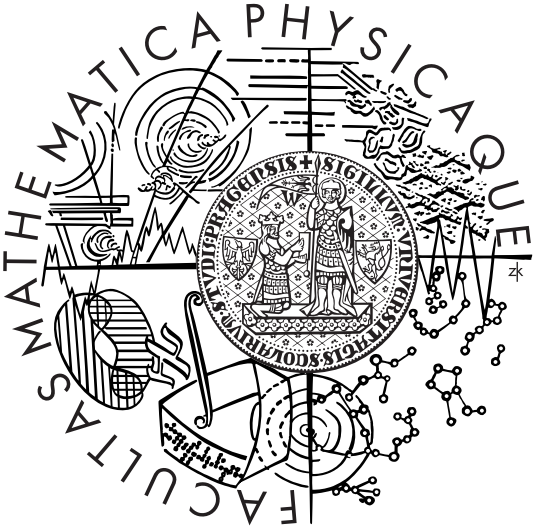
CHARLES UNIVERSITY, PRAGUE
FACULTY OF MATHEMATICS AND PHYSICS

DOCTORAL THESIS



PAVEL KŮS

Automatic *hp*-Adaptivity on Meshes with Arbitrary-Level Hanging Nodes in 3D

Institute of Mathematics, AS CR

Advisor: RNDr. Tomáš Vejchodský, PhD.
Co-advisor: RNDr. Pavel Šolín, PhD.

Study program: M6 – Scientific and technical computing

2011

ACKNOWLEDGMENTS

I wish to express my deep gratitude to all the people who supported me throughout my graduate studies. First, I want to thank my supervisor Dr. Tomáš Vejchodský not only for his valuable advice with the mathematical part of the work, but also for his great support, help and patience over the past four years. Apart from that all, which encouraged me to proceed with the work, I also owe him for the possibility to work at the Mathematical Institute and to visit many conferences, where I met interesting people and learned new inspiring ideas.

This work would not have been possible without Dr. Pavel Šolín, the leader of the HERMES project, who introduced me to the *hp*-FEM, taught me a lot about various aspects of scientific computing and was a constant source of inspiration and ideas for my work. I also feel indebted to him for the opportunity to study at the University of Texas at El Paso and to visit Sandia National Laboratories for a summer internship. It was the friendly approach of him and his wife Dagmar, that made my stay at El Paso such a pleasant experience.

I also want to thank Prof. Ivo Doležel for his support, kindness and for many valuable discussions regarding physical and engineering applications of our project. He gave me the opportunity to work at the Institute of Thermomechanics and although being an extremely busy person, he always found time to help and encourage me.

I also have to thank all the colleagues from our team that I have met at El Paso, namely David Andrš, Jakub Červený, Lenka Dubcová and Martin Zítka. I always enjoyed to cooperate with them all and I learned a lot from them, particularly about computers. I thank all other former and present members of the team for their joint effort that made HERMES such a capable computing system.

Finally, I wish to thank my family for ceaseless support during my studies and Míša for a strong encouragement during final stages of this work and for much more.

This work was supported by the grants GAČR P102/11/0498 and IAA100760702.

I declare that I carried out this doctoral thesis independently, and only with the cited sources, literature and other professional sources.

I understand that my work relates to the rights and obligations under the Act No. 121/2000 Coll., the Copyright Act, as amended, in particular the fact that the Charles University in Prague has the right to conclude a license agreement on the use of this work as a school work pursuant to Section 60 paragraph 1 of the Copyright Act.

Prague, 2011 Pavel Kůs

Abstract

The thesis is concerned with theoretical and practical aspects of the *hp*-adaptive finite element method for solving elliptic and electromagnetic problems described by partial differential equations in three spatial dimensions. Besides the standard element refinements, the *hp*-adaptivity allows independent adaptation of degrees of the polynomial approximation as well. This leads to exponentially fast convergence even for problems with singularities. The efficiency of the *hp*-adaptivity is enhanced even more by the ability of the algorithm to work with meshes with arbitrary-level hanging nodes. This generality, however, leads to great complexity of the implementation. Therefore, the thesis concentrates on the mathematical analysis of algorithms that have led to successful implementation of the method. In addition, the thesis discusses the numerical integration in 3D and the implementation of the method itself. Finally, numerical results obtained by this new implementation are presented. They confirm advantages of *hp*-adaptivity on meshes with arbitrary-level hanging nodes.

Abstrakt

Dizertační práce se zabývá teoretickými a praktickými aspekty *hp*-adaptivní metody konečných prvků pro řesení eliptických a elektromagnetických úloh popsaných parciálními diferenciálními rovnicemi ve třech prostorových dimenzích. Používaná *hp*-adaptivita umožňuje zjemňovat elementy v prostoru i zvyšovat jejich polynomiální řád, což vede k exponenciálně rychlé konvergenci i pro úlohy se singularitami. Efektivitu *hp*-adaptivity ještě zvyšuje schopnost algoritmu pracovat se sítěmi s visícími uzly libovolné úrovně. Tato obecnost však vede ke značné komplexnosti implementace. Jádrem této práce je proto matematická analýza algoritmů, které vedly k úspěšné implementaci metody. Dále jsou diskutovány možnosti numerické integrace ve 3D a samotná implementace metody. V závěru jsou předloženy numerické výsledky získané touto novou implementací, které potvrzují výhody *hp*-adaptivity na sítích s visícími uzly libovolné úrovně.

CONTENTS

CHAPTER

ONE

PREFACE

1.1 Motivation and history of the finite element method

A development of the finite element method started at the end of the first half of the 20th century. It was driven by a growing need for computer simulations of technical problems described by partial differential equations and enabled by the beginning of the development of computers.

In the following decades, the finite element method gained more and more popularity. It started to be used in variety of technical fields, e.g. civil engineering, electrical engineering, fluid dynamics and others. In some fields, completely new type of finite elements had to be developed, since standard treatment is not suitable for discretization of partial differential equations used there. For example, for discretization of Maxwell's equations, edge elements have been developed.

Apart from increasing number of fields, where FEM was used, another development took place. At the very beginning, simple linear elements were used to discretize the area and perform the calculation. When capabilities of computers increased, it was clear, that there are much more effective possibilities. A wide variety of adaptive algorithms have been developed in order to allow more precise solution of the problems. Namely, h-adaptivity appeared, allowing to refine elements in space and p-adaptivity, which introduced usage of higher-order elements. By combining those two approaches one can obtain hp-adaptivity, that employs possibilities to adjust both element size and polynomial degree.

Obviously this method is the most demanding variant from the above mentioned, not only from implementational point of view, but also for its analysis. However, many interesting results showing its superiority have been achieved. Some of

them can be found in [42], [29] and elsewhere.

1.2 Existing *hp*-FEM software

Hand in hand with theoretical development, practical implementations of the method started to appear. Their aim was at the beginning solely to verify theoretical results, but later, more sophisticated codes were developed with aspirations to be used for engineering problems. Nowadays there are more groups developing own *hp*-FEM code and it is hard to name all of them. Let us just mention few of them:

- The group of prof. L. Demkowicz at Texas Institute for Computational and Applied Mathematics at University of Texas at Austin has been working on *hp*-FEM software for a long time. Their package hp90 supports calculations of two and three dimensional problems in H^1, $\boldsymbol{H}^{\text{curl}}$ and $\boldsymbol{H}^{\text{div}}$ spaces using meshes with hanging nodes of the first level. For more information see web page[1] of the project or publications [14], [15] and [16].

- Software developed at Texas A&M University, at Institute of Aerodynamics and Flow Technology of the German Aerospace Center in Braunschweig and at the Numerical Methods Group at University of Heidelberg called deal.II implements the *hp*-adaptivity for two and three dimensional problems. The adaptivity is driven by various types of local error estimators. The library supports meshes with hanging nodes of the first level. Details can be found in [7] or on the project website[2].

- The group of prof. C. Schwab at Seminar for Applied Mathematics at Swiss Federal Institute of Technology in Zurich develops a set of classes for solution of elliptic partial differential equations. It comprises various methods including *hp*-adaptivity based on a-priori knowledge of the solution. For more information see the web page[3] or publication [22].

- Other software described in e.g. [39] and [24].

This is just a brief and incomplete list of the best known packages. Our group develops and uses the system HERMES, which is described in detail in the following section.

[1] `http://users.ices.utexas.edu/~leszek/projects.html`
[2] `http://www.dealii.org`
[3] `http://www.concepts.math.ethz.ch`

1.3 *hp*-FEM system HERMES

HERMES is a modular C++ library for development of adaptive *hp*-FEM solvers. It is focused on solving large range of problems from different fields that are described by partial differential equations, including difficult multi-physics systems and strongly coupled problems. Emphasis is put on effectivity of algorithms, but also on their robustness, universality and ease of use of the library. One of the key features is, that all algorithms are designed to be problem-independent and therefore do not have to be adjusted for each individual type of problem. HERMES is developed by a group around prof. Pavel Šolín at the University of Nevada at Reno (formerly at the University of Texas at El Paso), at the Institute of Thermomechanics in Prague and at the University of West Bohemia at Pilsen.

System HERMES has been developed for a long time. The first was its 2D version, that have already reached a very advanced state. It has been successfully applied on multi-physic problems arising from different fields. It includes novel features such as meshes with arbitrary-level hanging nodes, multi-mesh assembling technique allowing to use different refinements for each physical field in a coupled problem, dynamically changing meshes for nonstationary problems and much more (see, e.g., [45], [43], [20], [47], [19]).

A 3D version of the software has been developed since 2005. The authors active in that stage developed version capable of solution of elliptic problem on a fixed tetrahedral mesh (see [52]). As a practical part of work on this thesis, capabilities of HERMES 3D have been extended to hexahedral meshes with arbitrary-level hanging nodes, allowing adaptivity and enabling $\boldsymbol{H}^{\mathrm{curl}}$ calculations. Some of the newly added features had been strongly inspired by the 2D version (such as adaptivity for instance), others had to be developed independently (such as data structures used to deal with 3D mesh, algorithms handling hanging nodes and construction of basis functions, see [27], [28]).

The development of HERMES constantly continues and is carried out by many new contributors. New features are being added and attempts are made to unify 2D and 3D versions and also by developing several related projects and improving the user documentation. Details can be found on the projects website[4], which is regularly updated. Since HERMES is an open-source project, anyone is welcomed to download its source code, use it and modify it.

[4] `http://hpfem.org/hermes`

1.4 Structure of the thesis

This thesis is divided into several parts, that follow a process leading from the theoretical background of the hp-FEM method through the theory used for our rather demanding approach of meshes with arbitrary-level hanging nodes towards its practical implementation and numerical results. We start with a brief introduction to the field in Chapter 2, where essential definitions and concepts are formulated and the method is explained. We do not, however, intent to provide a comprehensive description, instead of that, several sources for a further detailed study are given.

In the following part of the text, which constitutes the core of the thesis, we address issues that arose during the development of the computer code. Even though some of the tough problems we had to overcome are very technical, we try to present them in a mathematically rigorous way. We have decided to organize this fundamental part into three chapters. In Chapter 3 we address problem-independent issues related to the geometry. It mostly deals with meshes with arbitrary-level hanging nodes and introduces structures used to describe relations in the mesh. In the next two chapters we proceed to description of implementation for elliptic problems (Chapter 4) and Maxwell's equations (Chapter 5). In both of them we first introduce a model problem, its weak formulation and the key properties. It is followed by a definition of local hierarchic basis functions of higher polynomial degrees, as they are used in Hermes. Further we describe in detail, how this local basis functions are used to construct global basis functions on meshes with arbitrary-level hanging nodes, using structures described in Chapter 3.

When we started to use the computer code for calculations, we realized, that numerical quadrature of higher-order basis functions in 3D becomes a big issue. The CPU time needed to evaluate the integrals and construct the stiffness matrix may significantly exceed the time needed for solving the discreet problem. In Chapter 6 we discuss possibilities of more effective implementation of the quadrature. One of the approaches was adapted to our setting and successfully implemented and tested.

The last part of the thesis is dedicated to the computer code. Of course, that the computer code is extensive and there is no use of describing it systematically. On the other hand, there are some details that may be found interesting, so we decided to present them in Chapter 7 in a rather unstructured way. Finally, in Chapter 8, several numerical examples are presented, as were solved by the Hermes 3D software. From the included figures we can see, how the meshes described in the previous chapters may look like in reality. We also present convergence compar-

isons, that show advantages of hp-FEM.

1.5 The Goals of this work

As was already mentioned, the work on this thesis included effort connected to computer implementation of algorithms and performing simulations verifying its correctness. A goal of this text is to deliver necessary theoretical background and to describe challenges, that have been encountered during the development, from the mathematical point of view. It includes presentation of several numerical experiments, that should prove usefulness of the selected approach.

