2008-01-01

Adaptive hp-FEM for Elliptic Problems in 3D on Irregular Meshes

David Andrs
University of Texas at El Paso, dandrs@miners.utep.edu

ADAPTIVE hp-FEM FOR ELLIPTIC PROBLEMS IN 3D ON IRREGULAR MESHES

DAVID ANDRS

Department of Mathematical Sciences

APPROVED:

.....
Pavel Šolín, Ph.D., Chair

.....
Granville E. Sewell, Ph.D.

.....
Piotr Wojciechowski, Ph.D.

.....
Vinod Kumar, Ph.D.

.....
Patricia D. Witherspoon, Ph.D.
Dean of the Graduate School

ADAPTIVE hp-FEM FOR ELLIPTIC PROBLEMS IN 3D ON IRREGULAR MESHES

by

DAVID ANDRS

THESIS

Presented to the Faculty of the Graduate School of
The University of Texas at El Paso
in Partial Fulfillment
of the Requirements
for the Degree of

MASTER OF SCIENCE

Department of Mathematical Sciences
THE UNIVERSITY OF TEXAS AT EL PASO
December 2008

Acknowledgement

First of all, I would like to thank my parents, Helena and Eduard Andrš, for their support during my whole young and reckless life.

Then, I would like to thank to Pavel Šolín, my adviser, for his leadership, advises and being great teacher. It was him who introduced me into the world of FEM.

Last, but not least, I would like to thank Lenka Dubcová, Jakub Červený, Pavel Kůs and Miroslav Šimko for creating a true research gang and participating on such amazing work like hp-FEM.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

The finite element method (FEM) is a numerical method for solving partial differential equations (PDE), which can describe a lot of macroscopic physical processes such as heat transfer, elasticity, electrodynamics, electrostatics, etc.

The hp-FEM is more general version of FEM based on piecewise-polynomial approximations that uses elements of variable size (h) and polynomial degree (p), which allows to achieve high accuracy and computational efficiency. The ultimate goal in hp-FEM is automatic hp-adaptivity, which by applying element refinements (h) and changing the polynomial degrees on elements (p) is trying to adapt the mesh to best approximate the solution.

The complexity of implementing hp-FEM, especially in 3D, is one of the reasons why it did not find its way into the commercial softwares. Another reason is that hp-FEM is not yet thoroughly explored–for example the choice of the shape functions with good conditioning properties, etc.

1.1 Second-order Elliptic PDEs

Let Ω be an open connected set in R^d, $d = 2, 3$. A fairly general form of a linear second-order partial differential equation (PDE) in n independent variables $z = (z_1, z_2, \ldots, z_n)^T$ is

$$-\sum_{i,j=1}^{n} \frac{\partial}{\partial z_i}\left(a_{ij}\frac{\partial u}{\partial z_j}\right) + \sum_{i=1}^{n}\left(\frac{\partial}{\partial z_i}(b_i u) + c_i\frac{\partial u}{\partial z_i}\right) + a_0 u = f \tag{1.1}$$

where $a_{ij} = a_{ij}(z), b_i = b_i(z), c_i = c_i(z), a_0 = a_0(z)$ and $f = f(z)$. For all derivatives to exist in the classical sense, the solution and the coefficients have to satisfy the following regularity requirements: $u \in C^2(\bar{\Omega}), a_{ij} \in C^1(\bar{\Omega}), b_i \in C^1(\bar{\Omega}), c_i \in C^1(\bar{\Omega}), a_0 \in C(\bar{\Omega}), f \in C(\bar{\Omega})$.

Definition 1.1 (Ellipticity) Consider a symmetric coefficient matrix $A(z) = \{a_{ij}\}_{i,j=1}^{n}$ corresponding to a second-order PDE of the form 1.1. The equation is said to be *elliptic* at a point $z \in R^n$ if $A(z)$ is positive definite.

1.2 Finite elements

Definition 1.2 (Finite element) *Finite element* in the sense of Ciarlet [2] is a triad $\mathcal{K} = (K, P, \Sigma)$, where

- K is a domain in $\mathbb{R}^d$ – we will limit ourselves to bricks ($d = 3$).
- P is a space of polynomials on K of dimension $\dim(P) = N_P$.
- $\Sigma = \{L_1, L_2, \ldots, L_{N_P}\}$ is a set of linear forms

$$L_i : P \to \mathbb{R}, \quad i = 1, 2, \ldots, N_P. \tag{1.2}$$

The elements of Σ are called *degrees of freedom* (and often abbreviated as DOF).

1.2.1 Function space H^1

Let $\Omega \subset \mathbb{R}^d$ be a bounded domain with Lipschitz-continuous boundary, d being the spatial dimension. The scalar Hilbert space of functions

$$H^1 = \{u \in L^2(\Omega); \partial u/\partial x_i \in L^2(\Omega),\ 1 \leq i \leq d\} \tag{1.3}$$

is the basic and most commonly used Sobolev space. Recall that the partial derivatives in (1.3) are understood in the sense of distributions.

1.2.2 Unisolvency of finite elements

Definition 1.3 introduces *unisolvency* as another expression for *compatibility* of the set of degrees of freedom Σ with the polynomial space P.

Definition 1.3 (Unisolvency of finite elements) The finite element $\mathcal{K} = (K, P, \Sigma)$ is said to be *unisolvent* if for every function $g \in P$ it holds

$$L_1(g) = L_2(g) = \ldots = L_{N_P}(g) = 0 \ \Rightarrow \ g = 0. \tag{1.4}$$

In other words, every vector of numbers

$$\mathbf{L}(g) = (L_1(g), L_2(g), \ldots, L_{N_P}(g))^T \in \mathbb{R}^{N_P}$$

uniquely identifies a polynomial g in the space P.

Definition 1.4 together with Theorem 1.1 offer a useful characterization of unisolvency of finite elements.

Definition 1.4 (δ-property) Let $\mathcal{K} = (K, P, \Sigma)$, $\dim(P) = N_P$, be a finite element. We say that a set of functions $\mathcal{B} = \{\theta_1, \theta_2, \ldots, \theta_{N_P}\} \subset P$ has the *δ-property* if

$$L_i(\theta_j) = \delta_{ij} \text{ for all } 1 \leq i, j \leq N_P. \tag{1.5}$$

Here δ_{ij} is the standard Kronecker delta, $\delta_{ij} = 1$ if $i = j$ and $\delta_{ij} = 0$ otherwise.

Theorem 1.1 (Characterization of unisolvency) Consider a finite element $\mathcal{K} = (K, P, \Sigma)$, $\dim(P) = N_P$. The finite element $\mathcal{K}$ is unisolvent if and only if there exists a unique basis $\mathcal{B} = \{\theta_1, \theta_2, \ldots, \theta_{N_P}\} \subset P$ satisfying the δ-property.

Proof to this theorem can be found in [1] on page 3.

Remark 1.1 (Nodal, hierarchic and dual basis) The expression *node* has in the finite element analysis several different meanings. To begin with, by *nodal* some authors denote the unique basis $\mathcal{B} \subset P$ that satisfies the δ-property (1.5). We will work with *nodal* and *hierarchic* bases of the space P which both satisfy the δ-property. Definitions will be given when appropriate. Existence and uniqueness of a basis $\mathcal{B} \subset P$ satisfying the δ-property is equivalent to the condition that the linear forms $L_1, \ldots, L_{N_P}$ form a *dual basis* to $\mathcal{B}$ in the space P' of linear forms over P.

Remark 1.2 (Selection of finite elements) The choice of a finite element (or their combination) depends upon the problem solved and upon the expectations that we put into the finite element scheme. It has a crucial effect on the behavior of the finite element scheme.

1.2.3 Finite element mesh

We assume that the bounded domain Ω with a Lipschitz-continuous boundary, where the underlying problem is investigated, is approximated by a computational domain Ω_h whose boundary is piecewise-polynomial.

Definition 1.5 (Finite element mesh) *Finite element mesh* $\mathcal{T}_{h,p} = \{K_1, K_2, \ldots, K_M\}$ over a domain $\Omega_h \subset \mathbb{R}^d$ with a piecewise-polynomial boundary is a geometrical division of Ω_h into a finite number of nonoverlapping (curved) open polygonal cells K_i such that

$$\Omega_h = \bigcup_{i=1}^{M} \overline{K}_i. \tag{1.6}$$

Each cell K_i, $1 \leq i \leq M$ is equipped with a polynomial order $1 \leq p(K_i) = p_i$.

Definition 1.6 (Hybrid mesh) If various types of cells are combined, the mesh is called *hybrid*.

Definition 1.7 (Regular mesh) The mesh is called *regular* if for any two elements K_i and K_j, $i \neq j$ only one of the following alternatives holds:

- $\overline{K}_i \cup \overline{K}_j$ is empty,
- $\overline{K}_i \cup \overline{K}_j$ is a single common vertex,
- $\overline{K}_i \cup \overline{K}_j$ is a single (whole) common edge,
- $\overline{K}_i \cup \overline{K}_j$ is a single (whole) common face.

By assuming that the mesh is regular we avoid *hanging nodes*, the existence of which substantially complicates the discretization procedure. We will use the word *node* as an abstraction for vertices, edges, faces and element interiors (the reasons for this soon become clear). Basically, hanging nodes can be grid vertices which lie in the interior of an edge or face of another cell, grid edges which lie in the interior of an edge or of a face of another cell, and grid faces which lie in the interior of a face of another cell. The constrained approximation technique in 2D and 3D will be discussed in more detail in Section 2.4. Various constellations involving hanging nodes are illustrated in Figure 1.1: A) One hanging vertex and 2 edges in a 2D mesh, B) 5 hanging vertices, 12 edges and 4 faces in a 3D mesh, C) One hanging edge and 2 faces in a 3D hybrid mesh. In the next text, we will consider only hexahedra.

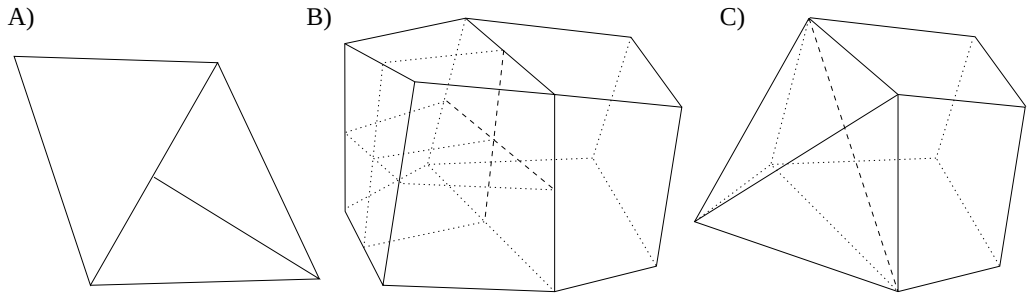


Figure 1.1: Examples of hanging nodes.

1.3 Reference Domains and Reference Maps

For piecewise-linear approximation, degrees of freedom are usually associated with the solution values at the grid vertices (here the nodal and hierarchic approaches coincide), and the variational formulation can be evaluated directly in the grid.

The situation changes dramatically with higher-order finite elements since they use a large amount of overlapping information, whose efficient management requires more structure to be imposed. For example, while first-order numerical quadrature can be implemented using coordinates of grid vertices only, higher-order quadrature schemes require many integration points per element (the actual amount depends on the order of accuracy, and in 3D it can easily achieve several hundreds). Both storing these values (d spatial coordinates and a weight per point) in all elements as well as reconstructing them periodically from a reference configuration would be extremely inefficient. The situation is analogous for higher-order basis functions.

Therefore, for higher-order finite element discretizations the mesh cells $K_i \in \mathcal{T}_{h,p}$ are mapped onto a *reference domain* $\hat{K}$ by means of smooth bijective *reference maps*

$$\mathbf{x}_{K_i} : \hat{K} \to K_i. \tag{1.7}$$

The maps $\mathbf{x}_K$, together with polynomial spaces on the reference domain $\hat{K}$, will be used for the definition of the space of functions $V_{h,p}(\Omega_h)$ where the finite element solution will be sought. An example of a reference map in the quadrilateral case in 2D is illustrated in Figure 1.2.

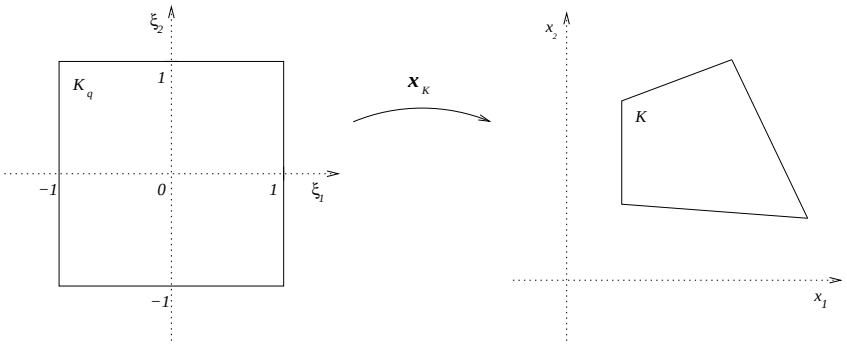


Figure 1.2: Reference map for a quadrilateral element.

The design of reference maps for all standard types of reference domains will be discussed in detail in section 2.3. There we also show how one uses them to transfer the integrals over elements K_i from the variational formulation to the reference domains. The higher-order finite element discretization is performed almost exclusively on the reference domains.

1.4 Finite Element Discretization

Consider a bounded domain $\Omega \subset \mathbb{R}^d$ with Lipschitz-continuous boundary, a partial differential equation (PDE) to be solved, and a set of conventional boundary conditions. Multiplying the PDE with a test function v from a suitable function space V, integrating over the domain Ω, applying the Green's theorem and incorporating the boundary conditions, one obtains a variational formulation

$$L(u^* + \bar{u}, v) = f(v) \text{ for all } v \in V. \tag{1.8}$$

Both the forms L and f are assumed linear in v. If nonhomogeneous Dirichlet conditions are present, the solution $u = u^* + \bar{u}$ is sought in an affine function space which is different from V (functions satisfying nonhomogeneous Dirichlet boundary conditions obviously cannot constitute any linear function space). The *lift function* u^* is chosen to satisfy nonhomogeneous Dirichlet conditions. Only the unknown component $\bar{u}$ satisfying *homogeneous* Dirichlet boundary conditions is sought. All functions $v \in V$ vanish on any Dirichlet part of the boundary $\partial\Omega$.

