|e_D(v)^{1/2} - e_D(w)^{1/2}\right| &= \left||v - P^{p_D}v|_{H^1(K_D)} - |w - P^{p_D}w|_{H^1(K_D)}\right| \\
&\leq \left|(v - w) - P^{p_D}(v - w)\right|_{H^1(K_D)} = \left|(I - P^{p_D})(v - w)\right|_{H^1(K_D)} \\
&\leq |v - w|_{H^1(K_D)}.
\end{aligned}$$

With this result, the proof follows easily:

$$\begin{aligned}
\left|E_{\mathcal{D}}(v)^{1/2} - E_{\mathcal{D}}(w)^{1/2}\right| &= \left|\sqrt{\sum_D e_D(v)} - \sqrt{\sum_D e_D(w)}\right| \\
&\leq \left|\sum_D \left(e_D(v)^{1/2} - e_D(w)^{1/2}\right)^2\right|^{1/2} \\
&\leq \left|\sum_D |v - w|^2_{H^1(K_D)}\right|^{1/2} = \|v - w\|_{H^1_0(\Omega)}.
\end{aligned}$$

□

Theorem 2.4.5 ([15, Thm. 2.1])**.** *Assume* (2.4.3)*. Then, for the sequences* $(u_k)_k$ *and* $(\mathcal{D}_k^*)_k$ *produced by Algorithm 2.4.1, we have*

$$\begin{cases} \|u-u_k\|_{H_0^1(\Omega)} \leq \varepsilon_k \quad (k \geq 0), \\ E_{\mathcal{D}_k^*}(u)^{1/2} \leq (\omega + C_{\mathsf{L}})\varepsilon_{k-1} \quad (k \geq 1), \\ \#\mathcal{D}_k^* \leq B\#\mathcal{D} \;\; \forall \mathcal{D} \in \mathbb{D} \text{ with } E_{\mathcal{D}}(u)^{1/2} \leq (b\omega - C_{\mathsf{L}})\varepsilon_{k-1} \quad (k \geq 1). \end{cases} \tag{2.4.6}$$

In words, Theorem 2.4.5 tells us that *hp*-AFEM is *instance optimal* for reducing $E_{\mathcal{D}}(u)$ over $\mathcal{D} \in \mathbb{D}$. It also shows linear convergence of the Galerkin solutions to the real solution u in terms of the number of iterations. By virtue of the equality $\varepsilon_k = \mu^k \varepsilon_0$, we can break out of the loop after the desired tolerance $\varepsilon_{\mathsf{tol}}$ has been achieved.

To prove this theorem, the following lemma is useful.

Lemma 2.4.7. *Given as input a conforming triangulation* $\mathcal{D} \in \mathbb{D}_c$*, hp-*NearBest *produces a near-best hp-triangulation* $\mathcal{D}_{\mathsf{NB}} = \mathcal{D}_{\mathsf{NB}}(\mathcal{D})$ *for which*

$$\left\| u - u_{\mathcal{C}(\mathcal{D}_{\mathsf{NB}})} \right\|_{H_0^1(\Omega)} \leq \|u - u_{\mathcal{D}}\|_{H_0^1(\Omega)} + C_{\mathsf{B}}(\mathcal{D}_{\mathsf{NB}}) E_{\mathcal{D}_{\mathsf{NB}}}(u_{\mathcal{D}})^{1/2}.$$

Proof. The result follows from Galerkin orthogonality and the triangle inequality:

$$\begin{aligned} \left\| u - u_{\mathcal{C}(\mathcal{D}_{\mathsf{NB}})} \right\|_{H_0^1(\Omega)} &= \inf_{w \in \mathcal{V}(C(\mathcal{D}_{\mathsf{NB}}))} \|u - w\| \\ &\leq \|u - u_{\mathcal{D}}\|_{H_0^1(\Omega)} + \inf_{w \in \mathcal{V}(C(\mathcal{D}_{\mathsf{NB}}))} \|u_{\mathcal{D}} - w\| \\ &\leq \|u - u_{\mathcal{D}}\|_{H_0^1(\Omega)} + C_{\mathsf{B}}(\mathcal{D}_{\mathsf{NB}}) E_{\mathcal{D}_{\mathsf{NB}}}(u_{\mathcal{D}})^{1/2}. \end{aligned}$$

□

Proof (of the theorem). The first statement holds, by construction, for $k = 0$. Now assume that it holds for $k-1$, in that $\|u - u_{k-1}\|_{H_0^1(\Omega)} \leq \varepsilon_{k-1}$. Then, after the kth call to *hp*-NearBest, the previous Lemma ensures that

$$\begin{aligned} \left\| u - u_{\mathcal{C}(\mathcal{D}_k^*)} \right\|_{H_0^1(\Omega)} &\leq \|u - u_{k-1}\|_{H_0^1(\Omega)} + C_{\mathsf{B}}(\mathcal{D}_k^*) E_{\mathcal{D}_k^*}(u_{k-1})^{1/2} \\ &\leq \varepsilon_{k-1} + C_{\mathsf{B}}(\mathcal{D}_k^*) E_{\mathcal{D}_k^*}(u_{k-1})^{1/2} \\ &\leq (1 + C_{\mathsf{B}}(\mathcal{D}_k^*)\omega)\varepsilon_{k-1}. \end{aligned}$$

The subsequent call to Reduce then ensures that

$$\|u - u_k\|_{H_0^1(\Omega)} \leq \frac{\mu}{1 + C_{\mathsf{B}}(\mathcal{D}_k^*)\omega} \left\| u - u_{\mathcal{C}(\mathcal{D}_k^*)} \right\|_{H_0^1(\Omega)} \leq \mu \varepsilon_{k-1} = \varepsilon_k.$$

Then, by the property of *hp*-NearBest and (2.4.3):

$$E_{\mathcal{D}_k^*}(u)^{1/2} \leq E_{\mathcal{D}_k^*}(u_{k-1})^{1/2} + C_{\mathsf{L}} \|u - u_{k-1}\|_{H_0^1(\Omega)} \leq (\mu + C_{\mathsf{L}})\varepsilon_{k-1} \quad (k \geq 1).$$

Let $\mathcal{D} \in \mathbb{D}$ with $E_{\mathcal{D}}(u)^{1/2} \leq (b\omega - C_{\mathsf{L}})\varepsilon_{k-1}$. Then again by Lipschitz continuity,

$$E_{\mathcal{D}}(u_{k-1})^{1/2} \leq (b\omega - C_{\mathsf{L}})\varepsilon_{k-1} + C_{\mathsf{L}}\varepsilon_{k-1} = b\omega\varepsilon_{k-1},$$

so by the near-best property, $\#\mathcal{D}_k^* \leq B\#\mathcal{D}$. □

Remark 2.4.8. Algorithm 2.4.1 requires a reduction factor $\rho := \mu/((1 + C_{\mathsf{B}}(\mathcal{D}_k^*))\omega)$ that depends on $C_{\mathsf{B}}(\mathcal{D}_k^*)$. Therefore, by $C_{\mathsf{B}}(\mathcal{D}) \simeq (1 + \log\|p_{\mathcal{D}}\|_\infty)^{3/2}$, the number of iterations required grows (very slowly) whenever the maximal polynomial degree increases. ◊

We end this section with a few notes on the convergence speed of *hp*-AFEM. In the current context, the near-best *hp*-adaptive approximation error σ_N^{hp} is a functional taking $v \in H_0^1(\Omega)$.

Corollary 2.4.9 (Algebraic decay)**.** *If $\sigma_N^{hp}(u)^{1/2}$ decays algebraically, in that*

$$\sup_N \sigma_N^{hp}(u)^{1/2} N^s < \infty \text{ for some } s > 0,$$

then the sequence of broken errors $E_{\mathcal{D}_k^*}(u)^{1/2}$ also decays algebraically, and even with optimal rate.

Proof. Write $\sigma_N := \sigma_N^{hp}(u)$ and $E_{\mathcal{D}} := E_{\mathcal{D}}(u)$. Define

$$Q := \sup_N \sigma_N^{1/2} N^s < \infty.$$

Then, by this property, for each $\varepsilon > 0$ there is a *smallest* $N_\varepsilon \in \mathbb{N}$ such that $\sigma_{N_\varepsilon}^{1/2} \leq \varepsilon$. This means that

$$Q \geq (N_\varepsilon - 1)^s \sigma_{N_\varepsilon - 1}^{1/2} > (N_\varepsilon - 1)^s \varepsilon \implies N_\varepsilon - 1 < Q^{1/s} \varepsilon^{-1/s}.$$

Therefore we must have

$$N_\varepsilon \lesssim Q^{1/s} \varepsilon^{-1/s} \eqsim \varepsilon^{-1/s}.$$

The near-best property ensures that for the sequence of triangulations $(\mathcal{D}_k^*)_k$ produced by *hp*-AFEM, we have $E_{\mathcal{D}_k^*}^{1/2} \leq (\omega + C_{\mathsf{L}})\varepsilon_{k-1}$, and $\#\mathcal{D}_k^* \leq B\#\mathcal{D}$ for any $\mathcal{D}$ with $E_{\mathcal{D}}^{1/2} \leq (b\omega - C_{\mathsf{L}})\varepsilon_{k-1}$; choose the $\mathcal{D}$ with smallest complexity. For this $\mathcal{D}$, by the previous result,

$$\#\mathcal{D} \lesssim [(b\omega - C_{\mathsf{L}})\varepsilon_{k-1}]^{-1/s} \eqsim \varepsilon_{k-1}^{-1/s}$$

which implies that

$$(\#\mathcal{D}_k^*)^s E_{\mathcal{D}_k^*}^{1/2} \leq (B\#\mathcal{D})^s (\omega + C_{\mathsf{L}})\varepsilon_{k-1} \lesssim (B\varepsilon_{k-1}^{-1/s})^s \varepsilon_{k-1} = B^s \varepsilon_{k-1}^{-1} \varepsilon_{k-1} = B^s.$$

In other words, we find $(\#\mathcal{D}_k^*)^s E_{\mathcal{D}_k^*}^{1/2} \lesssim B^s$, with the constants inside "$\lesssim$" independent of k. Therefore

$$\sup_k (\#\mathcal{D}_k^*)^s E_{\mathcal{D}_k^*}^{1/2} \lesssim B^s < \infty$$

so that the broken errors decay algebraically with rate s. □

The previous result was reminiscent of the algebraic decay of *h*-AFEM. In our *hp*-case, we can do even better.

Corollary 2.4.10 (Exponential decay [15, Rem. 2.2])**.** *Let $\sigma_N^{hp}(u)^{1/2}$ decay* exponentially*, in that*

$$\sup_N \left\{ \sigma_N^{hp}(u)^{1/2} e^{\eta N^\tau} \right\} < \infty \text{ for some } \eta, \tau > 0.$$

Then an approach similar to the above shows that

$$\sup_k \left\{ e^{B^{-\tau}\eta(\#\mathcal{D}_k^*)^\tau} E_{\mathcal{D}_k^*}(u)^{1/2} \right\} < \infty.$$

In other words, when $\sigma_N^{hp}(u)^{1/2}$ decays exponentially with parameters (η, τ), then the sequence of broken errors produced by hp-AFEM does as well, with parameters $(\tilde{\eta}, \tau)$, where $\tilde{\eta} = B^{-\tau}\eta$.

We will see in Chapter 5 that this exponential decay is observed in practice.

Remark 2.4.11. The exponential decay of the error norms found above is in terms of the number of degrees of freedom. Let us consider the behaviour in terms of computational cost. For now, forget about cost involved in the coarsening step, and rather focus on the computational cost of the error reduction step. If this reduction requires $F = CN^k$ flops, with $k > 1$ and $C > 0$, then $N = (F/C)^{1/k}$. With this, an exponential decay rate $\exp(-\eta N^\tau)$ in terms of the complexity results in a rate

$$\exp\left(-(\eta C^{-\tau/k})F^{\tau/k}\right)$$

in terms of the number of flops. This is still exponential (albeit with a suboptimal exponent τ/k). Therefore, exponential convergence with respect to computational efoort *can be expected* using equilibrated fluxes (cf. Remark 2.3.18). In light of Example 2.3.17, this cannot be expected for the Melenk-Wohlmuth reduction strategy. ◊

Conclusion

In this chapter, we studied a novel *hp*-adaptive finite element method. In Corollary 2.4.10, we saw that under mild circumstances, this *hp*-AFEM procedure exhibits *exponential decay* of the approximation error in terms of the size of the triangulations.

The results of this chapter are not fully satisfactory, for a multitude of reasons.

Firstly, the complexity of the smallest conforming refinement of an *hp*-triangulation $\mathcal{D}$ cannot be bounded in terms of the complexity of $\mathcal{D}$ itself; see Example 2.2.14.

Moreover, the coarsening routine *hp*-NearBest measures error in the *broken* norm as opposed to the $H^1(\Omega)$-seminorm. This introduces some problems, most notably the loss of a logarithmic factor; see §2.2.3.

Lastly, the choice of error estimator (and with it, the implementation of Reduce) depends on an assumption that introduces another unbounded quotient; see Example 2.3.12. A different error reduction strategy was recently introduced and uses equilibrated fluxes (see Remark 2.3.18), but its implementation is very cumbersome.

Most of these issues are not actually observed in practice. This invites further analysis: Maybe it is possible to show that the issues don't appear when confining our view to a smaller class of boundary value problems.

3. Bases for the finite element space

In §1.2.2, we derived a method for constructing the Galerkin solution on a given triangulation. In the process, we assumed having a global basis of the finite element space. In this chapter, we will discuss the construction of such a global basis. We will start off by assuming Lagrange elements and the global basis of the Galerkin space it induces, and identify why it is ill-suited for *hp*-adaptivity.

We will then introduce *hierarchical bases* and argue why they are a natural choice, before showing some of the implications of using *local* bases to induce a global basis. We will continue by giving an example of a widely used hierarchical basis that fares well with respect to these properties. Our final section will be devoted to studying a fairly novel application of *Bernstein-Bézier polynomials* within an *hp*-adaptive finite element context.

3.1. Lagrange elements

In an *h*-adaptive setting, one usually chooses a Lagrange element (cf. Example 1.2.11) as the foundation for a finite element method. Choose some degree p, and recall that on each triangle K in a conforming triangulation $\mathcal{K}$, the local degrees of freedom were defined as the point evaluations on the set of domain points $\mathcal{D}^p(K)$, defined by (cf. Definition 1.2.5)

$$\mathcal{D}^p(K) := \{\mathbf{v}_{\boldsymbol{\alpha}} : \boldsymbol{\alpha} \in \mathcal{I}^p\}, \quad \mathcal{I}^p := \left\{\boldsymbol{\alpha} \in \mathbb{N}_0^3 : |\boldsymbol{\alpha}| = p\right\}.$$

Any function $v|_K \in \mathbb{P}_p(K)$ is completely determined by its point evaluations on these domain points, so necessarily, each v in our finite element space $V(\mathcal{K})$ must be completely determined by its point evaluations on their union $\cup_{K\in\mathcal{K}}\mathcal{D}^p(K)$. In fact, it is overdetermined by this set: $v \in V(\mathcal{K}) \subset H_0^1(\Omega)$ must vanish on $\partial\Omega$, so it is even completely determined by point evaluations on the set

$$\mathcal{L}(\mathcal{K}) := \bigcup_{K\in\mathcal{K}} \mathcal{D}^p(K) \setminus \partial\Omega.$$

This gives rise to the set of *global degrees of freedom*

$$V(\mathcal{K})' \supset \mathcal{N}(\mathcal{K}) := \left\{N : V(\mathcal{K}) \to \mathbb{R} : v \mapsto v(\mathbf{v}) : \mathbf{v} \in \mathcal{L}(\mathcal{K})\right\}.$$

The following results show the role of the global degrees of freedom.

Proposition 3.1.1. *Take a conforming triangulation $\mathcal{K}$, and endow each triangle with a Lagrange element of degree p. Then the set of global degrees of freedom $\mathcal{N}(\mathcal{K})$ is a basis for $V(\mathcal{K})'$.*

Corollary 3.1.2. *Given a conforming triangulation $\mathcal{K}$, and on each triangle, a Lagrange element of degree p. The set $\boldsymbol{\Phi}_{\mathcal{K}}^{\mathsf{Lagr}} := \left\{\boldsymbol{\phi}_i : 1 \le i \le \dim V(\mathcal{K})\right\} \subset V(\mathcal{K})$ dual to $\mathcal{N}(\mathcal{K})$ (in that $N_i(\boldsymbol{\phi}_j) = \delta_{ij}$) is a global basis for $V(\mathcal{K})$.*

Proposition 3.1.3. *The global basis functions $\boldsymbol{\phi} \in \boldsymbol{\Phi}_{\mathcal{K}}^{\mathsf{Lagr}}$ have local support.*

Proof. There is a bijection between $\mathcal{L}(\mathcal{K})$ and $\boldsymbol{\Phi}_{\mathcal{K}}^{\mathsf{Lagr}}$, so each basis function is associated with a point. Its support is equal to the union of all triangles that contain this point. □

See Figure 3.1.4. The basis functions fall in one of three categories. If the function ϕ^{v} is associated with a vertex of some triangle, then the number of triangles in its support is exactly the valence of this vertex. When the function ϕ^{e} is associated with an edge between two triangles, then it must vanish outside the union of these two. Lastly, if ϕ^{f} is associated with some point in the interior of a triangle, then the support must be the single triangle containing this point.

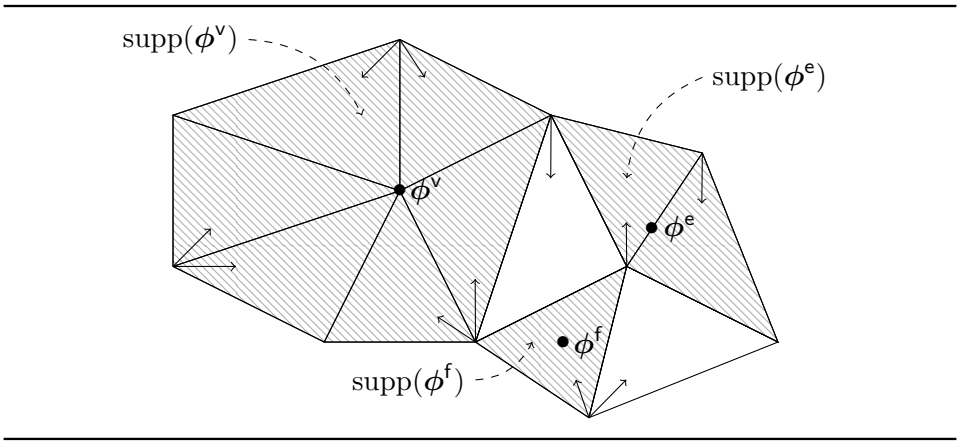


Figure 3.1.4.: Three example functions from the global Lagrange basis $\mathbf{\Phi}_{\mathcal{K}}^{\mathsf{Lagr}}$. Each basis function is associated with a specific point inside the domain. The support of this function is equal to the union of all triangles that contain this point.

We see that, in the *h*-adaptive case, this basis works beautifully. Let us look at an example of why this basis is not well-suited for the *hp* case.

Example 3.1.5. Imagine a triangulation $\mathcal{K}$ of the unit square $\Omega := (0,1)^2$ into two triangles K_1 and K_2. Equip K_1 with the linear polynomials $\mathbb{P}_1(K_1)$, and K_2 with quadratics $\mathbb{P}_2(K_2)$. On the interface $e := K_1 \cap K_2$ between these two triangles, these spaces do not agree:

$$\dim \mathbb{P}_1(K_1)|_e = 2 \neq 3 = \dim \mathbb{P}_2(K_2)|_e.$$

The finite element space $V(\mathcal{K})$ of this triangulation is spanned by continuous piecewise polynomials. Hence, along e, any $v \in V(\mathcal{K})$ must be a linear polynomial, so any global basis of $V(\mathcal{K})$ must not contain a quadratic component on e.

In contrast, the local shape space of K_2 *does* contain a quadratic component along this edge. Worse still, the three basis functions of $\mathbb{P}_2(K_2)$ along e are all of strict degree 2. It is unclear how we should proceed in finding a global basis for $V(\mathcal{K})$. ♢

This problem can be solved in a multitude of ways. The classical solution is through a *hierarchical* approach.

3.2. Hierarchical elements

We will construct a basis for the polynomials of given degree through a construction akin to that of the Lagrange basis, in that each basis function is associated with either a vertex, an edge, or a face of a triangle.

In the definition of a finite element in §1.2, we started with (a) an element domain, (b) a local shape space, and (c) a set of local degrees of freedom. Instead of these local degrees of freedom, one can also use a local basis for the shape space. The following result shows that this leads to an equivalent definition of a finite element.

Lemma 3.2.1. *Given an element domain K, a local shape space $\mathcal{P}(K)$, and a basis Φ of $\mathcal{P}(K)$, there is a set $\mathcal{N}(K)$—the local degrees of freedom—dual to Φ.*

Proof. Seeing $\Phi = \{\phi_r : 1 \le r \le \dim \mathcal{P}(K)\}$ as a vector of functions, any $w \in \mathcal{P}(K)$ can be written in terms of this basis using a vector $\mathbf{w}$ such that $w = \mathbf{w}^\top \Phi$. Then the set

$$\mathcal{N}(K) := \{\mathcal{P}(K) \to \mathbb{R} : w \mapsto \mathbf{w}_r : 1 \le r \le \dim \mathcal{P}(K)\}$$

is dual to Φ, and of the appropriate size, hence a basis for $\mathcal{P}(K)'$. □

Definition 3.2.2. Given an element domain K, a family of bases $(\Phi_{K,p})_{p\in\mathbb{N}}$ for the degree-p polynomials is *hierarchical* when it satisfies

$$\Phi_{K,1} \subset \Phi_{K,2} \subset \Phi_{K,3} \subset \dots;$$

the sequence of bases is nested. ◊

Corollary 3.2.3. *By the definition of a hierarchical basis, we see that for each p,*

$$\Phi_{K,p+1} \setminus \Phi_{K,p} = \mathbb{P}_{p+1}(K) \setminus \mathbb{P}_p(K),$$

or in words, that in going from degree p to $p+1$, we add polynomials of strict degree $p+1$.

In order to satisfy H^1*-conformity* of the global basis—meaning that global basis elements lie in $H^1(\Omega)$, or in other words, that they are continuous across element edges—we will make a few mild assumptions.

3.2.1. Local hierarchical basis

On a conforming *hp*-triangulation $\mathcal{D} \in \mathbb{D}_c$, let $\mathcal{E}(\mathcal{D})$ denote the set of its edges, and $\mathcal{V}(\mathcal{D})$ the set of its vertices.

Assume that on each edge $e \in \mathcal{E}(\mathcal{D})$, there is a sequence $(\phi_{e,q})_{q\ge 2}$ of polynomials such that

$$\operatorname{span}\{\phi_{e,2}, \dots, \phi_{e,p}\} = H^1_0(e) \cap \mathbb{P}_p(e) \quad (p \ge 2). \tag{3.2.4}$$

Moreover, assume that on each triangle $K \in \mathcal{K}(\mathcal{D})$, there are three types of functions:

- a set of *vertex functions*

$$\{\phi_{K,\mathbf{v}} : \mathbf{v} \in \mathcal{V}(\mathcal{D}) \cap T\} \subset \mathbb{P}_1(K) \tag{3.2.5}$$

  such that $\phi_{K,\mathbf{v}}(\mathbf{w}) = \delta_{\mathbf{vw}}$ for $\mathbf{w} \in \mathcal{V}(\mathcal{D}) \cap T$.

- on each edge $e \in \mathcal{E}(\mathcal{D}) \cap K$ and $q \ge 2$, an *edge function* $\phi_{K,e,q} \in \mathbb{P}_q(K)$ satisfying

$$\phi_{K,e,q}|_e = \phi_{e,q} \quad \text{and} \quad \phi_{K,e,q}|_{e'} = 0 \ \ (e \ne e' \in \mathcal{E}(\mathcal{D}) \cap K). \tag{3.2.6}$$

- for every $q \geq 3$, a collection $\Phi^\circ_{K,q} \subset \mathbb{P}_q(K)$ of *face functions* with cardinality $\#\Phi^\circ_{K,q} = q-2$ for which
$$\operatorname{span} \Phi^\circ_{K,q} = H^1_0(K) \cap \mathbb{P}_q(K) \setminus H^1_0(K) \cap \mathbb{P}_{q-1}(K). \tag{3.2.7}$$

These assumptions lead to a result locally on K.

Proposition 3.2.8. *For $(K,d) \in \mathcal{D}$ and some degree $p = p(d)$, the set*

$$\Phi_{K,p} := \left\{\phi_{K,\mathbf{v}} : \mathbf{v} \in \mathcal{V}(K) \cap K\right\} \cup \left\{\phi_{K,e,q} : e \in \mathcal{E}(K) \cap K, 2 \leq q \leq p\right\} \cup \bigcup_{q=3}^{p} \Phi^\circ_{K,q},$$

of all the above functions of degree $\leq p$ is a basis for $\mathbb{P}_p(K)$.

Proof. If the functions in $\Phi_{K,p}$ are all linearly independent from each other, and the cardinality of $\Phi_{K,p}$ is correct in that

$$\#\Phi_{K,p} = \dim \mathbb{P}_p(K) = (p+1)(p+2)/2,$$

then the result must hold. There are no duplicates in $\Phi_{K,p}$, hence

$$\#\Phi_{K,p} = 3 + \sum_{q=2}^{p} 3 + \sum_{q=3}^{p} q - 2 = 3 + 3(p-1) + (p-1)(p-2)/2,$$

so its cardinality satisfies the requirement.

Linear independence will first be assessed within one function type; after that, we will consider independence across them, completing the argument.

We know that $\dim \mathbb{P}_q(K) = (q+1)(q+2)/2$, and that $\dim \mathbb{P}_q(e) = q+1$ on every edge, so that

$$\dim(H^1_0(K) \cap \mathbb{P}_q(K)) = \dim \mathbb{P}_q(K) - 3 \dim \mathbb{P}_q(e) + 3 = \frac{(q+1)(q+2)}{2} - 3q.$$

Therefore, for each $3 \leq q \leq p$ separately, the face functions $\Phi^\circ_{K,q}$ span a space of dimension

$$\dim\left(H^1_0(K) \cap \mathbb{P}_q(K) \setminus H^1_0(K) \cap \mathbb{P}_{q-1}(K)\right) = \frac{(q+1)(q+2)}{2} - 3q - \frac{q(q+1)}{2} + 3(q-1) = q-2$$

and must therefore comprise a linearly independent set. Each $\phi \in \Phi^\circ_{K,q}$ is of strict degree q, so the union of all such sets must be linearly independent as well.

A similar argument holds for the edge functions: For each $2 \leq q \leq p$, there is exactly one edge function that does not vanish along a given edge. Each such edge function is of strict degree q, so their union must contain functions that are all linearly independent from each other.

The set of vertex functions is just the degree-1 Lagrange basis, this set must contain linearly independent functions as well.

Now, both edge- and face functions vanish in vertices, so these must all be independent from the vertex functions; face functions moreover vanish along edges, so the same must between edge- and face functions. We conclude that $\Phi_{K,p}$ is a set of functions all linearly independent from each other. □

Corollary 3.2.9. *The family of bases constructed as above is inherently hierarchical.*

3.2.2. Global hierarchical basis from local contributions

Given the definition of such a hierarchical element, we will now construct a global basis for $V(\mathcal{D})$.

Definition 3.2.10. For every interior vertex $\mathbf{v} \in \mathcal{V}_{int}(\mathcal{D}) := \mathcal{V}(\mathcal{D}) \setminus \partial\Omega$, define the global vertex function $\boldsymbol{\phi}_{\mathbf{v}}$ through $\boldsymbol{\phi}_{\mathbf{v}}|_K := \phi_{K,\mathbf{v}}$ when $\mathbf{v} \in K \in \mathcal{K}(\mathcal{D})$. Collect all of these functions in a set $\boldsymbol{\Phi}^V_{\mathcal{D}}$.

For every interior edge $e \in \mathcal{E}_{int}(\mathcal{D}) := \mathcal{E}(\mathcal{D})\backslash\partial\Omega$, define $p_e := \min\{p_D, p_{D'}\}$ when $e = K_D \cap K_{D'}$. Then, for $2 \leq q \leq p_e$, define the global edge function $\boldsymbol{\phi}_{e,q}$ through $\boldsymbol{\phi}_{e,q}|_K := \phi_{K,e,q}$ when $e \subset K \in \mathcal{K}(\mathcal{D})$. Denote the set of all such functions with $\boldsymbol{\Phi}^E_{\mathcal{D}}$.

For $D \in \mathcal{D}$, the (local) face functions $\phi \in \Phi^\circ_{K_D,q}$ for $3 \leq q \leq d_D$ all vanish on ∂K_D so they can easily viewed as functions on the entire domain, vanishing on all element domains except K_D. For the set of all global face functions, write $\boldsymbol{\Phi}^F_{\mathcal{D}}$. ◊

These definitions lead to the following important result.

Theorem 3.2.11. *Let $\mathcal{D} \in \mathbb{D}_c$ be a conforming hp-triangulation. Then*

$$\boldsymbol{\Phi}_{\mathcal{D}} := \boldsymbol{\Phi}^V_{\mathcal{D}} \cup \boldsymbol{\Phi}^E_{\mathcal{D}} \cup \boldsymbol{\Phi}^F_{\mathcal{D}}$$

is a basis for the Galerkin space

$$V(\mathcal{D}) = H^1_0(\Omega) \cap \prod_{D\in\mathcal{D}} \mathbb{P}_{p_D}(K_D).$$

Proof. We will prove this by showing that (1) $\boldsymbol{\Phi}_{\mathcal{D}} \subset V(\mathcal{D})$; (2) all functions in $\boldsymbol{\Phi}_{\mathcal{D}}$ are linearly independent; and (3) $V(\mathcal{D}) \subset \operatorname{span}\boldsymbol{\Phi}_{\mathcal{D}}$. By (1) and (3), $\operatorname{span}\boldsymbol{\Phi}_{\mathcal{D}} = V(\mathcal{D})$, and therefore by (2), it must be a basis.

(1) All our global functions $\boldsymbol{\phi} \in \boldsymbol{\Phi}_{\mathcal{D}}$ vanish on $\partial\Omega$, and are piecewise polynomials of appropriate local degree. We have to show global continuity.

Global continuity is obvious for the face functions. For each global edge function $\boldsymbol{\phi}_{e,q}$ with $e := K_D \cap K_{D'}$, the equality $\phi_{K_D,e,q}|_e = \phi_{e,q} = \phi_{K_{D'},e,q}|_e$ shows that it must be continuous across 'its' edge; the fact that it vanishes outside $K_D \cup K_{D'}$ and along any edge $e' \neq e$ shows its global continuity. The global vertex functions are just the hat functions shown in Chapter 1, and we already argued their global continuity.

(2) Linear independence follows from the fact that locally on each triangle, the functions are part of the basis defined in the previous proposition and hence linearly independent.

(3) Take a function $v \in V(\mathcal{D})$. Of course $v = Vv + (v - Vv)$, where Vv is its piecewise linear interpolant. Now, Vv is continuous and piecewise linear, and hence can be expressed in terms of the global hat functions $\boldsymbol{\Phi}^V_{\mathcal{D}}$.

Now, by $v(\mathbf{w}) = Vv(\mathbf{w})$ for vertices $\mathbf{w}$, we know that $v - Vv$ vanishes in vertices. Therefore, restricted to an edge $e \in \mathcal{E}_{int}(\mathcal{D})$, it holds that $(v - Vv)|_e \in H^1_0(e) \cap \mathbb{P}_{p_e}(e)$. In (3.2.4) we assumed having a basis $\{\phi_{e,2}, \ldots, \phi_{e,p_e}\}$ for this space, so there is a unique representation of $(v - Vv)|_e$ in terms of this local basis. Moreover, each $\phi_{e,q} \in \Phi_{e,p_e}$ corresponds with exactly one global basis function $\boldsymbol{\phi}_{e,q} \in \boldsymbol{\Phi}_{\mathcal{D}}$ that vanishes on all other edges. Therefore,

define Ev as the linear combination of global edge functions $\mathbf{\Phi}_{\mathcal{D}}^{E}$ for which $Ev|_e = (v - Vv)|_e$ along all interior edges.