CHAPTER

TWO

INTRODUCTION TO THE HIGHER-ORDER FEM

There are many books, that deal with the higher-order FEM, e.g. [46], [48] [40], [14], [12] and others. Therefore there is no need to repeat all the theory in this thesis. However, in this chapter we will give a brief introduction into the field, with emphasis on those aspects of the finite element method, that are not widely used in standard low-order approach and therefore may be worth recalling. We will mention specifics of hierarchic elements, *hp*-FEM formulation and the de Rham diagram, which plays essential role in the design of basis functions, especially for electromagnetic problem.

2.1 Function spaces

First let us define function spaces, that will be used to introduce weak formulation of partial differential equations, that we address in this work. Let Ω be a domain in $\mathbb{R}^3$. Space used for elliptic equations is

$$H^1 = \{u \in L^2(\Omega);\ \partial u/\partial x_i \in L^2(\Omega),\ \ i = 1,2,3\}, \tag{2.1}$$

which consists of functions from $L^2(\Omega)$, whose distributive partial derivatives lie in $L^2(\Omega)$ as well. It has been for long unclear, which space should be used for discretization of Maxwell's equations. Finally it has been shown (see, e.g., [9]), that the proper space is

$$\boldsymbol{H}^{\mathrm{curl}} = \{\boldsymbol{E} \in (L^2(\Omega))^3; \quad \boldsymbol{\nabla} \times \boldsymbol{E} \in (L^2(\Omega))^3\}, \tag{2.2}$$

which consists of square integrable vector-valued functions, whose distributional curls are square integrable as well. Those spaces can be connected by the de Rham diagram, described in the following section.

2.2 Finite element mesh

In the finite element approach, the notion of the finite element mesh is of the crucial importance. By finite element mesh we mean a set of non-overlapping open polytopic elements, that cover the domain Ω. There are many different types of elements, that are used for 3D calculations, such as tetrahedra, hexahedra (and also their combinations together with prisms and pyramids) or even more general shapes. In this work we deal with hexahedral meshes only. The advantages of hexahedral meshes for our approach are discussed at the beginning of Chapter 3, which is dedicated to mesh-related issues.

Let us start with a definition of the hexahedral mesh.

Definition 2.1. (Finite element mesh) Hexahedral finite element mesh $\mathscr{T}_{h,p} = \{K_1, K_2, \ldots, K_M\}$ over a polytope $\Omega \subset \mathbb{R}^3$ is a geometrical division of Ω into a finite number of non overlapping closed hexahedral elements K_i such that

$$\overline{\Omega} = \bigcup_{i=1}^{M} K_i. \tag{2.3}$$

Each cell K_i, $i = 1, \ldots, M$, is equipped with three orders $p_1^{K_i}$, $p_2^{K_i}$, $p_3^{K_i}$, different for each direction. If all three orders are the same and equal to p^{K_i}, we call p^{K_i} the order of the element.

Definition 2.2. (Regular mesh) A mesh is called *regular* if for any two elements $K_i, K_j \in \mathscr{T}_{h,p}$, $i \neq j$, the intersection $K_i \cap K_j$ is either empty or it is a single common vertex, edge or face.

Remark 2.1. The notion of regular mesh should not be confused with regular family of triangulations, which is related to the shape of individual elements. Also notice that some authors may use the term "conforming mesh" instead of "regular mesh".

Remark 2.2. Although we defined elements K_i to be closed, we use the same symbol K_i even in certain situations, where an open set is required. This is typically the case of the definitions of functional spaces on the element.

The concept of regular mesh is very important, since it is the only mesh that most codes allow. Although our approach is capable of handling much more general

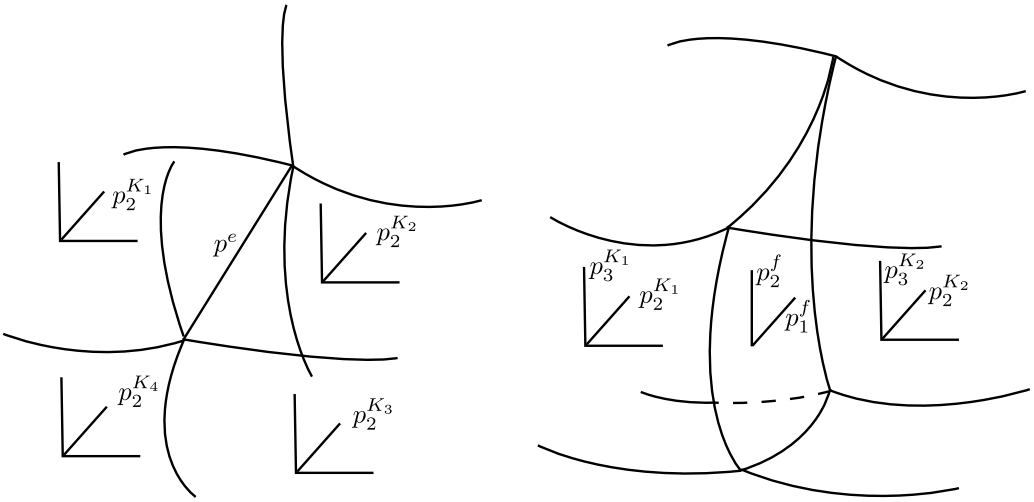


Figure 2.1: Minimum rule used to determine the order of edge e and face f. In this case the minimum rule will result in $p^e = \min\{p_2^{K_1}, p_2^{K_2}, p_2^{K_3}, p_2^{K_4}\}$, $p_1^f = \min\{p_2^{K_1}, p_2^{K_2}\}$ and $p_2^f = \min\{p_3^{K_1}, p_3^{K_2}\}$

meshes, regular mesh is always used as initial mesh, from which the calculation starts. The details will be explained in Chapter 3.

2.2.1 Minimum rule

In the definition of the mesh, we not only defined geometrical division of the space, but also assigned order to each element. This order will later be used to decide, which basis functions from the reference domain should be used to define local basis on the element. The basis function is a polynomial on a given element K and by the order of this basis function in K we mean the degree of this polynomial. Since we use hierarchic basis, we always use only basis functions with order smaller or equal to the order of the element (in each direction). The details will be described later.

In the hierarchical basis, apart from shape functions associated with element interiors (so called bubble functions), there are also shape functions associated with vertices, edges and faces. Therefore we also have to specify orders of edges and faces (orders of vertices are always one), so we are able to choose accordant basis functions related to them.

The key restriction, that has to be observed in order to obtain polynomial space of good properties, is that the orders of edges and faces are always smaller or equal

to appropriate element order, taken from all adjacent elements. The situations for edge and face in the mesh are shown in Figure 2.1.

2.3 De Rham diagram

Proper understanding of the de Rham diagram is essential for finite elements, especially for applications in electromagnetism (see [9]). In this section we want to recall main ideas of the topic. An excellent analysis of the problem can be found in [8], where authors elaborate ideas behind mimetic discretization of differential operators. They use a rather complicated theory of differential forms. A very good introduction to this topic, from a geometrical point of view, can be found in [6]. In this introductory text we will restrict ourselves to simple approach towards the de Rham diagram, as can be found in e.g. [48].

The de Rham diagram relates functional spaces H^1, $\boldsymbol{H}^{\mathrm{curl}}$, $\boldsymbol{H}^{\mathrm{div}}$ and L^2.

$$H^1 \xrightarrow{\boldsymbol{\nabla}} \boldsymbol{H}^{\mathrm{curl}} \xrightarrow{\boldsymbol{\nabla}\times} \boldsymbol{H}^{\mathrm{div}} \xrightarrow{\boldsymbol{\nabla}\cdot} L^2, \tag{2.4}$$

where symbols $\boldsymbol{\nabla}$, $\boldsymbol{\nabla}\times$ and $\boldsymbol{\nabla}\cdot$ stand for gradient, curl and divergence, respectively. The diagram is important for design of basis functions, as well as for stability and convergence of Maxwell's equations and related problems. Its meaning is, that images of functions from one space always lie in the following space of the sequence. Moreover, they form null space of the following operator.

2.3.1 Reference domain

As it has already been discussed, we use only hexahedral meshes in this work. Thus the only reference domain we consider is a cube, defined as $B = \{\boldsymbol{\xi} \in \mathbb{R}^3;\ -1 < \xi_1, \xi_2, \xi_3 < 1\}$. It is depicted in Figure 2.2 together with denotation of its vertices, edges and faces. The reference domain is used to define shape functions (see Sections 4.2.2 and 5.2.1 for definition of shape functions for H^1 and $\boldsymbol{H}^{\mathrm{curl}}$ space, respectively) , that can be transformed to the physical mesh using reference mapping. It is also used for numerical integration, when the weak formulation is transformed from the physical mesh to the reference domain and suitable integration scheme is used to evaluate the integrals. The discussion of numerical integration can be found in Chapter 6.

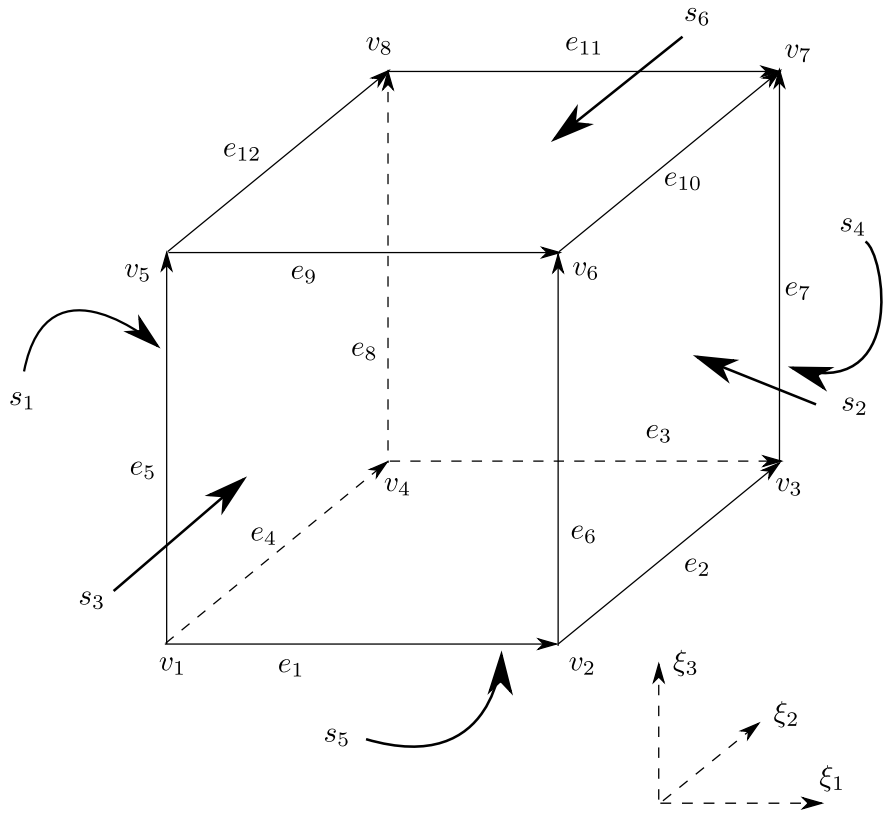


Figure 2.2: Reference cube B.

2.3.2 Reference mapping

Reference mapping maps the reference domain to each individual element in the mesh. As a reference mapping we use trilinear map $\boldsymbol{x}_K$, that maps vertices v_i of reference domain B to vertices v_i' of the element K. Using H^1 vertex functions, that will be defined in Section 4.2.2, we can define reference mapping as follows:

Definition 2.3. (Reference mapping) Let K be element of hexahedral mesh with vertices v_i', $i = 1, \ldots, 8$. Corresponding reference mapping $\boldsymbol{x}_K$ is defined as

$$\boldsymbol{x}_K(\boldsymbol{\xi}) = \sum_{i=1}^{8} v_i' \, \varphi^{v_i}(\boldsymbol{\xi}). \tag{2.5}$$

Reference mapping is a vector-valued function defined on the reference domain B. The use of vertex basis functions φ^{v_i} could be, of course, avoided. It is, however, good from the implementational point of view. Moreover, if we would like to

employ curvilinear elements (and we do not in this work), we could simply add higher-order local basis functions to the sum in the definition. Details of this approach can be found in [48].