Neumann and Newton (Robin) boundary conditions are enforced by substituting them directly into boundary integrals in (1.8) over the corresponding part of the boundary $\partial\Omega$. See

any basic finite element textbook for more details.

1.4.1 Approximation of weak forms and discretization

The finite element discretization of (1.8) is done in the following standard steps:

Step 1: Approximate the domain Ω with another domain Ω_h which is more convenient for meshing and computation.

Step 2: Cover the domain Ω_h with a finite element mesh $\mathcal{T}_{h,p}$. Choose appropriate reference domains for all geometrical types of elements and for each $K \in \mathcal{T}_{h,p}$ construct a smooth bijective reference map $\mathbf{x}_K$.

Step 3: Approximate the space V, using appropriate polynomial spaces on the reference domains and the reference maps, by a suitable subspace $V_{h,p} = \mathrm{span}(v_1, v_2, ..., v_N)$.

Remark 1.3 (Variational crimes) Let us remark that in reality usually $V_{h,p} \not\subset V$ since the domains Ω and Ω_h differ, and moreover often even $\Omega_h \not\subset \Omega$. Hence this step is sometimes classified as a *variational crime* in the FE community.

Step 4: Approximate the form L by another form $L_{h,p}$, replacing the exact integration over Ω and $\partial\Omega$ by numerical integration over Ω_h and $\partial\Omega_h$. Notice that boundary conditions are shifted from $\partial\Omega$ to $\partial\Omega_h$ in this step.

Step 5: Approximate the linear form f by another linear form $f_{h,p}$ in the same way as in Step 4.

Step 6: We arrive at a new, approximate variational formulation: The solution $u_{h,p}$ is sought in the form $u_{h,p} = u^*_{h,p} + \bar{u}_{h,p}$, $\bar{u}_{h,p} \in V_{h,p}$, satisfying

$$L_{h,p}(u^*_{h,p} + \bar{u}_{h,p}, v_{h,p}) = f_{h,p}(v_{h,p}), \tag{1.9}$$

for all $v_{h,p} \in V_{h,p}$. The function $u^*_{h,p}$ is a suitable piecewise polynomial (usually a simple piecewise-linear) approximation of the lift function u^* from (1.8).

Step 7: Express the function $\bar{u}_{h,p}$ as a linear combination of the basis functions v_i of the

space $V_{h,p}$ with unknown coefficients y_i,

$$u_{h,p}(\mathbf{x}) = u^*_{h,p}(\mathbf{x}) + \bar{u}_{h,p}(\mathbf{x}) = u^*_{h,p}(\mathbf{x}) + \sum_{j=1}^{N} y_j v_j(\mathbf{x}). \tag{1.10}$$

Step 8: Insert the construction (1.10) into the approximate weak form (1.9) and select $v_{h,p} := v_i$, $i = 1, 2, \ldots, N$. This turns (1.9) into a system of algebraic equations

$$L_{h,p}\left(u^*_{h,p} + \sum_{j=1}^{N} y_j v_j, v_i\right) = f_{h,p}(v_i), \quad i = 1, 2, ..., N. \tag{1.11}$$

If the form $L_{h,p}$ is bilinear, (1.11) represents a system of linear algebraic equations which can be written in a matrix form $\mathbf{SY} = \mathbf{F}$,

$$\sum_{j=1}^{N} \underbrace{L_{h,p}\left(v_j, v_i\right)}_{\mathbf{S}_{ij}} \underbrace{y_j}_{\mathbf{Y}_j} = \underbrace{f_{h,p}(v_i) - L_{h,p}\left(u^*_{h,p}\right)}_{\mathbf{F}_i}, i = 1, 2, ..., N.$$

Otherwise the algebraic system (1.11) is nonlinear.

Step 9: Solve the system (1.11) for the unknown coefficients $\mathbf{Y}$ of $\bar{u}_{h,p}$ with a suitable numerical scheme. Retrieve the approximate solution $u_{h,p}$ using (1.10).

1.5 Orthogonal Polynomials

Orthogonal polynomials find applications in diverse fields of mathematics, both for theoretical and numerical issues. In our case they will play an essential role in the design of optimal higher-order shape functions. The class of Jacobi polynomials,

$$P_{n,\alpha,\beta}(x) = \frac{(-1)^n}{2^n n!}(1-x)^{-\alpha}(1+x)^{-\beta}\frac{\mathrm{d}^n}{\mathrm{d}x^n}\left[(1-x)^{\alpha+n}(1+x)^{\beta+n}\right], \tag{1.12}$$

holds the prominent position among orthogonal polynomials.

1.5.1 Legendre polynomials

Legendre polynomials have a special importance among the descendants of the Jacobi polynomials $P_{n,\alpha,\beta}$, they are defined as

$$L_n(x) = P_{n,0,0}(x).$$

and form an orthonormal basis of the space $L^2(I)$. Originally, they were constructed by means of the Gram-Schmidt orthogonalization process, and later many useful properties of these polynomials were found. For all of them let us mention, e.g., that their roots are identical with integration points for higher-order Gauss quadrature rules in one spatial dimension.

There are several ways to define them, among which probably the most useful for the implementation of higher-order shape functions is the recurrent definition

$$\begin{aligned} L_0(x) &= 1, \\ L_1(x) &= x, \\ L_k(x) &= \frac{2k-1}{k} x L_{k-1}(x) - \frac{k-1}{k} L_{k-2}(x), \quad k = 2, 3, \ldots, \end{aligned} \tag{1.13}$$

Their orthogonality is exactly specified by

$$\int_{-1}^{1} L_k(x) L_m(x) \mathrm{d}x = \begin{cases} \frac{2}{2k+1} & \text{for } k = m, \\ 0 & \text{otherwise.} \end{cases} \tag{1.14}$$

It is not difficult to obtain explicit formulae for Legendre and also other sets of orthogonal polynomials up to very high orders using standard mathematical software. Let us list a few Legendre polynomials as a reference for computer implementation.

$$\begin{aligned} L_0(x) &= 1, \\ L_1(x) &= x, \\ L_2(x) &= \frac{3}{2}x^2 - \frac{1}{2}, \\ L_3(x) &= \frac{1}{2}x(5x^2 - 3), \\ L_4(x) &= \frac{1}{8}(35x^4 - 30x^2 + 3), \\ L_5(x) &= \frac{1}{8}x(63x^4 - 70x^2 + 15), \end{aligned} \tag{1.15}$$

The functions $L_0, L_1, \ldots, L_9$ are illustrated in Figure 1.3.

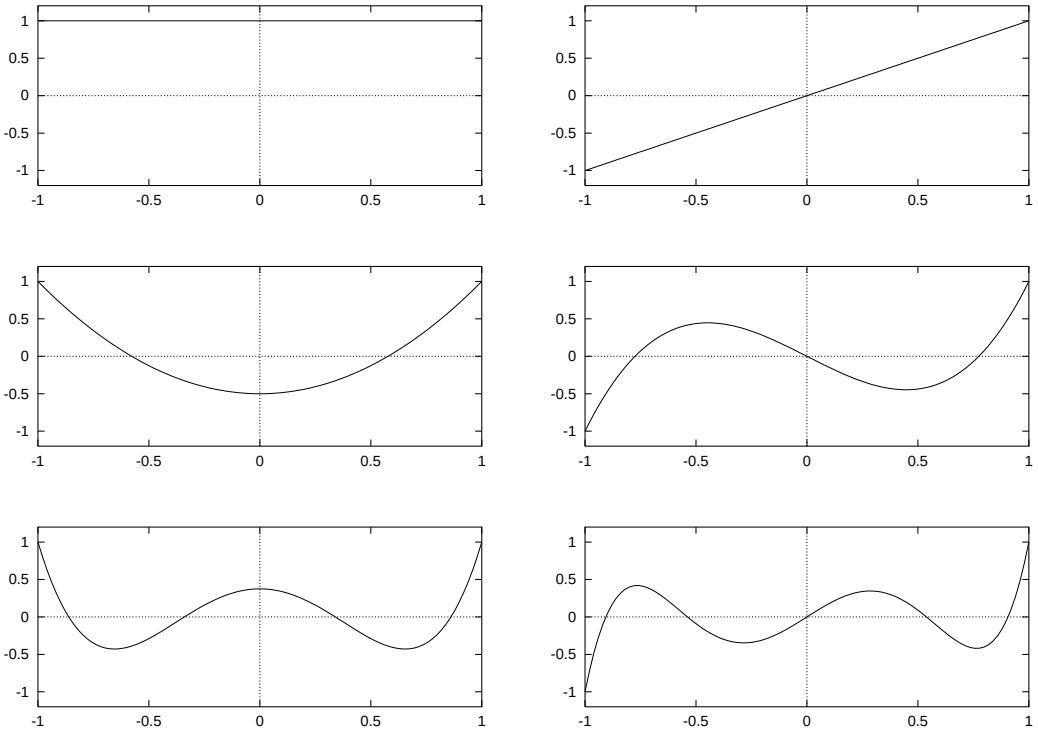


Figure 1.3: Legendre shape functions $L_0, L_1, L_2, L_3, L_4, L_5$

1.5.2 Lobatto shape functions

Let us define functions

$$\begin{aligned} l_0(x) &= \frac{1-x}{2}, \quad l_1(x) = \frac{x+1}{2}, \\ l_k(x) &= \frac{1}{\|L_{k-1}\|_2} \int_{-1}^{x} L_{k-1}(\xi)\, \mathrm{d}\xi, \quad 2 \le k, \end{aligned} \tag{1.16}$$

where $\|L_{k-1}\|_2 = \sqrt{2/(2k-1)}$ from (1.14). Obviously $l_k(-1) = 0$, $k = 2, 3, \ldots$. It follows from the orthogonality of higher-order Legendre polynomials L_k to $L_0 \equiv 1$,

$$\int_{-1}^{1} L_k(x)\, \mathrm{d}x = 0, \quad k \ge 1, \tag{1.17}$$

that also $l_k(1) = 0$, $k = 2, 3, \ldots$. The *Lobatto shape functions* $l_0, l_1, l_2, \ldots, l_p$ form a complete basis of the space $P_p(-1, 1)$ of polynomials of the order of at most p in the interval $(-1, 1)$.

Let us list some of them for reference:

$$
\begin{aligned}
l_2(x) &= \frac{1}{2}\sqrt{\frac{3}{2}}(x^2-1), \\
l_3(x) &= \frac{1}{2}\sqrt{\frac{5}{2}}(x^2-1)x, \\
l_4(x) &= \frac{1}{8}\sqrt{\frac{7}{2}}(x^2-1)(5x^2-1), \\
l_5(x) &= \frac{1}{8}\sqrt{\frac{9}{2}}(x^2-1)(7x^2-3)x,
\end{aligned}
\tag{1.18}
$$

The Lobatto shape functions will play an essential role in the design of hierarchic shape functions in Chapter 2. Some of them are illustrated in Figures 1.4 (notice the different scales).

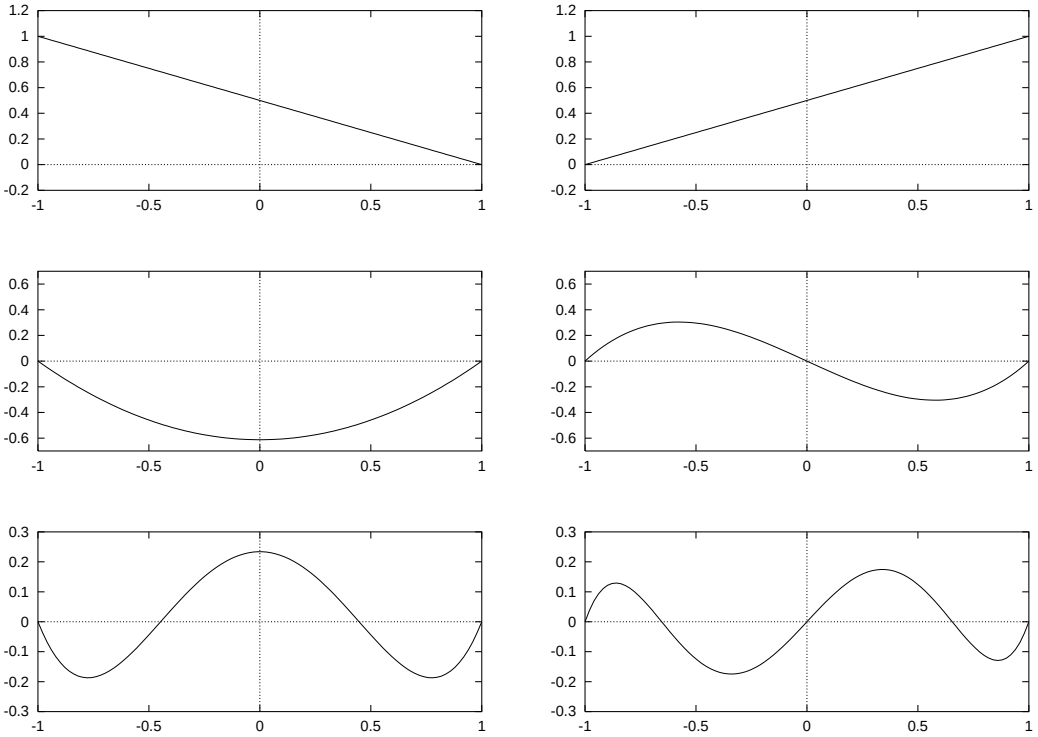


Figure 1.4: Lobatto shape functions $l_0, l_1, l_2, l_3, l_4, l_5$

Chapter 2

Higher-Order Finite Element Method

The construction of finite elements of arbitrary order for H^1-conforming approximations is relatively well known, and various options of hierarchic shape functions for all commonly used reference domains can be found in several textbooks (see, e.g., [3],[4],[5]) and numerous articles ([6],[8],[9],[10],[7],[11],[12],[13],[14],[15],[16],[17] and others). However, the question of the *optimal design* of shape functions is extremely difficult (already the formulation of optimality criteria is not at all trivial), and very few results stating any kind of optimality are available. The conditioning of the master element stiffness and/or mass matrix is a good indicator of quality of the shape functions.

2.1 Brick master element $\mathcal{K}_B^1$

We will be dealing with only one master element, which is master brick $\mathcal{K}_B^1$, that will be associated with the reference brick domain

$$K_B = \{\xi \in \mathbb{R}^3;\ -1 < \xi_1, \xi_2, \xi_3 < 1\}, \tag{2.1}$$

depicted in Figure 2.1.