Then define $Fv := v - Vv - Ev$. We see that Fv must vanish along all edges and in all vertices. Moreover, locally on each element (K, d), $v|_K$ is of degree $p = p(d)$, and so are both $(Vv)|_K$ and $(Ev)|_K$. Therefore $(Fv)|_K$ is of degree p locally on each element, so it is a member of $\mathbf{\Phi}_{\mathcal{D}}^{F}$ globally. We conclude that

$$v = Vv + v - Vv = Vv + Ev + Fv,$$

where each term is a linear combination of global functions of vertex, edge, or face type. □

In light of the previous theorem, each global basis function is constructed from local contributions. Viewing the local bases

$$\Phi_D = \left\{\phi_{D,r} : 1 \leq r \leq \lfloor d_D \rfloor_{\triangle}\right\} \quad (D \in \mathcal{D})$$

as vectors of basis functions, it is interesting to find out which tuples (D, r) contribute to a global basis function, and how. This gives rise to a *local-to-global mapping* that maps local degrees of freedom $\left\{(D, r) : D \in \mathcal{D}, 1 \leq r \leq \lfloor d_D \rfloor_{\triangle}\right\}$ to a global index

$$i_D(r) \in \varnothing \cup \left\{1, \ldots, \dim V(\mathcal{D})\right\}.$$

Now, $i_D(r) = \varnothing$ exactly when the local basis function $\phi_{D,r}$ does not contribute to the global basis $\mathbf{\Phi}_{\mathcal{D}}$. This happens when $\phi_{D,r}$ is

(a) a vertex basis function on the domain boundary;

(b) an edge basis function on the domain boundary;

(c) an edge basis function in the domain interior, and the local complexity of the neighbouring element is too low (e.g., the situation in Example 3.1.5).

Algorithm 3.2.12. We construct this local-to-global mapping by setting $N := 0$, and traverse the elements in some fixed order, assessing for each local degree of freedom r whether or not it should receive a global index (the conditions (a), (b), and (c) above). If not, we assign $i_D(r) := \varnothing$. Else, we distinguish two cases: Whether or not this local basis function contributes to the same global basis function as some other (D', r') that we have seen before. If it has, we set $i_D(r) := i_{D'}(r')$. Else, we assign $i_D(r) := N$, and increase N.

A precise formulation is in Algorithm B.0.1 in Appendix B. ◊

3.3. Local bases

We saw in the previous section that equipping each element in a triangulation $\mathcal{D}$ with local bases Φ_D can lead to a global basis $\mathbf{\Phi}_{\mathcal{D}}$ for the Galerkin space $V(\mathcal{D})$. Almost always, this local basis is constructed once on a *reference element domain* $\hat{K}$, and the other local bases are derived from this *reference basis* through the following construction.

Definition 3.3.1. Take a reference element domain $\hat{K}$. On this reference element, construct a family of bases $(\hat{\Phi}_p)_p$. For any element domain $K \in \mathfrak{K}$, we can find an invertible affine mapping $F_K : \hat{K} \to K$ such that $K = F(\hat{K})$. On an *hp*-element $D := (K, d)$, the set

$$\Phi_D := \left\{ \hat{\phi} \circ F_K^{-1} : \hat{\phi} \in \hat{\Phi}_{p(d)} \right\} \tag{3.3.2}$$

is a basis for $\mathbb{P}_{p(d)}(K)$. ◊

Corollary 3.3.3. *Any element constructed like this is automatically* affine equivalent *to the* reference element $(\hat{K}, d)$.

3.3.1. Elementwise decomposition

Definition 3.3.4. In (1.2.20), we saw that constructing the Galerkin solution $u_{\mathcal{D}}$ involves a *stiffness matrix* and a *load vector*. Let $\boldsymbol{\Phi}_{\mathcal{D}}$ be a basis for the finite element space $V(\mathcal{D})$ constructed through the local-to-global mapping; then these quantities are defined as

$$\boldsymbol{A} := \left[\langle \nabla \boldsymbol{\phi}_j, \nabla \boldsymbol{\phi}_i \rangle_{L^2(\Omega)} \right]_{i,j=1}^{\#\boldsymbol{\Phi}_{\mathcal{D}}}, \quad \boldsymbol{b} := \left[\langle f, \boldsymbol{\phi}_i \rangle_{L^2(\Omega)} \right]_{i=1}^{\#\boldsymbol{\Phi}_{\mathcal{D}}}. \qquad \text{◊}$$

Take the load vector as an example. Note that $\boldsymbol{b}$ decomposes into contributions on each element locally, in that

$$\boldsymbol{b} = \sum_{D \in \mathcal{D}} \tilde{b}^D, \quad \tilde{b}_i^D = \sum_{D \in \mathcal{D}} \langle f, \boldsymbol{\phi}_i \rangle_{L^2(K_D)}.$$

By the construction of $\boldsymbol{\Phi}_{\mathcal{D}}$, its basis functions have local support (cf. Figure 3.1.4), so these vectors $\tilde{b}^D$ are sparse. Moreover, we can associate with each $\tilde{b}^D$ an *element load vector*

$$b_r^D := \langle f, \phi_r^D \rangle_{L^2(K)} \implies \tilde{b}_{i_D(r)}^D = b_r^D \quad (1 \le r \le \lfloor d \rfloor_\triangle).$$

Note that in general, b^D—with only $\lfloor d \rfloor_\triangle$ entries—is much smaller than $\tilde{b}^D$.

If the forcing function $f|_K$ is locally a polynomial, then we can write it as $f|_K = \mathbf{f}_D^\top \Phi_D$. With this, we see that the element load vector reduces to

$$b_r^D = \mathbf{f}_D^\top \operatorname{col}_r M^D, \tag{3.3.5}$$

where M^D is the *element mass matrix*

$$M_{rs}^D = \langle \phi_r^D, \phi_s^D \rangle_{L^2(K)} \quad (1 \le r, s \le \lfloor d \rfloor_\triangle).$$

Similarly, one can derive the *element stiffness matrix* to be

$$A_{rs}^D = \langle \nabla \phi_r^D, \nabla \phi_s^D \rangle_{L^2(K)} \quad (1 \le r, s \le \lfloor d \rfloor_\triangle).$$

Proposition 3.3.6. *The element mass matrix is, for each hp-element, invertible. The element stiffness matrix is singular, for each hp-element.*

3.3.2. Expressing polynomials on refined triangulations

In (3.3.5), we saw that when a function v is a polynomial locally on $D = (K, d)$, there is an appropriate vector $\mathbf{v}_D \in \mathbb{R}^{\lfloor d \rfloor_\triangle}$ of its coefficients with respect to the basis Φ_D. Then v is also a polynomial on any (h- or p-)refinement $D' = (K', d')$ of D, so there is some vector $\mathbf{v}_{D'} \in \mathbb{R}^{\lfloor d' \rfloor_\triangle}$ that describes its coefficients with respect to this basis. We are interested in finding this vector, as it is needed in a number of places in our algorithm. Let us write $\triangle_p$ for the pth triangle number; then $\dim \mathbb{P}_p = \triangle_p$.

Definition 3.3.7. With $D := (K, d)$, define the *p-transfer mapping*

$$T_D^p : \mathbb{R}^{\triangle_p} \to \mathbb{R}^{\triangle_{p+1}} : \mathbf{v}_D \mapsto \mathbf{v}_{D'}, \quad D' := (K, \triangle_{p+1})$$

and for $k = 1, 2$ the *h-transfer mappings*

$$T_D^{h,k} : \mathbb{R}^{\triangle_p} \to \mathbb{R}^{\triangle_p} : \mathbf{v}_D \mapsto \mathbf{v}_{D^k}, \quad D^k := (K^k, \triangle_p)$$

where K^1, K^2 are the left resp. right child of K after bisection.

Any polynomial $v \in \mathbb{P}_p(K)$ can be written in terms of the induced local basis on any refinement of D by successive composition of these mappings. ♢

Example 3.3.8. These transfer mappings are used in a number of places inside the *hp*-adaptive algorithm. For instance, by defining the forcing function vectors $\mathbf{f}_D$ on an initial triangulation $\mathcal{D}_0$, we can derive the corresponding vectors on any *hp*-element D found from $\mathcal{D}_0$ through h- or p-refinement, allowing us to efficiently compute the element load vectors from (3.3.5).

The module *hp*-NearBest also relies on these mappings for computing the error functionals, as we will see in Chapter 4. ♢

Proposition 3.3.9. *The mappings T_D^p and $T_D^{h,k}$ are linear maps, so we can write them as matrices. Viewing a local basis Φ_D as a column vector of functions, the following equalities hold:*

$$(T_D^p)^\top \Phi_{D'} = \Phi_D, \quad (T_D^{h,k})^\top \Phi_{D^k} = \Phi_D|_{K^k} \quad (k = 1, 2, \quad D' \text{ and } D^k \text{ as above}).$$

Proof. The first assertion holds trivially. We will prove the equality for the p-transfer matrix; the argument for the h-transfer matrices is analogous.

We know that for any $v \in \mathbb{P}_p(K)$, there are vectors $\mathbf{v}_D, \mathbf{v}_{D'}$ such that

$$v = \mathbf{v}_D^\top \Phi_D = \mathbf{v}_{D'}^\top \Phi_{D'} \quad \text{and} \quad \mathbf{v}_{D'} = T_D^p \mathbf{v}_D,$$

so

$$v = (T_D^p \mathbf{v}_D)^\top \Phi_{D'}.$$

This is for all v, therefore also for all $\phi \in \Phi_D$, so

$$\Phi_D = (T_D^p)^\top \Phi_{D'}.$$ □

Proposition 3.3.10. *The matrices T_D^p and $T_D^{h,k}$ are invariant under D; we will write T^p and $T^{h,k}$ instead.*

Proof. We show the argument for the p-transfer matrix; the argument for the h-transfer matrices is analogous.

Using the above proposition, $(T_D^p)^\top \Phi_{D'} = \Phi_D$ for all hp-elements D. Moreover, Φ_D is found from the reference basis through $\Phi_D = \hat{\Phi}_p \circ F_K^{-1}$.

Then we see that

$$(T_D^p)^\top \Phi_{D'} = \Phi_D = \hat{\Phi}_p \circ F_K^{-1} \quad \text{and} \quad (T_D^p)^\top \Phi_{D'} = (T_D^p)^\top (\hat{\Phi}_{p+1} \circ F_K^{-1}) = ((T_D^p)^\top \hat{\Phi}_{p+1}) \circ F_K^{-1},$$

so that

$$\hat{\Phi}_p \circ F_K^{-1} = ((T_D^p)^\top \hat{\Phi}_{p+1}) \circ F_K^{-1} \implies \hat{\Phi}_p = (T_D^p)^\top \hat{\Phi}_{p+1}.$$

Now, we know that $\hat{\Phi}_p = (T^p_{\hat{D}_p})^\top \hat{\Phi}_{p+1}$ with the reference hp-element $\hat{D}_p := (\hat{K}, \triangle_p)$, so we conclude that $T_D^p = T^p_{\hat{D}_p}$ for each hp-element D. Therefore $T^p := T^p_{\hat{D}_p}$ is *the* p-transfer matrix, independent of D. □

Proposition 3.3.11. *The h-transfer matrices are invertible.*

Proof (sketch). Any polynomial on a child triangle can be uniquely extended to the parent triangle. Expressed in terms of local bases, this is exactly the inverse operation of applying the h-transfer matrix. Therefore this inverse must exist. □

3.3.3. Conditioning of a local basis

Definition 3.3.12. The condition number $\kappa(A)$ of an invertible matrix A gives a bound on how inaccurate the solution x of the linear system $Ax = b$ will be after approximation. It is defined as

$$\kappa(A) := \|A\|\|A^{-1}\|$$

for any matrix norm $\|\cdot\|$. In our case, we will use the 2-norm, and

$$\kappa(A) = \frac{\sigma_{\max}(A)}{\sigma_{\min}(A)};$$

the condition number is the quotient of the largest and smallest singular values. ♢

On a finite-memory machine, we express real numbers in *floating-point representation* which introduces rounding errors, so this condition number is instrumental in the analysis of numerical methods.

Corollary 3.3.13. *Note that by Propositions 3.3.6 and 3.3.11, the mass-, and h-transfer matrices are invertible, and hence have a condition number.*

The element stiffness matrix A^D is singular, so $\kappa(A^D) = \infty$. Still, we can get an idea of the "conditioning" of the element stiffness matrices by considering the *Moore-Penrose inverse* of A^D.

Theorem 3.3.14 ([28, Thm. 2.1])**.** *For each square matrix $A \in \mathbb{R}^{n\times n}$, there is a unique matrix A^+ such that*

$$AA^+A = A, \quad A^+AA^+ = A^+, \quad (AA^+)^\top = AA^+, \quad (A^+A)^\top = A^+A;$$

this matrix is the Moore-Penrose inverse *of A.*

Definition 3.3.15. We can extend the definition of the condition number to singular matrices by defining

$$\tilde{\kappa}(A) := \|A\|_2 \left\|A^+\right\|_2 .$$

Of course, when A is invertible, $A^+ = A^{-1}$ so that $\tilde{\kappa}(A) = \kappa(A)$.

One can show [28, (2.7)] that $\left\|A^+\right\|_2 = \max\left\{1/\sigma : \sigma \text{ singular value of } A, \sigma > 0\right\}$; its 2-norm is the reciprocal of the smallest positive singular value. Therefore

$$\tilde{\kappa}(A) = \frac{\sigma_{\max}(A)}{\sigma_{\min>0}(A)}.$$

In subsequent analyses, we will use $\tilde{\kappa}$ instead of κ to compute condition numbers. ◊

Definition 3.3.16. Given a reference triangle $\hat{K}$, a family of local reference bases $(\hat{\Phi}_p)_{p\in\mathbb{N}}$ on $\hat{K}$ is called *well-conditioned* when the condition number of the h-transfer matrices, and reference mass- and stiffness matrices have a *favourable* growth rate with respect to p.

There is no exact definition of favourable, but generally $\mathcal{O}(p^N)$ for some N is good. ◊

Remark 3.3.17. It must be noted that conditioning of the basis is often measured numerically, rather than through rigorous proofs. One generally computes condition numbers of the desired quantities up to a suitably high degree p, and extrapolates from this the behaviour for $p \to \infty$. ◊

3.4. Examples of hierarchical elements

We saw that certain families of local bases can induce a global basis for the Galerkin space, and have looked at properties of general local bases, but we haven't actually seen any examples. This is the topic of the remainder of this chapter.

3.4.1. The Szabó-Babuška basis

Between 1989 and 1991, Szabó and Babuška [6] [45, §6.2] derived the first hierarchical basis. The shape functions are based on the *Legendre polynomials*, which have favourable orthogonality properties.

Definition 3.4.1. The *Legendre polynomial* $P_p \in \mathbb{P}_p([-1,1])$ of strict degree p is recursively defined as

$$P_0(x) := 1, \quad P_1(x) := x, \quad p \cdot P_p(x) := 2p \cdot x \cdot P_{p-1}(x) - (p-1)P_{p-2}(x) \ \ (p \geq 2). \qquad \Diamond$$

Theorem 3.4.2. *The Legendre polynomials are orthogonal with respect to the $L^2([-1,1])$-inner product:*

$$\int_{-1}^{1} P_p(x)P_q(x)\,\mathrm{d}x = \frac{2}{2p+1}\delta_{pq}.$$

Recall the assumptions of a local hierarchical basis in §3.2.1. We will define the hierarchical Szabó-Babuška basis one type at a time, all in terms of the barycentric coordinates $\boldsymbol{\lambda}$ on a triangle K.

Definition 3.4.3. The first three basis functions are the degree-1 vertex functions defined as

$$\phi_{K,r}(\boldsymbol{\lambda}) := \lambda_r, \quad (1 \leq r \leq 3).$$

They are one in the rth vertex, and vanish in the others, and therefore necessarily satisfy (3.2.5). ◊

Proposition 3.4.4. *For $q \geq 2$, the function*

$$\mathfrak{E}_q : [-1,1] \to \mathbb{R} : x \mapsto -\frac{8\sqrt{4q-2}}{q(q-1)} P'_{q-1}(x)$$

is a polynomial of strict degree $q-2$.

Definition 3.4.5. On each edge $e \in \mathcal{E}(\mathcal{D})$, we fix a global orientation $e = \text{hull}(\mathbf{v}_1^e, \mathbf{v}_2^e)$ and local barycentric coordinates

$$\left\{(\lambda_1^e, \lambda_2^e) : 0 \leq \lambda_1^e, \lambda_2^e \leq 1, \lambda_1^e + \lambda_2^e = 1\right\}.$$

On this edge, we define the sequence $(\phi_{e,q})_{q\geq 2}$ through

$$\phi_{e,q}(\lambda_1^e, \lambda_2^e) := \lambda_1^e \lambda_2^e \mathfrak{E}_q(\lambda_2^e - \lambda_1^e).$$

By virtue of the $\lambda_1^e\lambda_2^e$-factor, these functions vanish in the vertices of e. Moreover, they are of strict degree q, so that (3.2.4) holds.

On a triangle $K := \text{hull}(\mathbf{v}_1^K, \mathbf{v}_2^K, \mathbf{v}_3^K)$, defining an edge shape function $\phi_{K,e_1,q}$ of the edge $e_1 := \text{hull}(\mathbf{v}_1^{e_1}, \mathbf{v}_2^{e_1})$ opposite vertex 1 requires some care. Noting that $e_1 = \text{hull}(\mathbf{v}_2^K, \mathbf{v}_3^K)$, it is very well possible that the global orientation of e_1 disagrees with its local orientation on K; take for example K_2 in Figure 3.4.6.

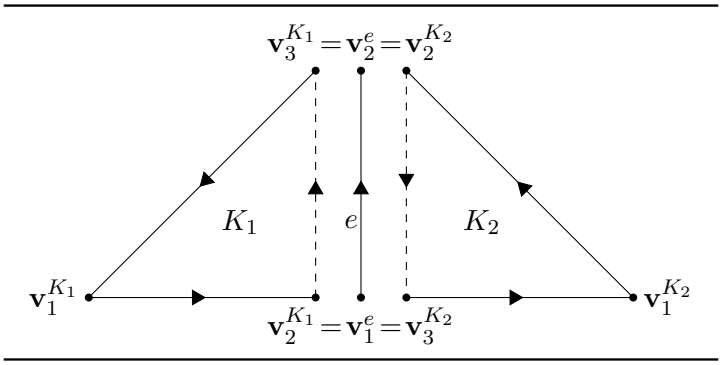


Figure 3.4.6.: Two neighbouring triangles with edge $e := K_1 \cap K_2$. Arrows indicate orientation. The global orientation of edge $e := \text{hull}(\mathbf{v}_1^e, \mathbf{v}_2^e)$ agrees with its local orientation on K_1, but not on K_2.

In order to guarantee that $\phi_{K,e_1,q}|_{e_1} = \phi_{e_1,q}$, we may have to correct for this by inverting the local orientation, effectively swapping the barycentric coordinates λ_2 and λ_3. More formally, we define

$$\phi_{K,e_1,q}(\boldsymbol{\lambda}) := \begin{cases} \lambda_2\lambda_3\mathfrak{E}_q(\lambda_3 - \lambda_2) & \text{when } \mathbf{v}_2^K = \mathbf{v}_1^{e_1}, \\ \lambda_2\lambda_3\mathfrak{E}_q(\lambda_2 - \lambda_3) & \text{when } \mathbf{v}_3^K = \mathbf{v}_1^{e_1}. \end{cases}$$

Note that $\phi_{K,e_1,q}$ vanishes on both edges emanating from vertex 1, and equals $\phi_{e_1,e}$ along e_1. The definitions of the other edge shape functions are analogous; this satisfies (3.2.6). ◊

Fixing an orientation along each edge has the effect that bases on neighbouring triangles are not necessarily affine equivalent to each other. We explore this in further detail in §4.3.

Definition 3.4.7. The collection of face functions of degree q is defined as

$$\Phi^{\circ}_{K,q} := \left\{\phi_{q,r}(\boldsymbol{\lambda}) := \lambda_1\lambda_2\lambda_3\mathcal{F}_{q,r}(\boldsymbol{\lambda}), \quad 0 \le r \le q-3\right\}$$

where

$$\mathcal{F}_{q,r}(\boldsymbol{\lambda}) := P_r(\lambda_2 - \lambda_1)P_{q-3-r}(2\lambda_3 - 1).$$

Each face function is a polynomial of strict degree q, and necessarily vanishes on all edges of the triangle. They are all linearly independent from each other; the set of all such face functions therefore satisfies (3.2.7). ◊

Even though their efforts were important in the development of later hierarchical bases, the Szabó-Babuška basis is not well-conditioned, as the following result shows.

Example 3.4.8. Looking at Figure 3.4.9, we see that for p between 1 and 20, the condition numbers exhibit quicker-than-polynomial growth. For large p, the matrices even break down numerically, and Mathematica [48] is no longer able to compute their condition numbers. We conclude that this family of bases is not well-conditioned.

The bottom graph shows that the stiffness matrix is, for large p, almost a full matrix without any sparsity. Thanks to orthogonality of the Legendre polynomials, the mass matrix exhibits a fair level level of sparsity, but this does not help dampen its condition number; the same holds for the h-transfer matrix. (The two h-transfer matrices have virtually equal condition number and sparsity percentage, so we chose to only consider one—in our case, $T_p^{h,1}$.) ◊

The efforts of Szabó and Babuška were more focused on exploring the theory of hierarchical finite elements, and in the original paper [6], it can be read that they acknowledge the fact that their example basis is imperfect. They mention an abstract "optimal" hierarchical basis "which exact structure is not known." Let us look at one improvement.

3.4.2. The Aiffa basis

In later years, multiple researchers have improved the results of the Szabó-Babuška basis in terms of conditioning. In 2001, Aiffa *et al.* [2] published a paper deriving the first hierarchical basis with well-conditioned reference stiffness matrix.

Their contribution is very simple, with enormous effects. Note that for large p, the majority of the Szabó-Babuška basis is made up of face shape functions. These face shape functions have a lot of interaction with each other. Removing these interactions by simply orthogonalizing the face shape functions (with respect to the bilinear form) decreases the condition number dramatically.

More formally, denote the Szabó-Babuška face shape functions of degree up to p with $\Phi^{\circ,\mathsf{SB}}_{K,\le p} := \cup_{q=3}^{p}\Phi^{\circ}_{K,q}$. Viewing it as a vector, we can write

$$\Phi^{\circ,\mathsf{SB}}_{K,\le p} = \left\{\phi^{\mathsf{SB}}_r : 1 \le r \le \triangle_{p-2}\right\}.$$

We define

$$\Phi^{\circ,\mathsf{Aif}}_{K,\le p} := G\Phi^{\circ,\mathsf{SB}}_{K,\le p}$$

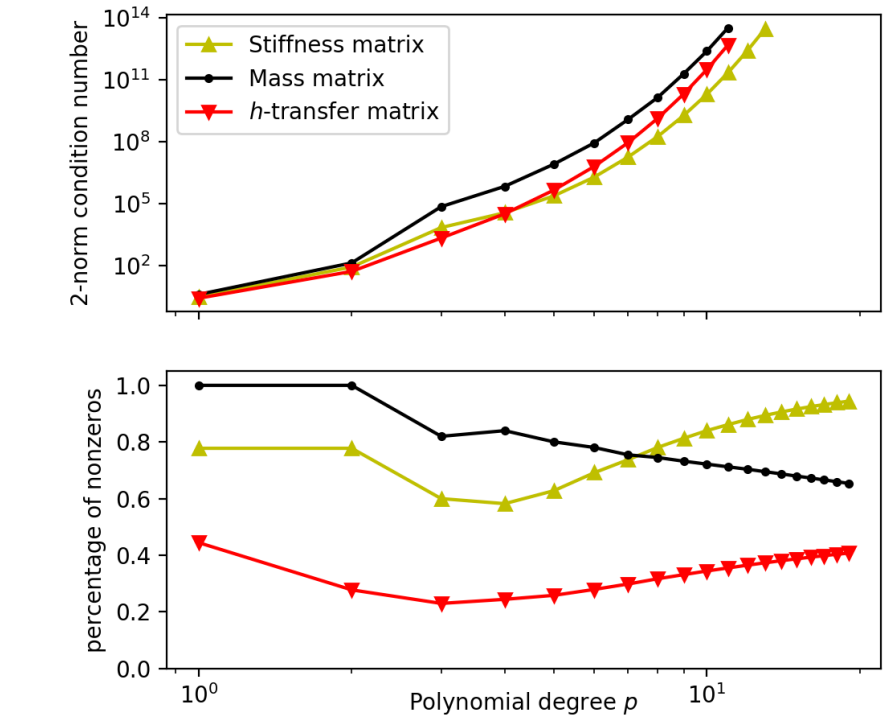


Figure 3.4.9.: Conditioning and sparsity of the Szabó-Babuška basis for $p = 1, \ldots, 20$. We investigate the h-transfer matrix, and the element mass- and stiffness matrices. Top: condition numbers of the three matrices of interest. Bottom: percentage of nonzero elements.

where G is a linear transformation such that

$$a(\phi_r^{\mathsf{Aif}}, \phi_s^{\mathsf{Aif}}) = \delta_{rs} \quad (1 \leq r, s \leq \triangle_{p-2}).$$

This is usually done though the Gram-Schmidt process; cf. Algorithm 3.4.10. This process retains the hierarchical structure of the basis when the input face shape functions are ordered from lower to higher degree.

Example 3.4.11. Look at Figure 3.4.12. We see that with respect to the Szabó-Babuška basis, condition numbers of the element mass- and stiffness matrix are greatly reduced and no longer grow exponentially. The percentage of nonzeros of the stiffness matrix is also greatly improved, at the cost of sparsity for the mass matrix. This is fine; the element mass matrices are only used in intermediate local computations. Element stiffness matrices however are used to build the global stiffness matrix, which we use to solve a very large linear system. Any gain in sparsity saves us memory; any gain in conditioning improves our final solution.

What we *do* see however, is the exponential growth of the h-transfer matrix condition number, **making the Aiffa basis not well-conditioned either**. ◊

The Aiffa basis looks like a strong candidate for becoming our local basis. By its hierarchicality, we satisfy the conditions of Theorem 3.2.11 so that this local basis leads to a global basis for $V(\mathcal{D})$.

1: $\phi_r := \phi_r^{\mathsf{SB}}$, for $1 \leq r \leq \triangle_{p-2}$;
2: $\phi_1^{\mathsf{Aif}} := \phi_1 / \sqrt{a(\phi_1, \phi_1)}$; ▷ Normalize ϕ_1
3: **for** $r = 2, \ldots, \triangle_{p-2}$ **do**
4: **for** $s = 1, \ldots, k-1$ **do**
5: $\beta_{rs} := a(\phi_r, \phi_s)$; ▷ Interaction between ϕ_r and ϕ_s
6: $\phi_r = \phi_r - \beta_{rs}\phi_s$; ▷ Orthogonalize ϕ_r wrt. ϕ_s
7: $\phi_r^{\mathsf{Aif}} := \phi_r / \sqrt{a(\phi_r, \phi_r)}$; ▷ Normalize ϕ_r
8: **return** $\Phi_{K,\leq p}^{\circ,\mathsf{Aif}}$.

Algorithm 3.4.10: The Gram-Schmidt orthogonalization process.

3.4.3. A downside to precomputing the matrices

Even though the basis is not well-conditioned, the previous example showed that it is, at least, a lot better than the previous Szabó-Babuška basis. There is however one glaring hole: It is really difficult to compute the element mass-, stiffness-, and h-transfer matrices. We therefore resort to precomputing them on a reference element, and apply linear transformations to these precomputed arrays in real-time to find the necessary quantities. We will look at this more carefully in Chapter 4, but it does mean that there is an upper limit on the maximal polynomial degree that can be present in an hp-triangulation. Any such limit immediately voids the optimality results of Chapter 2. This motivates the final section of this chapter.

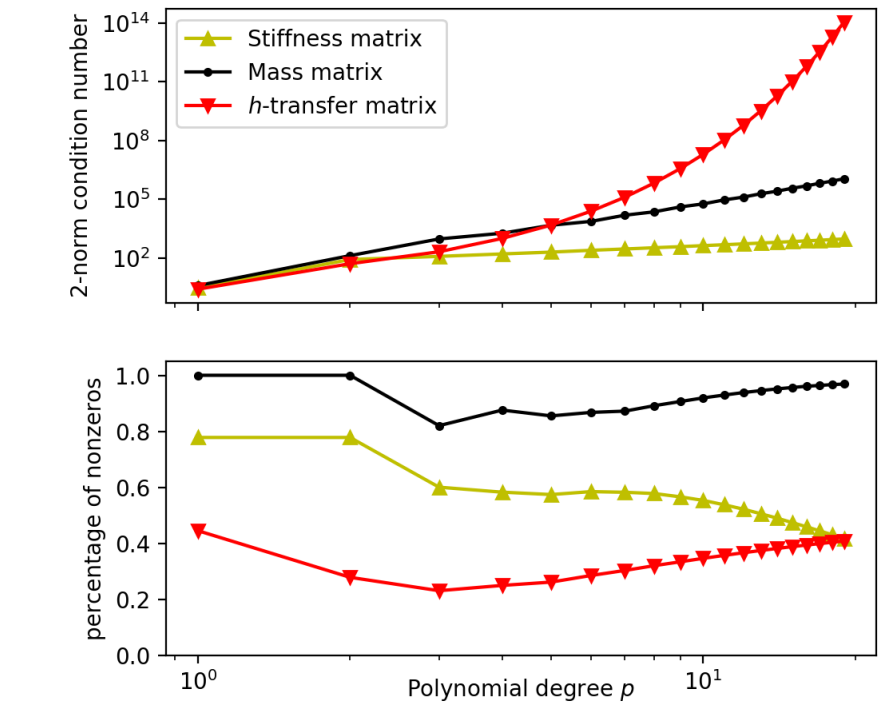


Figure 3.4.12.: Conditioning and sparsity of the Aiffa basis for $p = 1, \ldots, 20$. We investigate the h-transfer matrix, and the element mass- and stiffness matrices. Top: condition numbers of the three matrices of interest. Bottom: percentage of nonzero elements.

3.5. Bernstein-Bézier elements

Just over a hundred years ago, in 1912, Bernstein [8] introduced a new basis for the univariate polynomials of certain degree. The Bernstein polynomials were designed for a constructive proof of the Weierstrass approximation theorem. A sequence of such polynomials is constructed, each a better approximation to the intended function than its predecessor. The slow convergence rate of this sequence, and the lack of digital computers to efficiently construct the Bernstein polyomials, caused the Bernstein basis to be of mainly *theoretical* rather than *practical* importance.

In the early 1960s, two engineers in the car industry—Paul de Faget de Casteljau of Citroën and Pierre Étienne Bézier of Renault—found a different application of the Bernstein basis. Interested in formulating mathematical tools that would allow designers to intuitively construct and manipulate complex shapes such as automobile bodies, de Casteljau and Bézier came across the Bernstein polynomials as the foundation of the *Bézier curves* now ubiquitous in computer aided geometric design. De Casteljau was responsible for formulating a recursive algorithm allowing one to evaluate a multivariate Bernstein polynomial; Bézier popularized them.

More recently, Ainsworth [4] has expressed interest in using the Bernstein-Bézier or BB-basis in an *hp*-adaptive finite element context. The main problem with higher-degree hierarchical bases is their reliance on precomputed arrays—see §3.4.3, which voids guaranteed exponential convergence of *hp*-AFEM. The BB-basis allows computing any necessary matrix *in real-time, without the use of a reference element.* This sounds amazing, but it does come at the expense of hierarchicality, so p-enrichment becomes harder. In this section, we will first construct a basis for the full polynomial space $\mathbb{P}_p(K)$, and defer the nonuniform degree case to the final paragraph.