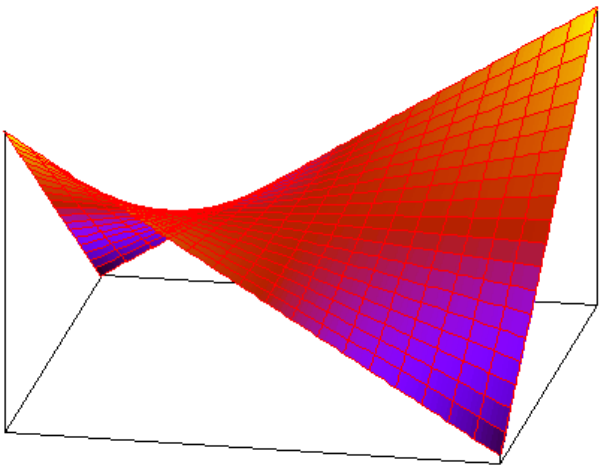


Figure 2.3: Example of face in the physical mesh, whose vertices are not in common plain.

Remark 2.3. Even though we do not use curvilinear reference mappings to construct elements of more general shape, element faces do not have to be parts of plane. If four vertices of the face do not lie in a common plane, resulting shape may look like an example depicted in Figure 2.3. The reason is, that the reference mapping is trilinear, therefore faces are mapped by bilinear mappings, the result is shown in the figure. Of course, line segments are mapped by linear mapping (1D restriction of the 3D trilinear reference mapping) and therefore are mapped to line segments, not curved lines. This remark is meant to show, what type of elements can reference mapping approach handle. In the reality, however, we usually use standard polytopes.

2.3.3 De Rham diagram for reference mappings

For the construction of reference mappings, another version of the de Rham diagram is important. It connects polynomial spaces on the reference domain $W_B^1 \subset H^1(B)$ and $\boldsymbol{W}_B^{\mathrm{curl}} \subset \boldsymbol{H}^{\mathrm{curl}}(B)$ with polynomial spaces on the element K from the physical mesh, $W_K^1 \subset H^1(K)$ and $\boldsymbol{W}_K^{\mathrm{curl}} \subset \boldsymbol{H}^{\mathrm{curl}}(K)$. The relevant part of the diagram can be written as follows:

$$\begin{array}{ccc} W_B^1 & \xrightarrow{\nabla_\xi} & \boldsymbol{W}_B^{\mathrm{curl}} \\ \downarrow \boldsymbol{\Phi}_K^1 & & \downarrow \boldsymbol{\Phi}_K^{\mathrm{curl}} \\ W_K^1 & \xrightarrow{\nabla_x} & \boldsymbol{W}_K^{\mathrm{curl}} \end{array} \tag{2.6}$$

It will be used for construction of transformations of polynomial spaces from the reference domain to the element in the physical mesh.

2.3.4 Transformation of polynomial spaces

Transformation of polynomials (e.g. basis functions) from the reference domain has to be done carefully, since it has to be done differently for different spaces. For the H^1 space, situation is simple. Function value of $\hat{w}$ at $\xi \in B$ has to be the same as function value of transformed function w at transformed point $x = \boldsymbol{x}_K(\xi)$. So the relation can be expressed as

$$w(\boldsymbol{x}_K(\xi)) = \hat{w}(\xi), \tag{2.7}$$

which means, that the mapping $\boldsymbol{\Phi}_K^1$ has the form

$$w = \boldsymbol{\Phi}_K^1(\hat{w}) = \hat{w} \circ \boldsymbol{x}_K^{-1}. \tag{2.8}$$

The situation is more complicated for the $\boldsymbol{H}^{\mathrm{curl}}$ space. The transformation $\boldsymbol{\Phi}_K^{\mathrm{curl}}$ has to be defined in such way, that the de Rham diagram (2.6) commutes. It means, that one obtains the same function after applying transformations in two possible ways in the diagram. In other words,

$$\nabla_x(\boldsymbol{\Phi}_K^1(\hat{w})) = \boldsymbol{\Phi}_K^{\mathrm{curl}}(\nabla_\xi(\hat{w})). \tag{2.9}$$

By application of chain rule we obtain

$$\boldsymbol{E} = \boldsymbol{\Phi}_K^{\mathrm{curl}}(\hat{\boldsymbol{E}}) = \left(\left(\frac{D\boldsymbol{x}_K}{D\boldsymbol{\xi}}\right)^{-T} \hat{\boldsymbol{E}}\right) \circ \boldsymbol{x}_K^{-1}. \tag{2.10}$$

When the weak formulation of an elliptic problem is transformed from the physical mesh to the reference element, we have to write gradient operator ∇_x in the physical mesh by means of the gradient operator ∇_ξ on the reference domain. It can be derived from the de Rham diagram, that

$$\nabla_x w = \left(\frac{D\boldsymbol{x}_K}{D\boldsymbol{\xi}}\right)^{-T} \nabla_\xi \hat{w}. \tag{2.11}$$

Similar relation may be derived for the curl operator, which is used in the weak formulation of electromagnetic problems:

$$\nabla_x \times \boldsymbol{E} = J_K^{-1}(\nabla_\xi \times \hat{\boldsymbol{E}}), \tag{2.12}$$

where

$$J_K(\xi) = \det\left(\frac{D\boldsymbol{x}_K}{D\boldsymbol{\xi}}\right) \tag{2.13}$$

stands for the Jacobian of the reference map $\boldsymbol{x}_K$.

2.4 Degrees of freedom

Degree of freedom (often abbreviated as DOF) has an important role in the finite elements. It is used in several contexts, but the meanings are connected. It will be defined in the following definition of the finite element. In our context of hierarchic elements, we use it also with connection to global basis functions and finally, when problem is transformed to the form of system of linear equations, DOFs may be related to its unknowns.

Definition 2.4. (Finite element) By finite element we mean triple (K, P, Σ), where K is a domain in $\mathbb{R}^d$, P is a space of polynomials on K of dimension n and Σ is a set of linear forms $L_i : P \to \mathbb{R}$, $i = 1, 2, \ldots, n$, called degrees of freedom.

Definition 2.5. (Unisolvent finite element) We say, that finite element (K, P, Σ) is unisolvent if arbitrary $p \in P$ satisfies the property

$$L_i(p) = 0 \; \forall i = 1, \ldots, n \quad \Rightarrow \quad p = 0.$$

It can be shown, that the unisolvency is equivalent with the fact, that there exists such a basis $q_1, \ldots, q_n$ of P such that $L_i(q_j) = \delta_{ij}$, where δ_{ij} is the Kronecker delta.

In the following we will show two particular ways how to interpret this rather theoretical definition. The nodal approach, where degrees of freedom are associated with values in certain points, is the most natural and the simplest one. It also has many favorable properties and therefore it is the most popular choice.

The only disadvantage that prevented us from using it is that in the nodal approach the bases corresponding to different polynomial degrees are constructed using completely different basis functions. Thus it is not possible to increase the polynomial order of an element space just by adding extra basis functions. Rather than that, one has to use completely different basis. This inconvenience that does not matter at all when the order of elements is constant, would bring us many troubles in the hp-adaptivity algorithm. Therefore we use hierarchic approach. Let us, however, briefly describe both possibilities.

2.4.1 Nodal elements

The most natural choice of degrees of freedom is to define $L_i(p)$ as the function value $p(x_i)$ in certain point $x_i \in K$ of the element. If points x_i are chosen correctly, finite element with degrees of freedom L_i is unisolvent and it is called nodal. The more points x_i of the element we choose, the higher polynomial degree can be. This choice is conventional from many reasons, for example for easy definition of the interpolation. It is, however, not optimal for hp-adaptivity, because in the standard construction of nodal basis, all the basis functions have the same polynomial degree. Hierarchic elements seem to be better choice for our needs.

2.4.2 Hierarchic elements

In the hierarchic approach we first select the basis $B^k = \{p_1, p_2, \ldots, p_{n_k}\}$ of the polynomial space P^k. Basis can be simply extended by adding more basis functions, so we can construct sequence of bases

$$B^k \subset B^{k+1}.$$

That is the hierarchic nature of the construction. Any polynomial $p \in P^k$ can be uniquely expressed as a linear combination

$$p = \sum_{i=1}^{n_k} \beta_i p_i$$

where β_i are real coefficients. Now if we define $L_i(p) = \beta_i$, L_i is evidently linear form. By this choice we obtain a set of linear forms (degrees of freedom) such that L_i is directly associated with an element of the basis p_i (rather than with a point as in the nodal approach) and finally we can form a unisolvent finite element (K, P, Σ), where $\Sigma = \{L_1, L_2, \ldots, L_{N_p}\}$.

The advantage is obvious. To obtain a finite element of higher order, one only has to enrich the basis with several new functions, the rest remain intact. There

is, however, one disadvantage when compared to the nodal case. Linear forms L_i are *not* defined outside of the local polynomial space $P(K)$. This means, that we can not use those forms to define standard interpolation operators. Technique called projection-based interpolation, which is used instead, is described in the following.

2.5 Projection-based interpolation

Interpolation on finite element is important from many practical and theoretical reasons, e.g. for the notion of conformity, that will be discussed in the following section. First let us define interpolation, that can be used under certain condition, that is fulfilled for most types of finite elements (including nodal).

Definition 2.6. (Finite element interpolant) Let (K, P, Σ) be unisolvent finite element and $B = \{q_1, q_2, \ldots, q_n\}$ basis of the space P such that $L_i(q_j) = \delta_{ij}$. Let $v \in V$, where $P \subset V$ such that all linear forms $L_1, L_2, \ldots, L_n$ are defined in v. Than the finite element interpolant is defined as

$$I_K(v) = \sum_{i=1}^{n} L_i(v) q_i.$$

The problem is, that for hierarchical elements the degrees of freedom L_i are not defined outside the space P. Thus we have to find a different way, how to define interpolation operators for hierarchical space. It is not at all straightforward and requires deeper mathematical analysis. In this text, we will present the main ideas only and skip the details, which can be found, e.g, in [48].

The key idea is, that we have to combine standard Lagrange interpolation with projection (that is the reason of calling this technique projection-based interpolation). Consider function $v \in V$, $P \subset V$, as in the previous definition. The interpolant is found by successive projecting v on spaces created by basis functions associated with vertices, edges, faces and the element interior. After each step, we subtract the partial interpolant from v and in the next step project only the difference. Details can be seen in proofs of Theorems 4.4 and 5.2, where projection-based interpolation technique is used, or in the book mentioned above.

2.6 Conformity requirements

In this section, let us discuss the notion of conformity requirements of function spaces defined in Section 2.1. We do not present the exact definition of conformity,

since it is rather complicated and we would not use it in the following text. It can be found in, e.g., [46]. The definition is related to the properties of the global interpolant (which is defined using local interpolants on individual elements). The idea is, that if the finite elements are conforming to a space V, global interpolants of functions from V (or from its dense subspace) stay in V.

In this thesis we are constructing conforming finite elements for spaces H^1 and $\boldsymbol{H}^{\mathrm{curl}}$. Thus, let us state two fundamental theorems, that will be used in the following. The details can be found in, e.g., [35], [36], [37] and [46].

Theorem 2.1. (Conformity requirements of the space H^1) *Let $\mathscr{T}_{h,p}$ be a finite element mesh on a domain $\Omega_h \subset \mathbb{R}^3$. The function $v : \Omega_h \to \mathbb{R}$ belongs to $H^1(\Omega_h)$ if and only if $v|_K \in H^1(K)$ for each element $K \in \mathscr{T}_{h,p}$ and the trace of $v|_{K_1}$ and $v|_{K_2}$ on f is the same for each common face $f = K_1 \cap K_2$, $K_1, K_2 \in \mathscr{T}_{h,p}$.*

Proof. The proof is done using the Green's theorem and can be found in [46]. □

Theorem 2.2. (Conformity requirements of the space $\boldsymbol{H}^{\mathbf{curl}}$) *Let $\mathscr{T}_{h,p}$ be a finite element mesh on a domain $\Omega_h \subset \mathbb{R}^3$. Consider a function $\boldsymbol{E} : \Omega_h \to \mathbb{R}^3$ such that $\boldsymbol{E}|_K \in (H^1(K))^3$ for each element $K \in \mathscr{T}_{h,p}$. Then $\boldsymbol{E} \in \boldsymbol{H}^{\mathrm{curl}}(\Omega_h)$ if and only if the traces of the tangential components $\boldsymbol{n} \times \boldsymbol{E}|_{K_1}$ and $\boldsymbol{n} \times \boldsymbol{E}|_{K_2}$ on f are the same for each common face $f = K_1 \cap K_2$, $K_1, K_2 \in \mathscr{T}_{h,p}$. The symbol $\boldsymbol{n}$ stands for a fixed normal vector to the face f.*

Proof. Again, the proof can be found in [46]. □

We have to keep those requirements in mind, when we design global basis functions in Chapters 4 and 5.