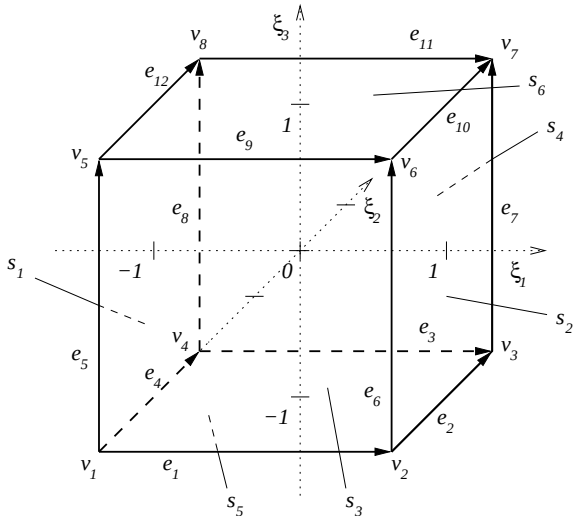


Figure 2.1: The reference brick K_B

Remark 2.1 This geometry is convenient for our purposes since it respects the interval of definition of the Jacobi polynomials in all three spatial directions. The one-dimensional affine coordinates appropriate for the geometry (2.1) have the form

$$\begin{aligned}\lambda_{1,B}(\xi_1,\xi_2,\xi_3)=\frac{\xi_1+1}{2},\lambda_{2,B}(\xi_1,\xi_2,\xi_3)=\frac{1-\xi_1}{2},\lambda_{3,B}(\xi_1,\xi_2,\xi_3)=\frac{\xi_2+1}{2},\\ \lambda_{4,B}(\xi_1,\xi_2,\xi_3)=\frac{1-\xi_2}{2},\lambda_{5,B}(\xi_1,\xi_2,\xi_3)=\frac{\xi_3+1}{2},\lambda_{6,B}(\xi_1,\xi_2,\xi_3)=\frac{1-\xi_3}{2}.\end{aligned} \tag{2.2}$$

To allow for anisotropic p-refinement of brick elements, we consider local directional orders of approximation $p^{b,1},p^{b,2},p^{b,3}$ in element interior (in directions ξ_1,ξ_2 and ξ_3, respectively).

In 3D there is the added possibility of anisotropic p-refinement of *faces*, for which we need to assign two local directional orders of approximation $p^{s_i,1},p^{s_i,2}$ to each face s_i, $i=1,\ldots,6$. These directional orders are associated with a *local two-dimensional system of coordinates* on each face, which matches an appropriate pair of global coordinate axes in lexicographic order. With this choice, based on [18], local coordinate axes on faces have the same orientation as the corresponding global ones, which simplifies sign-related issues in the formulae for face functions. Edges will be equipped as usual with local orders of approximation $p^{e_1},\ldots,p^{e_{12}}$, and their orientation will be used for the construction of edge functions only.

Remark 2.2 (Minimum rules in 3D) In 3D the minimum rule limits the local orders of approximation on both edges and faces. Local (directional) orders on mesh faces are not allowed to exceed the minimum of the (appropriate directional) orders of approximation associated with the interior of the adjacent elements. Local orders of approximation on mesh edges are limited by the minimum of all (appropriate directional) orders corresponding to faces sharing that edge.

The local orders $p^{b,1},\ldots,p^{b,3}$, $p^{s_i,1},p^{s_i,2}$, $i=1,\ldots,6$, and $p^{e_1},\ldots,p^{e_{12}}$ suggest that a finite element of the form $\mathcal{K}_B^1=(K_B,W_B,\Sigma_B^1)$ will be equipped with polynomial space

$$W_B=\left\{w\in\mathcal{Q}_{p^{b,1},p^{b,2},p^{b,3}};\, w\,|_{s_i}\in\mathcal{Q}_{p^{s_i,1},p^{s_i,2}},\, w\,|_{e_j}\in\mathcal{P}_{p^{e_j}}(e_j), i=1,\ldots,6,\ j=1,\ldots,12\right\}. \tag{2.3}$$

Here,

$$\mathcal{Q}_{p,q,r}=\text{span}\left\{\xi_1^i\xi_2^j\xi_2^k;\ (\xi_1,\xi_2,\xi_3)\in K_B, i=0,\ldots,p,\ j=0,\ldots,q,\ k=0,\ldots,r\right\}. \tag{2.4}$$

The set of degrees of freedom Σ_B^1 will be uniquely identified by a concrete choice of basis in W_B. H^1-conformity requirements, constraining function values at *vertices*, on *edges* and on *faces*, dictate that the hierarchic basis of space W_B will have to comprise *vertex*, *edge*, *face* and *bubble* functions.

Vertex functions $\varphi_B^{v_j}$, $j = 1, 2, \ldots, 8$, are associated with element vertices, and they provide the complete basis of a space W_B for lowest-order approximation. Recall functions $l_0, l_1, \ldots$ from (1.5.2). Vertex functions will be chosen in a conventional way, i.e., trilinear in the form

$$\varphi_B^{v_j} = l_{d_1}(\xi_1) l_{d_2}(\xi_2) l_{d_3}(\xi_3), \tag{2.5}$$

where

$$\mathbf{d} = (d_1, d_2, d_3)$$

is a vector index, whose components are related to axial directions ξ_1, ξ_2 and ξ_3, respectively. It is defined as follows: consider edges $e_{j_1}, e_{j_2}, e_{j_3}$, containing the vertex v_j, and lying in axial directions ξ_1, ξ_2, ξ_3, respectively. We put $d_k = 0$ if v_j lies on the left of edge e_{j_k} (with respect to the axial direction ξ_k), and $d_k = 1$ otherwise. Notice that the vertex functions $\varphi_B^{v_j}$ are equal to one at the vertex v_j, and vanish at all seven remaining vertices. Their traces are linear on all edges. An example of the vertex function is illustrated in Figure 2.2.

Edge functions $\varphi_{k,B}^{e_j}$, $j = 1, \ldots, 12$, $k = 2, 3, \ldots, p^{e_j}$, will be designed in such a way that the traces of $\varphi_{k,B}^{e_j}$ to the edge e_j match the Lobatto shape functions $l_2, \ldots, l_{p^{e_j}}$ (representing a basis of polynomial space $\mathcal{P}_{p^{e_j},0}(e_j)$), and vanish on all remaining edges. Consider a polynomial order k, $2 \le k \le p^{e_j}$, and define index $\mathbf{d} = (d_1, d_2, d_3)$ as follows: Put $d_m = k$, where ξ_m is the axis parallel to e_j. The remaining two components are set to either zero or one, depending on whether the edge lies on the left or right side of the reference brick, with respect to the remaining two axial directions. An edge function $\varphi_{k,B}^{e_j}$ of order k is defined by

$$\varphi_{k,B}^{e_j} = l_{d_1}(\xi_1) l_{d_2}(\xi_2) l_{d_3}(\xi_3), \tag{2.6}$$

as illustrated in Figure 2.2.

Face functions $\varphi_{n_1,n_2,B}^{s_i}$, $2 \le n_1 \le p^{s_i,1}$, $2 \le n_2 \le p^{s_i,2}$, corresponding to a face s_i, $i = 1, \ldots, 6$, will be constructed to have a trace of directional polynomial orders n_1, n_2 on the face s_i (with respect to its local coordinate system specified above), and to vanish on the five remaining faces. Appropriate components of the index $\mathbf{d} = (d_1, d_2, d_3)$ now contain directional orders

n_1, n_2, and the remaining component is set to either zero or one, depending on whether the face s_i lies on the left or right side of the reference brick with respect to the remaining axial direction. We define

$$\varphi^{s_i}_{n_1,n_2,B} = l_{d_1}(\xi_1) l_{d_2}(\xi_2) l_{d_3}(\xi_3),$$

and illustrate the construction in Figure 2.2.

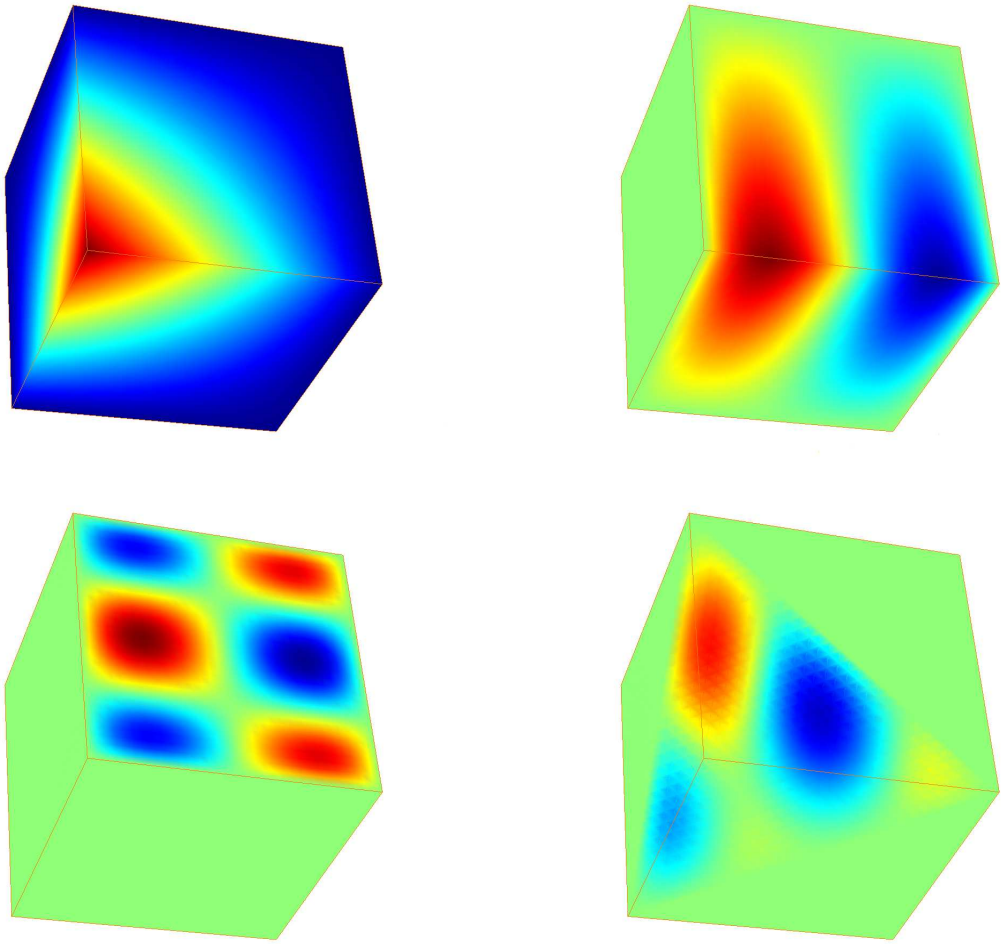


Figure 2.2: An example of the vertex, edge, face and bubble function

Remark 2.3 Notice that all face functions sharing the same face s_j are linearly independent, and obviously linearly independent of face functions corresponding to other faces. Moreover, all of the aforementioned face functions are linearly independent of edge and vertex functions.

Bubble functions are the last ones to be added into the hierarchic basis of the space W_B. They generate the space $\mathcal{Q}_{p^{b,1},p^{b,2},p^{b,3},0}$ of polynomials of directional orders at most $p^{b,j}$ in axial directions ξ_j, $j = 1, \ldots, 3$, that vanish everywhere on the boundary of the reference brick K_B,

$$\varphi^b_{n_1,n_2,n_3,B} = l_{d_1}(\xi_1) l_{d_2}(\xi_2) l_{d_3}(\xi_3), \quad 2 \le d_j \le p^{b,j},\ j = 1, \ldots, 3. \tag{2.7}$$

Numbers of hierarchic shape functions in the basis of the space W_B are presented in Table 2.1.

Table 2.1: Scalar hierarchic shape functions of $\mathcal{K}^1_B$.

Node type	**Polynomial order**	**Number of shape functions**	**Number of nodes**
Vertex	always	1	8
Edge	$2 \le p^{e_j}$	$p^{e_j} - 1$	12
Face	$2 \le p^{s_i,1}, p^{s_i,2}$	$(p^{s_i,1} - 1)(p^{s_i,2} - 1)$	6
Interior	$2 \le p^{b,1}, p^{b,2}, p^{b,3}$	$(p^{b,1} - 1)(p^{b,2} - 1)(p^{b,3} - 1)$	1

Remark 2.4 When the distribution of order of polynomial approximation in the finite element mesh is *uniform* ($p = p^{b,i} = p^{s_j,k} = p^{e_m}$ for all i, j, k, m), the above introduced basis of W_B reduces to a basis

$$l_{d_1}(\xi_1) l_{d_2}(\xi_2) l_{d_3}(\xi_3), \quad 0 \le d_1, d_2, d_3 \le p,$$

of the standard space $\mathcal{Q}_{p,p,p}$ of cardinality $\mathrm{card}(\mathcal{Q}_{p,p,p}) = (p+1)^3$.

2.2 Projection-based interpolation on reference domains

Projection-based interpolation on hierarchic elements is a nontrivial technique that forms an essential part of higher-order finite element methods. Recall from Section 1.2 that in contrast to nodal higher-order elements, the degrees of freedom $L_1, L_2, \ldots, L_{N_P}$ for hierarchic elements *are not defined outside of the local polynomial space* $P(K)$. One needs to combine the standard nodal (Lagrange) *interpolation* with *projection* on higher-order polynomial subspaces.

Given a sufficiently regular function $u \in V(\Omega_h)$, it is our aim to find an appropriate piecewise-polynomial interpolant $u_{h,p} \in V_{h,p}(\Omega_h)$, $V_{h,p} \subset V$. For every element $K \in \mathcal{T}_{h,p}$ with affine reference map $\mathbf{x}_K : \hat{K} \to K$ this is equivalent to the interpolation of the function $u \mid_K \circ \mathbf{x}_K$ *on the reference domain*. Therefore we will stay on the reference domain for a while.

Properties of projection-based interpolation operators

In order that the projection-based interpolation Π is algorithmically efficient, conforming and compatible with convergence theory, we request the following properties:

1. *Locality.* The projection-based interpolant Πu of a function u is constructed element-wise. Therefore we request that *within an element,* Π *uses function values of* u *from this element only.*
2. *Global conformity.* For every function $u \in V^0$, $\overline{V^0} = V$, the projection-based interpolant $u_{h,p} = \Pi u$ still lies in the space V.
3. *Optimality.* The interpolant $u_{h,p} \in V_{h,p}$ must have the minimum distance from the interpolated function $u \in V$ in an appropriate norm.