3.5.1. Bernstein polynomials

Recall the domain- and lattice points from Definition 1.2.5:

$$\mathcal{D}^p(K) := \{\mathbf{v}_{\boldsymbol{\alpha}} : \boldsymbol{\alpha} \in \mathcal{I}^p\}, \quad \mathcal{I}^p := \left\{\boldsymbol{\alpha} \in \mathbb{N}_0^3 : |\boldsymbol{\alpha}| = p\right\}$$

where $\mathbf{v}_{\boldsymbol{\alpha}}$ is point inside K corresponding with the barycentric coordinates $p^{-1}\boldsymbol{\alpha}$.

Definition 3.5.1. Each $\boldsymbol{\alpha} \in \mathcal{I}^p$ corresponds with a Bernstein polynomial on $\hat{K}$ defined in terms of the barycentric coordinates as

$$B^p_{\boldsymbol{\alpha}}(\boldsymbol{\lambda}) := \binom{p}{\boldsymbol{\alpha}} \boldsymbol{\lambda}^{\boldsymbol{\alpha}}, \quad \boldsymbol{\lambda}^{\boldsymbol{\alpha}} := \prod_{i=1}^{3} \lambda_i^{\alpha_i}$$

where

$$\binom{p}{\boldsymbol{\alpha}} := \frac{p!}{\alpha_1!\alpha_2!\alpha_3!}$$

is a multinomial coefficient.

By convention, when $\alpha_i = 0$, then $\lambda_i^{\alpha_i} := 1$, even when $\lambda_i = 0$. ◊

By definition, a Bernstein polynomial $B^p_{\boldsymbol{\alpha}}$ for which $\alpha_1, \alpha_2, \alpha_3 > 0$ must vanish on the boundary. Conversely, when $\alpha_i = 0$, the function $B^p_{\boldsymbol{\alpha}}$ does *not* vanish on the edge opposite vertex i. The following properties help our understanding of the general shape of these polynomials.

Proposition 3.5.2 ([32, Thm. 2.5])**.** *The Bernstein polynomial $B^p_{\boldsymbol{\alpha}}$ attains its unique maximum at the domain point $\mathbf{v}_{\boldsymbol{\alpha}}$ associated with its lattice point $\boldsymbol{\alpha}$. More formally:*

$$B^p_{\boldsymbol{\alpha}}(\mathbf{v}_{\boldsymbol{\alpha}}) > B^p_{\boldsymbol{\alpha}}(\boldsymbol{\lambda}) \quad (\boldsymbol{\lambda} \neq \mathbf{v}_{\boldsymbol{\alpha}} \in \mathcal{D}^p(\hat{K})).$$

Proposition 3.5.3 ([4])**.** *The Bernstein polynomials satisfy*

$$B^p_{\boldsymbol{\alpha}} \in \mathbb{P}_p(\hat{K}), \quad B^p_{\boldsymbol{\alpha}}(\boldsymbol{\lambda}) \geq 0 \quad (\boldsymbol{\lambda} \in \hat{K}, \boldsymbol{\alpha} \in \mathcal{I}^p) \quad \textit{and} \quad \sum_{\boldsymbol{\alpha} \in \mathcal{I}^p} B^p_{\boldsymbol{\alpha}}(\boldsymbol{\lambda}) = 1 \quad (\boldsymbol{\lambda} \in \hat{K}).$$

In words, $B^p_{\boldsymbol{\alpha}}$ is a polynomial of degree p that is nonnegative on its entire domain, and the set of all Bernstein polynomials of given degree sums to unity.

The following result makes the Bernstein polynomials useful in a finite element setting.

Theorem 3.5.4 ([32, Thm. 2.4])**.** *The set $\hat{\Phi}^{\mathsf{BB}}_p := \left\{ B^p_{\boldsymbol{\alpha}} : \boldsymbol{\alpha} \in \mathcal{I}^p \right\}$ of all Bernstein polynomials of given degree forms a basis for $\mathbb{P}_p(\hat{K})$.*

Definition 3.5.5. By the above theorem, $\hat{\Phi}^{\mathsf{BB}}_p$ is a basis for $\mathbb{P}_p(\hat{K})$. Its enumeration is through the lexicographic ordering of the multi-indices $\boldsymbol{\alpha} \in \mathcal{I}^p$. This defines the Bernstein-Bézier reference element. ◊

Definition 3.5.6. The usual invertible affine mapping $F_K : \hat{K} \to K$ can be used to transform the reference basis functions $B^p_{\boldsymbol{\alpha}}$ to Bernstein polynomials on arbitrary triangles through

$$B^{K,p}_{\boldsymbol{\alpha}} := B^p_{\boldsymbol{\alpha}} \circ F_K^{-1},$$

thereby defining the BB-basis Φ^{BB}_D for hp-elements $D := (K, d)$ when $p(d) = p$. ◊

3.5.2. De Casteljau algorithm

Imagine having found a finite element solution using the BB-basis, and wanting to visualize this solution. A common starting point is to evaluate the solution on a large number of points in the problem domain. How can we evaluate this Galerkin solution?

Of course, on each hp-element $D := (K, d)$ separately, it is a polynomial. We have its local coefficient vector $\mathbf{u}^{(p)}$ so that

$$u|_K = \sum_{\boldsymbol{\alpha} \in \mathcal{I}^p} \mathbf{u}^{(p)}_{\boldsymbol{\alpha}} B^{K,p}_{\boldsymbol{\alpha}}.$$

To evaluate u at a point $\mathbf{x} \in K$ with barycentric coordinates $\boldsymbol{\lambda}$, we use the *de Casteljau* algorithm. The algorithm is based on the recurrence relation

$$B^{K,p}_{\alpha}(\boldsymbol{\lambda}) = \sum_{n=1}^{3} \lambda_n B^{K,p-1}_{\boldsymbol{\alpha}-\boldsymbol{e}_n}(\boldsymbol{\lambda}) \tag{3.5.7}$$

where we drop any term involving negative multi-indices. Inserting this into the original formulation of $u|_K$, we see that

$$u|_K(\boldsymbol{\lambda}) = \sum_{\boldsymbol{\alpha} \in \mathcal{I}^{p-1}} \mathbf{u}^{(p-1)}_{\boldsymbol{\alpha}} B^{K,p-1}_{\boldsymbol{\alpha}}(\boldsymbol{\lambda}),$$

where

$$\mathbf{u}_{\boldsymbol{\alpha}}^{(p-1)} := \sum_{n=1}^{3} \lambda_n \mathbf{u}_{\boldsymbol{\alpha}+\boldsymbol{e}_n}^{(p)} \quad (\boldsymbol{\alpha} \in \mathcal{I}^{p-1}).$$

Note that the values $\mathbf{u}_{\boldsymbol{\alpha}}^{(q-1)} = \mathbf{u}_{\boldsymbol{\alpha}}^{(q-1)}(\boldsymbol{\lambda})$ are dependent on the barycentric coordinates $\boldsymbol{\lambda}$, and are not simply constants. The de Casteljau algorithm then consists of iterating this process to obtain the zeroth order representation

$$u|_K(\boldsymbol{\lambda}) = \mathbf{u}_{\mathbf{0}}^{(0)} B_{\mathbf{0}}^{K,0}(\boldsymbol{\lambda}) = \mathbf{u}_{\mathbf{0}}^{(0)},$$

which contains the evaluation of $u|_K$ in $\boldsymbol{\lambda}$ on the right-hand side.

We end this paragraph with a precise formulation of the *de Casteljau* algorithm in Algorithm 3.5.8.

```
1: function Evaluate(u^(p), λ)
2:     for q = p, ..., 1 do
3:         for α ∈ I^(q-1) do
4:             u_α^(q-1) := Σ_{n=1}^3 λ_n u_{α+e_n}^(q);
5:     return u_0^(0).
```

Algorithm 3.5.8: De Casteljau algorithm for evaluating a polynomial in Bernstein-Bézier form.

3.5.3. Computing mass- and stiffness matrices

In a previous paragraph, we derived a basis Φ_D^{BB} given an *hp*-element $D := (K, d)$. Inside the *hp*-AFEM algorithm, we don't really need the polynomials itself: It is the interactions between them that matter. Of course, it is possible to find, for instance, the element mass matrix cells by manually computing $\int \phi_r \phi_s$—this is the route we took in the hierarchical case. However, as it turns out, the Bernstein basis allows us to do something much more beautiful.

In this paragraph, we will express the element mass- and stiffness matrices on the *hp*-element D *directly*, without the involvement of a reference element. The result is that we don't have to rely on precomputed arrays and can therefore scale our algorithm to $p \to \infty$.

Definition 3.5.9. The *binomial coefficient* is defined in terms of multinomial coefficients as

$$\binom{p}{q} := \binom{p}{(q, p-q)}, \quad (p \in \mathbb{N}, \quad 0 \le q \le p).$$

Moreover, we define

$$\binom{\boldsymbol{\alpha}}{\boldsymbol{\beta}} := \prod_{n=1}^{3} \binom{\alpha_n}{\beta_n}, \quad (\boldsymbol{\alpha}, \boldsymbol{\beta} \in \mathbb{N}_0^3, \quad \beta_n \le \alpha_n \forall n). \qquad \Diamond$$

The following properties make the Bernstein polynomials truly remarkable.

Proposition 3.5.10 ([4, (2.6)])**.** *The product of Bernstein polynomials is again a (scaled) Bernstein polynomial:*

$$B_{\boldsymbol{\alpha}}^{K,p} B_{\boldsymbol{\beta}}^{K,q} = \frac{\binom{\boldsymbol{\alpha}+\boldsymbol{\beta}}{\boldsymbol{\alpha}}}{\binom{p+q}{p}} B_{\boldsymbol{\alpha}+\boldsymbol{\beta}}^{K,p+q}.$$

Proposition 3.5.11 ([4, (2.7)])**.** *The integral of a Bernstein polynomial on K satisfies*

$$\int_K B_{\boldsymbol{\alpha}}^{K,p}(\mathbf{x})\,\mathrm{d}\mathbf{x} = \frac{\mathrm{vol}(K)}{\binom{p+2}{2}}.$$

Lemma 3.5.12 ([4, (2.4)])**.** *If $K = \mathrm{hull}(\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3)$ with $\mathbf{v}_n = (x_n, y_n)$, the gradients of the barycentric coordinates satisfy*

$$\nabla\lambda_1 = \begin{bmatrix} y_3 - y_1 \\ x_1 - x_3 \end{bmatrix}, \quad \nabla\lambda_2 = \begin{bmatrix} y_1 - y_2 \\ x_2 - x_1 \end{bmatrix}, \quad \nabla\lambda_3 = \begin{bmatrix} y_2 - y_3 \\ x_3 - x_2 \end{bmatrix}.$$

Proposition 3.5.13. *The gradient of a Bernstein polynomial satisfies*

$$\nabla B_{\boldsymbol{\alpha}}^{K,p}(\mathbf{x}) = p \sum_{n=1}^{3} B_{\boldsymbol{\alpha}-\boldsymbol{e}_n}^{K,p-1}(\mathbf{x}) \nabla\lambda_n$$

where we again adopt the convention dropping any term involving negative multi-indices.

Corollary 3.5.14. *The element mass matrix of degree p satisfies*

$$M_{\boldsymbol{\alpha}\boldsymbol{\beta}}^{K,p} = \frac{\binom{\boldsymbol{\alpha}+\boldsymbol{\beta}}{\boldsymbol{\alpha}}}{\binom{2p}{p}} \frac{\mathrm{vol}(K)}{\binom{2p+2}{2}}.$$

Corollary 3.5.15. *The element stiffness matrix of degree p satisfies*

$$\begin{aligned} A_{\boldsymbol{\alpha}\boldsymbol{\beta}}^{K,p} &= \frac{p^2\,\mathrm{vol}(K)}{\binom{2p}{2}} \sum_{n,m=1}^{3} \frac{\binom{\boldsymbol{\alpha}-\boldsymbol{e}_n+\boldsymbol{\beta}-\boldsymbol{e}_m}{\boldsymbol{\alpha}-\boldsymbol{e}_n}}{\binom{2p-2}{p-1}} \nabla\lambda_n \cdot \nabla\lambda_m \\ &= \frac{2\,\mathrm{vol}(K)}{\binom{2p}{p}} \sum_{n,m=1}^{3} \binom{\boldsymbol{\alpha}-\boldsymbol{e}_n+\boldsymbol{\beta}-\boldsymbol{e}_m}{\boldsymbol{\alpha}-\boldsymbol{e}_n} \nabla\lambda_n \cdot \nabla\lambda_m. \end{aligned}$$

A result of the above corollaries is that the main difficulty in finding the element matrices is not numerical integration, but rather computation of the *multinomial coefficients*!

The binomial coefficients are most easily found by the following recursive formula.

Lemma 3.5.16 (Pascal triangle)**.** *The binomial coefficients satisfy*

$$\binom{p}{0} = 1 = \binom{p}{p}, \quad \binom{p}{q} = \binom{p-1}{q-1} + \binom{p-1}{q} \quad (1 \le q \le p-1).$$

Remark 3.5.17. These binomial coefficients grow quickly: It can be shown that the central binomial coefficient $\binom{2p}{p}$ grows like $\frac{4^p}{\sqrt{\pi p}}$.

In 32-bit integer arithmetic, $p = 17$ already causes overflow, and with 64 bits, this number increases to $p = 34$. Of course, our finite element solver works with floating point numbers rather

than integers, but storing the binomial coefficients in floating point representation introduces rounding errors that may become significant for large values of p.

A commonly used alternative is to represent the binomial coefficients in a *prime exponent vector*. The binomial coefficient $\binom{p}{q}$ clearly cannot possess a prime factor larger than p, so that if we order the m prime numbers $\leq p$ in a vector $(p_1, \ldots, p_m)$, there is a unique vector $(e_1, \ldots, e_m)$ such that

$$\binom{p}{q} = \prod_{i=1}^{m} p_i^{e_i}.$$

An efficient algorithm for finding this vector is described in [22]. This representation has more advantages: Multiplication and division of binomial coefficients become simple additions resp. subtractions of the prime exponent vector. ◊

3.5.4. Computing the transfer matrices

Subdivision of a triangle (h-refinement) and degree raising (p-enrichment) have many applications in Computer-Aided Geometric Design. Take for instance the visualization of a polynomial in Bernstein-Bézier representation. Being able to evaluate this polynomial in the vertices of a highly refined mesh can be hugely beneficial in this case. It is outside the scope of this thesis to fully dive into this topic, but some aspects will be useful within our finite element context as well.

Many of the algorithms with Bernstein polynomials are based on recurrence relations like (3.5.7). For p-enrichment, for instance, the following relation is useful.

Proposition 3.5.18 ([3, §3.2])**.** *Given some degree $p \in \mathbb{N}$, and some lattice point $\boldsymbol{\alpha} \in \mathcal{I}^p$, we can express the Bernstein polynomial $B^p_{\boldsymbol{\alpha}}$ in terms of the BB-basis of degree $p+1$ by*

$$B^p_{\boldsymbol{\alpha}} = \sum_{i=1}^{3} \frac{\alpha_i + 1}{p+1} B^{p+1}_{\boldsymbol{\alpha}+\boldsymbol{e}_i}$$

where $\boldsymbol{e}_i$ is the ith unit vector.

Proof. This follows from

$$\begin{aligned} B^p_{\boldsymbol{\alpha}}(\boldsymbol{\lambda}) &= \sum_{i=1}^{3} \lambda_i B^p_{\boldsymbol{\alpha}}(\boldsymbol{\lambda}) = \sum_{i=1}^{3} \lambda_i \frac{p!}{\prod_j \alpha_j!} \prod_j \lambda_j^{\alpha_j} \\ &= \sum_{i=1}^{3} \frac{\alpha_i + 1}{p+1} \frac{(p+1)!}{\prod_j (\alpha_j + \delta_{ij})!} \prod_j \lambda_j^{\alpha_j + \delta_{ij}} \\ &= \sum_{i=1}^{3} \binom{p+1}{\boldsymbol{\alpha}+\boldsymbol{e}_i} \boldsymbol{\lambda}^{\boldsymbol{\alpha}+\boldsymbol{e}_i} = \sum_{i=1}^{3} B^{p+1}_{\boldsymbol{\alpha}+\boldsymbol{e}_i}(\boldsymbol{\lambda}). \end{aligned}$$

□

Remark 3.5.19. Note that the above recursion fully characterizes the p-transfer matrix. Each column contains exactly three nonzero cells, making this matrix very sparse. For instance, a

little pen-and-paper algebra shows that

$$\begin{cases} B^1_{(1,0,0)} = B^2_{(2,0,0)} + \frac{1}{2}B^2_{(1,1,0)} + \frac{1}{2}B^2_{(1,0,1)} \\ B^1_{(0,1,0)} = B^2_{(0,2,0)} + \frac{1}{2}B^2_{(1,1,0)} + \frac{1}{2}B^2_{(0,1,1)} \\ B^1_{(0,0,1)} = B^2_{(0,0,2)} + \frac{1}{2}B^2_{(1,0,1)} + \frac{1}{2}B^2_{(0,1,1)} \end{cases} \implies T_1^p = \begin{bmatrix} 1 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 \\ \frac{1}{2} & 0 & \frac{1}{2} \\ 0 & 1 & 0 \\ 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & 0 & 1 \end{bmatrix}. \qquad \Diamond$$

We can approach the h-transfer matrices in a similar fashion. The De Casteljau algorithm (cf. §3.5.2) can be used to find a recurrence relation for the expression of a Bernstein polynomial on a triangle in terms of the BB-basis on its children.

Theorem 3.5.20 ([32, Thm. 2.38])**.** *Recall that $|\boldsymbol{\beta}| = q$ for $\boldsymbol{\beta} \in \mathcal{I}^q$. Given a triangle $K := \text{hull}(\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3)$, define its two children as*

$$K^1 := \text{hull}(\mathbf{w}, \mathbf{v}_3, \mathbf{v}_1), \quad K^2 := \text{hull}(\mathbf{w}, \mathbf{v}_1, \mathbf{v}_2), \quad \mathbf{w} := (\mathbf{v}_1 + \mathbf{v}_2)/2.$$

Let $\left\{ B_{\boldsymbol{\beta}}^{K^k,p} : \boldsymbol{\beta} \in \mathcal{I}^p \right\}$ be the BB-basis on triangle K^k, for $k = 1, 2$. Then

$$B_{\boldsymbol{\alpha}}^{K,p}|_{K^1} = \sum_{\boldsymbol{\beta} \in \mathcal{I}^p} \mathbf{b}_{\boldsymbol{\alpha},\boldsymbol{\beta}-\beta_2\boldsymbol{e}_2}^{(p-\beta_2)} B_{\boldsymbol{\beta}}^{K^1,p} \quad \textit{and} \quad B_{\boldsymbol{\alpha}}^{K,p}|_{K^2} = \sum_{\boldsymbol{\beta} \in \mathcal{I}^p} \mathbf{b}_{\boldsymbol{\alpha},\boldsymbol{\beta}-\beta_3\boldsymbol{e}_3}^{(p-\beta_3)} B_{\boldsymbol{\beta}}^{K^2,p},$$

where $\left\{ \mathbf{b}_{\boldsymbol{\alpha},\boldsymbol{\beta}}^{(q)} : \boldsymbol{\beta} \in \mathcal{I}^q \right\}$ are the quantities obtained in the qth step of the De Casteljau algorithm (cf. Algorithm 3.5.8).

As a result, these quantities satisfy

$$\mathbf{b}_{\boldsymbol{\alpha},\boldsymbol{\beta}}^{(p)} = \delta_{\boldsymbol{\alpha}\boldsymbol{\beta}}, \quad (\boldsymbol{\beta} \in \mathcal{I}^p), \quad \mathbf{b}_{\boldsymbol{\alpha},\boldsymbol{\beta}}^{(q-1)} := \frac{1}{2}\mathbf{b}_{\boldsymbol{\alpha},\boldsymbol{\beta}+\boldsymbol{e}_2}^{(q)} + \frac{1}{2}\mathbf{b}_{\boldsymbol{\alpha},\boldsymbol{\beta}+\boldsymbol{e}_3}^{(q)} \quad (1 \le q \le p, \boldsymbol{\beta} \in \mathcal{I}^{q-1}). \quad (3.5.21)$$

Remark 3.5.22. The above result fully characterizes the h-transfer matrices. Note that the initial quantity $\mathbf{b}_{\boldsymbol{\alpha},\cdot}^{(p)}$ of (3.5.21) is very sparse, having zeros in every position except $\boldsymbol{\beta}$. The recurrence relation of (3.5.21) then ensures that *every* such vector $\mathbf{b}_{\boldsymbol{\alpha},\boldsymbol{\beta}}^{(q-1)}$ is sparse, which in turn ensures sparsity of the final h-transfer matrices. $\Diamond$

3.5.5. Conditioning of the basis

Through the results in Corollaries 3.5.14 and 3.5.15, Proposition 3.5.18, and Theorem 3.5.20, it is obvious that the cells inside the element matrices are readily computed in real time. Let us now study the conditioning of these matrices in more detail.

Theorem 3.5.23 ([31, Cor. 4.3])**.** *The condition number of the reference mass matrix $M^{\hat{K},p}$, as a function of the degree p, is*

$$\kappa(M^{\hat{K},p}) = \binom{2p+2}{p}.$$

This number grows exponentially in p.

We therefore conclude that the BB-basis is not *well-conditioned.*

Example 3.5.24. See Figure 3.5.25. In terms of conditioning, the BB-basis is comparable to the previous bases, in that the condition number of either mass- or stiffness matrix shows exponential growth. In this case however, *all three* condition numbers grow quickly. Moreover, both mass- and stiffness matrices are full, so any sparsity in that regard is gone.

The sparsity of the h-transfer matrix is remarkable; see Theorem 3.5.20 and the remark thereunder for a short explanation. ◊

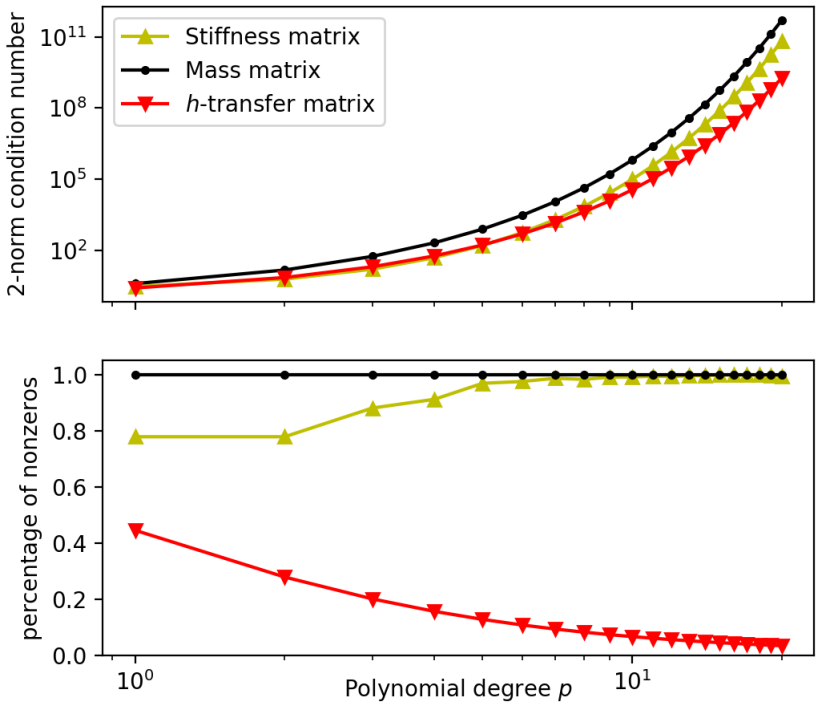


Figure 3.5.25.: Conditioning and sparsity of the BB-basis for $p = 1, \ldots, 20$. We investigate the h-transfer matrix, and the element mass- and stiffness matrices. Top: condition numbers of the three matrices of interest. Bottom: percentage of nonzero elements.

Remark 3.5.26. Remember that the condition number of the element stiffness matrices has a direct effect on the conditioning of the global stiffness matrix.

The BB-basis is ill-conditioned, and thus, the Galerkin solution $u_{\mathcal{K}}$ with respect to the global basis cannot be expected to bring $\|u - u_{\mathcal{K}}\|_{H^1_0(\Omega)}$ down to machine precision. Numerical findings by Kirby [30, Figs. 4.4–4.6] suggest that this problem is not seen in practice. In his words: "The accuracy we obtained in the finite element solution is surprising in light of the very large condition numbers, and we do not have an explanation for this. The condition number only provides a worst-case situation that seems, in this case, quite pessimistic." ◊

3.5.6. Bernstein-Bézier elements of nonuniform degree

Because the BB-basis is not hierarchical in nature, we cannot simply construct a basis for $\mathbb{P}_p(K)$ and throw away some basis functions hoping to get a global basis for the Galerkin space.

Ainsworth employs a completely different tactic in [3]. The theory is still in early stages of development, and its notation rather heavy, so we will not derive it here in full. Instead, we will try to explain the idea and why it works.

Definition 3.5.27. Given a triangle $K := \operatorname{hull}(\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3)$, for $0 \leq k \leq 2$, the convex hull of any $k+1$ of its vertices is a nondegenerate k-simplex in $\mathbb{R}^2$. We call a simplex like this a *subsimplex* of K, as it corresponds with either its face ($k = 2$), an edge ($k = 1$), or a vertex ($k = 0$). Denote the set of k-subsimplices with $\triangle_k(K)$, and the set of all subsimplices $\triangle(K)$. ◊

Domain points

On such a k-simplex $F := \operatorname{hull}(\mathbf{v}_1^F, \ldots, \mathbf{v}_{k+1}^F)$, we can define the *domain points*

$$\mathcal{D}_k^p(F) := \left\{ \mathbf{v}_{\boldsymbol{\alpha}}^F : \boldsymbol{\alpha} \in \mathcal{I}_k^p \right\}, \quad \mathbf{v}_{\boldsymbol{\alpha}}^F := \frac{1}{p} \sum_{n=1}^{k+1} \alpha_n \mathbf{v}_n^F$$

where $\mathcal{I}_k^p$ is the *lattice index set* defined by $\mathcal{I}_k^p := \left\{ \boldsymbol{\alpha} \in \mathbb{N}_0^{k+1} : |\boldsymbol{\alpha}| = p \right\}$. The *interior lattice set* $\mathring{\mathcal{I}}_k^p \subset \mathcal{I}_k^p$ is defined as $\mathring{\mathcal{I}}_k^p := \left\{ \boldsymbol{\alpha} \in \mathbb{N}^{k+1} : |\boldsymbol{\alpha}| = p \right\}$, inducing the *interior domain points* $\mathring{\mathcal{D}}_k^p(F) := \left\{ \mathbf{v}_{\boldsymbol{\alpha}}^F : \boldsymbol{\alpha} \in \mathring{\mathcal{I}}_k^p \right\}$. See Figure 3.5.28 for a visualization.

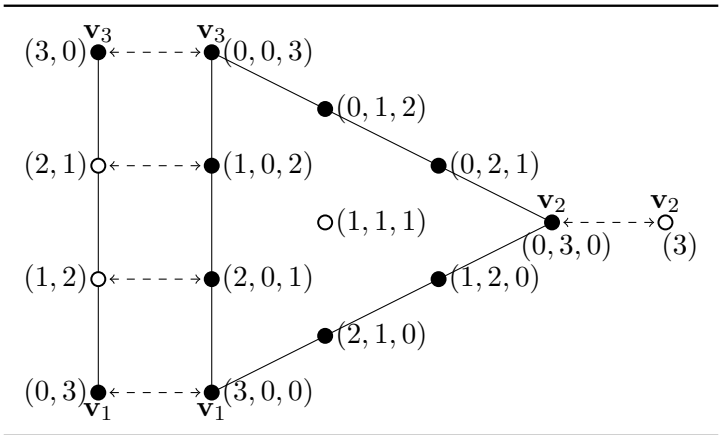


Figure 3.5.28.: A triangle $K := \operatorname{hull}(\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3)$ with the 1-subsimplex $\operatorname{hull}(\mathbf{v}_3, \mathbf{v}_1)$ and 0-subsimplex $\operatorname{hull}(\mathbf{v}_2)$, along with their lattice set and domain points, for $p = 3$. Interior domain points are indicated by open circles.

Remark 3.5.29. Figure 3.5.28 shows another important point. To make sure subsimplices are uniquely determined by their vertices, we order the vertices in a specific way. When F is

an edge, the direction of F is clockwise relative to the triangle K; when F is a 2-simplex, its vertices are ordered like K itself. ◊

When F is a k-subsimplex of K, the set of domain points $\mathcal{D}^p_k(F)$ is a subset of the domain points $\mathcal{D}^p_2(K)$ of the triangle itself. Therefore, given $\boldsymbol{\alpha}^F \in \mathcal{I}^p_k$, there is an index $\boldsymbol{\alpha}^K \in \mathcal{I}^p_2$ such that $\mathbf{v}^F_{\boldsymbol{\alpha}^F} = \mathbf{v}^K_{\boldsymbol{\alpha}^K}$. We can formalize the relationship between multi-indices of the domain points that F and K have in common by the following procedure.

Definition 3.5.30. Given $K := \text{hull}(\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3)$ and k-subsimplex $F := \text{hull}(\mathbf{v}^F_1, \ldots, \mathbf{v}^F_{k+1})$, we have

$$\boldsymbol{\alpha}^K = \varepsilon_{FK}\boldsymbol{\alpha}^F, \tag{3.5.31}$$

where $\varepsilon_{FK} : \mathcal{I}^p_k \to \mathcal{I}^p_2$ is the *extension mapping* defined by

$$\alpha^K_n = \begin{cases} \alpha^F_m & \text{when } \mathbf{v}_n = \mathbf{v}^F_m \text{ for some } m, \\ 0 & \text{else.} \end{cases} \tag{3.5.32}$$

◊

Example 3.5.33. For instance, in Figure 3.5.28, we see that

$$\begin{aligned} &\varepsilon_{FK}(3,0) = (0,0,3), \quad \varepsilon_{FK}(2,1) = (1,0,2), \\ &\varepsilon_{FK}(1,2) = (2,0,1), \quad \varepsilon_{FK}(0,3) = (3,0,0). \end{aligned}$$ ◊

The following result is one of the cornerstones of the argument in this section.

Proposition 3.5.34 ([3, Lem. 3])**.** *The set of domain points of the original triangle is equal to the disjoint union of the interior domain points on all subsimplices:*

$$D^p_2(K) = \bigsqcup_{k=0}^{2} \bigsqcup_{F \in \triangle_k(K)} \varepsilon_{FK}\mathring{\mathcal{D}}^p_k(F).$$

Barycentric coordinates

On any k-simplex $F \in \triangle_k(K)$, we can define local *barycentric coordinates* $\boldsymbol{\lambda}^F$, being an $(k+1)$-tuple of nonnegative scalars that sum to unity. The rule $F \ni \mathbf{x} \to \boldsymbol{\lambda}^F \in \mathbb{R}^{k+1}$ defines an affine mapping on F.