2.7 The Galerkin method

The variational formulations of partial differential equations are typically formulated in infinite dimensional spaces of functions. The idea of the Galerkin method is to replace the infinite dimensional space by its finite dimensional subspace. The problem in the finite dimension is then equivalent to a system of algebraic equations, which can be solved by standard methods. To be more specific, let us suppose, we want to find a solution $u \in V$ to a weak formulation

$$a(u, v) = l(v) \qquad \text{for all } v \in V,$$

where V is a space of infinite dimension, a stands for a bilinear and l for a linear form. In the Galerkin method we construct sequence of finite-element subspaces

V_n and find sequence of approximate solutions $u_n \in V_n$, such that

$$a(u_n, v) = l(v) \qquad \text{for all } v \in V_n.$$

According to Céa's lemma, the accuracy of the approximate solution u_n depends on approximation properties of the space V_n, or, more precisely, on ability of the space V_n to approximate the exact solution u. Our task therefore is to construct such spaces in an adaptivity process. It has been shown, that hp variant of the finite element method, allowing both spatial refinements and polynomial degree variability is extremely efficient in this effort.

Its advantage is that it allows us to construct such subspaces of V, that reflect different types of behavior of the solution. We can use large higher-order elements in areas, where the solution is smooth, small low-order elements around singularities, anisotropic elements and polynomial orders in boundary layers, etc. And, of course, the choice of element types is done fully automatically in the adaptivity process, see Section 7.3 for details.

CHAPTER

THREE

HEXAHEDRAL MESH WITH ARBITRARY-LEVEL HANGING NODES

In the finite element method, the concept of the finite element mesh is of a crucial importance. In three spatial dimensions, various types of meshes are used. The most common are tetrahedral, hexahedral or hybrid (where tetrahedral and hexahedral elements are used together with prisms and pyramids) meshes. Another fundamental distinction is whether we allow so-called hanging nodes in the mesh, or not.

The reason, why we are interested in hanging nodes, is that we want to use adaptive algorithms, that will refine some elements in order to obtain more precise solution in areas, where its behavior is complicated. There are some sophisticated methods that allow local refinements and keep the tetrahedral mesh regular, see e.g. [25], [11]. Of course, it is simpler to assemble stiffness matrix on the regular mesh. On the other hand, these methods are not general enough to allow development of fully automatic adaptivity algorithm.

In this work we will consider hexahedral meshes with arbitrary-level hanging nodes. Hexahedral elements have one advantage, which is essential for our intents. It is very easy to refine hexahedral elements. We can simply split them into two, four or eight sub-elements, which are again hexahedral. Division of tetrahedral elements into smaller tetrahedrons is more complicated, at least from the implementational point of view.

It is also possible to divide tetrahedron to eight smaller tetrahedra. However, dividing of hexahedron into two sub-elements is much simpler. It is not question

of dividing itself, but mainly of construction of basis functions on such meshes. In algorithms using hexahedral meshes we can handle all situations by successive application of simple splitting of hexahedron into two parts, as it is discussed in the following and also in chapter devoted to practical implementation.

The drawback of presence of hanging nodes in the mesh is, that construction of global basis functions is much more complicated. In [14] the authors describe algorithms, that are capable of handling hanging nodes, but only of the first level. Inspired by work [49], where a very sophisticated algorithm is described for 2D problems, we wanted to allow arbitrary-level hanging nodes, which would bring several advantages, that are discussed in the following.

It turned out, however, that the situation in 3D is much more complicated. The geometrical complexity of the setting can be appreciated from the following text. After introduction describing mesh refinements, we define in Section 3.3 structures used for description of complicated relations in the mesh and dependency of vertices, edges and faces. In Section 3.4 orientation handling of edges and faces is introduced. Those definitions are fundamental for the construction of global basis functions for elliptic and for electromagnetic problems described in Sections 4.3 and 5.3, respectively.

3.1 Element refinements

As it was already mentioned, one of the important advantages of hexahedral elements is the possibility to refine them easily. The only type of refinement, that we have to consider, is the division of an element into two halves. More complex refinements, such as natural isotropic refinement of an hexahedron to 8 sub-elements can be achieved by successive application, as shown in Figure 3.1.

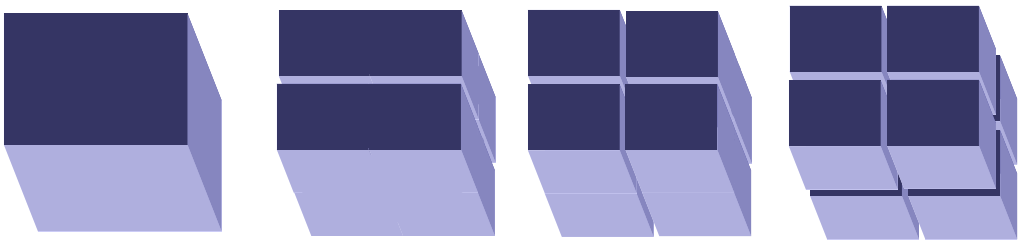


Figure 3.1: Isotropic refinement of a hexahedral element into 8 sub-elements by successive halving.

Now let us define the element splitting more precisely.

Definition 3.1. (Element splitting) Let K be element of the mesh obtained by the reference mapping $\boldsymbol{x}_K$. Let

$$\boldsymbol{x}^1_{\text{split}}(\xi_1,\xi_2,\xi_3) = (\frac{1}{2}\xi_1 - \frac{1}{2},\xi_2,\xi_3), \quad \tilde{\boldsymbol{x}}^1_{\text{split}}(\xi_1,\xi_2,\xi_3) = (\frac{1}{2}\xi_1 + \frac{1}{2},\xi_2,\xi_3).$$

By refining the element K in the direction ξ_1 we mean removing element K from the mesh and adding two elements $\bar{K}$ and $\tilde{K}$ obtained by reference mappings $\boldsymbol{x}_K \circ \boldsymbol{x}^1_{\text{split}}$ and $\boldsymbol{x}_K \circ \tilde{\boldsymbol{x}}^1_{\text{split}}$, respectively. In the same way we define splitting operators and refinements in directions ξ_2 and ξ_3.

3.2 Regularity of the mesh

Possibility of refinements of elements in the mesh brings up an issue of regularity of the mesh. First let us define several notions, that will be useful in the following:

Definition 3.2. (Mesh node) A node is an entity in the mesh, such that degrees of freedom may be associated with it. In our case, a node can be either vertex, edge, face or the element interior.

Definition 3.3. (Hanging node) A hanging node is such node, that is contained in interior of another node of the same or higher dimension.

Theorem 3.1. (Regular mesh) *The mesh is regular, if and only if there are no hanging nodes present.*

Proof. The statement of this theorem is obviously equivalent to the Definition 2.2. □

Our automatic adaptive process is always started with a regular mesh, which is called coarse (also base or initial) mesh. This mesh can be a product of some mesh generation tool, or, in the case of simple geometries, it may be created by hand. One of the advantages of adaptivity algorithms is that we do not have to start with perfect fine mesh. The only demand we have on the initial mesh is to capture the geometry of the domain.

When the automatic adaptivity process starts, elements are refined and hanging nodes appear, unless we use some strategy to get rid of them (and as was mentioned in the beginning of this chapter, there is no sufficiently general method available). Hanging nodes are parts of another nodes, therefore we have to treat them differently – we are not allowed to assign values on them freely. In order to satisfy the conformity requirements, values are imposed by the other nodes. The

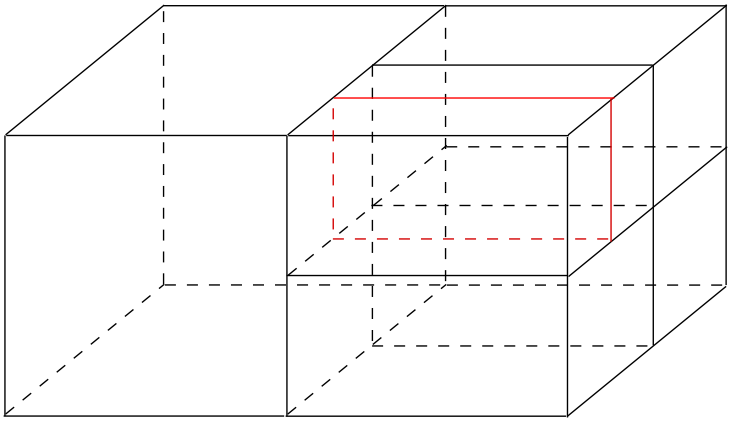


Figure 3.2: Comparison of irregularity rules. Additional splitting of the upper right element by the red face turns the mesh to 2-irregular.

presence of arbitrarily distributed hanging nodes brings lots of complications for the implementation of the method. Therefore some authors simplify the setting by only allowing hanging nodes of the first level, see e.g. [14].

Definition 3.4. (1-irregular mesh) Mesh is called 1-irregular, if each face is adjacent to either undivided face, two faces or four faces obtained by dividing face in both directions (the shape of "cross", as in Figure 3.1).

The difference between 1- and 2-irregular mesh is illustrated in Figure 3.2. If only black edges are present, the mesh is 1-irregular, according to the previous definition. When the refinement indicated in red appears, the mesh starts to be 2-irregular. By the same way we could define n-irregular meshes for arbitrary n.

Although a 1-irregular mesh is much more flexible than the regular, those algorithms that use it suffer from similar drawbacks as those, that use regular mesh (of course, in much less extent). If the process of automatic adaptivity results in mesh with hanging nodes of higher level, additional refinements have to be performed in order to obtain mesh with hanging nodes of first level only. It means more element splitting and adding more degrees of freedom, which do not help decrease the error and are present only in order to keep the mesh 1-irregular. Those extra degrees of freedom cause slow-down of the convergence of the method.

Another problem is, that if we introduce forced refinements in order to enforce 1-irregularity of the mesh, we lost locality of the adaptivity procedure. In other words instead of refining elements independently, only with respect to their contribution to the error, we have to check, whether neighboring elements should be refined as well. Sometimes it may not be sufficient to check neighboring elements

only, since forced refinements may spread through the mesh in a recursive manner.

From this reason we have chosen to overcome difficulties caused by handling meshes with arbitrary level hanging nodes in order to achieve best possible convergence of the method and also to avoid nonlocal refinement of elements.

3.2.1 Incompatible refinement

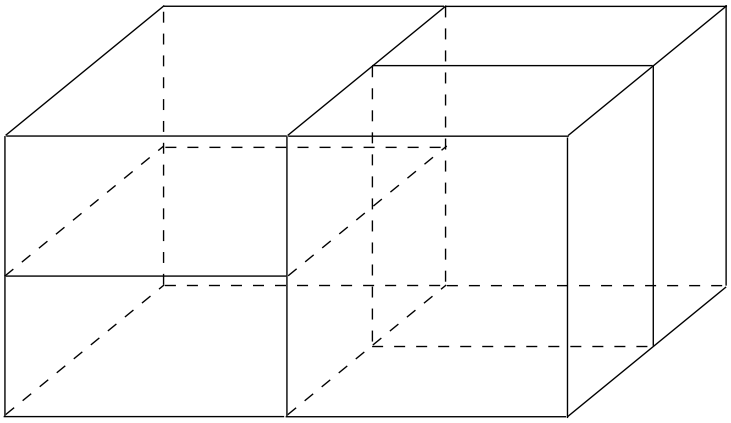


Figure 3.3: Incompatible element refinements.

The meshes with hanging nodes may not be completely arbitrary. There is one situation, that has to be avoided. Two elements sharing one face can not be divided in such way, that the common face would be split differently from either direction, as shown in Figure 3.3. If such situation occurs during the adaptivity process, one of the elements has to be further divided so that division of the face is not in conflict. The reason of this is, that in the situation from Figure 3.3 we cannot say, which nodes are constrained and we would not be able to construct global basis functions (at least the way we do it).

3.3 Constrained nodes

In Section 2.4.2 we introduced the idea of hierarchic elements. We stated, that in this approach, degrees of freedom are associated with mesh nodes. In a regular mesh, generally all nodes may have assigned degrees of freedom, their number depends on their order[1].

[1]Degrees of freedom are not always present, e.g. if a node lies on Dirichlet boundary, if the

When we deal with meshes with hanging nodes, the situation is different. If we want to keep basis functions conforming[2], we cannot place degrees of freedom into some of the nodes. Rather than that, local basis functions related to those nodes will contribute to construction of global basis functions related to constraining nodes, as will be shown in accordant sections of Chapters 4 and 5.

In the following we will describe types of constrains and the way to handle them. Although this is a mesh-related issue, if we want to define things meaningfully, we have to have in mind, what will be basis functions like. Indeed, the purpose of all these constrain handling is to be used for construction of global basis functions in the future. That is the reason, why we will state something about basis functions first, so that further definitions make sense.