2.3 Construction of reference maps

Now we will define suitable parametrizations for edges and faces of (generally arbitrarily curved) elements $K \in \mathcal{T}_{h,p}$, and apply the transfinite interpolation technique in order to design reference maps $\mathbf{X}_K(\xi) : \hat{K} \to K$ (where $\hat{K}$ is an appropriate reference domain).

Moreover, the reference maps $\mathbf{X}_K(\xi)$ are nonpolynomial (when the edges or faces are parametrized by nonpolynomial functions). As long as they are smooth and one-to-one, this is not a problem and in principle one can use them in the finite element code anyway. However, usually one prefers to construct their *isoparametric* approximations $\mathbf{x}_K(\xi) \approx \mathbf{X}_K(\xi)$, which are polynomial maps defined in terms of master element shape functions and can be easily stored and handled in the computer code.

2.3.1 Mapping (curved) brick elements onto K_B

Let the edges $\tilde{e}_j$, $j = 1, \ldots, 12$, of a brick $K \in \mathcal{T}_{h,p}$ be parametrized by continuous curves $\mathbf{X}^{e_j}(\zeta) \subset \mathbb{R}^3, \zeta \in [-1, 1]$. As usual the parametrization of edges has to be compatible with the orientation of edges of the reference domain K_B (depicted in Figure 2.1). In other words,

$$\begin{aligned} \mathbf{X}^{e_1}(1) = \mathbf{X}^{e_2}(-1) &= \mathbf{X}^{e_6}(-1) = \mathbf{x}_2, \\ \mathbf{X}^{e_2}(1) = \mathbf{X}^{e_3}(1) &= \mathbf{X}^{e_7}(-1) = \mathbf{x}_3, \\ \mathbf{X}^{e_3}(-1) = \mathbf{X}^{e_4}(1) &= \mathbf{X}^{e_8}(-1) = \mathbf{x}_4, \\ \mathbf{X}^{e_4}(-1) = \mathbf{X}^{e_1}(-1) &= \mathbf{X}^{e_5}(-1) = \mathbf{x}_1 \end{aligned} \tag{2.8}$$

and so on. Here $\mathbf{x}_1, \ldots, \mathbf{x}_8$ are vertices of the physical element K, ordered compatibly with the vertices $v_1, \ldots, v_8$ of the reference domain K_B.

New in 3D are parametrizations $\mathbf{X}^{s_i}(\zeta_1, \zeta_2), \zeta \in [-1, 1]^2$ for the faces $\tilde{s}_i \subset \partial K$, $i = 1, \ldots, 6$. Recall that local coordinate axes ζ_1, ζ_2 attached to each face are oriented accordingly to the coordinate axes ξ_1, ξ_2, ξ_3, following their lexicographic order. An essential new issue in 3D is the compatibility of face parametrizations $\mathbf{X}^{s_i}(\zeta_1, \zeta_2)$ with parametrizations of edges. For example, for the face s_1 this translates into the compatibility conditions

$$\begin{aligned} \mathbf{X}^{s_1}(\zeta, -1) &= \mathbf{X}^{e_4}(\zeta), \quad \zeta \in [-1, 1], \\ \mathbf{X}^{s_1}(\zeta, 1) &= \mathbf{X}^{e_{12}}(\zeta), \quad \zeta \in [-1, 1], \\ \mathbf{X}^{s_1}(-1, \zeta) &= \mathbf{X}^{e_5}(\zeta), \quad \zeta \in [-1, 1], \\ \mathbf{X}^{s_1}(1, \zeta) &= \mathbf{X}^{e_8}(\zeta), \quad \zeta \in [-1, 1]. \end{aligned} \tag{2.9}$$

Notice that compatibility conditions (2.9) together with conditions (2.8) yield compatibility of parametrizations of faces with vertices:

$$\begin{aligned} \mathbf{X}^{s_1}(-1, -1) &= \mathbf{x}_1, \\ \mathbf{X}^{s_1}(1, -1) &= \mathbf{x}_4, \\ \mathbf{X}^{s_1}(1, 1) &= \mathbf{x}_8, \\ \mathbf{X}^{s_1}(-1, 1) &= \mathbf{x}_5, \\ &\vdots \end{aligned} \tag{2.10}$$

The transfinite interpolation scheme will be defined using a vertex, edge and face contribution,

$$\mathbf{X}_K(\xi) = \mathbf{X}_K^v(\xi) + \mathbf{X}_K^e(\xi) + \mathbf{X}_K^s(\xi), \quad \xi \in K_B. \tag{2.11}$$

The vertex part $\mathbf{X}_K^v(\xi)$ is defined by combining the physical mesh vertex coordinates $\mathbf{x}_i$ and scalar vertex functions (2.5),

$$\mathbf{X}_K^v(\xi) = \sum_{i=1}^{8} \mathbf{x}_i \varphi_B^{v_j}(\xi).$$

Consider a reference edge $e_j = v_A v_B$ and the parametrization $\mathbf{X}^{e_j}$ of the corresponding physical mesh edge $\tilde{e}_j$. Its bubble part

$$\mathbf{X}_0^{e_j}(\zeta) = \mathbf{X}^{e_j}(\zeta) - \mathbf{X}^{e_j}(-1)\frac{1-\zeta}{2} - \mathbf{X}^{e_j}(1)\frac{\zeta+1}{2}, \quad \zeta \in [-1,1],$$

is bilinearly blended,

$$\mathbf{X}_K^{e_j}(\xi) = \mathbf{X}_0^{e_j}(\lambda_B(\xi) - \lambda_A(\xi))\lambda_C(\xi)\lambda_D(\xi), \tag{2.12}$$

and used for the definition of the edge part

$$\mathbf{X}_K^e(\xi) = \sum_{j=1}^{12} \mathbf{X}_K^{e_j}(\xi)$$

of the transfinite interpolant $\mathbf{X}_K(\xi)$. For each edge e_j the affine coordinates in (2.12) are chosen so that λ_A, λ_B vanish on faces perpendicular to e_j and are ordered so that $\lambda_A(v_A) = \lambda_B(v_B) = 1$. The affine coordinates λ_C, λ_D vanish on the faces $s_C, s_D \subset \partial K_B$, which do not share any vertex with the edge e_j, respectively.

In the same way, for each face s_i we first construct the bubble part

$$\mathbf{X}_0^{s_i}(\zeta) = \mathbf{X}^{s_i}(\zeta) - \mathbf{X}_K^e\,|_{s_i}\,(\zeta) - \mathbf{X}_K^v\,|_{s_i}\,(\zeta),$$

of the parametrization $\mathbf{X}^{s_i}(\zeta)$, which entirely vanishes on the boundary of the face s_i. Functions $\mathbf{X}_0^{s_i}(\zeta)$ are further linearly blended into the element interior,

$$\mathbf{X}_K^{s_i}(\xi) = \mathbf{X}_0^{s_i}(\lambda_B(\xi) - \lambda_A(\xi), \lambda_D(\xi) - \lambda_C(\xi))\lambda_E(\xi), \tag{2.13}$$

and contribute to the face part

$$\mathbf{X}_K^s(\xi) = \sum_{i=1}^{6} \mathbf{X}_K^{s_i}(\xi)$$

of the transfinite interpolant $\mathbf{X}_K(\xi)$.

The affine coordinates in (2.13) are chosen taking into account the local coordinate system on the face s_i: λ_A, λ_B correspond to faces perpendicular to the local axis ζ_1 and $\lambda_A(e_A) = \lambda_B(e_B) = 1$ where the edges e_A, e_B correspond to $\zeta_1 = -1$ and $\zeta_1 = 1$ on the face s_i, respectively. Similarly λ_C, λ_D are chosen for the second local axial direction ζ_2. The affine coordinate λ_E vanishes on the element-opposite face s_E.

2.3.2 Inversion of reference maps

Inversion of the reference maps $\mathbf{x}_K(\xi) : \hat{K} \to K$, where K is a physical mesh element and $\hat{K}$ is the corresponding reference domain, is only required if we need to locate a geometrical point $\xi^* \in \hat{K}$, given its image

$$\mathbf{x}^* = \mathbf{x}_K(\xi^*) \in K \in \mathcal{T}_{h,p}. \tag{2.14}$$

This might be the case, for example, when the user asks the value of the approximate solution at some specific point $\mathbf{x}$ in the computational domain.

Affine case

If the Jacobi matrix $D\mathbf{x}_K/D\xi$ of the map $\mathbf{x}_K$ is constant (i.e., K is either a triangle or tetrahedron with linear edges and/or faces), we have

$$\frac{D\mathbf{x}_K}{D\xi}(v_1)(\xi^* - v_1) = \mathbf{x}^* - \mathbf{x}_K(v_1),$$

which yields

$$\xi^* = v_1 - \left(\frac{D\mathbf{x}_K}{D\xi}\right)^{-1}(v_1)(\mathbf{x}_K(v_1) - x^*).$$

Here v_1 is (for example the first) vertex of the reference domain K_t or K_T and $\mathbf{x}_K(v_1)$ is the corresponding vertex of the physical mesh element K.

2.4 Constrained approximation

The constrained approximation technique has long been used for first-order elements in various applications. To our knowledge, for higher-order elements it was first introduced by Demkowicz, Oden et al. in [14]. It is necessary for an efficient resolution of phenomena that require a high local concentration of degrees of freedom – those are typically boundary and internal layers, regions with steep gradients, singularities, etc. Attempts to resolve these features with regular meshes can lead to inefficient distribution of degrees of freedom and even can spoil the convergence of the finite element scheme.

Hence, the main idea is to employ *irregular meshes* (i.e., meshes with hanging nodes in the sense of Paragraph 1.2.3) in such a way that the approximation still satisfies global conformity

requirements – continuity of the approximation across element interfaces for H^1-conforming approximations. Perhaps the easiest way to understand the constrained approximation is to view it in terms of change of basis in a given polynomial space.

2.4.1 Continuous constrained approximation in 3D

While the mathematical issues related to the constrained approximation technique in 3D are at a similar level of complexity as in two spatial dimensions, its algorithmic aspects become significantly more pronounced.

Naturally, for continuous approximations our task is to handle hanging nodes in such a way that continuity of the approximation across element interfaces is preserved. Thus the entire action will take place on mesh edges and faces and it will involve the vertex, edge and face functions only. For hexahedral elements, there are basically two situations we may find it useful to consider:

1. two quadrilateral faces constrained by one quadrilateral face,
2. four quadrilateral faces constrained by one quadrilateral face,

We will discuss only Case 1 in more detail since Case 2 is very similar to Case 1.

Two quadrilateral faces constrained by another quadrilateral face

Let us consider the model situation depicted in Figure 2.3. Assume that the edges v_1v_2 and v_1v_4 specify the global orientation of the face s_1. The same orientation, by definition, is inherited by the small faces s_2 and s_3, respectively (or, more precisely, with the only change that the edge v_1v_4 is replaced by its sons v_1v_5 and v_5v_4). The face s_1 comes with two local directional orders of approximation $p^{s_1,1}$, corresponding to the first direction v_1v_2, and $p^{s_1,2}$, associated with the other direction v_1v_4. By definition, these directional polynomial orders are inherited by the small faces s_2 and s_3.

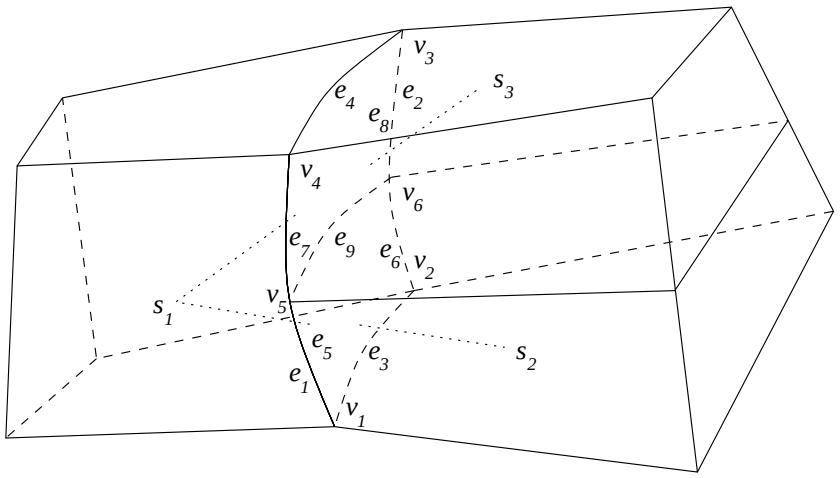


Figure 2.3: Constrained continuous approximation 3D.

The local orders of approximation on the edges $e_1, \ldots, e_4$ are denoted by $p^{e_1}, \ldots, p^{e_4}$, respectively. Each of these edges comes with a unique global orientation that is *independent of the orientation of the face* s_1. The edges e_5, e_7 and e_6, e_8 inherit local orders from the edges e_1 and e_2, respectively. Recall that the minimum rule guarantees that $p^{e_3} \leq p^{s_1,1}$, $p^{e_4} \leq p^{s_1,1}$. Hence, by definition, the edge e_9 is equipped with the directional order $p^{s_1,1}$ on the face s_1. In summary, we have

$$\begin{aligned} p^{s_2,1} = p^{s_3,1} &= p^{s_1,1} \\ p^{s_2,2} = p^{s_3,2} &= p^{s_1,2} \\ p^{e_5} = p^{e_7} &= p^{e_1}, \\ p^{e_6} = p^{e_8} &= p^{e_2}, \\ p^{e_9} &= p^{s_1,1}. \end{aligned} \tag{2.15}$$

Now one could strictly copy the 2D procedure, i.e., construct for each face s_2 and s_3 a transition matrix that converts the coefficients corresponding to DOF associated with the face s_1 to coefficients related to faces s_2 and s_3, respectively. However, due to the product structure of quadrilateral faces, these transition matrices would contain almost exclusively zeros. In other words, the number of constraining relations is dramatically less than $m_b m_s$ where m_b and m_s stand for the total number of DOF associated with the big (constraining) and small (constrained) face, respectively. Thus it is convenient to go deeper into the structure of the shape functions in order to avoid unnecessary extra algebraic equations.

Remark 2.5 Notice that the transition matrices are of the type $m_s \times m_b$ where generally

$m_s \neq m_b$. In our case, the inequality occurs for s_2 if $p^{s_1,1} \neq p^{e_4}$ and for s_3 if $p^{s_1,1} \neq p^{e_3}$.