In view of $F \subset K$, a point $\mathbf{x} \in F$ also belongs to K so both $\boldsymbol{\lambda}^K(\mathbf{x})$ and $\boldsymbol{\lambda}^F(\mathbf{x})$ are well-defined. Similar to (3.5.32), one can derive that

$$\lambda^K_n = \begin{cases} \lambda^F_m & \text{when } \mathbf{v}_n = \mathbf{v}^F_m \text{ for some } m, \\ 0 & \text{else.} \end{cases} \tag{3.5.35}$$

With a slight abuse of notation, we *also* denote the mapping defined through (3.5.35) by ε_{FK}, so that

$$\boldsymbol{\lambda}^K(\mathbf{x}) = \varepsilon_{FK}\boldsymbol{\lambda}^F(\mathbf{x}) \quad (\mathbf{x} \in F). \tag{3.5.36}$$

We can extend this definition to all $\mathbf{x} \in K$ by the *barycentric extension*: Viewing $\boldsymbol{\lambda}^F(\mathbf{x})$ as the three-tuple $(\lambda(\mathbf{x}), 1-\lambda(\mathbf{x}), 1)$ for $\mathbf{x} \in F$, there is $\mu(\mathbf{x}) \in [0,1]$ that satisfies

$$\boldsymbol{\lambda}^F(\mathbf{x}) = \big(\lambda(\mathbf{x})[1-\mu(\mathbf{x})], [1-\lambda(\mathbf{x})][1-\mu(\mathbf{x})], \mu(\mathbf{x})\big) \quad (\mathbf{x} \in K).$$

This yields

$$\boldsymbol{\lambda}^K(\mathbf{x}) := \varepsilon_{FK}\boldsymbol{\lambda}^F(\mathbf{x}) \quad (\mathbf{x} \in K). \tag{3.5.37}$$

Bernstein polynomials

With the barycentric coordinates defined on any subsimplex $F \in \triangle(K)$, the definition of a Bernstein polynomial is easily extended to F through

$$B_{\boldsymbol{\alpha}}^{F,p}(\mathbf{x}) := \binom{p}{\boldsymbol{\alpha}} \boldsymbol{\lambda}^F(\mathbf{x})^{\boldsymbol{\alpha}}, \quad (\boldsymbol{\alpha} \in \mathcal{I}_k^p, \mathbf{x} \in F).$$

Recall the extension of multi-indices and barycentric coordinates from F to K (cf. (3.5.31) resp. (3.5.37)), and note that

$$\boldsymbol{\alpha}^F! = \boldsymbol{\alpha}^K!, \quad \boldsymbol{\lambda}^F(\mathbf{x})^{\boldsymbol{\alpha}^F} = \boldsymbol{\lambda}^K(\mathbf{x})^{\boldsymbol{\alpha}^K}.$$

Then there is a natural extension of the Bernstein polynomials on F to K through

$$B_{\boldsymbol{\alpha}^F}^{F,p}(\mathbf{x}) = B_{\boldsymbol{\alpha}^K}^{K,p}(\mathbf{x}) = B_{\varepsilon_{FK}\boldsymbol{\alpha}^F}^{K,p}(\mathbf{x}), \quad (\mathbf{x} \in F, \boldsymbol{\alpha} \in \mathcal{I}_k^p).$$

The above obviously only holds for $\mathbf{x} \in F$. We can again extend the definition to all $\mathbf{x} \in K$ through a *barycentric extension mapping* E_{FK} similar to (3.5.37), yielding

$$E_{FK} B_{\boldsymbol{\alpha}}^{F,p}(\mathbf{x}) = B_{\varepsilon_{FK}\boldsymbol{\alpha}}^{K,p}(\mathbf{x}), \quad (\mathbf{x} \in K, \boldsymbol{\alpha} \in \mathcal{I}_k^p).$$

Nonuniform degree Bernstein-Bézier basis

There is a natural correspondence between Bernstein polynomials on F and the domain points $\mathcal{D}_k^p(F)$ of F. Denote by $\mathring{\mathbb{P}}_k^p(F)$ the space of functions spanned by the Bernstein polynomials associated with $\mathring{\mathcal{D}}_k^p(F)$:

$$\mathring{\mathbb{P}}_k^p(F) := \operatorname{span}\left\{ B_{\boldsymbol{\alpha}}^{F,p} : \boldsymbol{\alpha} \in \mathring{\mathcal{I}}_k^p \right\}.$$

Proposition 3.5.34 showed us that the domain points on a triangle can be written as the disjoint union of interior domain points on its subsimplices. In light of these arguments, it comes to no surprise that

$$\mathbb{P}_p(K) = \bigoplus_{k=0}^{2} \bigoplus_{F \in \triangle_k(K)} E_{FK} \mathring{\mathbb{P}}_k^p(F),$$

where

$$E_{FK}\mathring{\mathbb{P}}_k^p(F) := \operatorname{span}\left\{ E_{FK} B_{\boldsymbol{\alpha}}^{F,p} : \boldsymbol{\alpha} \in \mathring{\mathcal{I}}_k^p \right\} = \operatorname{span}\left\{ B_{\varepsilon_{FK}\boldsymbol{\alpha}}^{T,p} : \boldsymbol{\alpha} \in \mathring{\mathcal{I}}_k^p \right\}. \tag{3.5.38}$$

In words, the BB-basis of a triangle K decomposes into disjoint subsets associated with its subsimplices. In the uniform case, this decomposition is not very interesting. It is, however, *crucial* for finding a basis for the polynomial space of nonuniform degree.

The decomposition allows us to select an arbitrary local degree $p_F \in \mathbb{N}$, independently specified for each subsimplex $F \in \triangle(K)$. Defining the local degree vector $\overrightarrow{p} := \{p_F : F \in \triangle(K)\}$ and the nonuniform polynomial space $\mathbb{P}_{\overrightarrow{p}}(K)$ it induces, we find the decomposition

$$\mathbb{P}_{\overrightarrow{p}}(K) = \bigoplus_{k=0}^{2} \bigoplus_{F \in \triangle_k(K)} E_{FK} \mathring{\mathbb{P}}_k^{p_F}(F).$$

A basis for this space flows naturally from the equality (3.5.38):

$$\mathbb{P}_{\overrightarrow{p}}(K) = \bigoplus_{k=0}^{2} \bigoplus_{F \in \triangle_k(K)} \operatorname{span}\left\{ B^{T,p_F}_{\varepsilon_{FK}\boldsymbol{\alpha}} : \boldsymbol{\alpha} \in \mathring{\mathcal{I}}_k^{p_F} \right\}.$$

Writing

$$\mathcal{I}_K^{\overrightarrow{p}} := \bigsqcup_{k=0}^{2} \bigsqcup_{F \in \triangle_k(K)} \left\{ \varepsilon_{FK}\boldsymbol{\alpha} : \boldsymbol{\alpha} \in \mathring{\mathcal{I}}_k^{P_F} \right\},$$

and noting that $|\boldsymbol{\alpha}| = p_F$ for all $\boldsymbol{\alpha} \in \mathcal{I}_K^{\overrightarrow{p}}$, we can formulate this basis in a very concise form.

Definition 3.5.39. Given $\overrightarrow{p}$, the BB-basis for the space $\mathbb{P}_{\overrightarrow{p}}(K)$ is defined as

$$\Phi^{\mathsf{BB}}_{K,\overrightarrow{p}} := \left\{ B^{K,|\boldsymbol{\alpha}|}_{\boldsymbol{\alpha}} : \boldsymbol{\alpha} \in \mathcal{I}_K^{\overrightarrow{p}} \right\}. \qquad \Diamond$$

Example 3.5.40. Of course, the Bernstein polynomials in $\Phi^{\mathsf{BB}}_{K,\overrightarrow{p}}$ are associated with points in $K := \operatorname{hull}(\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3)$, just like their uniform-degree counterparts. In Figure 3.5.42 we visualize these points for the local degree vector

$$p_{123} = 4, p_{12} = 3, p_{23} = 2, p_{31} = 4, p_1 = 0, p_2 = 1, p_3 = 3. \tag{3.5.41}$$

Note that $p_1 = 0$, which means that no global degree of freedom should be assigned to $\mathbf{v}_1$—it lies on the domain boundary. $\Diamond$

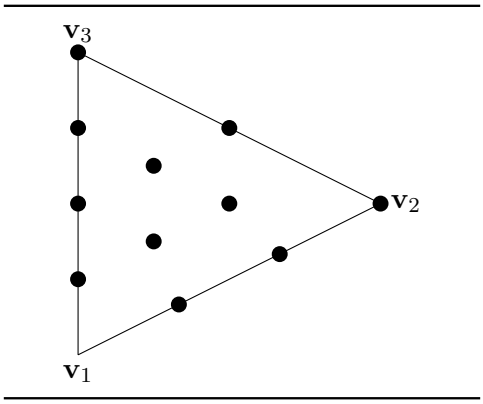


Figure 3.5.42.: Positions of the points on $K := \operatorname{hull}(\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3)$ corresponding with the lattice indices $\boldsymbol{\alpha} \in \mathcal{I}_K^{\overrightarrow{p}}$ of Example 3.5.40.

Global basis induced by the nonuniform BB-basis

The above paragraph shows how to construct the basis functions of the BB-basis of nonuniform order. However, we are mostly interested in applying this inside a FEM, in which case a global basis needs to be constructed as well. The reason that hierarchical elements are the prime choice for *hp*-FEM is the ease with which one constructs such a basis.

How does the nonuniform BB-basis fare in this setting? This is the crux of the matter: The element basis from Definition 3.5.39 *automatically* gives a basis for the space $V(\mathcal{D})$. For instance, if K and K' share a k-subsimplex $F \in \triangle_k(K)$, then each $\boldsymbol{\alpha} \in \mathring{\mathcal{I}}_k^{p_F}$ adds a Bernstein polynomial to the basis. On F, its value is given by the Bernstein polynomial $B_{\boldsymbol{\alpha}}^{F,p_F}$ in $k+1$ variables; on K (resp. K'), by $B_{\varepsilon_{FK}\boldsymbol{\alpha}}^{K,p_F}$ (resp. $B_{\varepsilon_{FK'}\boldsymbol{\alpha}}^{K',p_F}$) in 3 variables. The values of these Bernstein polynomials agree on the common subsimplex F.

The above argument stems from the fact that each triangle containing F takes its local basis function on this subsimplex from a basis function defined on F itself. Therefore, *continuity of the global basis functions is guaranteed by construction.*

Raising the local degree

The basis we constructed in the previous paragraph is very easy and elegant, and by virtue of Proposition 3.5.10, the element mass- and stiffness matrices are readily computed—we refer to [3, §3] for a thorough review.

To make this basis useful in our *hp*-AFEM setting, one detail needs to be addressed. In *hp*-NearBest, we construct a near-best approximation in a broken space of piecewise polynomials—no boundary condition or continuity across edges is imposed.

Therefore, locally, this near-best approximation is not necessarily a member of the nonuniform space $\mathbb{P}_{\overrightarrow{p}}(K)$, but rather the full space $\mathbb{P}_p(K)$. Computing the broken error requires us to compare the Galerkin solution with its near-best approximation; for this, we need an expression of $u_{\mathcal{D}}$ in terms of the local bases on full spaces $\mathbb{P}_{p_D}(K_D)$.

In the hierarchical case, we construct a basis for the full polynomials space $\mathbb{P}_p(K)$ and "throw away" basis functions along edges and in vertices to ensure that we span exactly $\mathbb{P}_{\overrightarrow{p}}(K)$. An expression of $u_{\mathcal{D}}$ in terms of the uniform local basis is then found by simply setting the coefficients for $\mathbb{P}_p(K) \setminus \mathbb{P}_{\overrightarrow{p}}(K)$ to zero.

Using the BB-basis however, we never construct the basis functions that span this complement $\mathbb{P}_p(K) \setminus \mathbb{P}_{\overrightarrow{p}}(K)$, so in order to express $u_{\mathcal{D}}$ in terms of the *uniform* BB-basis of degree $p := \max \overrightarrow{p}$, we need to raise its degree on each subsimplex. Starting from Proposition 3.5.18, one can derive that this is achieved by Algorithm 3.5.43; cf. [3, p.A559].

Require: polynomial $v = \mathbf{v}^\top \Phi_{K,\overrightarrow{p}}^{\mathsf{BB}}$ with coefficients $\mathbf{v}_{\boldsymbol{\alpha}}^{F,p_F}$ for $F \in \triangle_k(K)$ $(k = 0, 1, 2)$;

function RaiseDegree($\left\{\mathbf{v}_{\boldsymbol{\alpha}}^{F,p_F} : \boldsymbol{\alpha} \in \mathring{\mathcal{I}}_k^{p_F}\right\}$ for every $F \in \triangle(K)$)
  Initialize $\mathbf{v}_{\boldsymbol{\alpha}}^{(\min \overrightarrow{p}-1)} := 0$ for $\boldsymbol{\alpha} \in \mathcal{I}_2^{\min \overrightarrow{p}-1}$;
  for $p = \min \overrightarrow{p}, \dots, \max \overrightarrow{p}$ **do**
    `// Raise global degree;`
    $\mathbf{v}_{\boldsymbol{\alpha}}^{(p)} := \sum_{i=1}^{3} \frac{\alpha_i}{p} \mathbf{v}_{\boldsymbol{\alpha}-\boldsymbol{e}_i}^{(p-1)}$ for $\boldsymbol{\alpha} \in \mathcal{I}_2^p$;
    for all $F \in \triangle_k(K)$ with $k = 0, 1, 2$ such that $p_F = d$ **do**
      `// Consolidate coefficients on subsimplices of degree` p;
      $\mathbf{v}_{\varepsilon_{FK}\boldsymbol{\alpha}}^{(p)} = \mathbf{v}_{\varepsilon_{FK}\boldsymbol{\alpha}}^{(p)} + \mathbf{v}_{\boldsymbol{\alpha}}^{F,p_F}$ for $\boldsymbol{\alpha} \in \mathring{\mathcal{I}}_k^{p_F}$;
  return $\left\{\mathbf{v}_{\boldsymbol{\alpha}}^{(\max \overrightarrow{p})} : \boldsymbol{\alpha} \in \mathcal{I}_2^{\max \overrightarrow{p}}\right\}$.

Algorithm 3.5.43: Raising the degree of a polynomial in nonuniform BB-representation to uniform BB-representation; cf. [3, Alg. 3].

Conclusion

In this chapter, we studied multiple bases and looked at their properties. We found that the Lagrange basis of §3.1 was not able to easily construct a basis for a nonuniform Galerkin space, so we quickly dismissed it.

One obvious solution was to consider a *hierarchical basis*, where each basis for degree $p+1$ is created by adding elements to the basis of degree p. We looked at the Szabó-Babuška basis, and saw that a natural basis for the Galerkin space follows, but that its conditioning was unsatisfactory. We saw an improvement in conditioning by a slight modification, yielding the Aiffa basis. Both hierarchical bases however suffer from the fact that its functions are quite ugly, so that we have to resort to precomputing the induced matrices. This is again unsatisfactory; it voids the near-best results of *hp*-AFEM.

However, the best results were found using the Bernstein polynomials introduced in §3.4. We found that the uniform-degree BB-basis naturally allows computing these matrices on the fly, and that there is a nice extension to the nonuniform case allowing us to construct a global basis. Still, it is hard to control the conditioning, even though empirical evidence suggests this ill-conditioning is not seen in practice.

Although the BB-basis is the best-looking solution, we will continue with the hierarchical Aiffa basis, mainly because we found out about the BB-basis well after finishing the implementation. The next chapter will shed some light on a few interesting details from this implementation.

4. Practical considerations

The first two chapters of this thesis were purely theoretical, and the third chapter ventured into more practical territory with the search for a local basis with favourable properties. We found two promising examples; our implementation will be based on the hierarchical Aiffa basis discussed in §3.4.2.

In this chapter, we will cover the more practical challenges found when implementing the *hp*-AFEM algorithm using this hierarchical basis. We will first dive into the computation of the quantities that we need. After that, we will have a short intermezzo on ensuring interelement continuity of global basis functions. We finish with a discussion of the numerical implementation.

4.1. Computing the necessary quantities

For the Bernstein-Bézier basis, we were able to derive the element mass- and stiffness matrix in closed form on *any* triangle $K \in \mathfrak{K}$. Unfortunately, the construction of the Aiffa basis does not allow this. We will have to resort to computing them on a reference triangle $\hat{K}$, and use the affine transformation F_K to express the element matrices locally on K.

In the following, assume $K := \operatorname{hull}(\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3)$ is a triangle with vertices $\mathbf{v}_n = (x_n, y_n)$ for $n = 1, 2, 3$.

Definition 4.1.1. The *reference triangle* $\hat{K}$ we will be using is spanned by the vertices

$$\hat{\mathbf{v}}_1 := (0,0), \quad \hat{\mathbf{v}}_2 := (1,0), \quad \hat{\mathbf{v}}_3 := (0,1). \qquad \Diamond$$

This reference element has the nice property that the first two barycentric coordinates (λ_1, λ_2) coincide with the local coordinates $\hat{\mathbf{x}}$.

Proposition 4.1.2. *The affine mapping $F_K : \hat{K} \to K$ satisfies*

$$F_K(\hat{\mathbf{x}}) = \begin{bmatrix} x_2 - x_1 & x_3 - x_1 \\ y_2 - y_1 & y_3 - y_1 \end{bmatrix} \hat{\mathbf{x}} + \begin{bmatrix} x_1 \\ y_1 \end{bmatrix},$$

and for its Jacobian, we see

$$\left|\det(\mathrm{D}F_K)\right| = \frac{\operatorname{vol}(K)}{\operatorname{vol}(\hat{K})} = 2\operatorname{vol}(K).$$

4.1.1. Reference to element matrices

We can compute the element mass matrix M^D in terms of the *reference mass matrix* $\hat{M}^p := M^{(\hat{K}, \triangle_p)}$ by

$$M^D_{rs} = \int_K \phi^D_r \phi^D_s = \left|\det(\mathrm{D}F_K)\right| \int_{\hat{K}} \hat{\phi}_r \hat{\phi}_s = 2\operatorname{vol}(K) \hat{M}^p_{rs}, \quad (1 \le r, s \le \triangle_p) \tag{4.1.3}$$

where $\left|\det(\mathrm{D}F_K)\right|$ is the determinant of the Jacobian of F_K—cf. Proposition 4.1.2.

Moreover, we can write the element load vectors as

$$b_r^D = 2\,\mathrm{vol}(K)\int_{\hat{K}} \hat{f}\hat{\phi}_r, \quad \hat{f} := f \circ F_K \quad (1 \le r \le \triangle_p).$$

If the forcing function f is a polynomial on K in that $f|_K = \mathbf{f}^\top \Phi_D$ for some vector $\mathbf{f}$, then this load vector reduces to

$$b_r^D = 2\,\mathrm{vol}(K)\mathbf{f}^\top(\mathrm{col}_r\,\hat{M}^p) \quad (1 \le r \le \triangle_p). \tag{4.1.4}$$

Proposition 4.1.5. *The element stiffness matrix can be written as*

$$A^D = \frac{1}{2\,\mathrm{vol}(K)}\left(C_1\hat{A}_1^p + C_2\hat{A}_2^p + C_3A_3^p\right)$$

where

$$C_1 := (x_3-x_1)^2 + (y_3-y_1)^2, \qquad \hat{A}_1^p := \left[\int_{\hat{K}} \frac{\partial\hat{\phi}_r}{\partial\hat{x}}\frac{\partial\hat{\phi}_s}{\partial\hat{x}}\right]_{r,s=1}^{\triangle_p},$$

$$C_2 := (x_2-x_1)(x_1-x_3) + (y_2-y_1)(y_1-y_3), \qquad \hat{A}_2^p := \left[\int_{\hat{K}} \frac{\partial\hat{\phi}_r}{\partial\hat{x}}\frac{\partial\hat{\phi}_s}{\partial\hat{y}} + \frac{\partial\hat{\phi}_r}{\partial\hat{y}}\frac{\partial\hat{\phi}_s}{\partial\hat{x}}\right]_{r,s=1}^{\triangle_p},$$

$$C_3 := (x_2-x_1)^2 + (y_2-y_1)^2, \qquad \hat{A}_3^p := \left[\int_{\hat{K}} \frac{\partial\hat{\phi}_r}{\partial\hat{y}}\frac{\partial\hat{\phi}_s}{\partial\hat{y}}\right]_{r,s=1}^{\triangle_p}.$$

Proof. First, notice that by the chain rule

$$\nabla_{\mathbf{x}} := \begin{bmatrix}\frac{\partial}{\partial x}\\ \frac{\partial}{\partial y}\end{bmatrix} = \begin{bmatrix}\frac{\partial\hat{x}}{\partial x}\frac{\partial}{\partial\hat{x}} + \frac{\partial\hat{y}}{\partial x}\frac{\partial}{\partial\hat{y}}\\ \frac{\partial\hat{x}}{\partial y}\frac{\partial}{\partial\hat{x}} + \frac{\partial\hat{y}}{\partial y}\frac{\partial}{\partial\hat{y}}\end{bmatrix} = \begin{bmatrix}\frac{\partial\hat{x}}{\partial x} & \frac{\partial\hat{y}}{\partial x}\\ \frac{\partial\hat{x}}{\partial y} & \frac{\partial\hat{y}}{\partial y}\end{bmatrix}\begin{bmatrix}\frac{\partial}{\partial\hat{x}}\\ \frac{\partial}{\partial\hat{y}}\end{bmatrix} = (\mathrm{D}F_K^{-1})^\top\nabla_{\hat{\mathbf{x}}}.$$

With this, we see that

$$\begin{aligned}\int_K \nabla_{\mathbf{x}}\phi_r^D\cdot\nabla_{\mathbf{x}}\phi_s^D &= \left|\det(\mathrm{D}F_K)\right|\int_{\hat{K}}\nabla_{\mathbf{x}}\hat{\phi}_r\cdot\nabla_{\mathbf{x}}\hat{\phi}_s\\ &= 2\,\mathrm{vol}(K)\int_{\hat{K}}((\mathrm{D}F_K^{-1})^\top\nabla_{\hat{\mathbf{x}}}\hat{\phi}_r)\cdot((\mathrm{D}F_K^{-1})^\top\nabla_{\hat{\mathbf{x}}}\hat{\phi}_s).\end{aligned}$$

Defining the matrix

$$B := \left|\det(\mathrm{D}F_K)\right|(\mathrm{D}F_K^{-1})^\top = \begin{bmatrix}y_3-y_1 & y_1-y_2\\ x_1-x_3 & x_2-x_1\end{bmatrix}$$

(where the last equality follows from some simple pen-and-paper algebra), the above reduces to

$$A_{rs}^D = \frac{1}{2\,\mathrm{vol}(K)}\int_{\hat{K}}(B\nabla_{\hat{\mathbf{x}}}\hat{\phi}_r)\cdot(B\nabla_{\hat{\mathbf{x}}}\hat{\phi}_s).$$

Now noting that

$$B^\top B = \begin{bmatrix}C_1 & C_2\\ C_2 & C_3\end{bmatrix},$$

a little algebra yields the desired result. □

4.1.2. Other useful equalities

Besides being necessary for the actual construction of the Galerkin solution, these matrices are useful in computing a number of local quantities.

Proposition 4.1.6. *Given some element $D := (K, d)$. For $v \in \mathbb{P}_p(K)$, let $\mathbf{v} \in \mathbb{R}^{\lfloor d \rfloor_\triangle}$ be the vector that realizes $v = \mathbf{v}^\top \Phi_D$. Then*

$$|v|^2_{H^1(K)} = \mathbf{v}^\top A^D \mathbf{v}, \quad \|v\|^2_{L_2(K)} = \mathbf{v}^\top M^D \mathbf{v}.$$

Moreover, the Aiffa basis elements satisfy

$$\hat{\phi}_1 + \hat{\phi}_2 + \hat{\phi}_3 = 1; \tag{4.1.7}$$

this is called a *partition of unity*. Therefore, given an element $D := (K, d)$, the integral of a basis element satisfies

$$\int_K \phi_r^D = \int_K \phi_r^D \left(\sum_{s=1}^3 \phi_s^D \right) = \sum_{s=1}^3 \int_K \phi_r^D \phi_s^D = \sum_{s=1}^3 M_{rs}^D \quad (1 \le r \le \lfloor d \rfloor_\triangle). \tag{4.1.8}$$

4.1.3. Computing the h-transfer matrices

Given $p \in \mathbb{N}$, we will construct the h-transfer matrices $T_p^{h,k}$. By Proposition 3.3.10, these matrices are invariant under the choice of element domain K, so we will compute them on the reference triangle $\hat{K}$.

Let $\hat{\Phi}_p$ be a reference basis for the polynomials of degree p. Denote with $\hat{K}^1, \hat{K}^2$ the children of $\hat{K}$, and denote with $\hat{\Phi}_p^1$ and $\hat{\Phi}_p^2$ the bases on each child as induced by the affine mapping F_K of (3.3.2).

Recall that any polynomial in $\mathbb{P}_p(K)$ is uniquely determined by its point evaluations $\mathcal{N}^p(K)$ on the *domain points* $\mathcal{D}^p(K) = \{\mathbf{d}_r : 1 \le r \le \triangle_p\}$.

Lemma 4.1.9. *For $k = 1, 2$, the matrix $\mathcal{N}^p(\hat{K}^k)(\hat{\Phi}_p^k) = \left[\phi_s^k(\mathbf{d}_r)\right]_{r,s=1}^{\triangle_p}$ is invertible.*

Proof. Say there is a vector $\mathbf{c} := c_1, \ldots, c_{\triangle_p}$ such that

$$\mathcal{N}^p(\hat{K}^k)(\hat{\Phi}_p^k)\mathbf{c} = \left[\sum_s c_s \phi_s^k(\mathbf{d}_r)\right]_{1 \le r \le \triangle_p} = \mathbf{0}.$$

These point evaluations $\mathcal{N}^p(\hat{K}^k)$ are a basis for $\mathbb{P}_p(\hat{K}^k)'$. Then by Lemma 1.2.2, we see that $\sum_s c_s \phi_s^k$ must be the zero function. Because $\hat{\Phi}_p^k$ is a basis for $\mathbb{P}_p(\hat{K}^k)$, $\mathbf{c}$ must be the zero vector. So the matrix must be invertible. □

Definition 4.1.10. In light of the above, define Ψ_p^k to be the collection of functionals *dual* to $\hat{\Phi}_p^k$ in that

$$\Psi_p^k(\hat{\Phi}_p^k) := \left[\psi_r(\phi_s)\right]_{r,s=1}^{\triangle_p} = I_{\triangle_p}.$$

Then

$$\Psi_p^k = \left[\mathcal{N}^p(\hat{K}^k)(\hat{\Phi}_p^k)\right]^{-1} \mathcal{N}^p(\hat{K}^k). \qquad \lozenge$$

Proposition 4.1.11. *Choose $k \in \{1,2\}$. Write ϕ_r for the elements in the reference basis $\hat{\Phi}_p$, and ϕ_r^k for the functions in the basis $\hat{\Phi}_p^k$. Write $\mathbf{d}_r^k$ for the points in $\mathcal{D}^p(K^k)$. Then the h-transfer matrix $T_p^{h,k}$ satisfies*

$$\left[\phi_s^k(\mathbf{d}_r^k)\right]_{r,s=1}^{\triangle_p} T_p^{h,k} = \left[\phi_s(\mathbf{d}_r^k)\right]_{r,s=1}^{\triangle_p}. \tag{4.1.12}$$

Corollary 4.1.13. *This system can be solved uniquely thanks to the invertibility result of lemma 4.1.9.*

Remark 4.1.14. In light of Figure 3.4.12, the h-transfer matrices are very ill-conditioned. Therefore it is most natural to compute the above matrices in exact arithmetic by means of some symbolic package. This is a necessary evil: Symbolic inversion is prohibitively expensive for large p ◊

Example 4.1.15. Choose the element $D := (\hat{K}, 3)$. The degree-1 Aiffa basis on $\hat{K}$ is simply the nodal basis

$$\hat{\Phi}_1 = \left\{(x,y) \mapsto 1-x-y, (x,y) \mapsto x, (x,y) \mapsto y\right\}.$$

The two child triangles of $\hat{K}$ are then

$$\hat{K}^1 = \text{hull}\left(\left(\tfrac{1}{2},\tfrac{1}{2}\right),(0,0),(1,0)\right), \quad \hat{K}^2 = \text{hull}\left(\left(\tfrac{1}{2},\tfrac{1}{2}\right),(0,1),(0,0)\right),$$

and their local bases satisfy

$$\hat{\Phi}_1^1 = \left\{(x,y) \mapsto 2y, (x,y) \mapsto 1-x-y, (x,y) \mapsto x-y\right\},$$
$$\hat{\Phi}_1^2 = \left\{(x,y) \mapsto 2x, (x,y) \mapsto y-x, (x,y) \mapsto 1-x-y\right\}.$$

The domain points $\mathcal{D}^1(K)$ are just the vertices of the triangle K, so that $\left[\phi_s^k(\mathbf{d}_r^k)\right]_{r,s=1}^{\triangle_p} = I_3$. Some pen-and-paper computations then yield

$$T_1^{h,1} = \left[\phi_s(\mathbf{d}_r^k)\right]_{r,s=1}^{\triangle_p} = \begin{bmatrix} 0 & \frac{1}{2} & \frac{1}{2} \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}, \quad T_1^{h,2} = \left[\phi_s(\mathbf{d}_r^k)\right]_{r,s=1}^{\triangle_p} = \begin{bmatrix} 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}. \qquad \text{◊}$$

4.2. The error functional

Recall the definition of the error functional in (2.2.5) given an hp-element $D := (K_D, d_D)$:

$$e_D(v) := |v - P^{p_D} v|^2_{H^1(K_D)}, \quad (v \in H^1(K_D)),$$

where P^p is the orthogonal projector onto $\mathbb{P}_p(K_D)/\mathbb{P}_0(K_D)$. In order to compute this value, we choose the unique representative

$$w \in \mathbb{P}^{p_D}(K_D) \;\text{ with }\; [w]_{\mathbb{P}_0(K_D)} = P^{p_D} v \;\text{ so that }\; \overline{w} = \overline{v}$$

where $\overline{w} := \int_{K_D} w$ denotes the average value.

Inside hp-NearBest, we compute this error functional for a plethora of hp-elements D. Its argument v will be the Galerkin solution $u_{\mathcal{D}}$ on some conforming hp-triangulation $\mathcal{D}$ and is fixed throughout one call to hp-NearBest. It is important to note that almost always, $D \notin \mathcal{D}$: If $D \in \mathcal{D}$, then $e_D(u_{\mathcal{D}}) = 0$ because $u_{\mathcal{D}}$ is its own best approximation. We will touch on this in more detail below.

In a few special cases, it is relatively easy to compute the error functional.

4.2.1. Special case

Originally, our error functional was only defined on elements $D \in \mathfrak{K} \times \mathbb{N}$. In §A, we showed that for *hp*-NearBest to work on a triangulation with multiple roots, it was necessary to extend this definition to the case $d = 0$, in which case we approximate $u_{\mathcal{D}}$ on D with the zero function; $e_D(u_{\mathcal{D}}) := |u_{\mathcal{D}}|^2_{H^1(K)}$.

The value of $e_D(u_{\mathcal{D}})$ can for $d = 0$ be found through a simple recursive routine: If $u_{\mathcal{D}}$ is a polynomial on K, we can invoke Proposition 4.1.6; if not, then we are sure $u_{\mathcal{D}}$ is a polynomial on some set of descendants of K, so we recurse. See also Algorithm 4.2.1.

---

function SquaredSeminorm($u_{\mathcal{D}}$, $K \in \mathfrak{K}$)
  if $u_{\mathcal{D}}$ is a polynomial on K **then**
    $p :=$ degree of $u_{\mathcal{D}}$ on K; $D := (K, \triangle_p)$ the *hp*-element;
    find $\mathbf{u}$ such that $u_{\mathcal{D}}|_K = \mathbf{u}^\top \Phi_D$;
    return $\mathbf{u}^\top A^D \mathbf{u}$. ▷ See Prop. 4.1.6
  $K^1, K^2 :=$ children of K;
  return SquaredSeminorm($u_{\mathcal{D}}, K^1$) + SquaredSeminorm($u_{\mathcal{D}}, K^2$); ▷ Recurse

---

Algorithm 4.2.1: Recursive algorithm for the computation of $|u_{\mathcal{D}}|^2_{H^1(K)}$.

Noting that the $H^1(K)$-seminorm measures *gradients* of functions, we see that

$$e_{K,1}(u_{\mathcal{D}}) = |u_{\mathcal{D}} - \alpha|^2_{H^1(K)} = |u_{\mathcal{D}}|^2_{H^1(K)} = e_{K,0}(u_{\mathcal{D}}) \quad (\alpha \in \mathbb{R}).$$

In other words, we are back in the previous special case. By $\lfloor 1 \rfloor_\triangle = \lfloor 2 \rfloor_\triangle = 1$, this holds even for $d = 2$.