3.3.1 Restrictions on basis functions

In the following we want to describe, how we will deal with constrains in the mesh when constructing global basis functions. We will deal with basis functions in the following chapters, but here we would like to address issues, which are more or less related to the mesh itself and will be used for both H^1 and $\boldsymbol{H}^{\mathrm{curl}}$ space.

On the other hand, the following definitions would not be suitable for all types of basis functions. So if we want to see the sense in the following chapter, we will state in advance, that our local basis functions will be designed in the following way:

- Vertex functions, if any, are simply trilinear on the reference cube with value 1 in one vertex and 0 in others.
- Edge functions are polynomial on one edge, linear in both directions perpendicular to the edge and vanish on all other edges.
- Face functions are polynomial in both directions parallel to the face and linear in the other one and vanish in all other faces.
- Interior functions are supported in the element interior.

Note that $\boldsymbol{H}^{\mathrm{curl}}$ basis functions are vector-valued, but in our approach for each basis function only one component is nonzero and satisfies previous conditions.

order of the element used for discretization of elliptic problem is 1 (then there are only linear basis functions and therefore no degrees of freedom related to edges and faces) or in the case of $\boldsymbol{H}^{\mathrm{curl}}$ space, where there are no degrees of freedom associated with vertices.

[2]Continuous for H^1 space, tangential components continuous for $\boldsymbol{H}^{\mathrm{curl}}$ space, etc.

3.3.2 Types of constrains

Let us describe types of constrains that we have to take in account. We can divide them into two basic categories. *Direct constrains* occur when vertices, edges and faces lie on a constraining face and values on them are imposed by the values on the constraining face. *Indirect constrains* are, as the name suggests, caused by more successive constrains spreading through the mesh.

The idea of handling arbitrary level hanging nodes is taken from work [49], where it is described for 2D meshes. The 3D case, however, is much more complex. There are more types of nodes, more types of constrains, both vertex and edge nodes are subject to indirect constrains and overall geometrical situation and necessity of proper handling of orientations makes whole thing rather complicated to implement.

Direct constrains

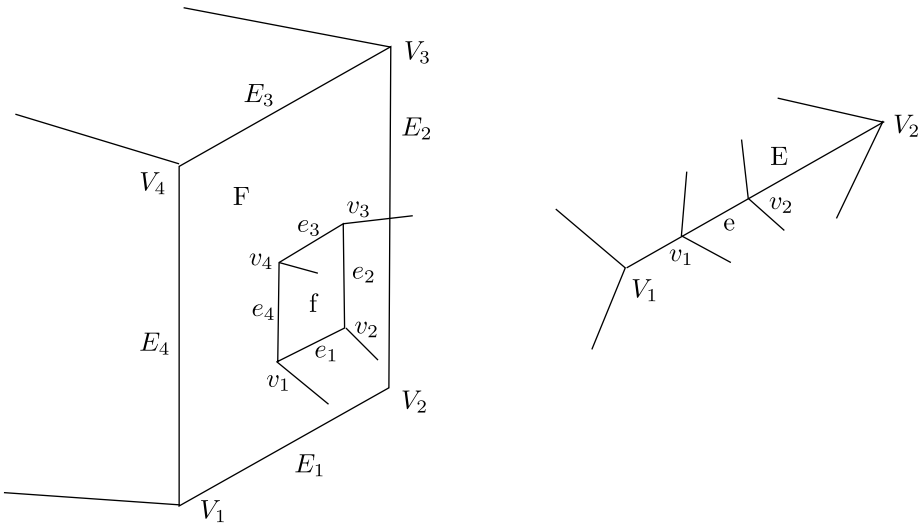


Figure 3.4: Direct constrains on face and edge.

Definition 3.5. (Direct constrains on face) Let face f be a part of face F, as in the first part of Figure 3.4. Than we say, that face f, edges $e_1, \ldots, e_4$ and vertices $v_1, \ldots, v_4$ are directly constrained by the face F. Moreover, edges e_1 and e_3 are constrained by edges E_1 and E_3, edges e_2 and e_4 are constrained by edges E_2 and E_4 and all vertices $v_1, \ldots, v_4$ are constrained by all edges $E_1, \ldots, E_4$ and all vertices $V_1, \ldots V_4$.

Definition 3.6. (Direct constrains on edge) Let edge e be a part of edge E, as in the second part of Figure 3.4. Than we say, that edge e and vertices v_1 and v_2 are directly constrained by edge E and vertices v_1, v_2 are constrained by vertices V_1 and V_2.

Definition 3.7. (Unconstrained nodes) Let n be a node (a vertex, edge or face) in the mesh. If there is no node N such that n is constrained by N, we say, that node n is unconstrained.

Unconstrained nodes and element interiors will be those, to whom degrees of freedom will be assigned. Values on constrained nodes are imposed by unconstrained nodes, so there are no degrees of freedom associated with them.

Indirect constrains

Assume we have this situation. Node n is constrained by node N, but in the same time, node N is constrained by node $\bar{N}$. In such situation there will be no degrees of freedom associated with the node N (it is constrained) and values on n will be imposed by values on $\bar{N}$.

Definition 3.8. (Indirect constrains) Consider nodes $n_1, \dots n_k$ such that n_i is constrained by n_{i-1} for $i = 2, \dots, k$. Than we say, that node n_k is *indirectly* constrained by the node n_1.

Remark 3.1. We can see, that faces may not be subjects of indirect constrains. It is clear from the fact, that faces can only be constrained by faces and we can not have three overlapping faces $f_1 \subset f_2 \subset f_3$, since there are always two elements adjacent to a face.

Definition 3.9. Consider vertex v_k. By symbol $N^{Vv}(v_k)$ we denote a set of all vertices v_l, such that v_k is constrained by v_l, no matter whether directly or indirectly. Similarly let us denote $N^{Ev}(v_k)$ and $N^{Fv}(v_k)$ sets of edges and faces that constrain v_k. Let us further define $N^{Ee}(e_k)$ and $N^{Fe}(e_k)$ sets of all edges and faces constraining edge e_k, respectively. Finally, let us define $N^{Ff}(f_k)$ set of all faces constraining face f_k.

Remark 3.2. According to the previous remark, $N^{Ff}(f_k)$ is either empty or contains directly constraining face.

The difference between constraining, directly constrained and indirectly constrained nodes can be seen in Figures 4.10 and 4.11, where a part of a face function from the space H^1 is depicted. Note the importance of tracking indirect constrains in order to preserve continuity of the global basis function.

3.3.3 Dependency of the nodes

In the previous section we introduced several sets, that describe constrains of individual nodes. We can think of it as if a constrained node was dependent on the constraining one. In the following we will describe algorithms, that are used to gather certain information about nodes. We call this process "resolving" nodes. When a node n is to be resolved we need to know, that all the nodes that constrain n have already been resolved. A question arises, whether this will be always possible, i.e. whether it is always true that either all nodes have been resolved (thus the algorithm can stop) or that we can find an unresolved node such that all nodes, that constrain it, have already been resolved.

If this is not true, it would mean, that there is a "circular" dependency of nodes. That there are nodes n_1, n_2,..., n_k, such that n_i constrains n_{i+1} for $i = 1,\ldots,k-1$ and furthermore n_k constrains n_1. We have to show, that this is impossible. The reason lies in the way, how the mesh is refined. At the beginning we have regular mesh with no hanging nodes, therefore no nodes are constrained. When any element is refined, some of the nodes (vertices, edges or faces), that are added, may be constrained by already present nodes. But there is no way, how newly added nodes may constrain those nodes, who were present before addition. This order based on time, when the node is added, creates natural hierarchy of nodes with no cycles. Therefore, in the process of resolving constrains, we can always find a node, that can be resolved (if we are not finished).

3.3.4 Resolving constrains

In this section we want to introduce additional data, which all constrained nodes can be equipped with. We will also define their values. Note, that the algorithm of finding those values is only dependent on the mesh itself and do not depend on concrete choice of basis functions. Of course, if we want it to be useful in what follows, we can use it only for basis functions satisfying restrictions stated in previous sections. Those data defined here will be later used to construct global basis functions in the mesh with hanging nodes.

Definition 3.10. (Vertex-vertex constrains) Let v_k be constrained vertex. Let us define $C^{Vv}(v_k)$ as a set of pairs

$$C^{Vv}(v_k) = \{(v_c, \gamma), v_c \in N^{Vv}(v_k)\},$$

where v_c is constraining vertex and γ is a coefficient, describing, how "strong" is the influence of constraining node on the constrained node, depending on its "distance" in the mesh.

Possibility of defining such coefficient is consequence of presence of linear components in local basis functions, as was explained in Section 3.3.1. Coefficient γ has a similar meaning in all the following definitions. If v_k is not constrained, we define $C^{Vv}(v_k) = \{(v_k, 1)\}$ for simplicity, so that in some of the following definitions we do not have to distinguish between constrained and unconstrained vertices (formally it seems, that unconstrained vertex is constrained by itself).

Definition 3.11. (Edge-vertex constrains) Let v_k be constrained vertex. Let us define $C^{Ev}(v_k)$ as a set of triples

$$C^{Ev}(v_k) = \{(e_c, \gamma, p), e_c \in N^{Ev}(v_k)\},$$

where e_c is constraining edge and $p \in (-1,1)$ is position on the constraining edge such that the value of edge function there will be used to determine values of constrained vertices.

Position $p \in (-1,1)$ is considered in the local parametrization of the edge and has a similar meaning in the following definitions.

Definition 3.12. (Face-vertex constrains) Let v_k be constrained vertex. Let us define $C^{Fv}(v_k)$ as a set of quadruples

$$C^{Fv}(v_k) = \{(f_c, \gamma, p_1, p_2), f_c \in N^{Fv}(v_k)\},$$

where f_c stands for constraining face and p_1 and p_2 are positions (in the local coordinate system) on the constraining face.

Definition 3.13. (Edge-edge constrains) Let e_k be constrained edge. Let us define $C^{Ee}(e_k)$ as a set of triples

$$C^{Ee}(e_k) = \{(e_c, \gamma, (q^1, q^2)), e_c \in N^{Ee}(e_k)\},$$

where e_c is constraining edge and (q^1, q^2) is a part of the constraining edge, to which corresponds edge e_k. The value q_1 represents position of its beginning and q_2 its end, respectively. Again we consider local parametrization, i.e. $q_i \in (-1,1)$.

If edge e_k is not constrained, we define for the sake of simplicity $C^{Ee}(e_k) = \{(e_k, 1, (-1,1))\}$. Here the part $(-1,1)$ consist of endpoints of the local parametrization and thus it corresponds to the whole edge.

Definition 3.14. (Face-edge constrains) Let e_k be constrained edge. Let us define $C^{Fe}(e_k)$ as a set of quintuple

$$C^{Fe}(e_k) = \{(f_c, \gamma, p, (q^1, q^2), d), f_c \in N^{Fe}(e_k)\},$$

where f_c is the constraining face and position p, part (q^1, q^2) and direction d determine the segment on the constraining face f_c, from where e_k is constrained.

This situation is probably the most difficult (and implementation of construction of such global basis functions is further complicated by the necessity of proper handling of orientation of faces and edges involved). For illustration see Figure 3.6 where we can see the propagation of the constrain from the constraining face to the indirectly constrained edge.

Definition 3.15. (Face-face constrains) Let f_k be constrained face. Let us define $C^{Fe}(f_k)$ as a set of triples

$$C^{Ff}(f_k) = \{(f_c, (q_1^1, q_1^2), (q_2^1, q_2^2)), f_c \in N^{Ff}(f_k)\},$$

where f_c is constraining face and (q_1^1, q_1^2), (q_2^1, q_2^2) are parts (in x and y coordinates), of the face f_c occupied by the face f_k.

This situation is quite simple, because there are no indirect constrains of faces. Therefore the set $C^{Ff}(f_k)$ has at most one element, given by constraining face f_c, on which f_k directly lies. Similarly as for vertices and edges, if face f_k is not constrained, we define $C^{Ff}(f_k) = \{(f_k, (-1,1), (-1,1))\}$.