More to the point, the coefficients $\alpha^{v_5}, \alpha_k^{e_5}, \alpha_k^{e_7}$, $2 \leq k \leq p^{e_1}$ corresponding to the vertex v_5 and edges e_5, e_7 are only constrained by the coefficients $\alpha^{v_1}, \alpha^{v_4}, \alpha_k^{e_1}$, $2 \leq k \leq p^{e_1}$ associated with the edge e_1. Therefore the transition matrices $M_L^{p^{e_1}}$, $M_R^{p^{e_1}}$ represent the algebraic relations between these coefficients. The global orientation of the edge e_1 can be either v_1v_4 or v_4v_1. Depending on this we relate the matrices $M_L^{p^{e_1}}$, $M_R^{p^{e_1}}$ to the edges e_5, e_7 in this or reverse order.

In the same way we proceed once more with the vertex v_6 and edges e_6, e_8 and e_2, expressing the short edge coefficients $\alpha^{v_6}, \alpha_k^{e_6}, \alpha_k^{e_8}$, $2 \leq k \leq p^{e_2}$ by means of the long edge coefficients $\alpha^{v_2}, \alpha^{v_3}, \alpha_k^{e_2}$, $2 \leq k \leq p^{e_2}$ by means of the transition matrices $M_L^{p^{e_2}}$, $M_R^{p^{e_2}}$.

The higher-order coefficients $\alpha_k^{e_9}$, $2 \leq k \leq p^{s_1,1}$ and $\alpha_{n_1,n_2}^{s_2}$, $\alpha_{n_1,n_2}^{s_3}$, $2 \leq n_1 \leq p^{s_1,1}$, $2 \leq n_2 \leq p^{s_1,2}$ are only constrained by the higher-order edge coefficients $\alpha_k^{e_3}$, $2 \leq k \leq p^{e_3}$ and $\alpha_l^{e_4}$, $2 \leq l \leq p^{e_3}$, and by the higher-order face coefficients $\alpha_{n_1,n_2}^{s_1}$, $2 \leq n_1 \leq p^{s_1,1}$, $2 \leq n_2 \leq p^{s_1,2}$. Without loss of generality let us assume the following compatibility between the edge and face parametrizations:

- The face s_1 is parametrized by a smooth bijective mapping from a master face $\hat{s}_q = [-1,1]^2$,
$$\mathbf{x}^{s_1} : \hat{s}_q \to \mathbb{R}^3.$$

- The edges e_3, e_4 and e_9 are parametrized by smooth bijective maps from a master edge $\hat{e} = [-1,1]$,
$$\mathbf{x}^{e_j} : \hat{e} \to \mathbb{R}^3, \quad j = 3,4,9,$$
such that
$$\begin{aligned} \mathbf{x}^{e_3}(o^{e_3}\zeta) &= \mathbf{x}^{s_1}(\zeta,-1) \quad \forall \zeta \in [-1,1], \\ \mathbf{x}^{e_4}(o^{e_4}\zeta) &= \mathbf{x}^{s_1}(\zeta,1) \quad \forall \zeta \in [-1,1], \\ \mathbf{x}^{e_9}(\zeta) &= \mathbf{x}^{s_1}(\zeta,0) \quad \forall \zeta \in [-1,1], \end{aligned} \tag{2.16}$$
where o^{e_3}, o^{e_4} are ± 1 orientation factors that compensate for the inconsistency between the orientations of the edges e_3, e_4 and the orientation of the face s_1. The edge e_9 is, by definition, oriented in harmony with the orientation of its father, the face s_1.

All operations are done on the quadrilateral reference face $\hat{s}_q$ as shown in Figure 2.4. This is a natural setting since the global orientation of the face s_1 depends on the global enumeration of grid vertices only.

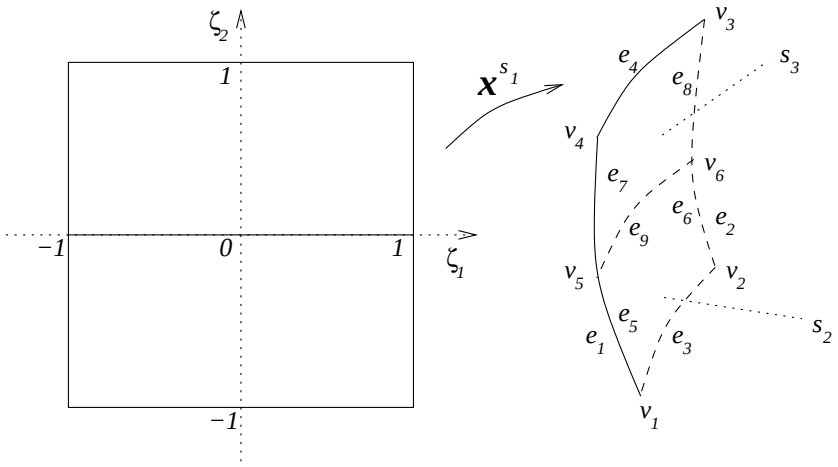


Figure 2.4: Quadrilateral reference face $\hat{s}_q = [-1,1]^2$, $s_1 = (\mathbf{x}^{s_1})(\hat{s}_q)$.

On the reference face $\hat{s}_q$, the higher-order part of the trace of the approximation reads

$$\underbrace{\sum_{k=2}^{p^{e_3}} o^{e_3}\alpha_k^{e_3} l_k(\zeta_1) l_0(\zeta_2)}_{\text{contribution of } e_3-\text{edge functions}} + \underbrace{\sum_{k=2}^{p^{e_4}} o^{e_4}\alpha_k^{e_4} l_k(\zeta_1) l_1(\zeta_2)}_{\text{contribution of } e_4-\text{edge functions}} \tag{2.17}$$

$$+ \underbrace{\sum_{k=2}^{p^{s_1,1}} \sum_{n_2=2}^{p^{s_1,2}} \alpha_{k,n_2}^{s_1} l_k(\zeta_1) l_{n_2}(\zeta_2)}_{\text{contribution of } s_1-\text{face functions}}$$

Restricted to the edge $(\mathbf{x}^{s_1})^{-1}(e_9)$, (2.17) yields

$$\sum_{k=2}^{p^{s_1,1}} \alpha_k^{e_9} l_k(\zeta_1) = \sum_{k=2}^{p^{e_3}} o^{e_3}\alpha_k^{e_3} l_k(\zeta_1) l_0(0) + \sum_{k=2}^{p^{e_4}} o^{e_4}\alpha_k^{e_4} l_k(\zeta_1) l_1(0)$$

$$+ \sum_{k=2}^{p^{s_1,1}} \sum_{n_2=2}^{p^{s_1,2}} \alpha_{k,n_1}^{s_1} l_k(\zeta_1) l_{n_2}(0) \quad \forall \zeta \in [-1,1].$$

Thus it is easy to write down the relations for all constrained higher-order edge coefficients for the edge e_9,

$$\alpha_k^{e_9} = o^{e_3}\alpha_k^{e_3} l_0(0) + o^{e_4}\alpha_k^{e_4} l_1(0) + \sum_{n_2=2}^{p^{s_1,2}} \alpha_{k,n_1}^{s_1} l_{n_2}(0), \quad k = 2, \ldots, p^{s_1,1}.$$

What remains to be done is to calculate the constrained higher-order coefficients $\alpha^{s_2}_{n_1,n_2}$, $\alpha^{s_3}_{n_1,n_2}$, $2 \leq n_1 \leq p^{s_1,1}$, $2 \leq n_2 \leq p^{s_1,2}$, corresponding to face functions on the small faces s_2, s_3.

Again it is the *minimum rule* that ensures that $p^{e_3} \leq p^{s_1,1}$ and $p^{e_4} \leq p^{s_1,1}$, allowing us to simplify the summation in (2.17) to

$$\sum_{k=2}^{p^{s_1,1}} l_k(\zeta_1) \left(o^{e_3} \alpha_k^{e_3} l_0(\zeta_2) + o^{e_4} \alpha_k^{e_4} l_1(\zeta_2) + \sum_{n_2=2}^{p^{s_1,2}} \alpha^{s_1}_{k,n_2} l_{n_2}(\zeta_2) \right) \tag{2.18}$$

(the α's with newly introduced indices are zero). Hence, for each $k = 0, \ldots, p^{s_1,1}$ we define a vector coefficient

$$\alpha_k = (o^{e_3} \alpha_k^{e_3}, o^{e_4} \alpha_k^{e_4}, \alpha^{s_1}_{k,0}, \ldots, \alpha^{s_1}_{k,p^{s_1,2}})$$

corresponding to the "long edge" basis functions $l_0, l_1, \ldots, l_{p^{s_1,2}}$. The vector

$$\alpha_k^L = (o^{e_3} \alpha_k^{e_3}, \alpha_k^{e_9}, \alpha^{s_2}_{k,0}, \ldots, \alpha^{s_2}_{k,p^{s_1,2}})$$

corresponding to the "short edge" basis functions $l_0^L, l_1^L, \ldots, l^L_{p^{s_1,2}}$ is obtained as

$$\alpha_k^L = M_L^{p^{s_1,2}} \alpha_k.$$

Analogously for all $k = 0, \ldots, p^{s_1,1}$ the vector

$$\alpha_k^R = (\alpha_k^{e_9}, o^{e_4} \alpha_k^{e_4}, \alpha^{s_3}_{k,0}, \ldots, \alpha^{s_3}_{k,p^{s_1,2}}),$$

corresponding to the "short edge" basis functions $l_0^R, l_1^R, \ldots, l^R_{p^{s_1,2}}$ is obtained as

$$\alpha_k^R = M_R^{p^{s_1,2}} \alpha_k.$$

2.5 Higher-Order Numerical Quadrature

As stated before, we are restricted only on hexahedra, whose geometry have a product form. It implies that the numerical quadrature will also have a product form. So, we can start with one-dimensional case and then extend it to 3D.

2.5.1 One-dimensional reference domain K_a

Let $g(y)$ be a function continuous in interval $[a,b]$, $a < b$. The numerical quadrature of order n on this interval is defined as

$$\int_a^b g(y)\mathrm{d}y \approx \sum_{k=0}^{n} A_{n,k} g(y_{n,k}), \tag{2.19}$$

where the symbols $A_{n,k}$ and $y_{n,k}$, $k = 0, 1, \ldots, n$ denote the quadrature coefficients and nodes, respectively. The nodes $y_{n,k}$, $k = 0, 1, \ldots, n$ are assumed distinct. Putting

$$y = c\xi + d, \qquad c = \frac{b-a}{2}, \qquad d = \frac{b+a}{2}, \tag{2.20}$$

substituting into (2.19) and rearranging, we get a formula corresponding to the one-dimensional reference domain $K_a = (-1, 1)$,

$$\int_{-1}^{1} f(\xi)\mathrm{d}\xi \approx \sum_{k=0}^{n} w_{n,k} f(\xi_{n,k}) \tag{2.21}$$

where $f(\xi) = g(c\xi + d)$ and $w_{n,k} = A_{n,k}/c$. Symbols $w_{n,k}$ are called *weights*.

There are a number of possibilities for choosing suitable weights $w_{n,k}$ and nodes $\xi_{n,k}$ for the numerical quadrature of the function $f(\xi)$. Specific characteristics will be discussed in the following.

Selection of Shape Functions

The problem of determining the integration points and weights is crucial for all types of quadrature rules. In general we can use various systems of linearly independent functions (not only polynomials) whose integrals can be determined analytically. The choice of such functions usually does not matter as long as the order of accuracy is reasonably small. In this case probably the easiest choice is the monomials ξ^i. For higher-order monomials, however, the inversion of the system matrix becomes problematic. The reason is roundoff errors in the evaluation of higher-order monomials for arguments close to zero. Probably the most natural choice is to use either the Legendre polynomials or H^1-hierarchic shape functions.

2.5.2 Gauss Quadrature

The quadrature rules of the Gauss type are based on the summation of weighted function values on nonequidistantly distributed integration points. The n-point Gauss quadrature rule for the one-dimensional reference domain $K_a = (-1, 1)$ reads

$$\int_{-1}^{1} f(\xi)\mathrm{d}\xi \approx \sum_{i=1}^{n} w_{n,i} f(\xi_{n,i}) \tag{2.22}$$

The integration points and weights can be obtained after inserting sufficiently many linearly independent functions with known integrals and resolving the resulting system of nonlinear algebraic equations. Since we have $2n$ unknown parameters at our disposal (n integration points $\xi_{n,i}$ and n weights $w_{n,i}$), the resulting formula will be accurate for all polynomials of order $2n - 1$ and lower.

2.5.3 Reference brick K_B

This section is devoted to higher-order numerical quadrature on the reference brick domain K_B. The product geometry of K_B is analogous to the geometry of the reference quadrilateral K_q, and therefore also the quadrature schemes exhibit common features. Again we will mention a simple and less efficient product scheme with practically unlimited order of accuracy, and several more economical Gaussian quadrature rules.

2.5.4 Composite Gauss quadrature

The simplest quadrature rules can be constructed by combining one-dimensional Gauss formulae in the three axial directions ξ_1, ξ_2, ξ_3. Let the quadrature rule

$$\int_{K_a} f(\xi)\,\mathrm{d}\xi \approx \sum_{i=1}^{M_a} w_{g_a,i} f(y_{g_a,i}),$$

where $y_{g_a,i}, w_{g_a,i}$ are Gauss integration points and weights corresponding to the one-dimensional reference domain K_a, integrate exactly all polynomials of the order p and lower. It is easy to see that the formula

$$\int_{K_a^3} g(\xi_1, \xi_2, \xi_3)\,\mathrm{d}\xi_1\,\mathrm{d}\xi_2\,\mathrm{d}\xi_3 \approx \sum_{i=1}^{M_a}\sum_{j=1}^{M_a}\sum_{k=1}^{M_a} w_{g_a,i} w_{g_a,j} w_{g_a,k} g(y_{g_a,i}, y_{g_a,j}, y_{g_a,k}) \tag{2.23}$$

has the order of accuracy p for functions of three independent variables ξ_1, ξ_2, ξ_3 defined in K_B. The formula (2.23) can easily be generalized to polynomials with different directional orders of approximation.

Similarly as for quadrilaterals, more efficient formulae can be used when integrating *complete polynomials* of the order p (with generally $n = (p+1)(p+2)(p+3)/6$ nonzero terms).

Chapter 3

Arbitrary Level Hanging Nodes in 3D

This chapter describes an arbitrary level hanging nodes, which are required for automatic mesh adaptivity.

3.1 Irregularity rules

Let us demonstrate the irregularity rules on 2D example, since it is easier to show. The extension to 3D is then straightforward. Figure 3.1 demonstrates how the regularity rule forces refinements that are not needed by the solution, which results in the larger number of degrees of freedom and also larger stiffness matrix to solve.