4.2.2. General case

When $u_{\mathcal{D}}$ is a polynomial locally on K_D, we write its degree as $p(u_{\mathcal{D}}, K_D)$.

Lemma 4.2.2. *When on an element $D = (K, d_D)$, the Galerkin solution is a polynomial of low enough degree—$p(u_{\mathcal{D}}, K) \le p_D$—the local error functional vanishes. This special case happens when for some $\tilde{D} := (K_{\tilde{D}}, d_{\tilde{D}}) \in \mathcal{D}$ we have $K \subset K_{\tilde{D}}$ and $d_D \ge d_{\tilde{D}}$, or in words, when D locally refines $\mathcal{D}$.*

Example 4.2.3. Let us look at a few typical examples. Let $\mathcal{D}$ be the triangulation depicted in the left of Figure 4.2.4; each triangle carries a local complexity of 10.

Take the second figure from the left. If we take a large triangle, say $K := K^2 \cup K^3$, then $u_{\mathcal{D}} \notin \mathbb{P}(K)$ so a global approximation of $u_{\mathcal{D}}$ on K of degree $4 = p(15)$ will have positive error.

The third figure shows a situation where $u_{\mathcal{D}}$ is a polynomial on a small triangle K, but best approximation of degree $2 = p(6)$ on the element $D := (K, 6)$ still has positive error.

The last figure shows the situation of Lemma 4.2.2: $u_{\mathcal{D}}$ is a polynomial on K, and the degree $4 = p(15)$ is $\ge p(u_{\mathcal{D}}, K) = p(10) = 3$. So in this case, we will achieve zero error. ◊

The above lemma gives us the case where computing the error functional vanishes. In other cases, we will compute $e_D(u_{\mathcal{D}})$ by noting that

$$e_D(u_{\mathcal{D}}) = \left|u_{\mathcal{D}} - P^{p_D}(u_{\mathcal{D}})\right|^2_{H^1(K_D)} = \min_{\left\{w \in \mathbb{P}_{p_D}(K_D) : \overline{w} = \overline{u_{\mathcal{D}}}\right\}} |u_{\mathcal{D}} - w|^2_{H^1(K_D)},$$

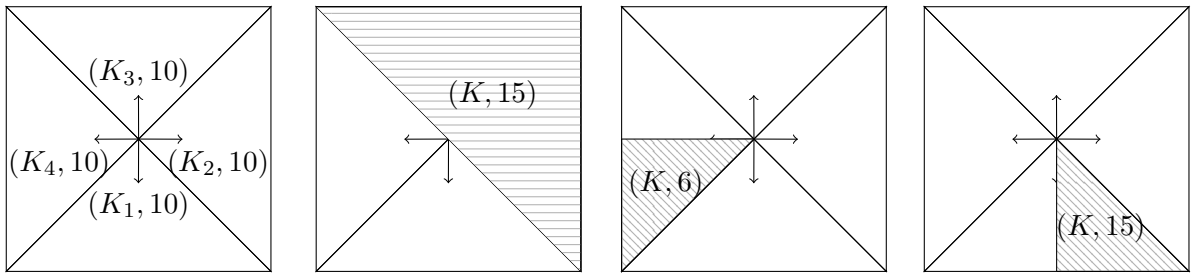


Figure 4.2.4.: Left: an hp-triangulation $\mathcal{D}$; imagine its Galerkin solution $u_\mathcal{D}$. To the right: three typical situations within hp-NearBest.

where we will explicitly construct the unique minimizer

$$m(u_\mathcal{D}, D) := \operatorname*{arg\,min}_{\{w \in \mathbb{P}_{p_D}(K_D) : \overline{w} = \overline{u_\mathcal{D}}\}} |u_\mathcal{D} - w|^2_{H^1(K_D)}.$$

Note that $m(u_\mathcal{D}, D) \in \mathbb{P}_{p_D}(K_D)$, so there is some $\mathbf{m} \in \mathbb{R}^{\lfloor d_D \rfloor_\triangle}$ so that $m(u_\mathcal{D}, D) = \mathbf{m}^\top \Phi_D$. Finding the minimizer is equivalent to finding

$$\mathbf{m} \in \mathbb{R}^{\lfloor d_D \rfloor_\triangle} \text{ s.t. } \Big| u_\mathcal{D}|_{K_D} - \mathbf{m}^\top \Phi_D \Big|^2_{H^1(K_D)} \text{ is minimal, given } \int_{K_D} \mathbf{m}^\top \Phi_D = \int_{K_D} u_\mathcal{D}.$$

This problem is again equivalent to finding the unique stationary point $(\mathbf{m}, \lambda)$ of the Lagrangian

$$\Lambda(\mathbf{m}, \lambda) := \frac{1}{2} \Big| u_\mathcal{D} - \mathbf{m}^\top \Phi_D \Big|^2_{H^1(K_D)} + \lambda \int_{K_D} (\mathbf{m}^\top \Phi_D - u_\mathcal{D}). \tag{4.2.5}$$

Proposition 4.2.6. *The stationary point* $(\mathbf{m}^*, \lambda^*)$ *of* (4.2.5) *is its saddlepoint. It holds that*

$$\nabla_{\mathbf{m},\lambda} \Lambda(\mathbf{m}^*, \lambda^*) = 0 \iff \underbrace{\begin{bmatrix} \langle \nabla \Phi_D, \nabla \Phi_D \rangle_{L^2(K_D)} & \int_{K_D} \Phi_D \\ \left(\int_{K_D} \Phi_D \right)^\top & 0 \end{bmatrix}}_{=:L} \begin{bmatrix} \mathbf{m}^* \\ \lambda^* \end{bmatrix} = \underbrace{\begin{bmatrix} \langle \nabla \Phi_D, \nabla u_\mathcal{D} \rangle_{L^2(K_D)} \\ \int_{K_D} u_\mathcal{D} \end{bmatrix}}_{=:\mathbf{b}}. \tag{4.2.7}$$

Proof. A little algebra yields that

$$\frac{\partial \Lambda}{\partial \lambda} = 0 \iff \int_{K_D} u_\mathcal{D} = \int_{K_D} \mathbf{m}^\top \Phi_D$$

and

$$\begin{aligned} \frac{\partial \Lambda}{\partial \mathbf{m}} &= -\langle \nabla u_\mathcal{D}, \nabla \Phi_D \rangle_{L^2(K_D)} + \langle \nabla \Phi_D, \nabla \Phi_D \rangle_{L^2(K_D)} \mathbf{m} + \lambda \int_{K_D} \Phi_D = 0 \\ &\iff \langle \nabla \Phi_D, \nabla \Phi_D \rangle_{L^2(K_D)} \mathbf{m} + \lambda \int_{K_D} \Phi_D = \langle \nabla u_\mathcal{D}, \nabla \Phi_D \rangle_{L^2(K_D)}. \end{aligned}$$

Combining yields the result. □

Remark 4.2.8. Given that the matrix and right-hand side are known, we can solve the above problem using a direct solver: The problem is local, hence its size is small ($\simeq d_D \times d_D$). In our implementation, a column-pivot Householder QR is used; MINRES is also possible. ◊

Remark 4.2.9. Even though a single computation is quick, we perform it around $N \log N$ to N^2 times (where N is the complexity of the triangulation found by *hp*-NearBest). In Chapter 5, we will see that these computations make up *the majority* of the total computation time of *hp*-AFEM; therefore, it makes sense to optimize the routine.

Given an *hp*-element D, one can show that for any triangle K' congruent to K_D (in that $K' = \alpha K_D + \mathbf{v}_0$ for some $\alpha > 0, \mathbf{v}_0 \in \mathbb{R}^2$), the following holds for the element $D' := (K', d_D)$:

$$\langle \nabla \Phi_{D'}, \nabla \Phi_{D'} \rangle_{L^2(K')} = \langle \nabla \Phi_D, \nabla \Phi_D \rangle_{L^2(K_D)}, \quad \int_{K'} \Phi_{D'} = \alpha^2 \int_{K_D} \Phi_D.$$

Therefore, with $L = L(D)$ the matrix from (4.2.7), we find that for the minimizer on D',

$$L(D') \begin{bmatrix} \mathbf{m}^* \\ \lambda^* \end{bmatrix} = \begin{bmatrix} \langle \nabla \Phi_{D'}, \nabla u_{\mathcal{D}} \rangle_{L^2(K')} \\ \int_{K'} u_{\mathcal{D}} \end{bmatrix} \iff L(D) \begin{bmatrix} \mathbf{m}^* \\ \alpha^2 \lambda^* \end{bmatrix} = \begin{bmatrix} \langle \nabla \Phi_{D'}, \nabla u_{\mathcal{D}} \rangle_{L^2(K')} \\ (\int_{K'} u_{\mathcal{D}})/\alpha^2 \end{bmatrix}.$$

In words, the problem of finding the minimizer on D' can be restated in terms of the Lagrangian matrix on D. Newest vertex bisection has the property that each element domain is in one of a small number of congruence classes ($\leq 4 \# \mathcal{K}(\mathcal{D}_0)$, with $\mathcal{D}_0$ the initial triangulation). We can explicitly precompute a factorization of the matrix for a representative in each such class, allowing us to solve for the minimizer much more quickly.

We chose not to implement this *yet*, because it adds another layer of complexity to an already highly complex implementation, but it does provide ideas for future versions. ◊

4.2.3. The Lagrangian system

The top-left block of the matrix in (4.2.7) is simply the element stiffness matrix A^D found through Proposition 4.1.5. The elements of the vector $\int_K \Phi_D$ are found through (4.1.8) and (4.1.3).

The right-hand side of the system in (4.2.7) is harder, as it relies on whether or not $u_{\mathcal{D}}$ is a polynomial locally on the element domain of $D := (K, d_D)$. When it *is*, the triangulation $\mathcal{D}$ that defines $u_{\mathcal{D}}$ must contain some element $\tilde{D}(D) := (K_{\tilde{D}}, d_{\tilde{D}})$ for which $K \subset K_{\tilde{D}}$—this is the middle case of Figure 4.2.4. Our computations require the local coefficient vector $\mathbf{u}$ of $u_{\mathcal{D}}|_{K_{\tilde{D}}}$ (in that $u_{\mathcal{D}}|_{K_{\tilde{D}}} = \mathbf{u}^\top \Phi_{\tilde{D}(D)}$).

Without loss of generality, assume $K = K_{\tilde{D}}$; otherwise we can bisect $\tilde{D}(D)$ until the domains *do* coincide, at each bisection transfering the local coefficient vector using the h-transfer matrices. This yields a tactic similar to Algorithm 4.2.1.

By Lemma 4.2.2, the case where $d_{\tilde{D}} \leq d_D$ is trivial. Therefore, we can assume that $p_{\tilde{D}} = p(u_{\mathcal{D}}, K) > p_D$.

In our hierarchical case, element mass- and stiffness matrices on D are the top-left block of those on $\tilde{D}(D)$. This implies the following result.

Proposition 4.2.10. *Take some element $D := (K, d_D)$. If $u_{\mathcal{D}}$ is a polynomial, then it is defined on some element $\tilde{D}(D)$ with $K_{\tilde{D}(D)} = K$ and $p_{\tilde{D}(D)} > p_D$—this is the middle case of Figure 4.2.4.*

Take $\mathbf{u} \in \mathbb{R}^{\lfloor d_{\tilde{D}(D)} \rfloor_\triangle}$ *the local coefficient vector of* $u_\mathcal{D}$ *on* K*. Then*

$$\langle \nabla \Phi_D, \nabla u_\mathcal{D} \rangle_{L^2(K)} = \tilde{A}^{\tilde{D}(D)} \mathbf{u},$$

where $\tilde{A}^{\tilde{D}(D)}$ *is the top-left block of* $A^{\tilde{D}(D)}$ *of size* $(\lfloor d_D \rfloor_\triangle \times \lfloor d_{\tilde{D}(D)} \rfloor_\triangle)$*.*

Moreover,

$$\int_K u_\mathcal{D} = \mathbf{u}^\top E^{\tilde{D}(D)}$$

where

$$E_r^{\tilde{D}(D)} := \sum_{s=1}^{3} M_{rs}^{\tilde{D}(D)}, \quad (1 \le r \le \lfloor d_{\tilde{D}(D)} \rfloor_\triangle).$$

Remark 4.2.11. Note that the above result only holds for our specific hierarchical basis. ♢

When the Galerkin solution $u_\mathcal{D}$ is *not* a polynomial on K—corresponding to the first case of Figure 4.2.4—things get a little trickier. The recursive relation

$$\int_K u_\mathcal{D} = \left(\int_{K^1} + \int_{K^2} \right) u_\mathcal{D}, \quad \left\{ K^1, K^2 \right\} \text{ children of } K \tag{4.2.12}$$

shows that we may traverse down the binary tree until an element on which $u_\mathcal{D}$ is a polynomial is found, thus finding $\int_K u_\mathcal{D}$. This algorithm is akin to Algorithm 4.2.1.

We can employ a similar strategy for the inner product vector. The following result connecting bases on children elements will prove useful. Given $D := (K, d_D)$, define its two child elements by $D^1 := (K^1, d_D)$ and $D^2 := (K^2, d_D)$, where K^1, K^2 are the children of K. By linearity of the h-transfer matrices, we deduce from Proposition 3.3.9 that

$$\nabla \Phi_D|_{K^k} = (T^{h,k})^\top \nabla \Phi_{D^k}, \quad (k = 1, 2).$$

With this result, the inner product vector decomposes to a sum of those vectors on child elements:

$$\begin{aligned}
\langle \nabla \Phi_D, \nabla u_\mathcal{D} \rangle_{L^2(K)} &= \langle \nabla \Phi_D, \nabla u_\mathcal{D} \rangle_{L^2(K^1)} + \langle \nabla \Phi_D, \nabla u_\mathcal{D} \rangle_{L^2(K^2)} \\
&= \langle (T^{h,1})^\top \nabla \Phi_{D^1}, \nabla u_\mathcal{D} \rangle_{L^2(K^1)} + \langle (T^{h,2})^\top \nabla \Phi_{D^2}, \nabla u_\mathcal{D} \rangle_{L^2(K^2)} \\
&= (T^{h,1})^\top \langle \nabla \Phi_{D^1}, \nabla u_\mathcal{D} \rangle_{L^2(K^1)} + (T^{h,2})^\top \langle \nabla \Phi_{D^2}, \nabla u_\mathcal{D} \rangle_{L^2(K^2)}.
\end{aligned}$$

A traversal analogous to Algorithm 4.2.1 then yields the result.

4.2.4. From minimizer to error functional

Assume we have our minimizer $\mathbf{m}$ (the Lagrange multiplier λ plays no role). Producing $e_D(u_\mathcal{D})$ from $\mathbf{m}$ is again done using a recursive tactic like Algorithm 4.2.1.

When $u_\mathcal{D}|_K = \mathbf{u}^\top \Phi_{\tilde{D}(D)}$ is a polynomial, we transfer the coefficient vector $\mathbf{m}$ to the element $\tilde{D}(D)$ through repeated application of the p-transfer matrix—in our hierarchical case, this reduces to appending $\mathbf{m}$ with zeros—yielding a vector $\tilde{\mathbf{m}}$. Then, through Proposition 4.1.6, we find

$$\begin{aligned}
e_D(u_\mathcal{D}) &= \left| u_\mathcal{D} - \mathbf{m}^\top \Phi_D \right|_{H^1(K)}^2 = \left| \mathbf{u}^\top \Phi_{\tilde{D}(D)} - \mathbf{m}^\top \Phi_D \right|_{H^1(K)}^2 \\
&= \left| \mathbf{u}^\top \Phi_{\tilde{D}(D)} - \tilde{\mathbf{m}}^\top \Phi_{\tilde{D}(D)} \right|_{H^1(K)}^2 = (\mathbf{u} - \tilde{\mathbf{m}})^\top A^{\tilde{D}(D)} (\mathbf{u} - \tilde{\mathbf{m}}).
\end{aligned}$$

Summary

We summarize this section in the flowchart of Figure 4.2.13.

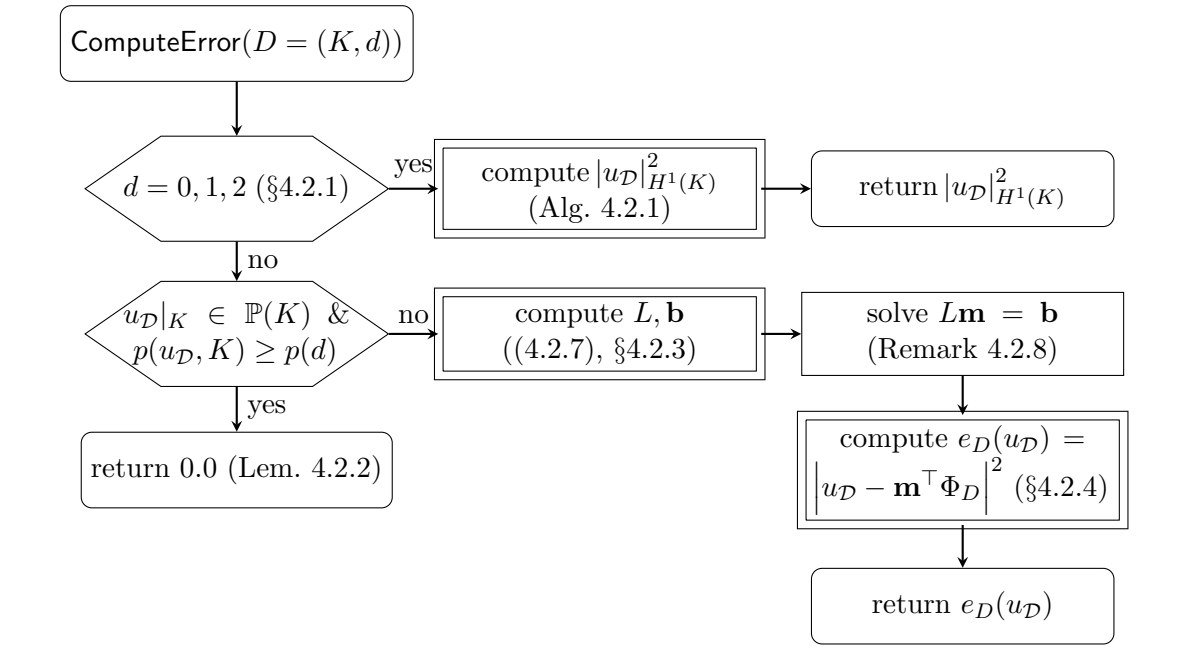


Figure 4.2.13.: Computing the error functional. Each doubly-bordered rectangle calls a recursive subroutine similar to Algorithm 4.2.1.

4.3. Interelement continuity

In Definition 3.4.5, we saw that the edge functions of the Szabó-Babuška- and Aiffa basis depend on the orientation of the edge it is associated with. From the definition:

$$\phi_{K,e_1,q}(\boldsymbol{\lambda}) := \begin{cases} \lambda_2\lambda_3\mathfrak{E}_q(\lambda_3-\lambda_2) & \text{when } \mathbf{v}_2 = \mathbf{v}_1^{e_1} \\ \lambda_2\lambda_3\mathfrak{E}_q(\lambda_2-\lambda_3) & \text{when } \mathbf{v}_3 = \mathbf{v}_1^{e_1} \end{cases}, \quad \mathfrak{E}_q(x) = -\frac{8\sqrt{4q-2}}{q(q-1)}P'_{q-1}(x).$$

We effectively negate the argument of $\mathfrak{E}_q$ when the global orientation of e_1 does not agree with its local orientation on K. Noting that each edge can be in one of two orientations, with three such edges, we end up with a total of $2^3 = 8$ different arrangements or *element types*. The effect is that, for instance, the element stiffness matrices of two neighbouring triangles have to be computed from *different* reference matrices.

Before we continue discussing this problem, let us first explore an elegant solution mentioned by Demkowicz in [18, p. 170]. Noting that Lagrange derivatives satisfy

$$P'_p(-x) = (-1)^p P'_p(x),$$

we see that $\mathfrak{E}_q$ is an even function when q is even, and odd when q is odd. Therefore, the following holds:

$$\phi_{K,e_1,q}(\boldsymbol{\lambda}) = \begin{cases} \lambda_2\lambda_3\mathfrak{E}_q(\lambda_3-\lambda_2) & \text{when } \mathbf{v}_2 = \mathbf{v}_1^{e_1} \text{ or } q \text{ is even} \\ -\lambda_2\lambda_3\mathfrak{E}_q(\lambda_3-\lambda_2) & \text{when } \mathbf{v}_3 = \mathbf{v}_1^{e_1} \text{ and } q \text{ is odd.} \end{cases}$$

This insight yields that the reference matrices differ *in sign only*.

After coming across this problem, and not yet having read the above book, we came up with our own solution. Our solution is less practical and more error-prone but it is so ingrained in the implementation that we cannot change it any more.

Given an element D, every possible element type can be associated with a unique binary 3-vector $t(D) \in \{0,1\}^3$. We can then assign element types $t(D)$ to each element D in a triangulation in such a way that the triangulation is *correctly typed*—that the orientation of an edge shared between two elements is different on both elements locally (in terms of $t(D)$, a 0 must be matched with a 1).

Proposition 4.3.1. *For each conforming triangulation $\mathcal{D} \in \mathbb{D}_c$, one can choose the element types $\{t(D) : D \in \mathcal{D}\}$ so that the triangulation is correctly typed.*

Proof. The proof is simple and constructive. Order the set of elements into a list $(D_i)_{i=1}^{\#\mathcal{K}(\mathcal{D})}$. We then iterate for each i, for each $j \in \{1,2,3\}$. If there is no neighbour across edge e_j, or the neighbour D_k has $k > i$, just set $t(D)_j = 0$. Else, set $t(D)_j = 1$.

By conditioning over $k \lesseqgtr i$, we change the orientation on exactly one of the two local edges. □

Recomputing element types after each subdivision can void the Galerkin solution coefficients. Instead, assign types to children elements D^1, D^2 in terms of their parents element type $t(D) = \{x, y, z\}$, by setting

$$t(D^1) := \{\neg x, x, y\}, \quad t(D^2) := \{y, z, x\}, \tag{4.3.2}$$

cf. Figure 4.3.3. We have the following result.

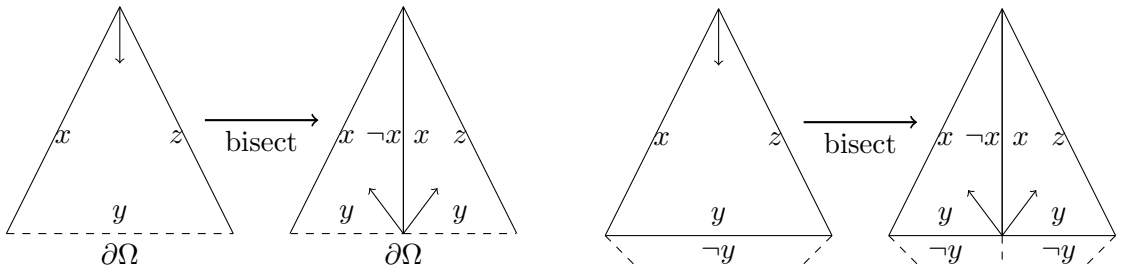


Figure 4.3.3.: Visual aid for (4.3.2). Left: case where D has its refinement edge along $\partial\Omega$. Right: D has a neighbour across this edge.

Proposition 4.3.4. *Given a correctly typed triangulation $\mathcal{D} \in \mathbb{D}_c$, any conforming refinement of $\mathcal{D}$ in which $D \in \mathcal{D}$ is bisected through* (4.3.2)*, is again correctly typed.*

Proof. The Refine routine of Algorithm 1.5.17 (constructing the smallest conforming refinement given a set of marked elements) is recursive in nature, and has two possible base cases: Either the refinement edge of D is on the domain boundary, or there is a neighbour across from this edge. In the first case, application of (4.3.2) results in a correctly typed refinement: exactly one of $x, \neg x$ is equal to 1.

In the second case, applying (4.3.2) to both refined children at once creates two extra edges, both carrying y and $\neg y$; the same argument applies. Therefore, by recursion, the statement holds. □

4.4. Implementation

This project features a fully working implementation of *hp*-AFEM. Broadly speaking, there are three independent components to this software.

The precomputation module creates the h-transfer matrices, and the element mass- and stiffness matrices; it is written in C++11 (using the symbolic algebra package GiNaC [7]) and Mathematica [48];

The library implements *hp*-AFEM in C++11 (using the linear algebra package Eigen [23]), and includes a few proof-of-concept applications;

The plotter is capable of parsing output generated by the library, allowing the creation of plots of the triangulation and solution. It is written in Python 2.7 (using, mainly, matplotlib [1]).

The library code is inspired by the finite element package FEniCS [5], but it does not share the code. The basic requirements (hierarchical basis, triangular elements, coarsening step) seemed too specific at the time.

4.4.1. The precomputation module

Earlier versions of this project were based on precomputations in MATLAB, for its simplicity and relative speed. However, we saw that with high-degree polynomials, MATLAB is too

inaccurate for our needs: The h-transfer matrices are very ill-conditioned (cf. Figure 3.4.12), so their computation through Proposition 4.1.11 falls apart with just 16 digits of precision.

For this reason, the precomputation module was rewritten in C++ with a symbolic algebra package to compute the matrices in exact arithmetic. We however saw that solving (4.1.12)—on top of being very ill-conditioned—is very computationally expensive. Even degree 9—involving symbolic systems of size 55×55—was already intractable using the various symbolic packages we tried. A more sophisticated approach was necessary.

The final version works by computing the mass- and stiffness matrices in C++, and also computes the known matrices of (4.1.12) symbolically. These symbolic matrices are then loaded into Mathematica, known for its excellent symbolic algorithms, which can solve the system for degrees up to 20 (size 231×231) in a few hours. This is of course fine—a basis only needs to be computed once.

The fact that our basis is hierarchical makes it possible to compute only the highest-degree matrices and take the top-left blocks for the lower-degree counterparts.

Expressing piecewise polynomial forcing functions in the hierarchical basis

To humans, the preferred polynomial basis is often the monomial basis. This allows expressing, for instance, the forcing function f as

$$f : \Omega \to \mathbb{R} : (x, y) \mapsto x(1-x)y(1-y).$$

Our *hp*-AFEM library however requires expressing all input in terms of the hierarchical basis. Using a construction similar in taste to that of the h-transfer matrices, we are able to precompute a matrix allowing fast conversion between the hierarchical and monomial basis. The forcing function, expressed in the hierarchical basis, is stored with the initial triangulation and serves as input for *hp*-AFEM.

File format

A triangulation $\mathcal{D}$ is uniquely determined by the tuple $(\mathcal{V}, \mathcal{T})$ where $\mathcal{V} \in \mathbb{R}^{V \times 2}$ defines the location of each vertex inside $\mathcal{D}$, and $\mathcal{T} \in \mathbb{N}_0^{\#\mathcal{K}(\mathcal{D}) \times 5}$ defines the triangles as follows. Each row represents an element $D \in \mathcal{D}$; the first three numbers represent the indices inside $\mathcal{V}$ of its three vertices, the fourth number its initial local complexity d_D, and the last number its element type $t(D)$.

A piecewise polynomial defined on this triangulation is stored through a vector $\mathtt{f}_D \in \mathbb{R}^{\lfloor d_D \rfloor_\triangle}$ of the basis coefficients for each $D \in \mathcal{D}$.

4.4.2. The library *hp*-AFEM

The library is written in C++11, making heavy use of object-oriented programming principles. It is partitioned into the following submodules.

I/O handles reading and writing the triangulations, matrices, and other various input/output;

ElementTree stores and manages *hp*-triangulations in-memory;

DoFHandler manages local and global degrees of freedom;

NearBest implements *hp*-NearBest from Algorithm 2.2.12;

Reduce implements Reduce from §2.3;

FEM builds the finite element system, calls the solver, and manages the solution;

See Figure 4.4.1 for a dependency graph of the modules inside the library, together with the *h*-AFEM and *hp*-AFEM applications.

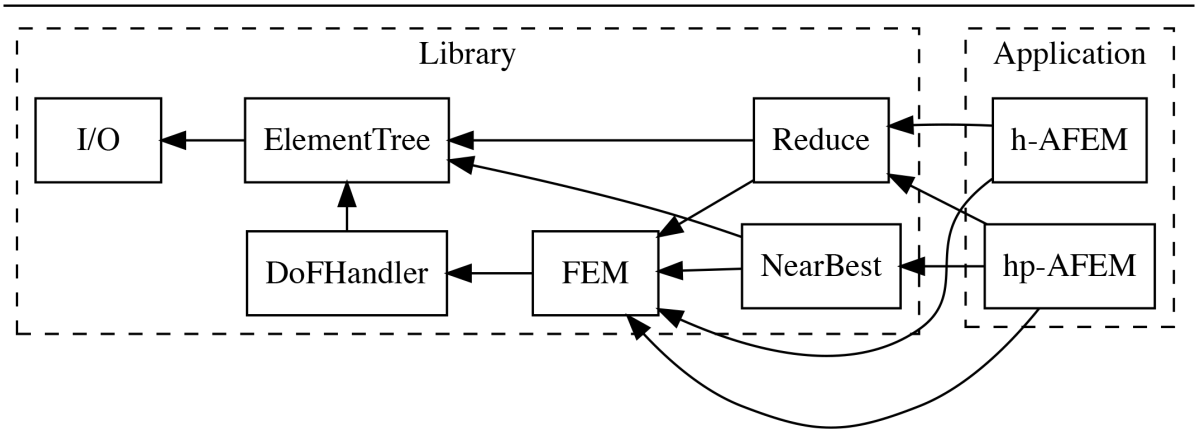


Figure 4.4.1.: Dependencies of the different modules within the *hp*-AFEM library, together with two applications.

Numerical accuracy

As we will see in the next Chapter, the *hp*-AFEM algorithm needs quite a few iterations to get into the *convergent region*. As a result, the finite element spaces we will be looking at can grow quite large, with up to 400K degrees of freedom. The main cause of deterioration at this point is, besides the time needed to solve systems this big, numerical accuracy. We therefore chose to work with *extended precision* floating point numbers, using 80 bits of memory, as opposed to the standard *double precision* with 64 bits. This grows the mantissa size from 53 bits to 64, thus decreasing the machine epsilon from $2^{-53} \approx 1.11 \cdot 10^{-16}$ to $2^{-64} \approx 9.63 \cdot 10^{-19}$.

Finite element solver

From a theoretical standpoint, solving the Galerkin system is easy. From a more implementational point of view, this is far from trivial. Assembling and storing the stiffness matrix has to be done carefully as to not go out of memory; sparse matrix classes are provided by the Eigen linear algebra package [23].

The linear algebra library stores matrices in the form of a list of (row, column, value)-tuples, allowing repetition of row and value. This list is constructed by appending every nonzero entry of every local stiffness matrix. We use a direct solver which employs the $LDL^\top$-decomposition of the global stiffness matrix; see [24, Solving Sparse Linear Systems]. It is also possible to use an iterative solver, but one has to take care defining a good initial guess and stopping criteria. We opted not to take this route.

Storing elements

An instance of *hp*-AFEM will, during its lifetime, create millions of elements in-memory. To avoid going out of memory, we employ the *Flyweight* design pattern—cf. [20]—storing only the essential information of an element. The rest (element stiffness matrix, for instance) is recomputed at each method call.

Development

Valgrind [36] was used to profile the memory and CPU-cycle usage of each method inside the running application. This was very important in the development phase, helping to find critical loops and memory leaks. GDB [41] was used to debug the software and helped development velocity significantly.

4.4.3. Plotter

The plotter is coded mainly in Python. Its main use is to plot triangulations, but it is also capable of producing 3D plots of the Galerkin solution and convergence graphs. It imports the files that the applications output, and was invaluable spotting mistakes in the algorithm.

5. Numerical results

In the previous chapters, we laid out all the necessities for a correct implementation of *hp*-AFEM. In this chapter, we will study some of the numerical results found using this implementation.

All coming paragraphs will in some way solve the Poisson problem of (1.1.9). We will study different (polygonal) domains $\Omega \subset \mathbb{R}^2$; even for the simple forcing function $f = 1$, some domains carry very interesting problems. The canonical domain in this case is L-shaped; the re-entrant corner introduces a singularity in the solution—see below for a more detailed analysis.