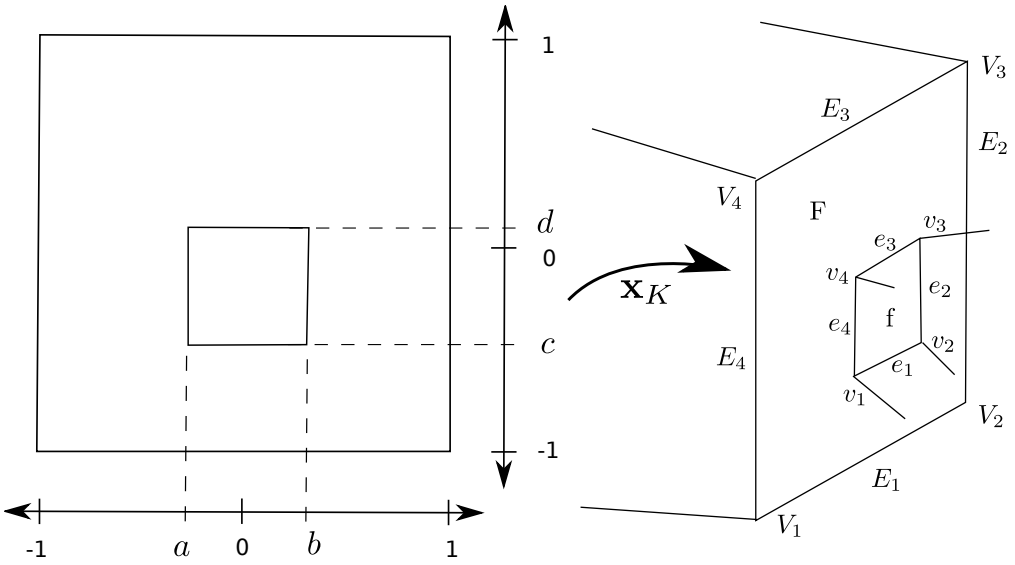


Figure 3.5: Situation on a face. Notice, that $a = x(v_1) = x(v_4) = x(e_4)$, $b = x(v_2) = x(v_3) = x(e_2)$, $c = y(v_1) = y(v_2) = y(e_1)$ and $d = y(v_3) = y(v_4) = y(e_3)$. These coordinates are used as positions p_i and parts (q_j^1, q_j^2).

Having in mind restrictions we imposed on basis functions in Section 3.3.1, let us introduce further notation. By $x(n)$ and $y(n)$ we denote coordinates of image of node n mapped by $\boldsymbol{x}_K{}^{-1}$ to the reference face, as it is shown in Figure 3.5. Here,

by node n we mean any vertex v_i, V_i or edge e_i or E_i. The definition is meaningful for edges as well, because we will always use only that of two coordinates, which does not change for all points of the edge.

Now we define function Γ, which can be viewed as measure of how much does constraining node influence constrained node, depending on its position ("distance" in the mesh). First for a constraining vertex V_i and constrained vertex v_k, $k = 1,2,3,4$ (as depicted in Figure 3.5) we have

$$\Gamma(V_i, v_k) = \frac{2 - |x(V_i) - x(v_k)|}{2} \cdot \frac{2 - |y(V_i) - y(v_k)|}{2},$$

for $i = 1,2,3,4$. Similarly for constraining edge

$$\Gamma(E_i, n) = \frac{2 - |y(E_i) - y(n)|}{2}$$

for $i = 1,3$, where n may represent e_1, e_3 or v_k and

$$\Gamma(E_i, n) = \frac{2 - |x(E_i) - x(n)|}{2}$$

for $i = 2,4$, where n may represent e_2, e_4 or v_k.

Now we are ready to start defining data for constrained nodes. It has to be done in such an order, that the data for the node n will be defined after the data for all nodes on that n depends have already been defined. For discussion of dependency of nodes see Section 3.3.3. In all following definitions we consider situation depicted in Figure 3.5 and use its notation without further mentioning.

Constrained vertices

We start with vertices being constrained by other vertices. If vertices V_i, $i = 1,2,3,4$ from Figure 3.5 are unconstrained, each vertex v_k is constrained by V_i only. But if it is not the case, v_k are constrained by all vertices constraining vertices V_i. It may happen, that more of V_i are constrained by the same vertex V_C. In such case we add up contributions of V_C "through" all V_i:

$$C^{Vv}(v_k) = \biguplus_{i=1}^{4} \Big\{ (v_c, \gamma') : \ (v_c, \gamma) \in C^{Vv}(V_i), \gamma' = \Gamma(V_i, v_k)\ \gamma \Big\}, \tag{3.1}$$

where $\uplus$ works as a set union with the exception of the case where there are two or more pairs (V_c, γ_1), (V_c, γ_2),..., (V_c, γ_n) to be included. Then instead of including them all we include one pair $(V_c, \gamma_1 + \gamma_2 + \cdots + \gamma_n)$ only.

For vertices constrained by edges the situation is more complicated. First, we have to store not only coefficient, but also position on constraining edge, from where the vertex is constrained. Second is that we have to take into consideration not only edges E_i and edges constraining those edges, but also edges constraining vertices V_i, as we can see from the definition:

$$C^{Ev}(v_k) = \biguplus_{i=1}^{4}\Big\{(e_c,\gamma',p):\ (e_c,\gamma,p)\in C^{Ev}(V_i),\gamma'=\Gamma(V_i,v_k)\,\gamma\Big\}\uplus$$
$$\uplus\ \biguplus_{i=1}^{4}\Big\{(e_c,\gamma',p):\ (e_c,\gamma,(Q^1,Q^2))\in C^{Ee}(E_i),$$
$$p=\frac{Q^2-Q^1}{2}\mathrm{coord}(v_k)+\frac{Q^1+Q^2}{2},\ \gamma'=\Gamma(E_i,v_k)\,\gamma\Big\}, \quad (3.2)$$

this time we add up contributions with the same e_c and p (similarly as in the previous case). We set $\mathrm{coord}(v_k)=x(v_k)$ for $i=1,3$ (constraining edges E_1 and E_3 parallel with x on the reference domain) and $\mathrm{coord}(v_k)=y(v_k)$ for $i=2,4$ (constraining edges E_2 and E_4 parallel with y). Calculation of p has a meaning of finding a position on the edge *constraining* edge E_i and it is calculated according to which part of that edge corresponds E_i to (given by (Q^1,Q^2)). If E_i is not constrained, then $(Q^1,Q^2)=(-1,1)$ and therefore $p=\mathrm{coord}(v_k)$.

Now let us proceed to the situation of vertices constrained by faces. Here we have to take into account faces constraining vertices V_i, faces constraining edges E_i and finally face F itself. This time we need to store two positions p_1 and p_2, that describe coordinates on constraining face, from the value of constraining face function will be taken to calculate the value of the constrained vertex (also according to the coefficient γ.

$$C^{Fv}(v_k) = \biguplus_{i=1}^{4}\Big\{(f_c,\gamma',p_1,p_2):\ (f_c,\gamma,p_1,p_2)\in C^{Fv}(V_i),\gamma'=\Gamma(v_k,V_i)\,\gamma\Big\}\uplus$$
$$\biguplus_{i=1}^{4}\Big\{(f_c,\gamma,p_1,p_2):\ (f_c,\gamma,P,(Q^1,Q^2),d)\in C^{Fe}(E_i),\gamma'=\Gamma(v_k,E_i)\,\gamma,$$
$$p_1=P,p_2=\frac{Q^2-Q^1}{2}\mathrm{coord}(v_k)+\frac{Q^1+Q^2}{2}\text{ for }d=D_x\text{ and}$$
$$p_1=\frac{Q^2-Q^1}{2}\mathrm{coord}(v_k)+\frac{Q^1+Q^2}{2},p_2=P\text{ for }d=D_y\Big\}\uplus$$
$$\uplus\ \Big\{(F,1,x(v_k),y(v_k))\Big\}, \quad (3.3)$$

where again $\mathrm{coord}(v_k)=x(v_k)$ for $i=1,3$ and $\mathrm{coord}(v_k)=y(v_k)$ for $i=2,4$. Symbols D_x and D_y stand for the directions V_1V_2 and V_2V_3, respectively. The

definition may seem rather complicated, but it has the same principle as those preceding. First, faces constraining vertices V_i are taken into account. They all will also constrain vertices v_k, we only have to adjust coefficient γ according to the position of v_k on F. For faces f_c constraining edges E_i we have to distinguish in which direction d on the constraining face f_c lies the edge, which the constrain propagates from. According to that, appropriate position on the constraining face is calculated. Finally we have to consider constraining face F on which v_k lies. Again we add up contributions with equal f_c, p_1 and p_2.

Constrained edges

First let us address situation of edges constrained by another edges. The only thing that contributes to a list of constrains of e_k are edges E_i if unconstrained or those edges that constrain them:

$$C^{Ee}(e_k) = \biguplus_{i \in i_E} \Big\{ (e_c, \gamma', (q^1, q^2)) : \ (e_c, \gamma, (Q^1, Q^2)) \in C^{Ee}(E_i), \\ q^1 = \frac{Q^2 - Q^1}{2} \mathrm{coord}_1(e_k) + \frac{Q^1 + Q^2}{2}, \ q^2 = \frac{Q^2 - Q^1}{2} \mathrm{coord}_2(e_k) + \frac{Q^1 + Q^2}{2}, \\ \gamma' = \Gamma(e_k, E_i)\, \gamma \Big\}, \quad (3.4)$$

where $i = \{1,3\}$, $\mathrm{coord}_1(e_k) = x_1(e_k)$ and $\mathrm{coord}_2(e_k) = x_2(e_k)$ for $k = 1,3$ and $i = \{2,4\}$, $\mathrm{coord}_1(e_k) = y_1(e_k)$ and $\mathrm{coord}_2(e_k) = y_2(e_k)$ for $k = 2,4$, respectively. The important task is to determine, to what part (q^1, q^2) of the constraining edge e_c would the edge e_k correspond to. It is determined using the part (Q^1, Q^2) (to which E_i corresponds in respect to e_c) and coordinates of e on the face F. Note that if E_i is not constrained, then $(Q^1, Q^2) = (-1, 1)$ and thus $(q^1, q^2) = (\mathrm{coord}_1(e_k), \mathrm{coord}_2(e_k))$. As usually we add up contributions with the same e_c and (q^1, q^2).

Information of faces constraining edges is more complex, since we have to store both position and part on a constraining face and also information, in which direction edge lies (with respect to the constraining face).

$$C^{Fe}(e_k) = \biguplus_{i \in i_E} \Big\{ (f_c, \gamma', p, (q^1, q^2), d) : \ (f_c, \gamma, p, (Q^1, Q^2), d) \in C^{Fe}(E_i), \\ q^1 = \frac{Q^2 - Q^1}{2} \mathrm{coord}_1(e_k) + \frac{Q^1 + Q^2}{2}, \ q^2 = \frac{Q^2 - Q^1}{2} \mathrm{coord}_2(e_k) + \frac{Q^1 + Q^2}{2}, \\ \gamma' = \Gamma(e_k, E_i)\, \gamma \Big\} \ \uplus \ \Big\{ (F, 1, \mathrm{coord}'(e_k), (\mathrm{coord}_1(e_k), \mathrm{coord}_2(e_k)), D) \Big\}, \quad (3.5)$$

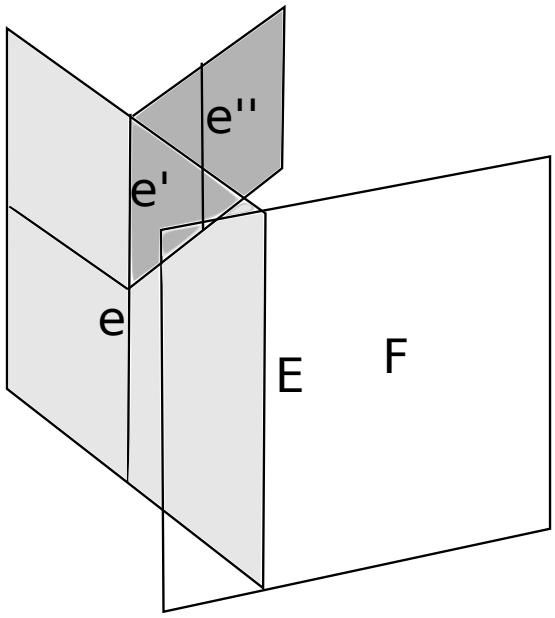


Figure 3.6: Illustration of indirect constrains. Suppose face F is unconstrained. Contributions of E to $C^{Ee}(e'')$ and F to $C^{Fe}(e'')$ depends on how reference element is mapped to physical elements involved. But if we assume mapping which is most "natural" from our point of view, it can be said, that $(E, 1/4 \cdot 1/2 \cdot 1/2, (0,1)) \in C^{Ee}(e'')$ and $(F, 1/2 \cdot 1/2, -1/2, (0,1), D_y) \in C^{Fe}(e'')$.

where $i = \{1,3\}$, $\text{coord}_1(e_k) = x_1(e_k)$, $\text{coord}_2(e_k) = x_2(e_k)$, $\text{coord}'(e_k) = y(e_k)$ and $D = D_x$ for $k = 1,3$ and $i = \{2,4\}$, $\text{coord}_1(e_k) = y_1(e_k)$, $\text{coord}_2(e_k) = y_2(e_k)$ $\text{coord}'(e_k) = x(e_k)$ and $D = D_y$ for $k = 2,4$, respectively. The notation is rather complicated, since we have to distinguish in which direction the edge lies and adjust variables accordingly. Important thing is, that the direction d is meant with respect to the constraining face. It is set from F for a direct constrain, but for indirect constrains through edges E_i it is copied without any change and does not depend on the direction of e_k with respect to F. Contributions with the same f_c, p, (q^1, q^2) and d are added up.