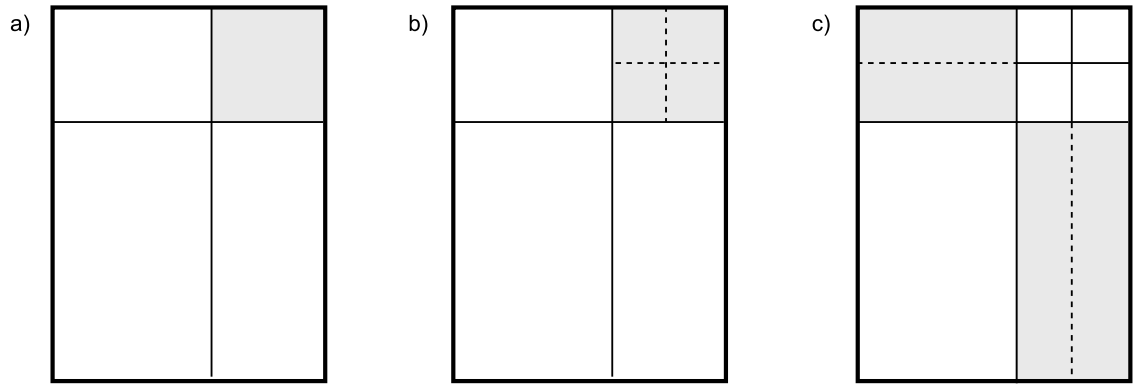


Figure 3.1: Forced refinements

To avoid such forced refinements, we can introduce 1-irregularity rule, which prevents the 2 forced refinements of the upper left and the lower right element. This is all right until we want to refine one of the four upper right elements (3.1c), then the force refinements appear again. We could introduce 2-irregularity rule, but obviously this in not a good way how to deal with forced refinement. The solution to this is a completely irregular mesh, thus k-irregularity rule with $k = \infty$.

The figure 3.2 shows regular (a), 1-irregular (b) and 2-irregular mesh (c). The vertices

marked A, B are hanging nodes of the first order, vertices marked C, D are hanging nodes of the second order.

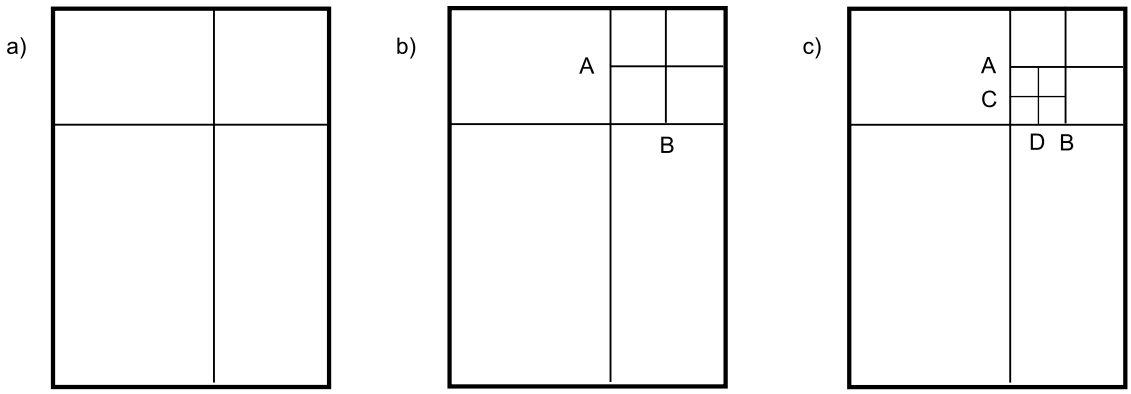


Figure 3.2: Irregular meshes

3.2 Multiple-level Hanging Nodes

We use a hexahedron as the base mesh element, because it is easy to refine and the numerical quadrature has the product form. The hexahedron can be refined in 7 ways: 3 refinements into 2 sub-elements (cutting plane parallel with x-, y-, z-plane), 3 refinements into 4 sub-elements (cutting planes parallel with x,y or x,z or y,z planes) or into 8 sub-elements (cutting planes are x,y,z planes), see Figure 3.3.

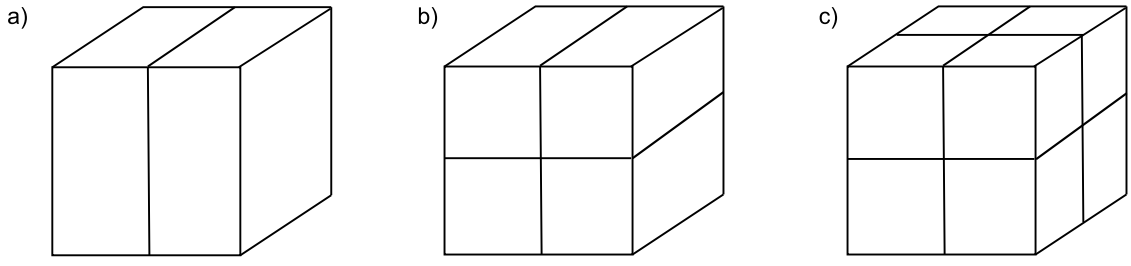


Figure 3.3: Refining a hexahedron

The requirement of continuity of approximation across a face which contains hanging nodes yields in three situations on faces that has to be solved, see Figure 3.4.

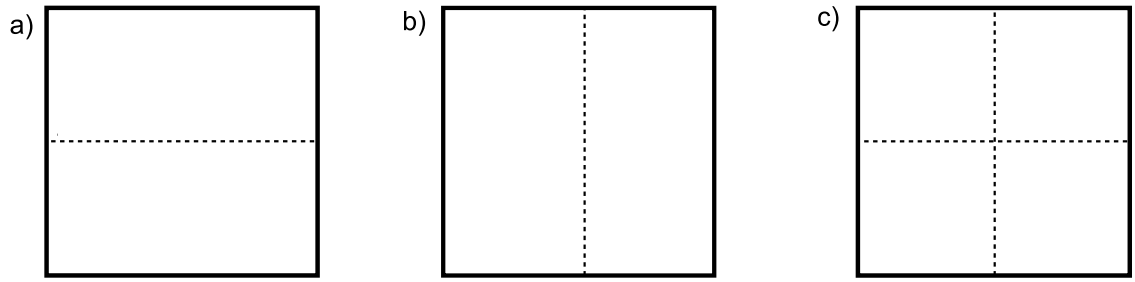


Figure 3.4: Possible refinements on hexahedron faces

In 3D, there are several types of constraints that has to be calculated:

- vertex by vertex, vertex by edge, vertex by face,
- edge by edge, edge by face,
- face by face.

All cases are depicted on the Figure 3.5. Cases (a), (b), (c), (d) occurs when the face is divided either horizontally or vertically. If the face is divided in both direction (e), the situation is lot more complicated. First complication is, that edges EF and GH are not physically present in the mesh, so to calculate associated constraints these edges have to be artificially created in the code (they are needed only for calculations of constraints). The case (e) is simplified for the illustration, for example, it does not include the constraints where edge AB is constraining the edges AE, EB—such situation is shown in case (b).

Cases (a) and (b) are the same as in 2D. New situation is case (c), where the edge on the side (AC) is constraining the edge in the middle of the face (EF). This type of the constraint is simple, because shape functions are in the product form, so we need to decompose the shape function to the directional functions and remember the appropriate function value.

Another new situation is the face by edge constraint. We also use the advantage of the product form of shape functions (face functions are the product of edge function and the linear function). It makes the constraint a lot easier to solve then if they were not product based.

The last 3D constraint is face by face type. This constraint is very simple to solve.

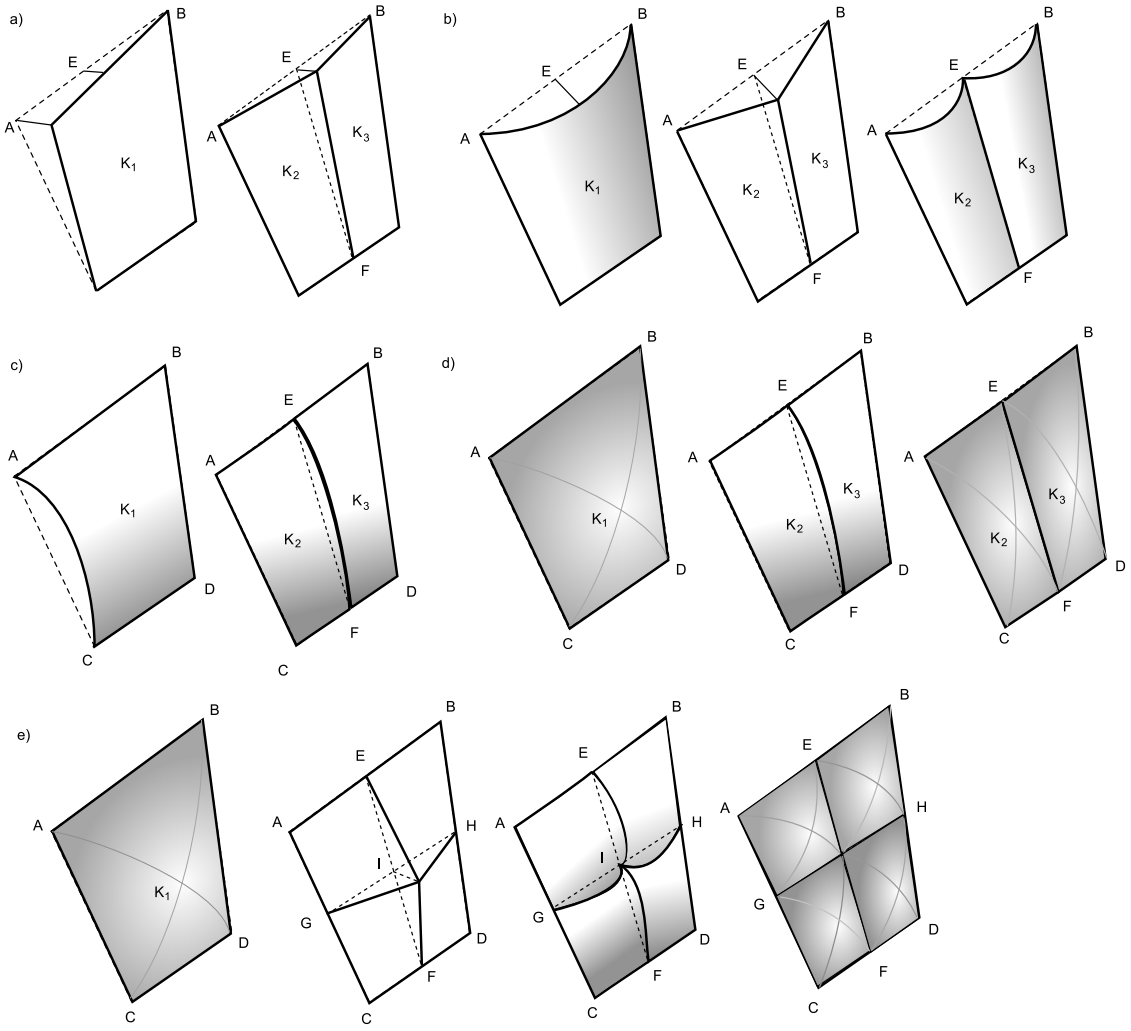


Figure 3.5: Constraints on the hexahedron face.

Indirect Constraints

So far, we dealt only with direct constraints. If one is using 1-irregularity rule, these are the only constraints that have to be solved. Using k-irregularity rule, where $k > 1$ (in our case $K = \infty$), the indirect constraints are introduced.

Indirect constraint is a situation where the function is constrained by another constrain function (as opposed to the direct constraint where the function is constrained by a base function).

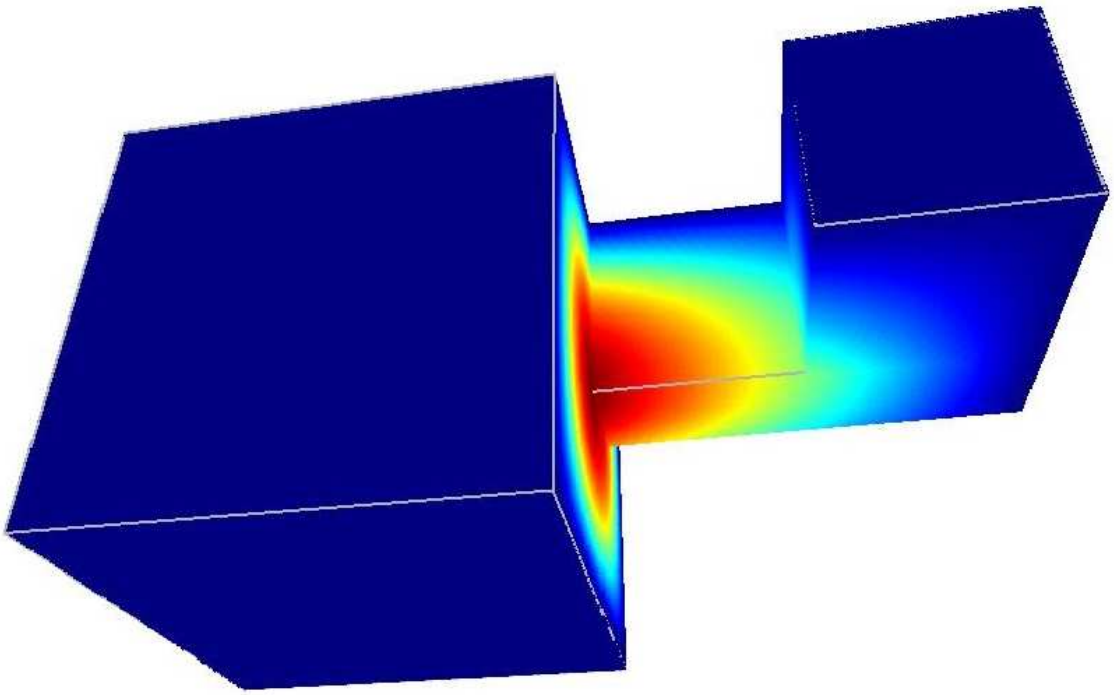


Figure 3.6: Part of the basis function.