We will look at various forcing functions $f \in L^2(\Omega)$, but we will confine ourselves to the piecewise polynomials, and in many cases even global polynomials $f \in \mathbb{P}(\Omega)$.

To overcome some of the numerical instabilities, we will use *extended precision* floating point numbers (also known as *long doubles*), which pairs nicely with the direct solver of choice. See §4.4.2 for a short discussion on these two considerations.

It has to be noted that the implementation currently suffers from the problem discussed in §3.4.3 regarding a maximum degree. In our case, this maximum is 20. This is high enough for *hp*-AFEM to exhibit the desired exponential decay in all considered cases—as we will see in the coming paragraphs—but it is problematic from a theoretical point of view.

We saw in Corollary 2.4.10 that this decay—the error norms decay like $\exp(-\eta(\#\,\mathrm{DoF})^\tau)$—is in terms of two parameters, η and τ. In their seminal paper, Guo and Babuška [27, Thm. 2.1] show that in 2 dimensions, $\tau = 1/3$; plotting the logarithm of the error against $\#\,\mathrm{DoF}^{1/3}$ should then show us a straight line. The slope η of this line is problem-dependent, and therefore usually not known in closed form.

Another shortcoming of this implementation is the choice of error estimator: The simple *refinement error estimator* of §1.4.1 is not reliable, so we have little theory to back up our error estimations. Still, in practice, the results are very good—see [46, §5.3] for a short review in the uniform h-case.

The default algorithm parameters are as follows.

- The **refinement error estimator** reference solution is found by two uniform h-refinements;
- In **Reduce**, the Dörfler parameter is set to $\theta := 0.8$;
- In *hp*-**AFEM**, the coarsening factor is $\omega := 4$, the reduction parameter is $\mu := \frac{1}{2}$, and the final tolerance is $\varepsilon := 0$, as to never stop iterating.

Finally, the choice of reduction factor $\rho := \mu/(1 + C_{\mathsf{B}}(\mathcal{D}_k^*)\omega)$ inside *hp*-AFEM is necessary for proving the convergence behaviour and is the result of compounded worst-case assumptions. In practice, we found that simply picking $\rho := \mu$ results in exponential decay as well.

5.1. Known solution

Our first numerical example is to test our *hp*-AFEM application with a few known solutions of the form

$$u^n : \Omega \to \mathbb{R} : (x, y) \mapsto (x\, y\, (1-x)\, (1-y))^n, \quad \Omega := (0,1)^2. \tag{5.1.1}$$

We choose $n \in \{1, \dots, 5\}$; $n > 5$ is not supported by the implementation (cf. §3.4.3).

Triangulate Ω with two triangles,

$$\mathcal{K} := \left\{ \overline{\Omega} \cap \{x + y \leq 1\}, \overline{\Omega} \cap \{x + y \geq 1\} \right\}$$

and equip both triangles with $d = 6$ degrees of freedom (in view of the homogenous Dirichlet boundary conditions, this ensures one global degree of freedom) yielding an *hp*-triangulation $\mathcal{D}_0$.

For each n, our Poisson solution u^n is a polynomial of degree $4n$, so we can expect the approximation error to be zero locally when $p_D \geq 4n$ for some element D in a triangulation $\mathcal{D} \geq \mathcal{D}_0$. The *hp*-NearBest routine will then decrease this degree, hopefully coarsening it to $p_D = 4n$. We expect that after a modest number of iterations of *hp*-AFEM, the estimated error drops to machine precision. The minimal number of global degrees of freedom we need to achieve true zero error is $(4n-1)^2$, corresponding with two triangles of degree $4n$.

We run *hp*-AFEM, at each iteration storing the Galerkin solution u_k^n on the *hp*-triangulation $\mathcal{C}_k^n := \mathcal{C}(\mathcal{D}_k^*)^n$, which is defined as the smallest conforming refinement of the output triangulation of *hp*-NearBest.

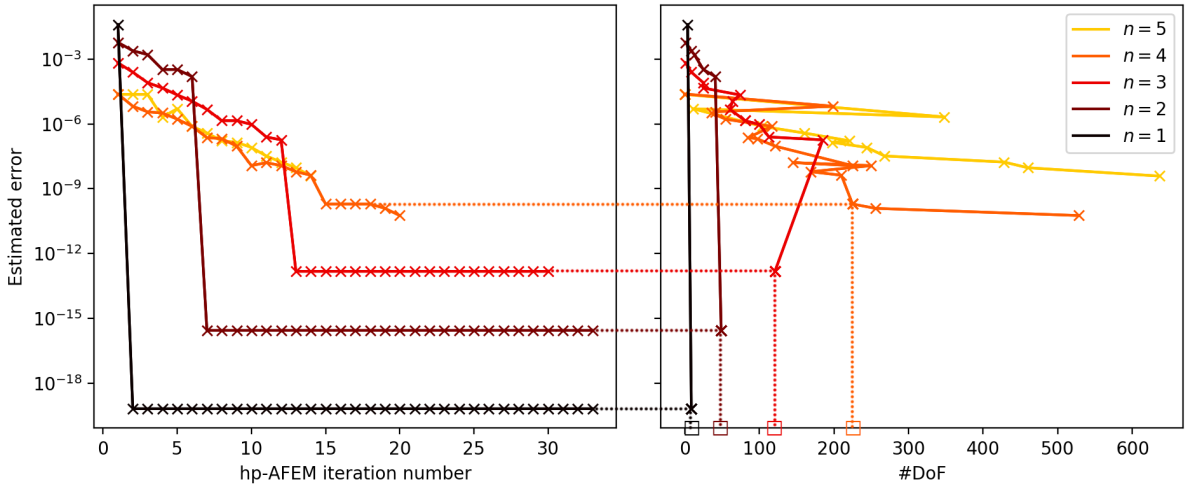


Figure 5.1.2.: Progression of the (estimated) error norms of the *hp*-triangulations $\mathcal{C}(\mathcal{D}_k^*)$ produced by *hp*-AFEM, for the Poisson problems specified by (5.1.1). Left: error norms as a function of k. Right: error norms as a function of $\#\,\mathrm{DoF}$. The boxes on the $\#\,\mathrm{DoF}$-axis denote the special values $(4n-1)^2$.

In Figure 5.1.2, the (estimation of the) error $\left\|u^n - u_k^n\right\|_{H_0^1(\Omega)}$ is plotted as a function of (left) k, and (right) $\dim V(\mathcal{C}_k^n) = \#\,\mathrm{DoF}(\mathcal{C}_k^n)$. In the left figure, we see that this progression plateaus after a number of iterations, and following the dotted lines to the right, we see that

these triangulations carry exactly $\#\,\mathrm{DoF}(\mathcal{C}_k^n) = (4n-1)^2$ degrees of freedom. In light of the previous, this means that *hp*-AFEM finds these *optimal* triangulations for which the true error is zero! Positivity of the estimated error stems from the fact that we are solving a perturbed problem which carries a non-polynomial true solution. This perturbation is due to instability of the h-transfer matrix: We transfer the load vector to refined triangulations, thereby applying the (ill-conditioned) h-transfer matrix multiple times.

In the right figure, we see that the complexity of the triangulations is not always monotonic in k; the whimsical nature of *hp*-NearBest makes its output unpredictable.

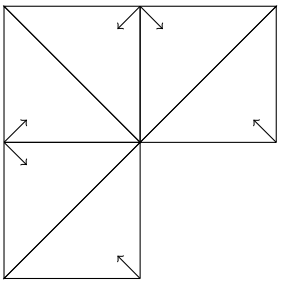
It is interesting to see that *hp*-AFEM finds the true solution in 2 iterations for $n = 1$, in 7 iterations for $n = 2$, in 13 for $n = 3$, in 15 for $n = 4$, and doesn't converge at all for $n = 5$. For $n = 4$, we see that a lower estimated error is achieved in iteration $k > 19$, *after* the plateau; this is again a side-effect of solving a perturbed problem.

5.2. L-shaped domain and first run of *hp*-AFEM

In the remainder of this chapter, we will solve the Poisson problem with forcing function $f = 1$ on the L-shaped domain. In formula form:

$$\begin{cases} -\triangle u = 1 & \text{on } \Omega \\ u = 0 & \text{on } \partial\Omega \end{cases} \qquad \text{with } \Omega := (-1,1)^2 \setminus [0,1) \times (-1,0].$$

We define the initial triangulation $\mathcal{D}_0$ as six triangles that are simple translations and rotations of each other, each equipped with quadratic polynomials. See the illustration on the right.



This domain has a re-entrant corner which induces a strong singularity in the solution, so that $u \notin H^2(\Omega)$. The solution u also exhibits mild singularities in the *salient* (non-reentrant) corners. Owing to the fact that $u \in H^2(\Sigma)$ for all Σ with $\overline{\Sigma} \subset \Omega$, we expect little refinement away from the corners, with strong h-refinement towards the re-entrant corner and milder h-refinement towards the others.

5.2.1. Error progression

Let us run the application using the default settings. See Figure 5.2.1 for two graphs of its error progression. The right graph shows that the most time-consuming step within each iteration is the call to *hp*-NearBest (denoted by the segment connecting a $\bullet$ with a $\times$). In fact, further analysis shows that in each iteration, upwards of 90% of the time is spent within *hp*-NearBest. This routine is the obvious bottleneck of the algorithm.

Looking at the left graph, we see a clear partition of the results. Below the dotted line, a clear exponential decay in terms of $\#\,\mathrm{DoF}^{1/3}$ is visible: Connecting the $\times$-markers yields a straight line. This is the *asymptotic regime*. Within the asymptotic regime, the slope of the line is -0.312: The error decays like $\exp(-0.312(\#\,\mathrm{DoF})^{1/3})$.

Above the dotted line is the *preasymptotic regime*. The triangulations produced here are just too small to be approximated by the algorithm in a true near-best way. This is underlined by the fact that $\#\,\mathrm{DoF}$ is not monotonically increasing.

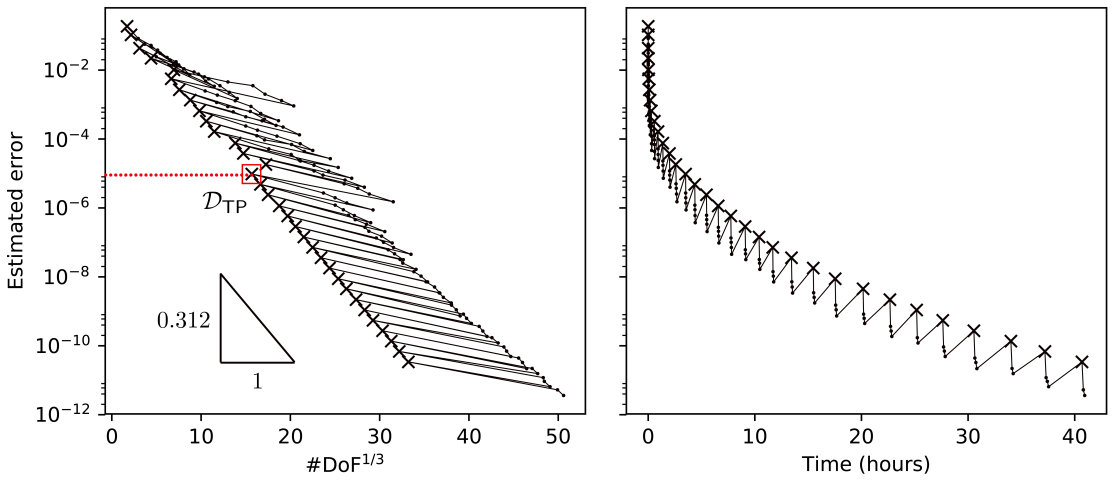


Figure 5.2.1.: Error progression of the triangulations produced by *hp*-AFEM on the model problem, using default parameters. Left: progression as a function of $\#\,\mathrm{DoF}^{1/3}$; right: as a function of time. An $\times$ denotes the output triangulation of *hp*-NearBest; $\bullet$ a triangulation within Reduce.

5.2.2. Addressing some problems in the theory

To get a feel for the qualitative properties of these near-best triangulations, let us look at a few in more detail. In Chapter 2, we identified a couple of issues with the theory. More specifically, we saw that

- We cannot bound the increase in total complexity when computing the smallest conforming refinement of an *hp*-triangulation;
- We lose a logarithmic factor in going from the broken error on a near-best triangulation to the energy error norm on its smallest conforming refinement;
- Had we used the Melenk-Wohlmuth error estimator, we would have had to restrict our definition of *conformity* to *comparable* triangulations, where the quotient of local complexities of neighbouring elements is bounded;
- Had we defined our reduction factor as $\rho := \mu/(1 + C_{\mathsf{B}}(\mathcal{D}_k^*)\omega)$, we would have seen a (slight) rise in the number of iterations inside Reduce.

We investigate these problems by looking at a few associated quantities, for a few triangulations found by *hp*-AFEM. See Table 5.2.2; it is partitioned into three segment, each discussing a different set of properties.

The first segment

The top segment shows basic properties of the triangulations. The triangulation of $k = 15$ jumps out; we will discuss it in a second. Apart from this oddity, we see that both error and total number of degrees of freedom are monotonic, and that the maximum polynomial degree is monotonic inside the asymptotic regime.

	$k=1$	$k=5$	$k=15$	$k=16$	$k=25$	$k=33$
		Preasymptotic regime		*Asymptotic regime*		
$\#\,\mathrm{DoF}$	5	336	5139	4615	16314	36655
$\left\|u-u_{\mathcal{C}(\mathcal{D}_k^*)}\right\|_{H_0^1(\Omega)}$	1.86e−1	1.02e−2	1.82e−5	4.82e−6	8.81e−9	3.41e−11
$\left\|p_{\mathcal{D}_k^*}\right\|_\infty$	2	11	20	9	12	16
$\max\{\eta(D)\}/\min\{\eta(D)\}$	2.17	46.1	601	41.9	20.5	8.66
% time in NearBest	99%	73%	92%	96%	95%	93%
$\#\mathcal{C}(\mathcal{D}_k^*)/\#\mathcal{D}_k^*$	1.42	1.49	1.63	1.23	1.16	1.12
Broken error quotient	0.523	0.162	0.172	0.105	0.130	0.115
Comparability quotient	1	5.5	10	4.5	6	8
# iters in Reduce	5	6	4	4	3	2

Table 5.2.2.: Various interesting quantities for a selection of the triangulations produced by *hp*-AFEM with default settings on the model problem.

The middle segment

In the middle segment, two quantities are shown. Firstly, we see that $\max(\eta)/\min(\eta)$—the quotient of maximum and minimum local estimated errors—is fairly small, suggesting that *hp*-NearBest is doing a good job in homogenizing local errors over the domain. Moreover, this becomes better for larger triangulations.

Concurrently, we see that within this single iteration of *hp*-AFEM, at least 90% of the time is spent within the *hp*-NearBest routine. This again underlines that we may expect the application to become 10 to 20 times faster if we optimize this step.

The final segment

The bottom segment quantifies the concerns raised previously. The top row suggests that in going from $\mathcal{D}_k^*$ to its smallest conforming refinement, we only see a *modest* increase in total number of DoFs; certainly not unbounded.

Theorem 2.2.16 shows that

$$\inf_{w\in V(\mathcal{C}(\mathcal{D}))}\|v-w\|_{H_0^1(\Omega)}\leq C_{\mathsf{B}}(\mathcal{D})E_{\mathcal{D}}(v)^{1/2}\quad(v\in H_0^1(\Omega)),\quad C_{\mathsf{B}}(\mathcal{D})\eqsim(1+\log\|p_{\mathcal{D}}\|_\infty)^{3/2}.$$

From this, we deduce that

$$\frac{\inf_{w\in V(\mathcal{C}(\mathcal{D}_k^*))}\left\|u_{\mathcal{D}_{k-1}}-w\right\|_{H_0^1(\Omega)}}{\left(1+\log\left\|p_{\mathcal{D}_k^*}\right\|_\infty\right)^{3/2}E_{\mathcal{D}}(u_{\mathcal{D}_{k-1}})^{1/2}}\lesssim 1,$$

with smaller values being more favourable. The second row of the final segment inside Table 5.2.2 shows this quotient. We see that these values are all well below one, suggesting that this "logarithmic increase in error" is a worst-case rather than typical-case scenario.

The third row displays the largest quotient of neighbour complexities. This maximum is attained between an element on a salient corner (endowed with a very high degree) and its

(degree-2) non-corner neighbour; see Figures 5.2.3 and 5.2.4 for examples. This would pose a definite problem when using the Melenk-Wohlmuth error estimator and truly taking $k \to \infty$; in our case, it is of little interest.

The final row shows the number of iterations within Reduce. Of course, our reduction parameter is fixed at $\rho := \mu$ and not dependent on $\|p_{\mathcal{D}}\|_\infty$, but still, it is remarkable to see that the number of iterations *decreases* rather than increases as triangulations become bigger.

The case $k = 15$

Looking at this table, $k = 15$ immediately pops out. Visual inspection of the triangulation shows why its values are different from the others: *hp*-NearBest returns a nonconforming triangulation with a very large and high-degree element, so its smallest conforming refinement contains a lot more degrees of freedom and is much less near-best. This is essentially the case of Example 2.2.14, thus explaining the (relatively) large $\#\mathcal{C}(\mathcal{D}_k^*)/\#\mathcal{D}_k^*$, as well as the large difference in local estimated errors.

5.2.3. Two near-best triangulations

Let us look at two triangulations visually: The tipping point triangulation—see the red box in Figure 5.2.1—and the final triangulation found by *hp*-AFEM. We plot the full triangulation, with zoom-ins into the corners that show strong h-grading.

Properties of the tipping point triangulation

Denote with $\mathcal{D}_{\mathsf{TP}}$ the triangulation on the tipping point between these two regimes. This *tipping point triangulation* is found at iteration $k = 16$ of *hp*-AFEM, after around 3 hours of runtime. It carries a total of $\#\,\mathrm{DoF}(\mathcal{D}_{\mathsf{TP}}) = 4615$ degrees of freedom, with estimated error $\left\|u - u_{\mathcal{D}_{\mathsf{TP}}}\right\|_{H_0^1(\Omega)} \approx 4.825 \times 10^{-6}$. Its maximum polynomial degree is $\left\|p_{\mathcal{D}_{\mathsf{TP}}}\right\|_\infty = 9$. In Figure 5.2.3, this triangulation is shown: The top shows the *hp*-triangulation itself, and the bottom shows the (estimated) error on each element.

We see from the top figure that $\mathcal{D}_{\mathsf{TP}}$ is very symmetrical. It has large, high-degree elements in the interior of the domain, which corresponds with a smooth local solution. In the salient corners, we see slight h-refinement, which is due to the (mild) singularities of u. Stronger h-grading is seen near the re-entrant corner—there is a strong singularity here. As we get closer to a corner, the local solution gets progressively more singular and hence cannot be approximated well by (high-order) polynomials.

We note that, remarkably, the algorithm opts for strong p-enrichment in the triangles directly adjacent to a corner. This is behaviour we cannot explain; conventional wisdom tells us these degrees of freedom would be better spent on h-refinement.

From the bottom figure, it is obvious that this total error is governed by the errors of the few triangles touching the corners. Looking at the color bar, we see that the errors are distributed fairly homogenously, in that the errors are all within two orders of magnitude from each other. This signals to us that the routine *hp*-NearBest is doing a good job distributing the total error evenly across all elements.

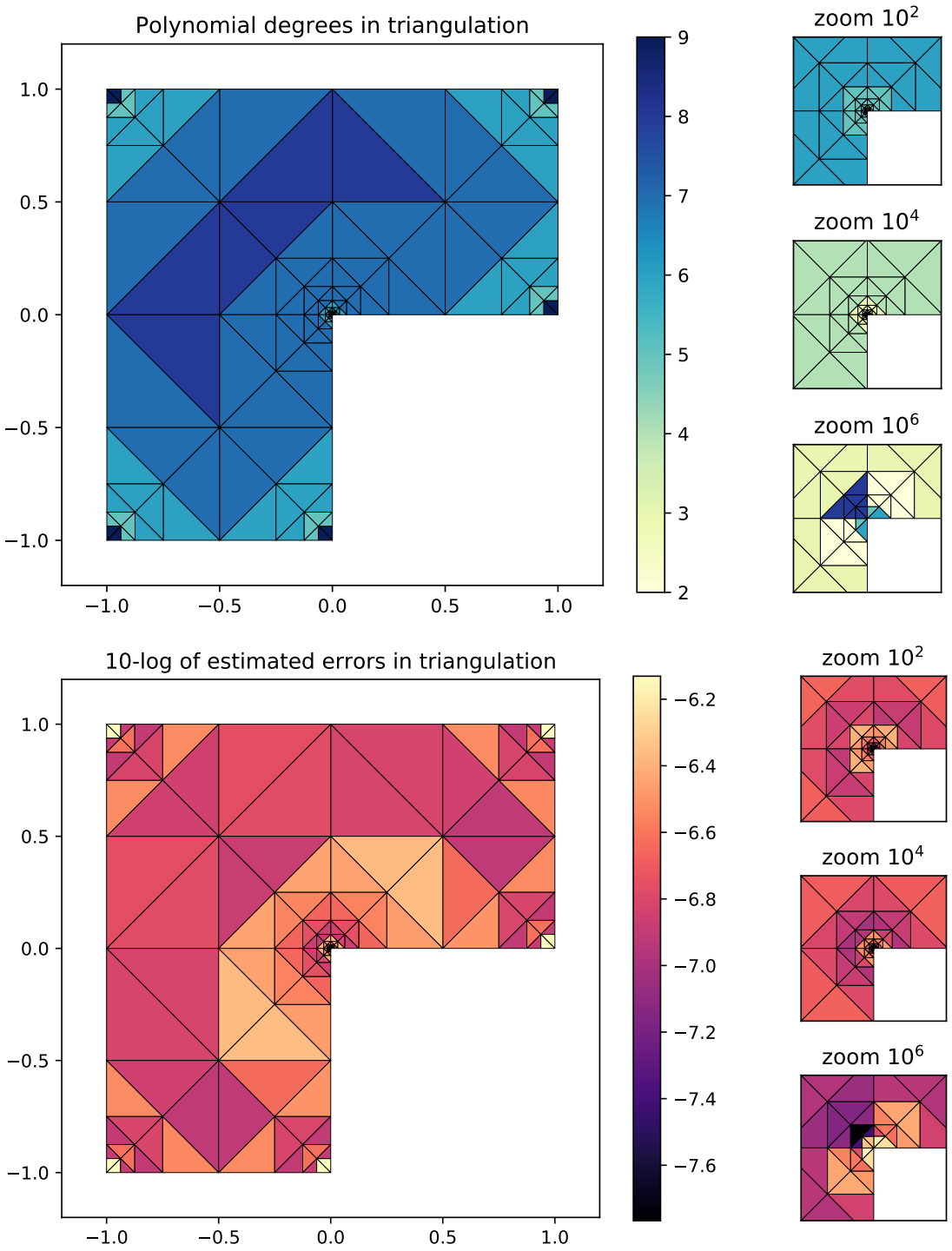


Figure 5.2.3.: The tipping point triangulation $\mathcal{D}_{\mathsf{TP}}$. Left: the full triangulation. Right: the reentrant corner, for various zoom levels.

Properties of the final triangulation

For completeness, let us also consider the triangulation found in the very last iteration of *hp*-AFEM—we killed the application after this result. It is found after 33 iterations, requiring upwards of 40 hours to compute. The total number of degrees of freedom is 36655, with estimated error 3.407×10^{-11}. Its maximum polynomial degree is 16. See Figure 5.2.4 for a visualization.

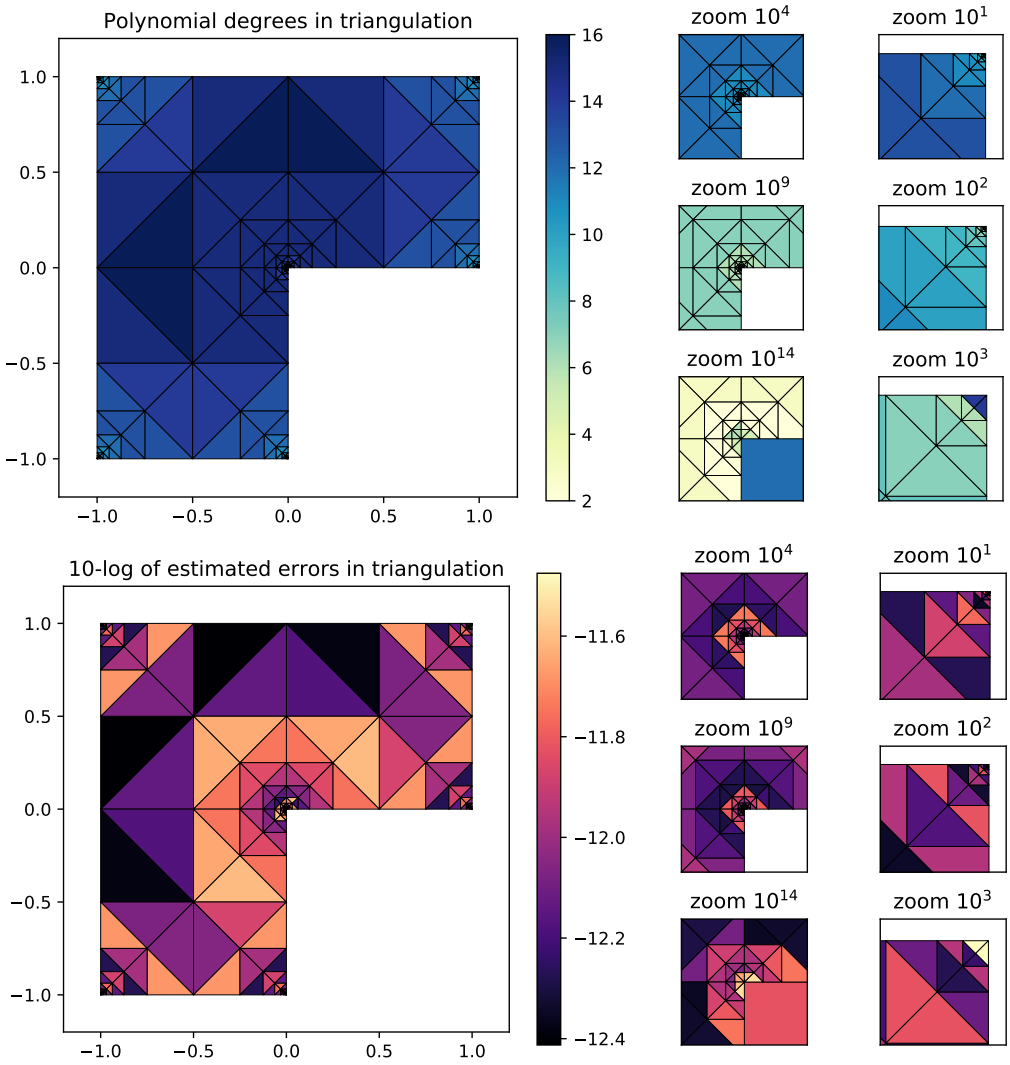


Figure 5.2.4.: The final triangulation found in *hp*-AFEM with default parameters. Left: the full triangulation. Middle: the reentrant corner, for various zoom levels. Right: the top-right salient corner, for various zoom levels.

There are a lot of similarities between the final triangulation and the tipping point triangulation of the previous paragraph. Again we see h-refinement towards the corners of the domain, and p-enrichment for the triangles directly adjacent to a corner.

Remarkable is the very strong h-grading towards the origin; a zoom level of 10^{14} was needed

to display the smallest triangles. The size of this triangulation required us to zoom in to a salient (not re-entrant) corner as well. Here, we see a strong p-grading (elements of degree 16), much stronger than for the re-entrant corner (degree 8). This is in contrast with the tipping point triangulation, where we saw elements of degree 9 in the salient corners with degree 8 in the re-entrant one.

From the bottom graph, we see that the total error is again governed by local errors in the corners. The distribution is even better than in the tipping point triangulation, with errors all within one order of magnitude from each other. This again shows that hp-NearBest takes a long time warming up and its results improve as triangulations get more complex.

5.3. Comparing hp-AFEM with other research

Now that we have a feeling of how our algorithm performs on the model problem, let us compare its results with some other results found in literature.

5.3.1. Conditioning of global stiffness matrices

In Chapter 3, we spent a considerable amount of time inspecting properties of the element matrices. Let us turn to the global view this time. Running hp-AFEM with default parameters on the model problem, we record the global stiffness matrix at each step. We employ the eigenvalue package Spectra [39] to estimate the smallest and largest eigenvalues of the matrices; because our global stiffness matrix is symmetric, we have

$$\kappa_2(A) := \frac{\sigma_{\max}}{\sigma_{\min}} = \frac{|\lambda_{\max}|}{|\lambda_{\min}|}.$$

Figure 5.3.1 shows the 2-norm condition number of these matrices.

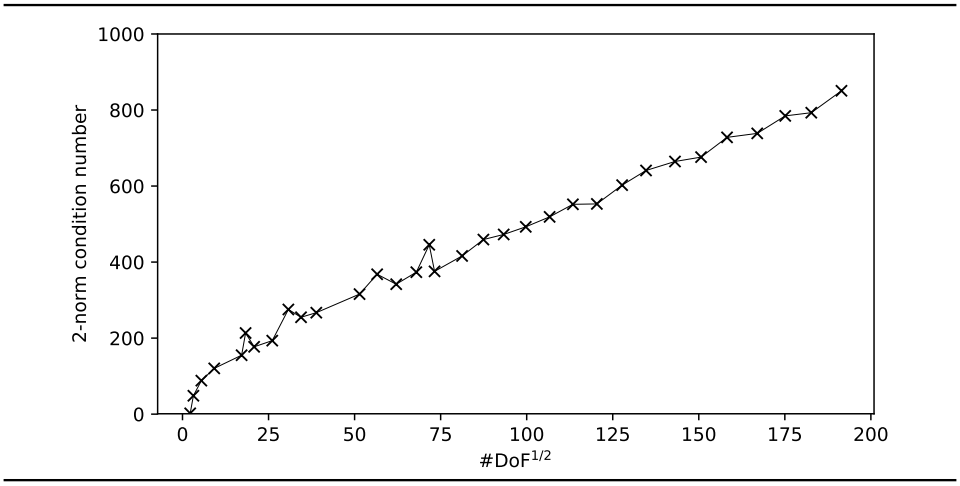


Figure 5.3.1.: Conditioning of the global stiffness matrices found when running hp-AFEM with default parameters on the model problem.

This figure shows that these stiffness matrices are extremely well-conditioned, even for very large systems. In [49], Xin *et al.* conduct numerical experiments for a range of problems. Each

experiment consists of uniform p-refinement of an initial triangulation, for the Lagrange- and Aiffa bases. Xin *et al.* conclude that the Lagrange basis exhibits exponential growth in terms of $p \eqsim (\#\,\mathrm{DoF})^{1/2}$, whereas the hierarchical Aiffa basis can show just linear growth. Our figure clearly reflects this conclusion, and shows that it generalizes to our hp-adaptive strategy as well.

5.3.2. Comparison with hp-REFINE

It is seen in practice [19, 35] that exponential decay for the model problem can be achieved using a priori information of the solution. One such method (often called IDEAL in literature) is driven by a Dörfler marking strategy, opting for h-refinement of a marked element when it contains a singularity, and choosing p-enrichment for all other elements.

IDEAL is driven by a Dörfler marking; we arbitrarily chose $\theta = 0.8$ (but lower values showed very similar results). Our algorithm is invoked with default parameters.

Looking at Figure 5.3.2, we see that both methods exhibit exponential convergence, and that their decay rates are virtually indiscernable. Our conclusion is that the triangulations produced by hp-AFEM are not even near-best, but practically ideal as well.