Constrained faces

This final step is simple, since there are no indirect constrains of faces. A face f can be only constrained if it lies on a bigger face F as depicted in Figure 3.5:

$$C^{Ff}(f) = \Big\{ (F,\ (x_1(f), x_2(f)),\ (y_1(f), y_2(f)) \Big\}$$

This concludes rather technical definitions of the constrain information of nodes. It will be used in the following chapters to design the global basis functions.

3.4 Orientation handling

Proper treatment of orientations is a surprisingly difficult part of the implementation of the higher-order basis functions for 3D meshes. Without good mathematical preparation it is almost impossible to have all situations correct in the computer code. From that reason we want to give a detailed description of the way how orientations are dealt with.

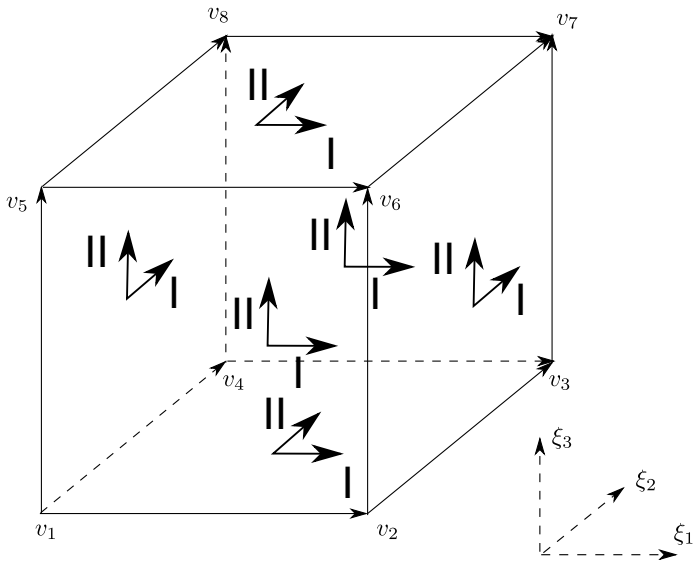


Figure 3.7: Local orientation of edges and faces on the reference domain B.

Global basis functions in the physical mesh are constructed by "gluing" images of shape functions defined on the reference domain and mapped by the reference mapping, which was described in Section 2.3.2. The problem is, that there are more possibilities how a reference cube may be mapped to the element in the mesh, depending on which vertices in the mesh are images of which vertices of the reference domain. And since higher-order shape functions may be unsymmetrical, we have to make sure to use correct mapping in order to make neighboring parts of the global basis functions "match" together (in order to satisfy conformity requirements of the finite element space).

We start with the reference element B. All edges and faces has to be oriented. We

have chosen orientation depicted in Figure 3.7. All edges are oriented in direction of that axis ξ_1, ξ_2, ξ_3, with which is the edge parallel. For faces, the situation is more complicated. There are two axis parallel to each face, orientations I and II are taken such that I is in the direction of the axis with the lower index.

3.4.1 Orientation of edges

Each edge in the mesh has to be equipped with unique orientation. It would be possible to store it as an additional information, but it is not necessary if we define it using unique indices $i(v)$ of vertices v in the mesh.

Figure 3.8: Global orientation of edge v_1v_2 in the physical mesh. Example of situation where $i(v_1) < i(v_2)$ and therefore the orientation is from v_1 to v_2.

Definition 3.16. (Edge orientation) Consider edge $e \in \mathscr{T}_{h,p}$ with endpoints v_1, v_2. Suppose $i(v_1) < i(v_2)$. Then we say, that e is oriented in the direction from v_1 to v_2.

Now suppose we want to construct global basis functions. We do it by mapping local basis function from the reference domain to elements adjacent to the edge e. But in order to ensure conformity requirements, it may be necessary to "adjust" local basis function in such way, that after mapping to the mesh, images "match" on elements involved.

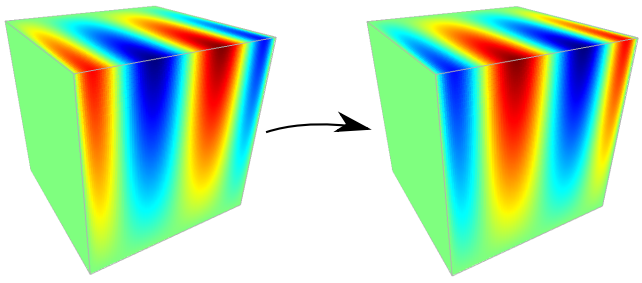


Figure 3.9: Illustration of construction of edge orientation change map. The coordinate in the direction of the edge is "turned around".

The adjustment will be performed by application of mapping, that switches direction of one of the coordinates, if necessary. This mapping is defined in the following:

Definition 3.17. (Orientation mapping) Let us assume element K with it's edge e. The mapping corresponding to the element, $\boldsymbol{x}_K$, will map local orientation from the reference domain (as shown in Figure 3.7) to e. If this mapped orientation is in the same direction as the global orientation of e, as defined in Definition 3.16, we define $\boldsymbol{x}_o^{K,e}$ as identity. In the other case, we define

- $\boldsymbol{x}_o^{K,e}(\xi_1,\xi_2,\xi_3) = (-\xi_1,\xi_2,\xi_3)$ for $\boldsymbol{x}_K^{-1}(e) \in \{e_1,e_3,e_9,e_{11}\}$
- $\boldsymbol{x}_o^{K,e}(\xi_1,\xi_2,\xi_3) = (\xi_1,-\xi_2,\xi_3)$ for $\boldsymbol{x}_K^{-1}(e) \in \{e_2,e_4,e_{10},e_{12}\}$
- $\boldsymbol{x}_o^{K,e}(\xi_1,\xi_2,\xi_3) = (\xi_1,\xi_2,-\xi_3)$ for $\boldsymbol{x}_K^{-1}(e) \in \{e_5,e_6,e_7,e_8\}$

The meaning of the definition is, that if local and global orientation of the edge do not match, we "switch" the corresponding coordinate to fix it.

Remark 3.3. Note, that the definition of orientation mapping depends not only on the edge e in the physical mesh, but also on the element, from which we "look" on the edge. Indeed, this is the purpose of the orientation handling. The same edge has to be treated differently, depending on from which element we "look" on it.

3.4.2 Orientation of faces

Now let us proceed to faces. Orientation of faces of the reference domain is depicted in Figure 3.7. Orientation of faces in the physical mesh is given by the following definition and shown in Figure 3.10.

Definition 3.18. (Face orientation) Consider face $f \in \mathscr{T}_{h,p}$ given by vertices v_1, v_2, v_3, v_4 of the physical mesh and assume, that for global indices $i(v_k)$ of vertices v_k, $k = 1,\ldots,4$, the following properties hold: $i(v_1) = \min(i(v_1),i(v_2),i(v_3),i(v_4))$ and $i(v_2) < i(v_4)$. Than we say, that the first orientation I' goes from v_1 to v_2 and the second orientation II' from v_1 to v_4. Orientation of face f is given by combination of I' and II'.

When orientation of reference face (I,II) is mapped to the physical face, there are eight possible situations of (I,II) and (I',II'). That is much more complicated than for edges, where there were only two possibilities - orientation was either the same, or the opposite. Possible situations are shown in Figure 3.11.

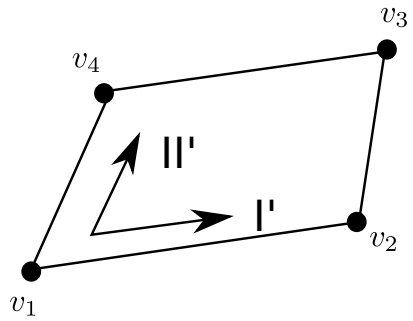


Figure 3.10: Global orientation of a face $v_1v_2v_3v_4$ in the physical mesh. Here $i(v_1) = \min(v_1, v_2, v_3, v_4)$ and $i(v_2) < i(v_4)$.

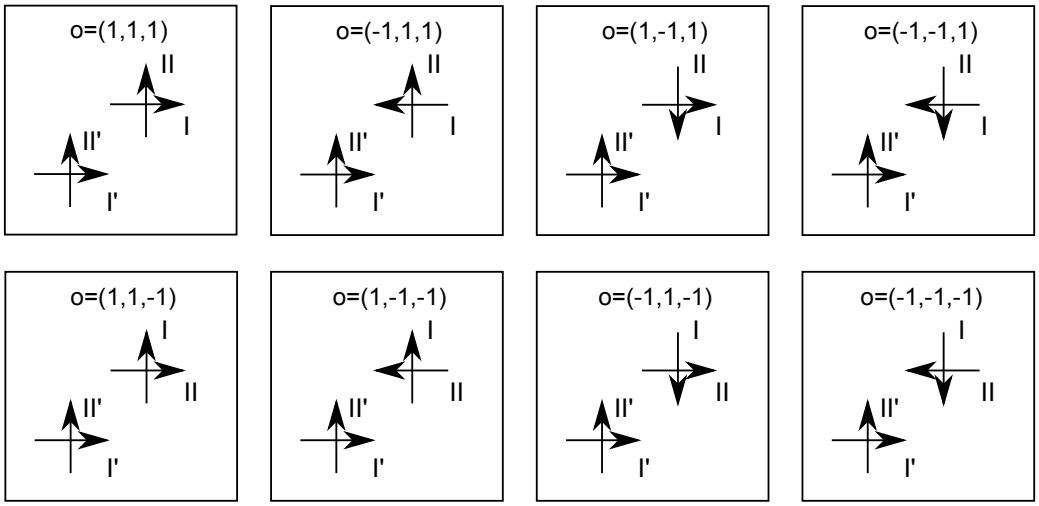


Figure 3.11: Possible combinations of global (I', II') and local (I, II) orientation of a face.

Similarly as for edges, we have to define mappings that will "adjust" local basis functions to different orientations. As we can see from Figure 3.11, the orientation of the face can be described by three orientation flags o_1, o_2, o_3, where $o = (o_1, o_2, o_3)$. Each flag may be either 1 or -1. Meaning of the flags is the following. If $o_1 = -1$, we have to "reverse" the direction of coordinate ξ_i, the first of the two directions of the face (the one with the lower index i). Flag o_2 has a similar meaning, if $o_2 = -1$ we have to "reverse" direction in the second coordinate of the face. Finally, the third flag o_3 has the meaning of swapping of the two face coordinates. Described mappings are shown in Figure 3.12.

In the following we elaborate this idea in more rigorous way. First let us define

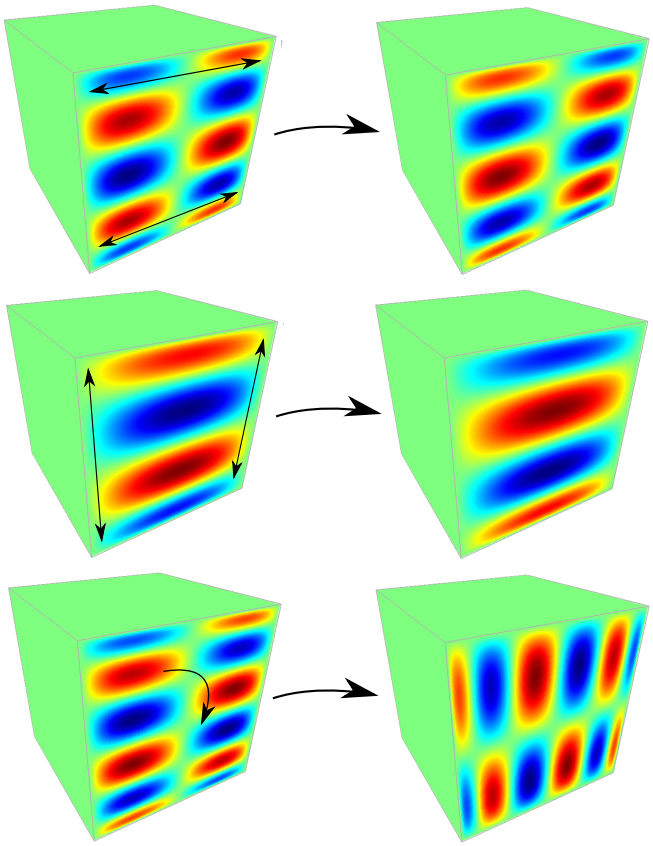


Figure 3.12: An illustration of the construction of the orientation mapping. The top and the middle figures illustrate the switching of the direction in the two coordinates of the face, the bottom figure illustrates swapping of them.

simple mappings, that we will use to construct the final orientation map.