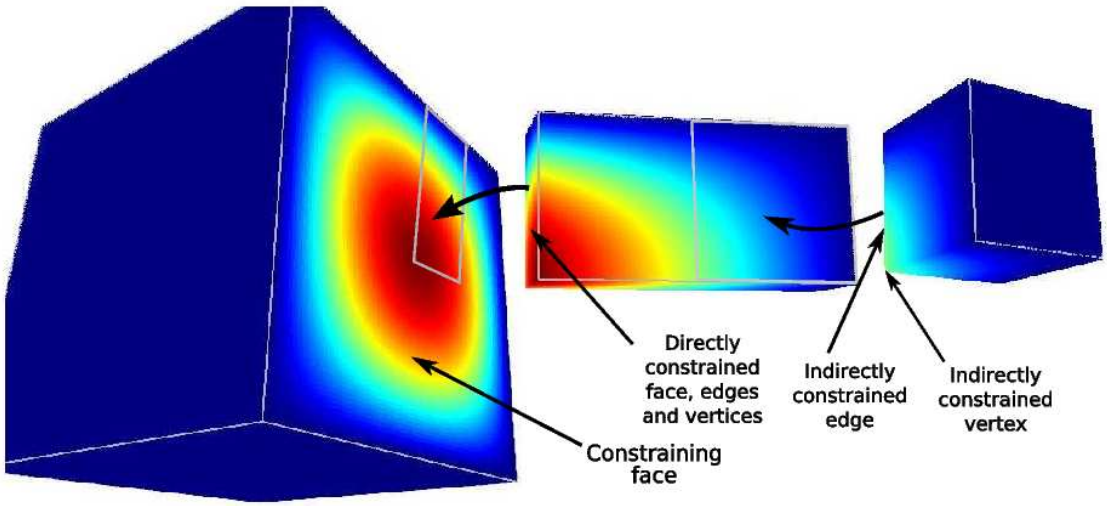


Figure 3.7: Example of a indirect constraint.

The figures 3.6 and 3.7 show how complicated the indirect constraints can be. On the figure a face function that is constraining another face function along with edge functions on the

face. These edge functions are constraining other edge functions, which can constrain a vertex function. Thus, we see another issue related to indirect constraints which is the large support of the basis function— it can go through several elements. So, the handling of the arbitrary level hanging nodes is not a trivial problem to solve.

Chapter 4

Automatic hp-Adaptivity

4.1 Reference Solution

The goal of the adaptivity is to find such a space $V_{h,p}$, which approximates the solution the best. To find the solution $u_{h,p}$ the adaptivity has to have an error estimate, so it "knows" where the mesh should be refined or where the polynomial order should be changed–in other words how to modify the space $V_{h,p}$ to reduce the approximation error. Our approach for this is to introduce the reference solution, i.e. an approximation which is at least one order more accurate than the coarse solution $u_{h,p}$. Then the hp-adaptivity is guided by an a-posteriori error estimate

$$e_{h,p} = u_{ref} = u_{h,p} \tag{4.1}$$

We construct the enriched finite element space by uniform subdivision of all mesh elements and by increasing the polynomial order of all elements by 1, ie. $u_{ref} = u_{h/2,p+1}$.

Such approach is virtually equation-independent, but it assumes that the source of the error is on the same element, which has the largest error. This is not always true. In such a case, different approach has to be chosen.

4.2 Algorithm of hp-Adaptivity

The following paragraph outlines the algorithm which is used for the hp-adaptivity. It is formally similar to algorithm described by Demkowicz et al. in [15] and Cerveny in [19].

1. Assume initial coarse mesh $\mathcal{T}_{h,p}$. User input includes prescribed tolerance $TOL > 0$ for the norm of the approximate error function 4.1 and the threshold $ERRT$ specifying how many elements will be refined in each step.
2. Compute coarse mesh approximation $u_{h,p} \in V_{h,p}$ on $\mathcal{T}_{h,p}$.

3. Find reference solution $u_{ref} \in V_{ref}, V_{h,p} \subset V_{ref}$, where V_{ref} is an enriched finite element space.
4. Construct the approximate error function 4.1, calculate its norm ERR_i on every element K_i in the mesh, $i = 1, 2, \ldots, M$ and calculate the global error estimate

$$ERR = \sum_{i=1}^{M} ERR_i.$$

5. if $ERR < TOL$, stop computation.
6. Sort all element into a list L by their value ERR_i, in decreasing order.
7. Determine the maximum of element errors, $ERR_{max} = \max ERR_i$, by taking the first item of L.
8. Let $NDOF$ be the current number of degrees of freedom of $\mathcal{T}_{h,p}$. Repeat the following cycle:
   (a) Take the next element K from the list L.
   (b) If $ERR_K < ERRT \cdot ERR_{max}$ break the cycle
   (c) Perform hp-refinement of K (to be described in more detail below).
   (d) If the total number of added DOF is greater than $NDOF \cdot MAXDOF$, break the cycle
9. Adjust polynomial degrees on elements.
10. Continue with step 2.

As described in [19], the crucial issue is the number of elements that should be refined. The reference solution has to be obtained in every iteration, but it is very computationally-intensive, especially for 3D problems, see Table 4.1 for the comparison of the coarse and the reference solution sizes. It show the number of elements, number of degrees of freedom and also the time required to assemble the stiffness matrix and time required for solving it.

Table 4.1: Comparison of coarse and reference solution sizes.

Coarse solution				Reference solution			
Elements	DOF	t_{assm} [s]	t_{solve} [s]	Elements	DOF	t_{assm} [s]	t_{solve} [s]
8	27	0.27	0.000687	72	1331	20.41	0.072584
72	27	0.43	0.000565	584	3375	36.98	0.131171
328	155	3.62	0.001247	2632	16031	276.91	0.823985
776	415	11.65	0.002947	6216	38863	661.31	2.428407
2376	1519	43.46	0.013273	19016	124079	2350.14	9.161934

4.3 Selecting Optimal hp-Refinement

Very problematic issue in 3D problems is the number of possible refinements that can be applied on one element. Lets consider increasing the polynomial order by one and two. Then we have 2 p-refinements. We do not do pure h-refinements, since they do not have mathematical sense. The h-refinements have to be connected with redistribution of the polynomial order, then we speak about hp-refinements. There are 3 options how to refine a hexahedron into 2 sub-elements, that gives us 9 possible refinements, then 3 options how to refine a hexahedron into 4 sub-elements, that gives us 81 more refinements and 1 options how to refine a hexahedron into 8 sub-elements, which gives us $6561(3^8)$ additional refinements. Altogether we have 6653 choices how to refine one element.

Hexahedron elements have one speciality—directional orders. Each element can carry a different order in each direction. Allowing this feature results in a better approximations for boundary layer problems, in general where the solution has a big changes in one direction but not in others. Taking in account the directional orders we get 26 p-refinements, 27^2 choices for diving into 2 sub-elements, 27^4 for dividing into 4 sub-elements and 27^8 choices for dividing into 8 sub-elements, which altogether is over 282 billion of choices.

Apparently all the choices can not be tried, so we have to reduce the number of possible refinements to some reasonable number. It is also very important that the error estimation procedure is very fast, since we are not refining only one element per one adaptivity iteration.

The candidate with the smallest projection error with respect to the number of degrees of freedom used is selected for refining the element K.

4.4 Electrostatic Problem

As a first example for the demonstration of the hanging nodes, let us use a problem from electrostatics—distribution of the electrostatic potential in the Fichera corner domain $\Omega = (-1, 1)^3 \setminus [0, 1]^3$. We solve the problem:

$$-\Delta u = f \text{ in } \Omega$$

$$u = u_D \text{ on } \partial\Omega$$

where f and u_D are chosen to comply with the exact solution:

$$u(x, y, z) = (x^2 + y^2 + z^2)^{\frac{1}{4}}$$

The solution of this problem is smooth (see Figure 4.1a) and the gradient of the solution has a singularity at $(0, 0, 0)$ (see Figure 4.1b).

a) b)

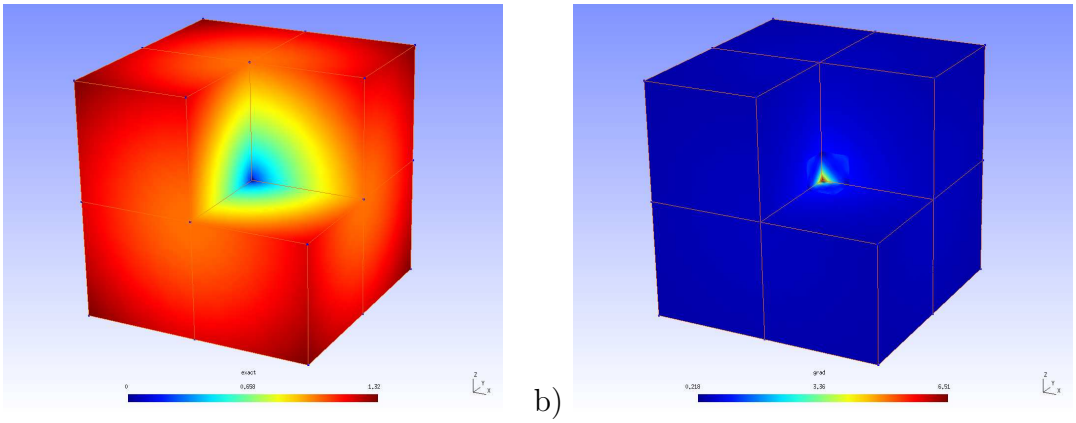


Figure 4.1: Distribution of the electrostatic potential, (a) the exact solution and (b) the magnitude of the gradient

4.4.1 Comparison with h-adaptivity

For the comparison, we ran the h-adaptivity with the element polynomial degree set to 1 and two (standard linear and quadratic elements). We allowed only the division to 8 sub-elements, polynomial order was unchanged during the adaptivity process.

The convergence graph (Figure 4.2) shows the speed of convergence for hp-adaptivity and h-adaptivity with polynomial orders 1 and 2.

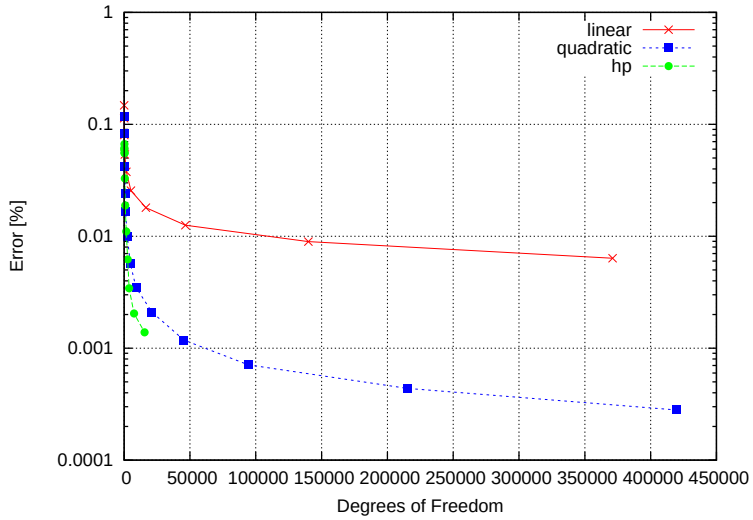
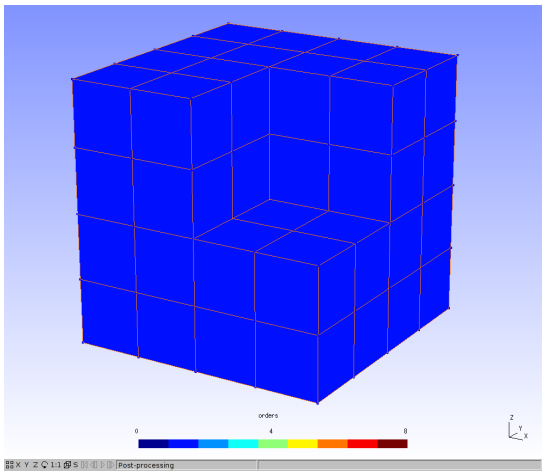


Figure 4.2: Comparison of h-adaptivity with hp-adaptivity

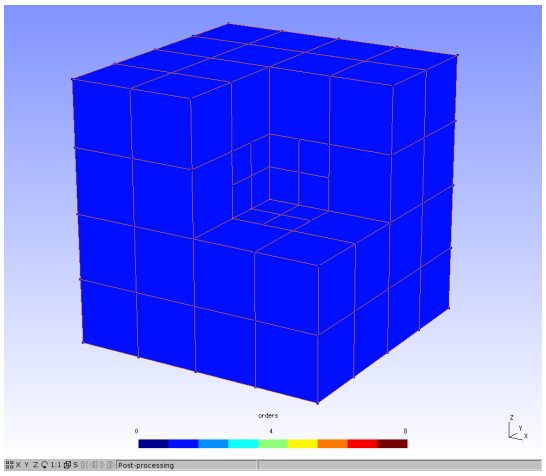
The figure 4.3 show meshes obtained with h-adaptivity and linear elements along with the number of degrees of freedom and error measured in H^1 norm. The figure shows 4.4 the same, but for hp-adaptivity. Table 4.2 show the number of elements in the mesh, number of degrees of freedom, time needed for error estimation (t_{error}), time needed by the adaptivity step (t_{adapt}) and the number of elements refined.

Table 4.2: Statistical data gathered during the run of automatic adaptivity.