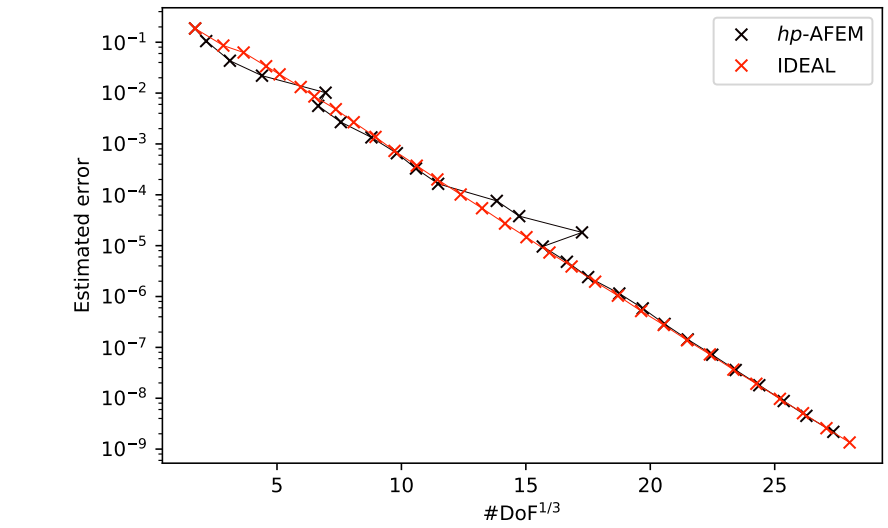


Figure 5.3.2.: Error progressions of (black) hp-AFEM versus (red) IDEAL on the model problem. The slope of a triangle correspond with that of the linear least-squares fit to the data.

On the other hand, it would be entirely unreasonable to compare computation times: hp-REFINE finishes in mere minutes, whereas hp-AFEM requires upwards of a day to complete. Of course, for our method, no a priori information is necessary for the exponential decay, whereas hp-REFINE explicitly requires knowing the singularities.

An example triangulation

For completeness, let us consider the triangulation $\mathcal{D}_{54}$ found at iteration 54 of IDEAL. It carries $\#\,\mathrm{DoF}(\mathcal{D}_{54}) = 4774$ degrees of freedom, which is very close to that of the tipping point triangulation $\mathcal{D}_{\mathsf{TP}}$ of Figure 5.2.3. Its error is $\left\|u - u_{\mathcal{D}_{54}}\right\|_{H^1_0(\Omega)} \approx 3.89 \times 10^{-6}$, with maximum polynomial degree $\left\|p_{\mathcal{D}_{54}}\right\|_\infty = 8$.

In Figure 5.2.3, we saw sharp p-enrichment on the elements directly adjacent to a corner. By construction, $\mathcal{D}_{54}$ has h-refinement here, but apart from this behaviour, the triangulations are very similar. We do see that the errors are distributed in a slightly more homogenous way; the estimator quotient is 10.2 for this triangulation, whereas it was 41.9 for $\mathcal{D}_{\mathsf{TP}}$ (cf. Table 5.2.2).

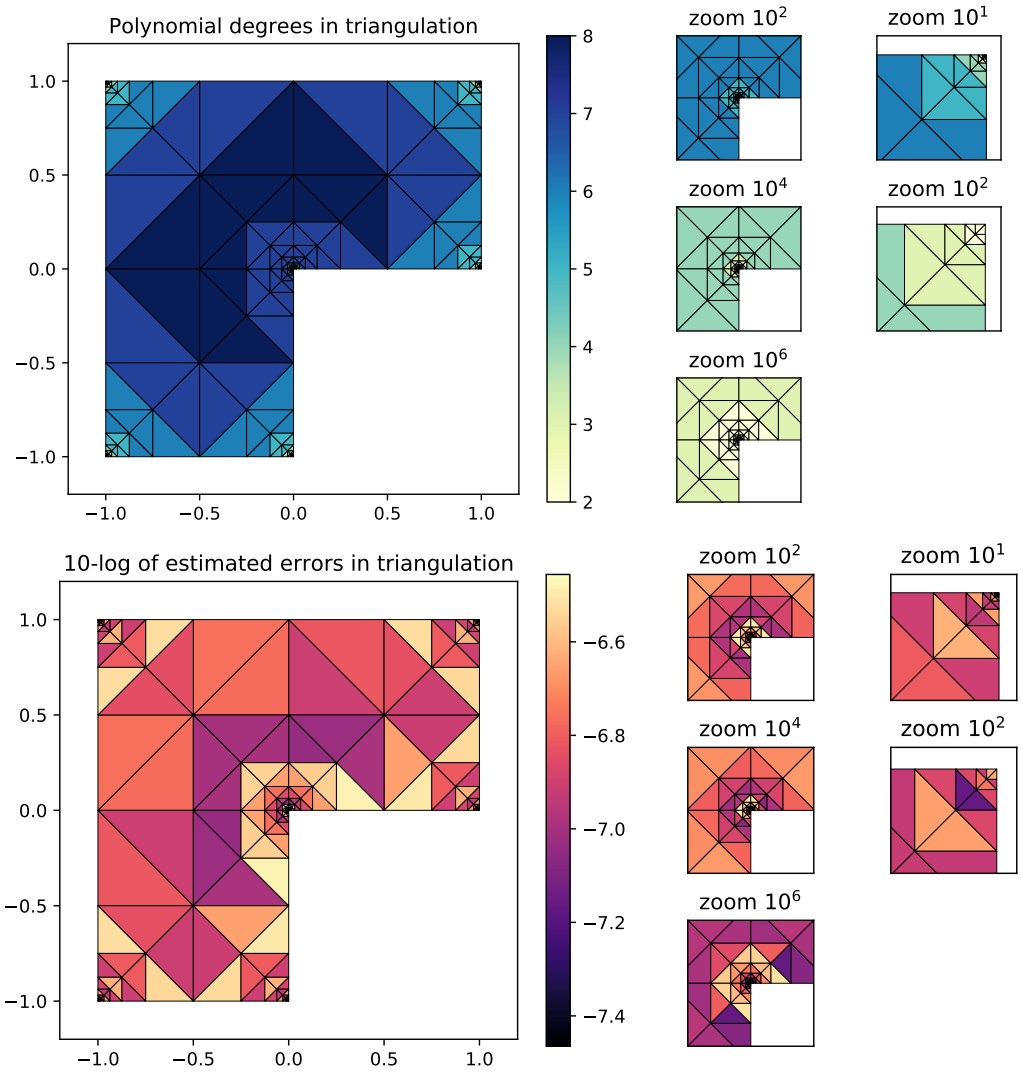


Figure 5.3.3.: The triangulation $\mathcal{D}_{54}$ produced at iteration 54 of IDEAL, with marking parameter $\theta = 0.8$ on the model problem. Left: the full triangulation. Middle: the reentrant corner, for various zoom levels. Right: the top-right salient corner, for various zoom levels.

5.3.3. Comparison with h-AFEM

We see from Figure 5.3.4 that the comparison of our novel algorithm with the well-established h-AFEM are very much in our favour. We start both algorithms from the tipping point triangulation $\mathcal{D}_{\mathsf{TP}}$ from Figure 5.2.1.

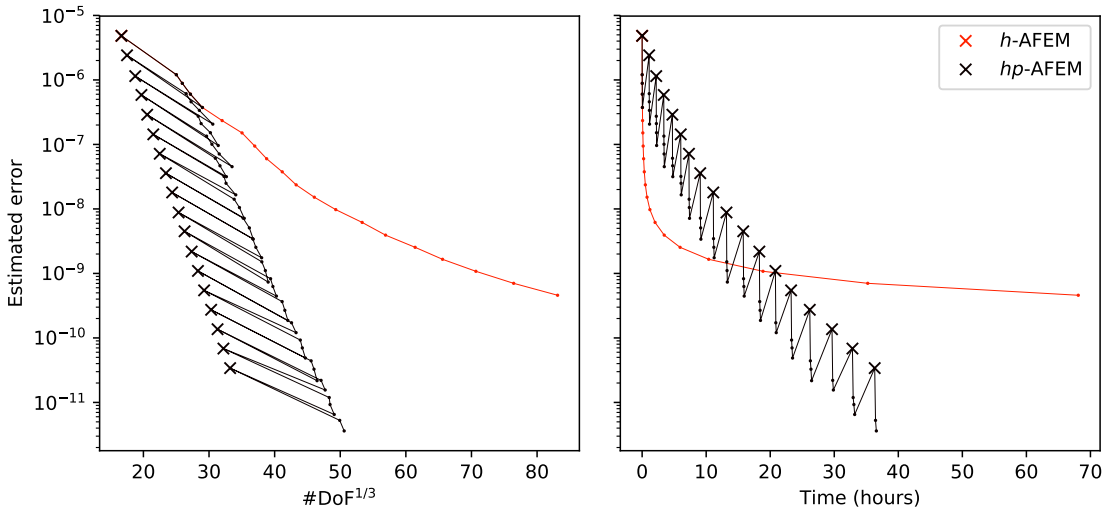


Figure 5.3.4.: Error progressions of (red line) h-AFEM and (black line) hp-AFEM on the model problem, starting from the tipping point triangulation in Figure 5.2.1.

The first call to Reduce inside hp-AFEM creates triangulations identical to those of h-AFEM, explaining the overlap of the first few data points. We see that h-AFEM does not exhibit exponential decay in terms of the number of degrees of freedom, leading to a clear distinction between the two algorithms.

The computational cost of hp-NearBest leads to a head start of h-AFEM in the first 20-or-so hours, albeit with much larger systems. The computer was able to perform 21 iterations of h-AFEM before running out of memory. From this, we gather that the main advantage of the current implementation of hp-AFEM over h-AFEM is the (smaller) size of the triangulations it produces, and not (yet) the speed. We think that optimizing hp-NearBest (as described in Remark 4.2.9) will help our case tremendously.

5.3.4. Comparison with hp-DECAY

Dolejší *et al.* [19, §6.1] conduct experiments with an hp-adaptive FEM in a setting similar to ours, the main differences being that their method is based on a heuristic (and hence cannot be proven to work optimally in every case) with discontinuous Galerkin (allowing hanging nodes to appear in the triangulations) with the equilibrated fluxes error estimator (which is provably reliable and efficient in a p-robust way, in contrast with our *refinement* error estimator).

Their problem consists of solving a Poisson problem on the L-shaped domain Ω, with prescribed solution $u(r,\theta) := r^{2/3}\sin(2\theta/3)$, forcing function $f = 0$, and the induced inhomogenous Dirichlet condition on $\partial\Omega$.

A result by Brenner [12] shows that for our model problem, the solution u can be expressed around the origin as

$$u(r,\theta) = w(r,\theta) + \kappa_1\phi(r)r^{2/3}\sin(2\theta/3) + \kappa_2\phi(r)r^{4/3}\sin(4\theta/3).$$

where: w is a smooth function vanishing in the origin; (r,θ) are polar coordinates around the origin; ϕ is a smooth cut-off function, being one in a neighbourhood of 0 with support small enough that u vanishes on $\partial\Omega$; and κ_1, κ_2 are real numbers.

In other words, u is the sum of a smooth function w and some functions around the re-entrant corner. Brenner then numerically finds estimates for the coefficients in [12, Tbl. 6]:

$$\kappa_1 \approx 0.4019, \quad \kappa_2 \approx 0.$$

We conclude that, around the origin, our solution u satisfies

$$u \approx w + 0.4019\phi(r)r^{2/3}\sin(2\theta/3). \tag{5.3.5}$$

Note the similarity between their solution and our observation in (5.3.5): If we assume that *hp*-AFEM has little trouble approximating the smooth function w well (a sensible assumption), and that the approximation error accumulates around the origin (a shaky assumption at best), our errors should lie somewhere close to κ_1 times theirs.

Quantitatively, a comparison of their error decay graph [19, Fig. 3] with ours yields the left of Figure 5.3.6. Their results are supplemented with the IDEAL method—note that in their case, the solution has only a single singularity so it will perform h-refinement in only the reentrant corner.

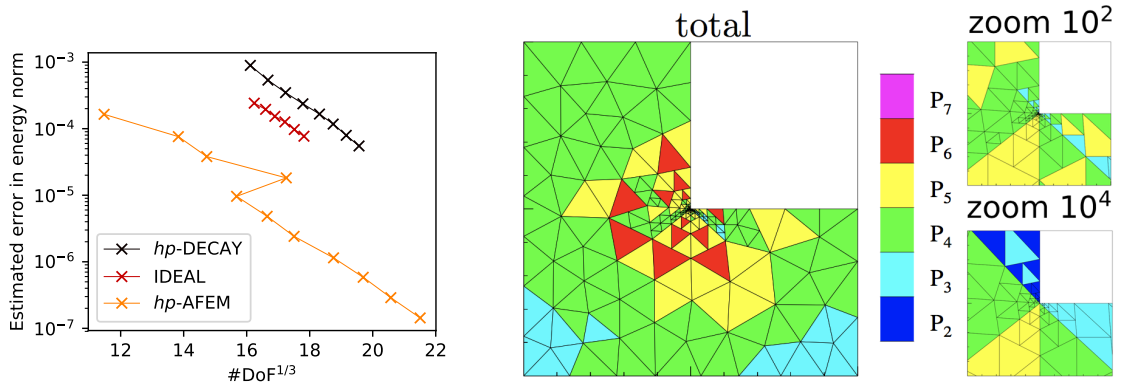


Figure 5.3.6.: Left: Error progression of the *hp*-DECAY and IDEAL methods described in [19], together with our own *hp*-AFEM. Right: The final triangulation of [19].

The figure reveals that our (estimated) errors are about two orders of magnitude (100 times) smaller than those of *hp*-DECAY—much smaller than the predicted factor of $\kappa_1 \approx 0.4019$—but also that this result should be taken with a grain of salt; even the errors of the IDEAL heuristic are worse than ours. (On the other hand, one can see that the results of *hp*-DECAY are worse than those of IDEAL, whereas in §5.3.2, we saw that on our model problem, error progression

of *hp*-AFEM was virtually the same as IDEAL.) As for the slope of the decay, we see that *hp*-AFEM and *hp*-DECAY are similar.

Qualitatively, their *final triangulation* (depicted right) shows characteristics similar to our triangulations—*h*-refinement towards the origin, with high degree elements in the center of the domain and lower degree towards the re-entrant corner. There are also some clear differences: Firstly, their solution has no intricacies in the salient corners of the domain, so there is hardly any *p*- or *h*-refinement there. This is in stark contrast with our situation. Moreover, *hp*-AFEM tends to do strong *p*-enrichment on the elements touching the corners, whereas neither the IDEAL- nor the *hp*-DECAY-triangulation carry this property.

5.4. Varying the algorithm parameters

It must be noted that the near-best result of Theorem 2.4.5 remains valid, regardless of the chosen parameter set. Therefore, the error progression of any valid choice of parameters is very similar to the left graph of Figure 5.2.1, as we will now see. For computation time and triangulation quality however, some parameter sets are more sensible than others.

5.4.1. Varying the Dörfler marking parameter

For instance, a high value of the Dörfler marking parameter $\theta \in (0, 1]$ results in very big systems, choking the computer. Figure 5.4.1 shows the error progression of the Galerkin solutions after each call to *hp*-NearBest, for a few chosen values of θ. The left graph tells us that the choice of θ has very little effect on the (approximation error of the) near-best triangulations that are found.

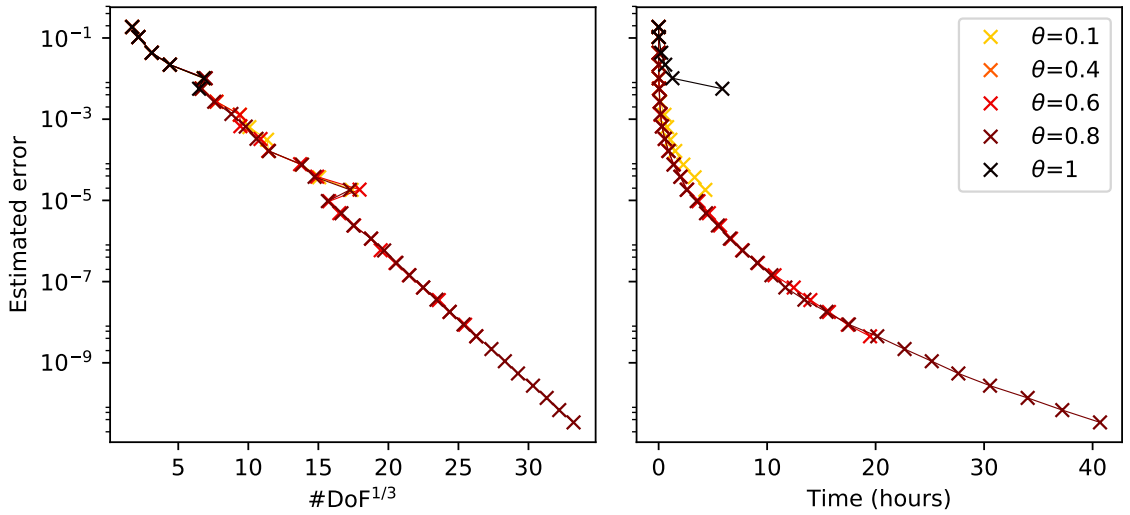


Figure 5.4.1.: Error progression for the solutions found by *hp*-AFEM on the model problem, for a few different values of θ.

The right shows us that $\theta = 1$—corresponding with uniform *h*-refinement—results in very long computation times and large systems; the computer ran out of memory after just 4 iterations. For other values, varying θ has little effect on computation time.

5.4.2. Varying the reduction and coarsening parameters

Varying the reduction parameter $\mu \in (0, 1)$ and the coarsening parameter $\omega \in (1, \infty)$ have little effect on the error progressions as well; see the left of Figure 5.4.2. We see that smaller μ results in faster computation times; this is thanks to the smaller number of calls to *hp*-NearBest.

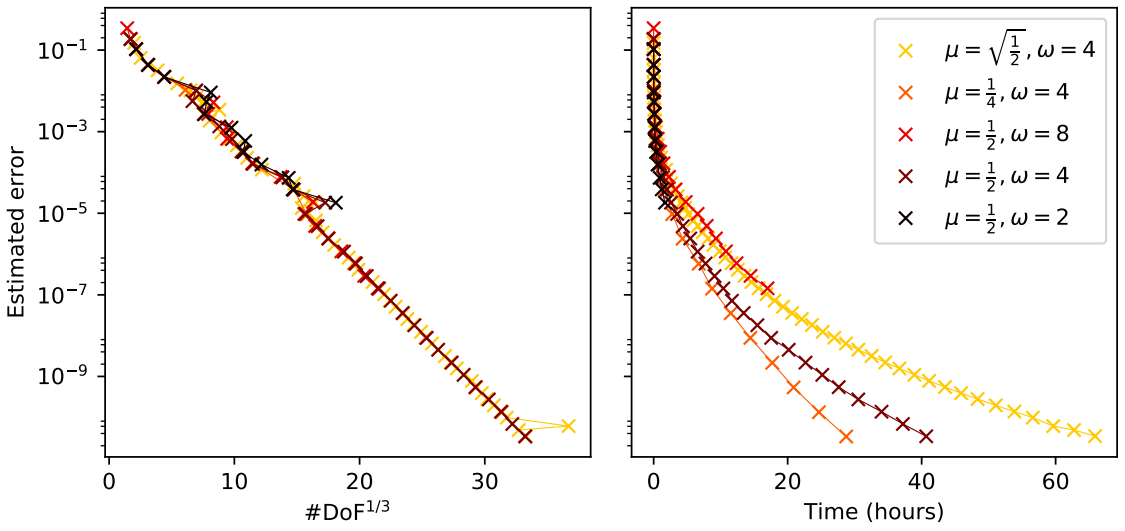


Figure 5.4.2.: Error progression for the solutions found by *hp*-AFEM on the model problem, for a few different values of μ and ω.

Of course, increasing the coarsening factor ω gives *hp*-NearBest more leeway to find a near-best approximation, thus increasing the quality of the triangulations—but also the computation time.

5.4.3. Quality of the triangulations under parameter changes

Let us try to quantify the strength of the different parameter sets by looking again their performance with respect to a few quantities. For each model that reached the asymptotic regime, we select the triangulation closest to the tipping point triangulation $\mathcal{D}_{\mathsf{TP}}$ of Figure 5.2.3. The results are in Table 5.4.3, again divided into three segments.

The (estimated) errors and number of degrees of freedom are all very close to each other, so we can make a good comparison. In fact, we see that most of the numbers are so close that no specific value is worth mentioning; this again underlines the robustness of *hp*-AFEM.

The only true difference is seen in local indicators. We see that the parameters $\omega = 8, \mu = \frac{1}{2}$ result in very homogenous spread of the error; the largest local errors (found in the salient corners) are just 10.2 times as big as the smallest ones.

Conclusion

From the results in this chapter, we conclude that the *hp*-AFEM is excellent at finding near-best triangulations that look beautiful. The error progressions produced by this algorithm show

Non-default parameter *hp*-AFEM iter	n/a $k = 15$	$\theta = 0.6$ $k = 15$	$\mu = 0.25$ $k = 8$	$\omega = 8$ $k = 16$	$\mu = \sqrt{0.5}$ $k = 29$
$\#\,\mathrm{DoF}$	3855	3929	3929	3783	3975
$\left\|u - u_{\mathcal{C}(\mathcal{D}_k^*)}\right\|_{H_0^1(\Omega)}$	9.62e−6	9.53e−6	9.53e−6	9.65e−6	9.54e−6
$\left\|p_{\mathcal{D}_k^*}\right\|_\infty$	12	12	12	12	12
$\max\{\eta(D)\} / \min\{\eta(D)\}$	20.3	60.2	60.2	10.2	60.2
% time in NearBest	90%	84%	90%	87%	96%
$\#\mathcal{C}(\mathcal{D}_k^*)/\#\mathcal{D}_k^*$	1.22	1.23	1.23	1.21	1.24
Broken error quotient	0.146	0.143	0.172	0.144	0.136
Comparability quotient	6	6	6	6	6
# iters in Reduce	4	6	5	5	3

Table 5.4.3.: The quantities of Table 5.2.2 for a selection of similar triangulations produced by *hp*-AFEM with different parameter values.

very clear exponential decay, and are very robust against changes in the parameters. Moreover, the choice of basis ensures we have a well-conditioned stiffness matrix, which in turn allows for high-accuracy solutions.

In terms of the quality of the found triangulations, the results are generally in our favour: On the model problem, *hp*-AFEM exhibits exponential decay, much better than the algebraic decay of *h*-AFEM. This error progression is indiscernable from the well-established heuristic IDEAL (which produces "ideal" triangulations, but requires a priori information about the solution and is therefore impractical). A direct comparison with the *hp*-adaptive heuristic *hp*-DECAY of Dolejší proved hard, but did show promising results.

The main bottleneck of this algorithm remains its computation time. Requiring roughly 5 hours to find a scientific tolerance of 10^{-6} on our model problem, its practicality is limited in realistic settings. However, the proposed solution looks promising.

Conclusion

In this thesis, we thoroughly investigated the novel algorithm *hp*-AFEM, both from a theoretical and a more practical standpoint.

In Chapters 1 and 2, we developed a framework for the theoretical analysis of *hp*-adaptive finite elements. We proved that the triangulations produced by *hp*-AFEM will, under mild conditions, exhibit an exponential convergence rate in terms of the number of degrees of freedom. This strongly improves over *h*-adaptive finite elements, where "only" algebraic decay can be expected.

In Chapter 3, our heads turned towards finding a local basis from which a global basis for the Galerkin space may be formed. We saw that the Lagrange basis usually chosen for *h*-adaptivity is ill-suited for our needs, and saw why a hierarchical local basis—where the basis of degree $p+1$ is formed by adding functions to the basis of degree p—offers a beautiful solution. We explored two examples of hierarchical bases, and the novel *Bernstein-Beézier* basis, and looked at their strengths and weaknesses within finite element context.

In Chapter 4, we looked at details one must consider when implementing *hp*-AFEM. Chapter 5 then discusses numerical experiments using this algorithm. We saw that it stacks very favourably against existing methods, and that its theoretical *near-best* property actually translates to practically ideal triangulations.

Future work

There are many interesting directions to follow in future work. Firstly, the current situation suffers from multiple theoretical problems—see the conclusion of Chapter 2. It is worthwhile to see if we can, by considering a smaller class of boundary value problems, mitigate these problems or maybe even solve them completely.

Practically, there are also a few details left unresolved. Firstly, these seems to be no unanimously optimal basis—see the conclusion of Chapter 3. In any case, the current hierarchical basis does not allow for quick computation of the required matrices *up to any degree*; therefore, the implementation relies on precomputed matrices which is a serious deficit of this choice. More investigation is needed to see if the Bernstein-Bézier basis is viable for our cause.

The implementation itself is far from perfect as well. In a subsequent version, one would likely want to resolve the current bottleneck—computing the error functionals inside *hp*-NearBest. We propose a solution in Remark 4.2.9. Moreover, it would benefit heavily from using some form of unit testing to make sure each routine works in typical situations (and edge cases!) The Google Test framework [29] has proven useful in the past. Better still, one should consider rewriting the library as a module of, for instance, the DUNE project; cf [17]. This could help robustness against errors, speed up development, and increase adoption within the finite element community.

In any case, there is loads of work still to be done. It is a good thing I will be continuing research as part of the NWO project *New challenges in adaptivity* under supervision of Rob Stevenson, together with colleague and long-time friend Raymond van Venetië. Hopefully, the many hours spent discussing this thesis with you will prove fruitful in the future.

Appendices

A. Near-best tree generation

Triangulations are *the* central point of this thesis. In Definition 1.5.5, we saw that given some initial triangulation $\mathcal{D}_0$, there is an equivalence between triangulations $\mathcal{D}$ found from $\mathcal{D}_0$ through bisection, and (leaves of) subtrees $\mathcal{T}$ of a binary tree $\mathfrak{K}$ with roots $D \in \mathcal{D}_0$. In this appendix, we will look at those subtrees rather than the triangulations.

A.1. Error mapping

Definition A.1.1. Central to our approach will be an abstract *error mapping*

$$e : \mathfrak{K} \times \mathbb{N} \to \mathbb{R} : D \mapsto e_D$$

that assigns an error to each possible *hp*-element D.

This error mapping is assumed to be decreasing under both h-refinement and p-enrichment in that

$$\begin{cases} e_{D^1} + e_{D^2} \leq e_D & K_{D^1}, K_{D^2} \text{ the children of } K_D, \text{ and } d_{D^1} = d_{D^2} = d_D, \\ e_{D'} \leq e_D & K_{D'} = K_D, \text{ and } d_{D'} \geq d_D. \end{cases} \tag{A.1.2}$$

◊

Definition A.1.3. On such an *hp*-triangulation $\mathcal{D} \in \mathbb{D}$, we define the *global hp-error*

$$E_{\mathcal{D}} := \sum_{D \in \mathcal{D}} e_D. \qquad \text{◊}$$

Remark A.1.4. With the global *hp*-error, (A.1.2) is equivalent to

$$E_{\tilde{\mathcal{D}}} \leq E_{\mathcal{D}} \quad (\tilde{\mathcal{D}} \geq \mathcal{D}). \qquad \text{◊}$$

We can use this error mapping to formulate a notion of optimality among classes of subtrees from Definition 1.5.5. We will start our discussion looking at the theory of h-subtree generation by Binev *et al.*; cf. [10]. After a short introduction, we will present the main result of this chapter: *hp*-subtree generation, again introduced by Binev; cf. [9].

A.2. Near-best h-subtree generation

Remark A.2.1. Every node of a subtree can be identified with an element in $\mathfrak{K}$. In this paragraph, focusing on h-refinement only, we will identify an element D with its domain K_D and often write K instead. ◊

Assumption A.2.2. In this paragraph, we will assume $\mathfrak{K}$ to have a single root; this corresponds with the initial triangulation $\mathcal{K}_0$ having a single element, implying that Ω is necessarily a triangle. We will mend this issue in the final section. ◊

In this h-adaptive case, the properties in (A.1.2) reduce to the subadditivity property:

$$e_K \geq e_{K^1} + e_{K^2}, \quad (K \in \mathfrak{K}) \tag{A.2.3}$$

when K^1 and K^2 are the children of K. The global h-error for a subtree $\mathcal{T}$ is then

$$E_{\mathcal{T}}^h := \sum_{K \in \mathcal{L}(\mathcal{T})} e_K,$$

where $\mathcal{L}(\mathcal{T})$ is the set of leaves of the subtree $\mathcal{T}$.

Definition A.2.4. We can define the *best N-term h-adaptive approximation error* as

$$\sigma_N^h := \inf_{\{\mathcal{T} \text{ subtree} : \#\mathcal{L}(\mathcal{T}) \leq N\}} E_{\mathcal{T}}^h. \tag{A.2.5}$$
$\Diamond$

Remark A.2.6. In the quantity σ_N^h, the infimum can be replaced by a minimum, because the set over which the infimum is taken is finite. However, the cardinality of this set is equal to the Nth *Catalan number*, which is known to grow like $4^N/N^{3/2}$; trying every tree in this set is intractable. $\Diamond$

Definition A.2.7. A sequence $(\mathcal{T}_N)_N$ of subtrees with $\#\mathcal{L}(\mathcal{T}_N) = N$ is *near-best* when there are $0 \leq b \leq 1 \leq B$ independent of N such that

$$E_{\mathcal{T}_N}^h \leq B\sigma_{bN}^h \quad (N \in \mathbb{N}). \qquad \Diamond$$

We can derive an iterative algorithm for finding trees $\mathcal{T}_N$ by greedily bisecting the leaf with largest error. We saw in [47, App. C] that—by studying its behaviour on a sequence of heaviside-like functions—this does not necessarily provide a near-best approximation.

It is remarkable, though, that we can alter this basic algorithm slightly and *gain* the near-best property. We modify the errors as below, and and up with the following algorithm.

Definition A.2.8. Define the *modified errors* as

$$\begin{cases} \tilde{e}_K := e_K & \text{when } K \text{ is a root;} \\ \frac{1}{\tilde{e}_K} := \frac{1}{e_K} + \frac{1}{e_{K^*}} & \text{when } K^* \text{ is the parent of } K. \end{cases} \tag{A.2.9}$$
$\Diamond$

Remark A.2.10. Let us look at the difference between e and $\tilde{e}$ in a little more detail. First, we see that when $e_{K^*} \approx e_K$—no error reduction from parent to child—$\tilde{e}_K \approx e_K/2$. When $e_K \ll e_{K^*}$, we see that $\tilde{e}_K \approx e_K$.

Thus, the modified error penalizes the absence of error reduction by a factor of up to $\frac{1}{2}$. If this persists, the modified error $\tilde{e}_K$ decays exponentially so the algorithm will refine elsewhere. $\Diamond$

Algorithm A.2.11 (Near-best subtree)**.** Define $\mathcal{T}_1$ as the tree containing *only* the root. Receive $\mathcal{T}_{N+1}$ from $\mathcal{T}_N$ by subdividing a leaf $K \in \mathcal{L}(\mathcal{T}_N)$ with largest $\tilde{e}(K)$ among all leaves. $\Diamond$

Theorem A.2.12 ([9, Thm. 2.1])**.** *Let the local errors e_K satisfy* (A.2.3)*, and build trees $\mathcal{T}_N$ by means of Algorithm A.2.11. Then the sequence $(\mathcal{T}_N)_N$ provides a near-best h-adaptive approximation, in that*

$$E_{\mathcal{T}_N}^h \leq \frac{N}{N-n+1}\sigma_n^h \tag{A.2.13}$$

for any $n \leq N$.

Assuming that finding e_K is an $\mathcal{O}(1)$ operation, the complexity for obtaining $\mathcal{T}_N$ is $\mathcal{O}(N)$, except for the sorting of $\{\tilde{e}_K : K \in \mathcal{K}\}$ which requires $\mathcal{O}(N \log N)$ operations.

Remark A.2.14. We can avoid sorting the local errors by *binary binning*: For each K, find a $\kappa \in \mathbb{Z}$ such that $2^\kappa \leq \tilde{e}_K < 2^{\kappa+1}$. Take any of the leaves from the nonempty bin with largest κ. This will increase the constant in (A.2.13) by 2 but lowers the total complexity of the algorithm to $\mathcal{O}(N)$. In applications however, the time spent computing the e_K will outweigh sorting a list by several orders of magnitude. ◊

Corollary A.2.15. *Theorem A.2.12 tells us that with $n = \lceil N/2 \rceil$, we have*

$$E^h_{\mathcal{T}_N} = \frac{N}{\lfloor N/2 \rfloor + 1} \sigma^h_{\lceil N/2 \rceil} < 2\sigma^h_{\lceil N/2 \rceil}$$

so that the subtrees produced by Algorithm A.2.11 are, at worst, half as good as the best possible subtree with half the number of leaves.

A.3. Near-best *hp*-subtree generation

The preceding paragraph showed that we can achieve near-best h-subtree approximation by a simple greedy algorithm that runs in linear time. Unfortunately, the hp case is much more involved.

First, let us recall the equivalence between h-triangulations created by newest vertex bisection from $\mathcal{K}_0$, and (the leaves of) h-subtrees. We can extend this equivalence to the hp case by equipping all nodes K of such a subtree $\mathcal{T}$ with a natural number, so that $\big\{D = (K, d) : K \in \mathcal{L}(\mathcal{T}), d \in \mathbb{N}\big\}$ becomes an hp-triangulation. This yields an *hp-subtree*, but in the current context, the construction of a *subordinate hp-subtree* will prove fruitful.

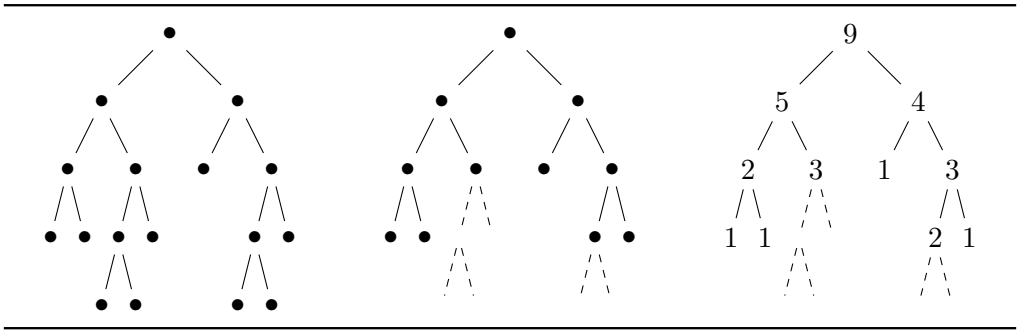


Figure A.3.1.: Left: An h-subtree $\mathcal{T}$. Middle: an h-subtree $\mathcal{T}' \subset \mathcal{T}$. Right: the subordinate hp-subtree $\mathcal{P}$.

Definition A.3.2. See Figure A.3.1 for a visualization. Let $\mathfrak{K}_K$ be the infinite binary tree rooted at K. Given an h-subtree $\mathcal{T}$, any subtree $\mathcal{T}' \subset \mathcal{T}$ can be converted to an *hp-subtree*

$$\mathcal{P}(\mathcal{T}', \mathcal{T}) := \big\{D = (K, d_K(\mathcal{T})) : K \in \mathcal{T}'\big\}$$

subordinate to $\mathcal{T}$ by defining its local complexities as

$$d_K(\mathcal{T}) := \#\mathcal{L}(\mathfrak{K}_K \cap \mathcal{T}) \quad \big(K \in \mathcal{T}'\big)\,.$$

Its total complexity then is defined as

$$\#\mathcal{P} := \sum_{D \in \mathcal{L}(\mathcal{P})} d_K = \#\mathcal{L}(\mathcal{T}). \qquad \text{◊}$$

We describe an algorithm that builds an h-subtree $\mathcal{T}_N$ with N leaves, and examines all possible hp-subtrees subordinate to $\mathcal{T}_N$ to find the $\mathcal{P}_N$ with minimal total error

$$\mathcal{P}_N := \operatorname*{arg\,min}_{\mathcal{P}\subset\mathcal{T}_N} E_{\mathcal{P}}, \quad E_{\mathcal{P}} := \sum_{D\in\mathcal{L}(\mathcal{P})} e_D.$$

We will first describe finding this hp-subtree $\mathcal{P}_N$, and then growing the h-subtree $\mathcal{T}_N$.

Subtree

Definition A.3.3. The *local hp-errors* $E_K := E_K(\mathcal{T})$ for nodes K of an h-subtree $\mathcal{T}$ are defined in terms of the local error as

$$E_K := \begin{cases} e_{K,1} & \text{when } K \in \mathcal{L}(\mathcal{T}); \\ \min\left\{E_{K^1} + E_{K^2}, e_{K,d_K(\mathcal{T})}\right\} & \text{when } K = K^1 \cup K^2. \end{cases}$$

This local hp-error encapsulates the competition between a piecewise approximation of lower local complexity, and a "global" approximation of higher complexity. ◊

Algorithm A.3.4 (Subtree)**.** To find $\mathcal{P}_N$ from $\mathcal{T}_N$, we start from the leaves of $\mathcal{T}_N$ and *trim* the tree every time $E_K = e_{K,d_K(\mathcal{T}_N)}$, i.e., every time a global approximation is preferable over a subdivision of the elements. ◊

Remark A.3.5. The dependency of $E_K(\mathcal{T}_N)$ on the h-subtree $\mathcal{T}_N = \{K\}$ is only through the number of leaves in the tree rooted at K, rather than the entire subtree $\mathcal{T}_N$. The value $E_K(\mathcal{T}_N)$ will change only when $\mathfrak{K}_K \cap \mathcal{T}_N$ is enlarged due to an increase in N. Therefore, it is convenient to define

$$E_{K,d} := \begin{cases} e_{K,1} & \text{when } K \in \mathcal{L}(\mathcal{T}); \\ \min\left\{E_{K^1} + E_{K^2}, e_{K,d}\right\} & \text{when } K = K^1 \cup K^2, \end{cases}$$

which coincides with $E_K(\mathcal{T}_N)$ on $D = (K,d) \in \mathcal{P}_N$ whenever $d_K(\mathcal{T}_N) = d$. ◊

Expand

Definition A.3.6. Analogous to the h-case (cf. (A.2.9)), we define the modified h-error $\tilde{e}_K$ for triangles K as

$$\begin{cases} \tilde{e}_K := e_{K,1} & \text{when } K \text{ is a root,} \\ \frac{1}{\tilde{e}_K} := \frac{1}{e_{K,1}} + \frac{1}{e_{K^*,1}} & \text{when } K^* \text{ is the parent of } K, \end{cases} \quad (K \in \mathcal{T}_N)\,. \qquad \text{◊}$$

Definition A.3.7. To monitor local error behaviour on the inner nodes of the tree, we define modified local hp-errors as

$$\begin{cases} \tilde{E}_{K,1} := \tilde{e}_K, \\ \frac{1}{\tilde{E}_{K,d}} := \frac{1}{E_{K,d}} + \frac{1}{\tilde{E}_{K,d-1}} & \text{when } d > 1. \end{cases}$$

Large modified local hp-errors correspond with nodes that are "high-potential" for efficiently driving the error down.

To locate the leaf of $\mathcal{T}_N$ that will (hopefully) drive the *hp*-error down the most, we find, at each node $K \in \mathcal{T}_N$, the quantity

$$q(K) := \begin{cases} \tilde{E}_{K,1} & \text{when } K \in \mathcal{L}(\mathcal{T}_N), \\ \min\left\{\max\left\{q(K^1), q(K^2)\right\}, \tilde{E}_{K,d_K(\mathcal{T}_N)}\right\} & \text{when } K = K^1 \cup K^2, \end{cases}$$

which defines the *leaf of highest potential* $s(K) \in \mathcal{L}(\mathfrak{K}_K \cap \mathcal{T}_N)$ through

$$s(K) := \begin{cases} K & \text{when } K \in \mathcal{L}(\mathcal{T}_N), \\ s(\arg\max\left\{q(K^1), q(K^2)\right\}) & \text{when } K = K^1 \cup K^2. \end{cases} \qquad \Diamond$$

The algorithm for incrementally growing $\mathcal{T}_N$ is then as follows.

Algorithm A.3.8 (Expand)**.** Define $\mathcal{T}_1$ as the tree containing only the domain root K_0. For each N, receive $\mathcal{T}_{N+1}$ from $\mathcal{T}_N$ by subdividing the leaf $s(K_0)$. $\Diamond$

Near-best *hp*-subtree

Algorithm A.3.9 (Near-best *hp*-subtree)**.** The full Near-best *hp*-subtree algorithm alternately calls Expand and Subtree, after each iteration receiving an *hp*-subtree $\mathcal{P}_N$ with complexity $\#\mathcal{P}_N = N$. For the self-contained description, see Algorithm A.3.10. A maximum number of iterations $N_{\max}$ can be specified. $\Diamond$

Definition A.3.11. In light of (A.2.5), we can define the *best N-term hp-adaptive approximation error* as

$$\sigma_N^{hp} := \inf_{\{\mathcal{P}\ hp\text{-subtree} \,:\, \#\mathcal{P} \le N\}} E_{\mathcal{P}}. \qquad \Diamond$$

Theorem A.3.12 ([9, Thm. 3.3])**.** *The h-subtree $\mathcal{T}_N$ and subordinate hp-subtree $\mathcal{P}_N$ produced by Algorithm A.3.9 provide near-best hp-approximation in the sense that*

$$E_{\mathcal{P}_N} \le \frac{2N-1}{N-n+1}\sigma_n^{hp}$$

for any $n \le N$.

Assuming we can compute the local hp-errors $e_{K,d}$ in $\mathcal{O}(1)$ operations, the algorithm obtains $\mathcal{P}_N$ in

$$\mathcal{O}\left(\sum_{D \in \mathcal{T}_N} d_K(\mathcal{T}_N)\right)$$

operations.

Corollary A.3.13. *Substituting $n = \lceil N/2 \rceil$ in the near-best estimate of Theorem A.3.12 yields*

$$E_{\mathcal{P}_N} \le \frac{2N-1}{\lfloor N/2 \rfloor + 1}\sigma_{\lceil N/2 \rceil}^{hp} < 4\sigma_{\lceil N/2 \rceil}^{hp};$$

the hp-subtrees produced by Algorithm A.3.9 are, at worst, a quarter as good as the best possible hp-subtree with half the complexity.

1: **procedure** Near-best *hp*-subtree($N_{\max} \in \mathbb{N}$)
2: $N := 1$; $\mathcal{T}_1 := \{K_0\}$; $\tilde{e}_{K_0} := e_{K_0,1}$; $E_{K_0,1} := e_{K_0,1}$; $\tilde{E}_{K_0,1} := \tilde{e}_{K_0}$;
3: $q(K_0) := \tilde{E}_{K,1}$, $s(K_0) := K_0$;
4: Subdivide $s(K_0)$ into K^1, K^2 yielding $\mathcal{T}_{N+1}$;
5: **for both** $K := K^1, K^2$ **do**
6: $\frac{1}{\tilde{e}_K} := \frac{1}{e_{K,1}} + \frac{1}{\tilde{e}_{s(K_0)}}$; $E_{K,1} := e_{K,1}$; $\tilde{E}_{K,1} := \tilde{e}_K$;
7: $q(K) := \tilde{e}_K$; $s(K) := K$;
8: $K := s(K_0)$;
9: $N := N + 1$;
10: **if** $N \geq N_{\max}$ **then**
11: **return**.
12: $d_K(\mathcal{T}_N) := d_K(\mathcal{T}_{N-1}) + 1$; calculate $e_{K,d_K(\mathcal{T}_N)}$;
13: $\{K^1, K^2\} :=$ children of K;
14: $E_{K,d_K(\mathcal{T}_N)} := \min\left\{E_{K^1,d_{K^1}(\mathcal{T}_N)} + E_{K^2,d_{K^2}(\mathcal{T}_N)}, e_{K,d_K(\mathcal{T}_N)}\right\}$;
15: $\frac{1}{\tilde{E}_{K,d_K(\mathcal{T}_N)}} := \frac{1}{E_{K,d_K(\mathcal{T}_N)}} + \frac{1}{\tilde{E}_{K,d_K(\mathcal{T}_N)-1}}$;
16: $X := \arg\max\{q(K^1), q(K^2)\}$; $q(K) := \min\left\{q(X), \tilde{E}_{K,d_K(\mathcal{T}_N)}\right\}$; $s(K) := s(X)$;
17: **if** $K = K_0$ **then**
18: **goto** line 4;
19: **else**
20: replace K with its parent;
21: **goto** line 12;

Algorithm A.3.10: The Near-best *hp*-subtree algorithm from [9, §3].

Corollary A.3.14. *The complexity estimate in Theorem A.3.12 varies between $\mathcal{O}(N \log N)$ for well-balanced trees and $\mathcal{O}(N^2)$ for highly skewed ones.*

Remark A.3.15. The complexity estimate in Theorem A.3.12 assumes that we can compute the local error functionals $e_{K,d}$ in $\mathcal{O}(1)$ time; this assumption is almost always violated, as one solves a linear system of size $\eqsim d$ to find this quantity. In fact, empirical results show that the bulk of the computation time of the total *hp*-AFEM algorithm is spent finding the local error functionals. ◊

A.4. Multiple roots

The algorithm as proposed by Binev works only for an initial triangulation with a single root element. However, in applications, this is never the case.

Remark A.4.1. Our first attempt at extending the algorithm to the case of multiple roots (cf. [47]) was acceptable at best: If the triangulation contains R roots, and we reduce the global error functional to ε/R on each root through Algorithm A.3.9, then the total error must be below ε. This however eliminates any adaptivity across roots; you can imagine that this is suboptimal. ◊

Canuto *et al.* [15, Rem. 3.1] propose a solution to the multiple root problem. We unify the R roots pairwise, at each step creating a *combination root*. We repeat this until a single

root remains; if this is not possible (because R is not a power of two; see Figure A.4.2) we create up to $\lceil \log_2 R \rceil - 1$ *virtual roots* which contain an empty element domain. We denote the augmented singly-rooted tree by $\hat{\mathfrak{K}}$.

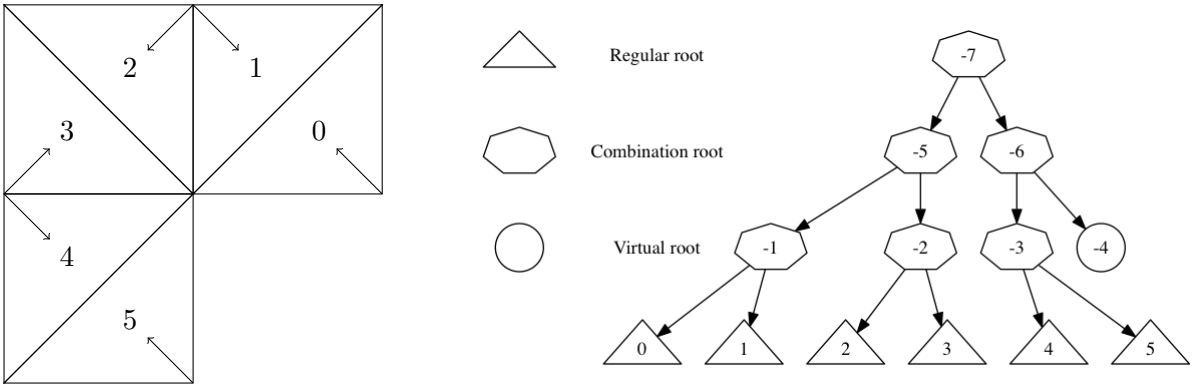


Figure A.4.2.: Combining elements to resolve the issue of multiple roots. Left: the original domain and triangulation. Right: the augmented tree $\hat{\mathfrak{K}}$.

We extend the definition of the approximation error $e_{K,d}$ as follows. For natural numbers d, we had already defined $e_{K,d}$; we define $e_{K,0}(v) := \|v\|^2_{H^1(K)}$ as the error of approximating v with the zero function.

For virtual roots K, we define $e_{K,d} := 0$ for any $d \geq 0$. For combination roots K with children K' and K'', we define $e_{K,d} := \min_{\{d',d'' \geq 0 : d'+d''=d\}} e_{K',d'} + e_{K'',d''}$. It is mostly trivial to show that this error functional conforms to the assumptions in (A.1.2). We will prove one property here.

Proposition A.4.3. *The extended error mapping e_D to the multiple-root case is again decreasing under p-enrichment.*

Proof. Let $K = K^1 \cup K^2$ be a combination root, and let d be given. With $\delta \in \mathbb{N}_0$, we see that

$$\begin{aligned} e_{K,d+\delta} &= \min_{\{d' \geq 0, d'' \geq 0 : d'+d''=d+\delta\}} e_{K^1,d'} + e_{K^2,d''} \\ &\leq \min_{\{d' \geq \delta, d'' \geq 0 : d'+d''=d+\delta\}} e_{K^1,d'} + e_{K^2,d''} \\ &= \min_{\{d'-\delta \geq 0, d'' \geq 0 : d'-\delta+d''=d\}} e_{K^1,d'-\delta} + e_{K^2,d''} \\ (\delta' := d' - \delta) &= \min_{\{\delta' \geq 0, d'' \geq 0 : \delta'+d''=d\}} e_{K^1,\delta'} + e_{K^2,d''} = e_{K,d}. \end{aligned}$$

□

Remark A.4.4. It must be noted that with the introduction of these combination- and virtual roots, we no longer have a one-to-one correspondence between leaves from subtrees and triangulations. Any such root appearing as a leaf of a subtree found through Algorithm A.3.9 must be subdivided. This introduces extra degrees of freedom, but for large enough N—so that the output subtree subdivides these roots organically—Theorem A.3.12 still holds. We chose not to investigate too much. ◊

B. Local-to-global mapping

This algorithm is a more in-depth version of Algorithm 3.2.12.

1: $N := 0$; ▷ # DoF seen until now
2: $V := \varnothing$; ▷ Array of global DoFs, indexed by vertex
3: **for all** $D_j \in \mathcal{D}$ **do** ▷ Runs over $i = 1, \ldots$
4: $r := 0$; ▷ Current local DoF
5: **for all** vertices v of D_j **do**
6: **if** $v \notin V$ **then**
7: **if** $v \in \partial\Omega$ **then**
8: $V[v] := \varnothing$; ▷ Vertex on domain boundary; no DoF
9: **else**
10: $V[v] := N$; ▷ Interior vertex; assign global DoF.
11: $N = N + 1$; ▷ Increase #DoFs seen
12: $i_{D_j}(r) := V[v]$; $r = r + 1$;
13: **if** $p_{D_j} < 2$ **then**
14: **continue**; ▷ Degree too low; no edge- or face DoFs
15: **for** $k = 1, \ldots, p_D$ **do**
16: **if** $k \geq 3$ **then** ▷ Assign face DoFs
17: **for** $r = 0, \ldots, k - 3$ **do**
18: $i_{D_j}(r) := N$; $r = r + 1$; $N = N + 1$;
19: **if** $k < p_{D_j}$ **then**
20: **for all** edges e of D_j **do**
21: **if** no neighbour across e **then** ▷ Edge on $\partial\Omega$; no DoF
22: $i_{D_j}(r) := \varnothing$; $r = r + 1$;
23: **continue**; ▷ No global DoF; continue
24: $D_k :=$ neighbour of D_j across e;
25: **if** $k \geq p_{D_k}$ **then** ▷ Neighbour complexity too low; no DoF
26: $i_{D_j}(r) := \varnothing$; $r = r + 1$;
27: **continue**; ▷ No global DoF; continue
28: $s :=$ index of the local DoF on D_k along e corresponding with r;
29: **if** $i < j$ **then**
30: $i_{D_j}(r) := N$; $i_{D_k}(s) := N$; ▷ Assign both locals the same global
31: $r = r + 1$; $N = N + 1$;
32: **else**
33: $i_{D_j}(r) := i_{D_k}(s)$; $r = r + 1$; ▷ Copy the global DoF

Algorithm B.0.1: Construction of the local-to-global mapping.

C. Popular summary

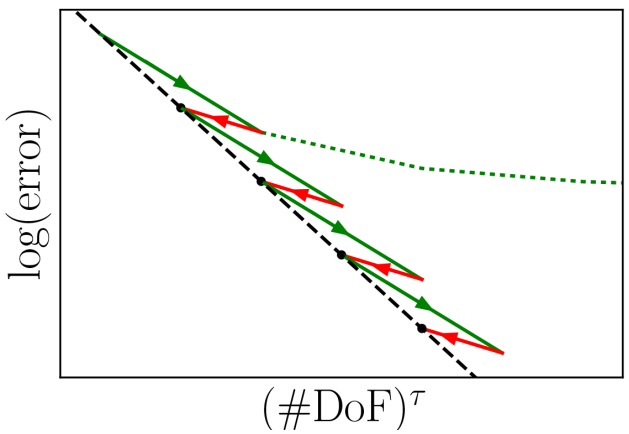
Partial differential equations (PDEs) describe many processes in nature, from the flow of water to the shape of a soap bubble. Often, it is hard (or even impossible) to find the function that solves such a PDE. In such cases, one looks for numerical solutions that approximate the true solution. In this thesis, we look at a *finite element method*: The domain of the function is partitioned into a large number of *elements*—in our two-dimensional case, we will subdivide a *polygon* into triangular elements. Endowing each triangle with a fixed polynomial degree, our finite element method aids in finding an approximate solution to the PDE that is continuous globally, and a polynomial on each triangle locally.

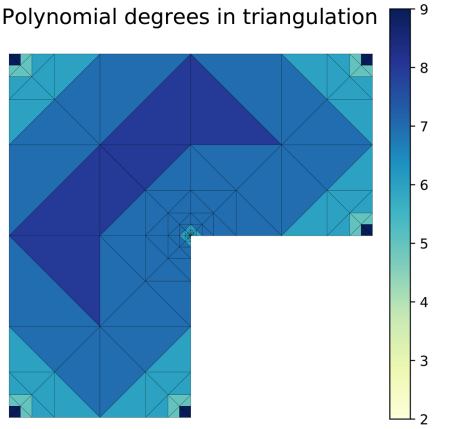
Given such an approximate solution, we often want to *refine* some of the triangles into smaller ones, so that we may construct a better solution on this refined grid. Often, it is natural to not refine *every* triangle, but only those on which the error (being the difference of true and approximate solution) is large. This is called an *h-adaptive* finite element, for it adaptively chooses which triangles to refine, thereby reducing their diameter h. Iterating such an h-adaptive FEM allows finding a sequence of approximate solutions for which the size of the global error can be shown to decay *algebraically* (like N^{-s} for some $s > 0$, where N is the total number of triangles).

In this thesis, we analyse a novel algorithm for an even more complex case—*hp*-adaptive finite elements—where we allow increasing the polynomial degree on each triangle separately. We will prove that, under mild circumstances, the size of the global error will decay *exponentially* (like $\exp(-N^\tau)$ for some $\tau > 0$) in the total number of degrees of freedom.

The algorithm works by alternating two routines—Reduce and NearBest. The former reduces the error size through a few cycles of h-adaptive FEM, and the latter increases the efficiency of the solution by throwing away near-redundant degrees of freedom, thereby sacrificing some accuracy. See the left Figure: the red line corresponds with a call to NearBest, and the green with a call to Reduce. The dotted green line shows the error progression of a pure h-adaptive FEM, whereas the black line—found by connecting all points from a call to NearBest—shows the possibly exponential convergence (corresponding with a straight line in a loglinear graph) of our *hp*-adaptive method.

Our implementation makes heavy use of an *hierarchical basis* for the polynomial space on each triangle, meaning that the basis for degree $p+1$ is found by adding some functions to the one for degree p. This implementation produces beautiful triangulations; see the right Figure.





Bibliography

[1] *Matplotlib: A 2D graphics environment*, Computing In Science & Engineering **9** (2007), no. 3, 90–95.

[2] S Adjerid, M Aiffa, and JE Flaherty, *Hierarchical finite element bases for triangular and tetrahedral elements*, Computer Methods in Applied Mechanics and Engineering **190** (2001), no. 22, 2925–2941.

[3] Mark Ainsworth, *Pyramid Algorithms for Bernstein-Bézier Finite Elements of High, Nonuniform Order in Any Dimension*, SIAM Journal on Scientific Computing **36** (2014), no. 2, A543–A569.

[4] Mark Ainsworth, Gaelle Andriamaro, and Oleg Davydov, *Bernstein–Bézier Finite Elements of Arbitrary Order and Optimal Assembly Procedures*, SIAM Journal on Scientific Computing **33** (2011), no. 6, 3087–3109.

[5] Martin S. Alnæs, Jan Blechta, Johan Hake, August Johansson, Benjamin Kehlet, Anders Logg, Chris Richardson, Johannes Ring, Marie E. Rognes, and Garth N. Wells, *The FEniCS Project Version 1.5*, Archive of Numerical Software **3** (2015), no. 100.

[6] I. Babuška, M. Griebel, and J. Pitkäranta, *The problem of selecting the shape functions for a p-type finite element*, International Journal for Numerical Methods in Engineering **28** (1989), no. 8, 1891–1908.

[7] Christian Bauer, Alexander Frink, and Richard year=2012 Kreckel, *GiNaC is Not a CAS*.

[8] Serge Bernstein, *Démonstration du théorème de Weierstrass fondée sur le calcul des probabilités*, Comm. Soc. Math. Kharkow **13** (1912), no. 1, 1–2.

[9] Peter Binev, *Tree approximation for hp-adaptivity*, IMI Preprint, University of South Carolina (2014, to appear).

[10] Peter Binev, Wolfgang Dahmen, and Ron DeVore, *Adaptive finite element methods with convergence rates*, Numerische Mathematik **97** (2004), no. 2, 219–268.

[11] D. Bræss, J. Schöberl, and V. Pillwein, *Equilibrated residual error estimates are p-robust*, Computer Methods in Applied Mechanics and Engineering **198** (2009), no. 13, 1189–1197.

[12] Susanne C. Brenner, *Multigrid Methods for the Computation of Singular Solutions and Stress Intensity Factors I: Corner Singularies*, Mathematics of Computation **68** (1999), no. 226, 559–583.

[13] Susanne C Brenner and Ridgway Scott, *The mathematical theory of finite element methods*, vol. 15, Springer Science & Business Media, 2008.

[14] Claudio Canuto, Ricardo H. Nochetto, Rob Stevenson, and Marco Verani, *On p-Robust Saturation for hp-AFEM*, (2016).

[15] ———, *Convergence and optimality of hp-AFEM*, Numerische Mathematik **135** (2017), no. 4, 1073–1119.

[16] P.G. Ciarlet, *The finite element method for elliptic problems*, Studies in Mathematics and its Applications, Elsevier Science, 1978.

[17] Andreas Dedner, Robert Klöfkorn, Martin Nolte, and Mario Ohlberger, *A generic interface for parallel and adaptive discretization schemes: abstraction principles and the dune-fem module*, Computing **90** (2010), no. 3, 165–196.

[18] L. Demkowicz, *Computing with hp-adaptive finite elements*, Chapman & Hall/CRC, 2007.

[19] Vít Dolejší, Alexandre Ern, and Martin Vohralík, *hp-Adaptation Driven by Polynomial-Degree-Robust A Posteriori Error Estimates for Elliptic Problems*, SIAM Journal on Scientific Computing **38** (2016), no. 5, A3220–A3246.

[20] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides, *Design Patterns: elements of reusable object-oriented software*, Pearson Education, 1995.

[21] David Gilbarg and Neil S. Trudinger, *Elliptic partial differential equations of second order*, Grundlehren der mathematischen Wissenschaften, Springer-Verlag, Berlin, New York, 1983, Cataloging based on CIP information.

[22] P. Goetgheluck, *Computing binomial coefficients*, Am. Math. Monthly **94** (1987), no. 4, 360–365.

[23] Gaël Guennebaud, Benoît Jacob, et al., *Eigen v3*, `http://eigen.tuxfamily.org`, 2010.

[24] ———, *Eigen v3: Solving Sparse Linear Systems*, `https://eigen.tuxfamily.org/dox-devel/group__TopicSparseSystems.html`, 2010.

[25] W. Gui and I. Babuška, *The h, p, and h-p versions of the finite element method in 1 dimension. II. The error analysis of the h- and h-p versions*, Numerische Mathematik **49** (1986), no. 6, 613–657.

[26] ———, *The h, p, and h-p versions of the finite element method in 1 dimension. III. The adaptive h-p version*, Numerische Mathematik **49** (1986), no. 6, 659–683.

[27] B. Guo and I. Babuška, *The h-p version of the finite element method*, Computational Mechanics **1** (1986), no. 3, 203–220.

[28] Roland Hagen, Steffen Roch, and Bernd Silbermann, *C^*-algebras and numerical analysis*, Marcel Dekker, 2001.

[29] Google Inc., *Google Test: A C++ Testing Framework*, `https://github.com/google/googletest`.

[30] Robert C. Kirby, *Low-Complexity Finite Element Algorithms for the de Rham Complex on Simplices*, SIAM Journal on Scientific Computing **36** (2014), no. 2, A846–A868.

[31] ———, *Efficient discontinuous Galerkin finite element methods via Bernstein polynomials*, (2015).

[32] M.J. Lai and L.L. Schumaker, *Spline functions on triangulations*, Encyclopedia of Mathematics an, no. v. 13, Cambridge University Press, 2007.

[33] Jens Markus Melenk and Barbara I Wohlmuth, *On residual-based a posteriori error estimation in hp-FEM*, Advances in Computational Mathematics **15** (2001), no. 1-4, 311–331.

[34] William F. Mitchell, *A comparison of adaptive refinement techniques for elliptic problems*, ACM Transactions on Mathematical Software (TOMS) **15** (1989), no. 4, 326–347.

[35] _____, *How High a Degree is High Enough for High Order Finite Elements?*, Procedia Computer Science **51** (2015), 246–255.

[36] Nicholas Nethercote and Julian Seward, *Valgrind: A program supervision framework*, In Third Workshop on Runtime Verification (RV'03, 2003.

[37] R.H. Nochetto, K.G. Siebert, and A. Veeser, *Theory of adaptive finite element methods: an introduction*, Multiscale, nonlinear and adaptive approximation, Springer, 2009, pp. 409–542.

[38] Ricardo H. Nochetto and A. Veeser, *Primer of adaptive finite element methods*, Multiscale and adaptivity: modeling, numerics and applications, Springer, 2011, pp. 125–225.

[39] Yixuan Qiu, *Spectra: C++ Library For Large Scale Eigenvalue Problems*, `https://spectralib.org`, 2010.

[40] L. Ridgway Scott and S. Zhang, *Finite element interpolation of nonsmooth functions satisfying boundary conditions*, Mathematics of Computation **54** (1990), no. 190, 483–483.

[41] Richard Stallman et al., *GDB: The GNU Project Debugger*, `https://www.gnu.org/software/gdb/`, 1986.

[42] Rob Stevenson, *Some notes with adaptive Finite Elements*, `https://staff.fnwi.uva.nl/r.p.stevenson/notes1.pdf`.

[43] _____, *Optimality of a standard adaptive finite element method*, Foundations of Computational Mathematics **7** (2007), no. 2, 245–269.

[44] _____, *The completion of locally refined simplicial partitions created by bisection*, Mathematics of Computation **77** (2008), no. 261, 227–241.

[45] B.A. Szabó and I. Babuška, *Finite element analysis*, A Wiley-Interscience publication, Wiley, 1991.

[46] Raymond van Venetië, *Equilibrated flux estimator for the adaptive finite element method*, MSc Thesis, University of Amsterdam (2016).

[47] Jan Westerdiep, *On h- and hp-type near-optimal tree generation and piecewise polynomial approximation in 1 and 2 dimensions*, BSc Thesis, University of Amsterdam (2014).

[48] Steven Wolfram, *Wolfram Mathematica*, `http://www.wolfram.com/mathematica/`.

[49] J. Xin, K. Pinchedez, and J.E. Flaherty, *Implementation of hierarchical bases in FEMLAB for simplicial elements*, ACM Trans. on Math. Soft. **31** (2005), no. 2, 187–200.