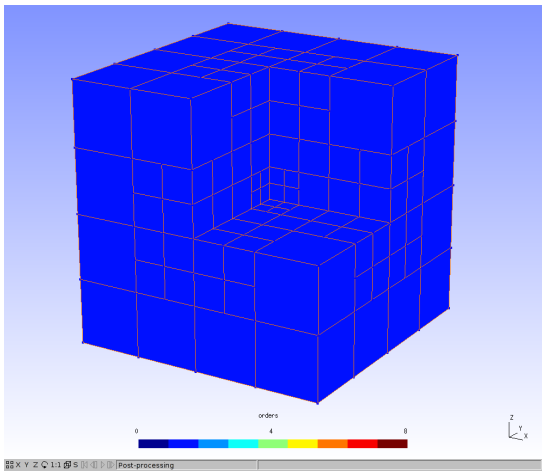
Number of Elements	DOF	t_{error} [s]	t_{adapt} [s]	Refined Elements
7	19	0.342877	1.441635	6
55	19	2.419289	2.562463	7
111	61	4.156678	7.202118	25
167	284	6.836485	12.231948	34
223	525	10.029116	7.310666	29
279	797	13.135698	18.000802	58
383	1518	21.273660	16.869027	76
607	2881	38.668589	24.866616	88



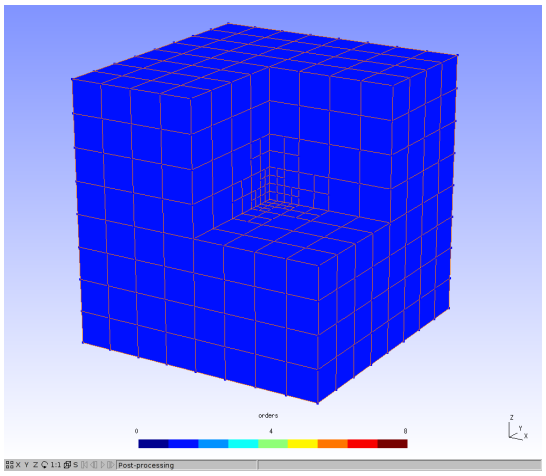
19 DOF, 14.79 %



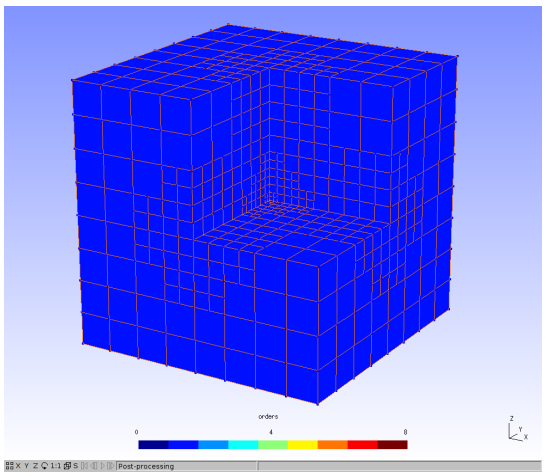
38 DOF, 11.34 %



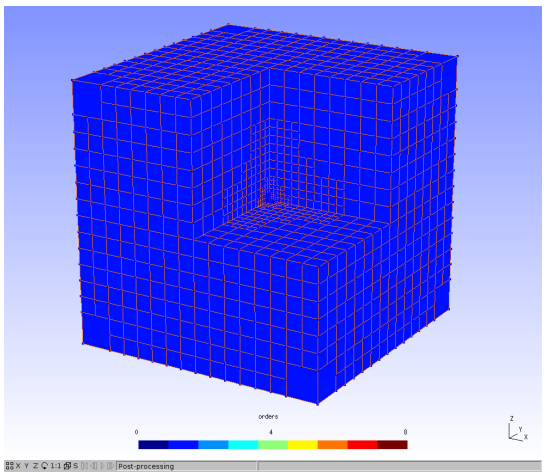
138 DOF, 8.03 %



509 DOF, 5.38 %

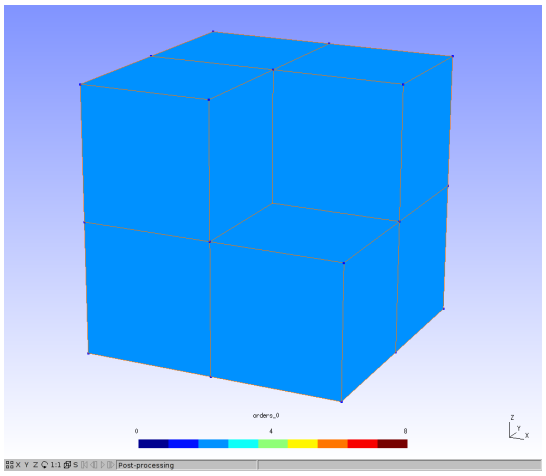


1660 DOF, 3.76 %

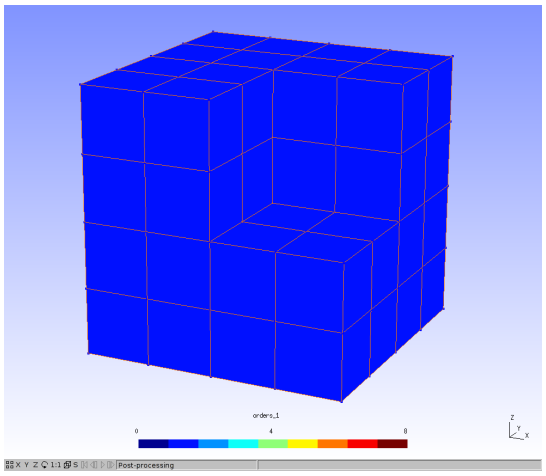


5161 DOF, 2.55 %

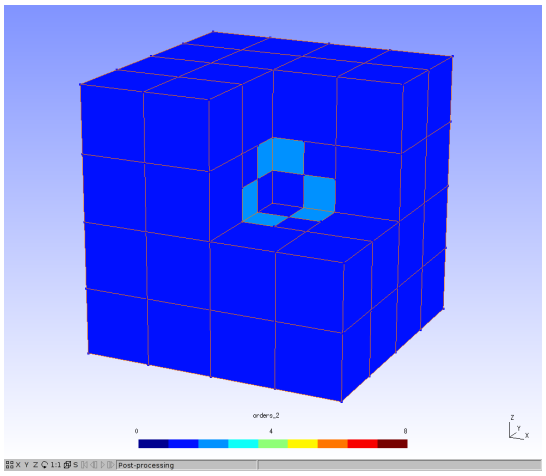
Figure 4.3: First 6 iterations obtained by h-adaptivity



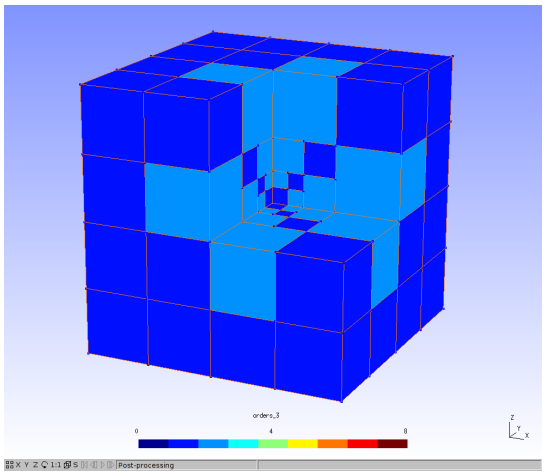
19 DOF, 11.64 %



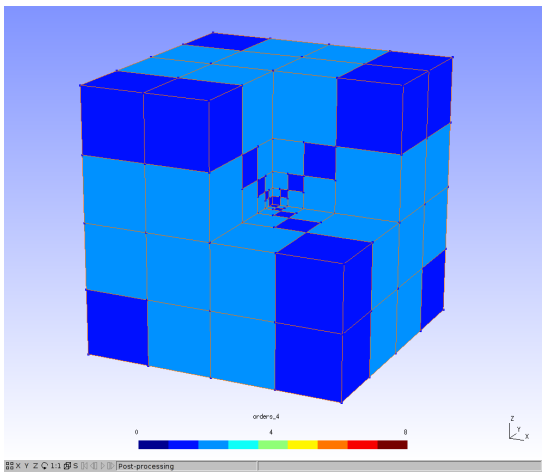
19 DOF, 13.97 %



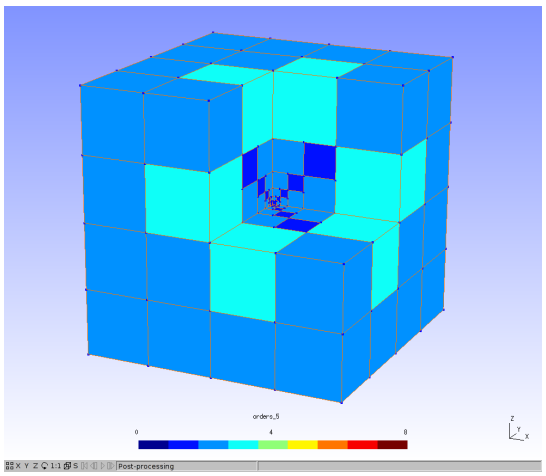
61 DOF, 13.20 %



284 DOF, 9.01 %



525 DOF, 4.32 %



797 DOF, 2.17 %

Figure 4.4: First 6 iterations obtained by *hp*-adaptivity

Chapter 5

Conclusions

This thesis was concerned with adaptive hp-FEM for elliptic problem in 3D. The necessary background was described in detail in Chapters 1 and 2, showing that dealing with numerical solutions of PDEs in 3D is not trivial at all.

Chapter 3 describes how to handle arbitrary level hanging nodes in 3D. They are crucial part of the program code, that can handle automatic hp-adaptivity, which is described in Chapter 4. This chapter also includes one numerical example to demonstrate the exponential convergence of the hp-adaptivity. The comparison with the h-adaptivity is included, too. It clearly proves one of the advantages of the hp-adaptivity—reaching very high accuracy, using a small number of DOF comparing to the classical FEM and h-adaptivity.

Presented algorithm of hp-adaptivity is using so called reference solution to estimate the error of the numerical solution. Table 4.1 shows that the reference solution corresponding to the coarse solution is very demanding in the sense of the number of DOF. Which means that to solve a very simple problem in 3D one needs a huge amount of computer memory just to fit the reference solution in. The reference solution is important for guiding the hp-adaptivity. Because of the huge amount of the memory required, the numerical experiment presented in this thesis used the analytical solution instead.

5.1 Future Work

This thesis served as the proof-of-the-concept that hp-adaptivity is capable of solving PDEs in 3D. It clearly revealed several problems that have to be solved in the future.

The first problem is the high demand on the computer memory. It is not the data structures required by the hp-adaptivity algorithm that would cause problems. It is the resulting stiffness matrix that has to be assembled. There are several techniques how to deal with this problem, one of them is the domain decomposition. This was not implemented, because it is beyond the scope of this thesis. The domain decomposition portion up the computation

domain into a several parts which are handled by a separate machine–for example on nodes of the computer cluster.

Another problem is the assembling of the stiffness matrix. Because of the higher-order shape functions and 3D numerical quadrature, the assembling of the elements with higher order takes much longer than for example assembling of only linear of quadratic elements. This issue can also be solved by the domain decomposition technique.

The next problem revealed during the working on this thesis was the problem of inverting the resulting stiffness matrix. We tried both iterative and direct solvers. The advantage of iterative solvers is that they can handle large amount of equations, but on the other side, they can be used only for elliptic problems—otherwise they wont converge. Direct solvers can be used on any type of problems, but they require a large amount of memory for storing the inverse of the stiffness matrix (the inverse of the sparse matrix is full matrix). That results in handling only relatively small problems with very low accuracy, which means that the advantage of hp-FEM (the high computational accuracy) is not used.

References

[1] P. Šolín, K. Segeth, I. Doležel, *Higher-Order Finite Element Method*, FIXME, 2004.

[2] P.G.Ciarlet, *The Finite Element Method for Elliptic Problems*, North- Holland, Amsterdam, 1979.

[3] I. Babuška, T. Strouboulis, *Finite Element Method and Its Reliability*, Clarendon Press, Oxford, 2001.

[4] G.E. Karniadakis, S.J. Sherwin, *Spectral/hp Element Methods for CFD*, Oxford University Press, Oxford, 1999.

[5] B. Szabó, I. Babuška, *Finite Element Analysis*, John Wiley & Sons, New York, 1991.

[6] M. Ainsworth, B. Senior, Aspects of an *hp*-adaptive finite element method: Adaptive strategy, conforming approximation and efficient solvers, Technical Report 1997/2, Department of Mathematics and Computer Science, University of Leicester, England, 1997.

[7] I. Babuška, T. Strouboulis, K. Copps, *hp*-optimization of finite element approximations: Analysis of the optimal mesh sequences in one dimension, Comput. Methods Appl. Mech. Engrg. 150 (1997), 89–108.

[8] I. Babuška, M. Suri, The optimal convergence rate of the *p*-version of the finite element method, SIAM J. Numer. Anal. 24 (1987), 750–776.

[9] I. Babuška et al., Efficient preconditioning for the *p*-version finite element method in two dimensions, SIAM J. Numer. Anal. 28 (1991), 624–661.

[10] I. Babuška, B.Q. Guo, Approximation properties of the *hp* version of the finite element method, Comput. Methods Appl. Mech. Engrg. 133 (1996), 319–346.

[11] I. Babuška, M. Suri, The *p*- and *h-p* versions of the finite element method, Comput. Methods Appl. Mech. Engrg. 80 (1990), 5–26.

[12] L. Demkowicz et al., De Rham diagram for *hp*-finite element spaces, Comput. Math. Appl. 39 (2000), 29–38.

[13] L. Demkowicz, J.T. Oden, Application of *hp*-adaptive BE/FE methods to elastic scattering, Comput. Methods Appl. Math. Engrg. 133 (1996), 287–318.

[14] L. Demkowicz et al., Toward a universal *hp*-adaptive finite element strategy. Part 1: constrained approximation and data structure, Comput. Methods Appl. Math. Engrg. 77 (1989), 79–112.

[15] L. Demkowicz, W. Rachowicz, P. Devloo, A fully automatic *hp*-adaptivity, TICAM Report 01-28, The University of Texas at Austin, 2001.

[16] W. Rachowicz, J.T. Oden, L. Demkowicz, Toward a universal *hp*-adaptive finite element strategy. Part 3. Design of *hp* meshes, Comput. Methods Appl. Mech. Engrg. 77 (1989), 181–212.

[17] P. Šolín, L. Demkowicz, Fully automatic goal-oriented *hp*-adaptivity for elliptic problems, TICAM Report 02-32, The University of Texas at Austin, August 2002, accepted by Comput. Methods Appl. Mech. Engrg.

[18] L. Demkowicz, D. Pardo, W. Rachowicz, 3D *hp*-adaptive finite element package (3Dhp90). Version 2.0. The ultimate (?) data structure for three-dimensional, anisotropic *hp*-refinements, TICAM Report 02-24, The University of Texas at Austin, June 2002.

[19] J.Červený, Higher-Order Adaptive FEM for Non-linear Problems, Master Thesis, 2007.

Curriculum Vitae

David Andrš was born on February 4, 1980 in Litoměřice, Czech Republic as the second son of Eduard Andrš and Helena Andršová. He graduated from Josef Jungmann's High School, Litoměřice, Czech Republic in the spring of 1998.

In 1998 he entered the College of Computer Science at the University of West Bohemia in Plzeň, Czech Republic, to pursue a master's degree in Computer Science. After receiving his master's degree in the spring 2003, he continued at the same university in pursuing a PhD degree in computer science, which he defended in the summer 2007.

In January 2007, he entered the Graduate school at The University of Texas at El Paso. While pursuing a master's degree in Applied Mathematics, he worked as a Teaching Assistant at the department of Mathematical Sciences.

He gave a presentation at the 1st Joint NMSU/UTEP Workshop on Mathematics and Computer Science.

Permanent address:

Seifertova 16
Litoměřice
412 01
Czech Republic