Definition 3.19. Let us define mappings, that change the direction of one coordinate:

$$\begin{aligned} \boldsymbol{x}_1(\xi_1,\xi_2,\xi_3) &= (-\xi_1,\xi_2,\xi_3), \\ \boldsymbol{x}_2(\xi_1,\xi_2,\xi_3) &= (\xi_1,-\xi_2,\xi_3), \\ \boldsymbol{x}_3(\xi_1,\xi_2,\xi_3) &= (\xi_1,\xi_2,-\xi_3) \end{aligned}$$

and mappings, that swap two coordinates:

$$\begin{aligned} \boldsymbol{x}_{1,2}(\xi_1,\xi_2,\xi_3) &= (\xi_2,\xi_1,\xi_3), \\ \boldsymbol{x}_{2,3}(\xi_1,\xi_2,\xi_3) &= (\xi_1,\xi_3,\xi_2), \\ \boldsymbol{x}_{3,1}(\xi_1,\xi_2,\xi_3) &= (\xi_3,\xi_2,\xi_1). \end{aligned}$$

Now we can proceed to a concrete element K and its face f. Depending on what face of the reference domain corresponds f to, we select appropriate mappings.

Definition 3.20. Let us assume element K and its reference mapping $\boldsymbol{x}_K$. Now let us define directions of faces on the reference cube B

- $d_1 = 2, d_2 = 3$ for $\boldsymbol{x}_K^{-1}(f) \in \{s_1, s_2\}$
- $d_1 = 1, d_2 = 3$ for $\boldsymbol{x}_K^{-1}(f) \in \{s_3, s_4\}$
- $d_1 = 1, d_2 = 2$ for $\boldsymbol{x}_K^{-1}(f) \in \{s_5, s_6\}$

as shown in Figure 2.2. With this in hand we can define atomic mappings

$$\boldsymbol{x}_{o,1}^{K,f} = \boldsymbol{x}_{d_1}, \qquad \boldsymbol{x}_{o,2}^{K,f} = \boldsymbol{x}_{d_2}, \qquad \boldsymbol{x}_{o,3}^{K,f} = \boldsymbol{x}_{d_1,d_2},$$

that will be used to construct the final mapping, as follows.

Definition 3.21. (Orientation mapping) Let us assume an element K and its face f. Let (I, II) be orientation mapped from the reference cube B (as shown in Figure 3.7) and (I', II') be the global orientation of the face f (as defined in Definition 3.18). Let $o = (o_1, o_2, o_3)$ be appropriate orientation triple, as defined in Figure 3.11. Let us define orientation mapping

$$\boldsymbol{x}_o^{K,f} = L\big(\boldsymbol{x}_{o,1}^{K,f}, o_1\big) \circ L\big(\boldsymbol{x}_{o,2}^{K,f}, o_2\big) \circ L\big(\boldsymbol{x}_{o,3}^{K,f}, o_3\big),$$

where $L\big(\boldsymbol{x}, o_i\big) = \boldsymbol{x}$ for $o_i = -1$ and $L\big(\boldsymbol{x}, o_i\big) = I$ for $o_i = 1$, where I is an identity mapping. In other words, we only want to apply those orientation adjustments $\boldsymbol{x}_{o,i}^{K,f}$, $i = 1, 2, 3$, for which $o_i = -1$.

In this rather long process we first selected appropriate mappings, according to which reference face is our face f image of and then we included or did not include them in construction of the final mapping $\boldsymbol{x}_o^{K,f}$, taking into consideration orientation flags o_1, o_2 and o_3.

Remark 3.4. Again we stress out, that just defined orientation mapping may be different for the same face f, when considered from two different elements, as is suggested by the notation itself.

CHAPTER

FOUR

HP-FEM IN 3D FOR ELLIPTIC PROBLEMS

Many physical phenomena (distribution of electrostatic potential, stationary heat conduction, etc.) can be modeled by second order elliptic partial differential equations. Thanks to this and their relative simplicity, the elliptic problems are the most often investigated. There is a huge amount of well-known sources dealing with elliptic problems and addressing their properties, so there is no need to list them. Publications dedicated to *hp*-FEM method for elliptic problems include (but are not at all restricted to) works [3], [4], [5], [24], [18], [17], [46], [48] and others.

In this chapter we want to apply geometrical properties and definitions discussed in the previous chapter on solution of elliptic problems. The handling of arbitrary-level hanging nodes is inspired by [49], where the process is described for two dimensional problems. It has to be stated, however, that the algorithmization is much more complicated in 3D. Although the algorithm presented in [49] is very smart, its elegance is allowed by the fact, that the whole setting in 2D is relatively simple, when compared to 3D case. For example, in 2D there are only few types of dependency of vertices and edges, that have to be handled. Orientation of edges is also not a big issue in 2D, while in 3D it becomes serious problem even for regular meshes and starts to be very unpleasant for meshes with hanging nodes. Although the principle is the same, number of different situations and combinations is incomparable. That is why we need more elaborate and systematic approach in 3D, which is presented in this (and the previous) chapter.

In the following text we first introduce a model elliptic problem and briefly revise its properties. As a next step we define higher-order shape functions on the reference cube. We use Lobatto functions for this purpose. The basis of our approach

including the definition of higher-order shape functions on the reference domain, reference mapping technique and others are described in book [48], which is a fundamental reference for this work and for the computer implementation.

The main part of this chapter is dedicated to the issue of construction of higher-order global basis functions on meshes with arbitrary-level hanging nodes. We have to overcome all suggested complications in order to design the algorithm, that will "glue" the contributions to the global basis function from different elements properly in order to satisfy conformity requirements of the space.

4.1 Model elliptic problem

To keep things simple, we will restrict ourselves to the following model elliptic problem. Of course, the presented theory and also the computer implementation would work the same way, even for a more general setting.

4.1.1 Classical formulation

Let us consider a liner second-order elliptic problem of finding $u \in C^1(\overline{\Omega}) \cap C^2(\Omega)$ such that

$$-\nabla \cdot \left(\tilde{a}(\boldsymbol{x})\nabla u\right) = f \quad \text{in } \Omega, \tag{4.1}$$

$$u = g_{\mathrm{D}} \quad \text{on } \Gamma_{\mathrm{D}}, \tag{4.2}$$

$$\frac{\partial u}{\partial \boldsymbol{\nu}} = g_{\mathrm{N}} \quad \text{on } \Gamma_{\mathrm{N}}, \tag{4.3}$$

where $\Omega \subset \mathbb{R}^3$, is a bounded domain with Lipschitz boundary, $\boldsymbol{\nu}$ is the unit outer normal to the boundary $\partial\Omega$, the sets Γ_{D} and Γ_{N} are relatively open in $\partial\Omega$, $\overline{\Gamma}_{\mathrm{D}} \cap \overline{\Gamma}_{\mathrm{N}} = \emptyset$, and $\overline{\Gamma}_{\mathrm{D}} \cup \overline{\Gamma}_{\mathrm{N}} = \partial\Omega$.

4.1.2 Weak formulation

We need to introduce so called *Dirichlet lift* G, a function, that equals the boundary condition on the Dirichlet boundary and that is sufficiently smooth on the domain Ω:

$$G = g_D \quad \text{on } \Gamma_{\mathrm{D}}, \tag{4.4}$$

$$G \in C^2(\Omega) \cap C(\overline{\Omega}). \tag{4.5}$$

The particular choice of the Dirichlet lift G is not important, as shown later, as long as it fulfills the above conditions.

Instead of seeking function u satisfying equations (4.1)–(4.3) we are looking for a function $\overline{u} = u - G$. We can rewrite equations (4.1)–(4.3) using new unknown function $\overline{u}$ as

$$-\nabla \cdot \big(\tilde{a}(\boldsymbol{x})\nabla(\overline{u}+G)\big) = f \quad \text{in } \Omega, \tag{4.6}$$

$$u = 0 \quad \text{on } \Gamma_{\mathrm{D}}, \tag{4.7}$$

$$\frac{\partial(\overline{u}+G)}{\partial \boldsymbol{\nu}} = g_{\mathrm{N}} \quad \text{on } \Gamma_{\mathrm{N}}. \tag{4.8}$$

To obtain the weak formulation we multiply the equation (4.6) by a smooth test function v and integrate over the domain Ω:

$$\int_\Omega -\nabla \cdot (\tilde{a}(\boldsymbol{x})\nabla(\overline{u}+G))v \,\mathrm{d}\boldsymbol{x} = \int_\Omega f v \,\mathrm{d}\boldsymbol{x}.$$

By using the Green's theorem and the linearity of the ∇ operator we obtain the form

$$\int_\Omega \tilde{a}(\boldsymbol{x})\nabla\overline{u}\cdot\nabla v \,\mathrm{d}\boldsymbol{x} = \int_\Omega f v \,\mathrm{d}\boldsymbol{x} - \int_\Omega \tilde{a}(\boldsymbol{x})\nabla G\cdot\nabla v \,\mathrm{d}\boldsymbol{x} + \int_{\Gamma_{\mathrm{N}}} \tilde{a}(\boldsymbol{x}) g_N v \,\mathrm{d}\boldsymbol{s}. \tag{4.9}$$

As it is usual, we will use the notation of the bilinear and linear forms:

$$a(\overline{u}, v) = \int_\Omega \tilde{a}(\boldsymbol{x})\nabla\overline{u}\cdot\nabla v \,\mathrm{d}\boldsymbol{x}, \tag{4.10}$$

$$l(v) = \int_\Omega f v \,\mathrm{d}\boldsymbol{x} + \int_{\Gamma_{\mathrm{N}}} \tilde{a}(\boldsymbol{x}) g_N v \,\mathrm{d}\boldsymbol{s}. \tag{4.11}$$

Formulation of the problem

Now we are ready to formulate the weak formulation of the problem. We do it by using functions from appropriate Sobolev spaces in the equation (4.9).

Definition 4.1. (Weak solution) Let $V = \{v \in H^1(\Omega), v|_{\Gamma_{\mathrm{D}}} = 0\}$, $G \in H^1(\Omega)$, $G|_{\Gamma_{\mathrm{D}}} = g_D$ in the sense of traces, $f \in L^2(\Omega)$, $\tilde{a} \in L^\infty(\Omega)$ and the function $\overline{u} \in V$ satisfies

$$a(\overline{u}, v) = l(v) - a(G, v) \quad \forall v \in V. \tag{4.12}$$

Than the function $u = \overline{u} + G$ is called the *weak solution* of problem (4.1)–(4.3).

Properties of the bilinear form a and the solution

Theorem 4.1. (Energetic norm) *The function $\left(a(v,v)\right)^{\frac{1}{2}}$ defines a norm in the space V which is equivalent to the standard norm in the space H^1.*

This norm is called *energetic* norm and is denoted by $\|\cdot\|_e$.

Theorem 4.2. *The solution $\overline{u}$ exists and is unique.*

Proof. The bilinear form a and the linear form l satisfy the assumptions of the Lax-Milgram lemma. □

Theorem 4.3. *The solution u is independent on the choice of the Dirichlet lift G*

Proof. It follows simply by considering two different Dirichlet lifts and corresponding weak solutions and subtracting them. □

4.2 Higher-order shape functions

As it was mentioned before, the advantage of higher-order global basis functions is the ability of good approximation of smooth solutions using significantly less degrees of freedom. First, let us define a set of higher-order shape functions on the reference cube. Those functions will be used to construct global basis functions in the physical mesh. A transformation used to map the functions from the reference domain to the physical mesh is called the reference mapping and it has already been introduced in Section 2.3.2.

A terminology may differ, so let us introduce this convention. When we speak about basis functions on the reference domain, we refer to them as *shape* functions or *local basis* functions. On the other hand basis functions in the physical mesh are mentioned as *global basis* functions or just *basis* functions.

4.2.1 Shape functions on reference domain

In the following we describe the set of shape functions, that are considered on reference cube, that was introduced in Section 2.3.1. As we use the hierarchic basis (which has been introduced in Section 2.4.2), each shape function is related to one of the nodes. There are 8 vertex, 12 edge, 6 face and 1 interior nodes. Most of the notation used is taken from [48].

Order of the element

There are several advantages when using hexahedral elements. One of them is the possibility of easy division into two sub-elements, which are hexahedral again. The other advantage is the possibility to use different polynomial degree in each direction. For this purpose we consider three local orders of approximation $p^{b,1}$, $p^{b,2}$, $p^{b,3}$ in directions of ξ_1, ξ_2, ξ_3. Element orders may vary from 1 up to 10 in our code, even though there is no principle upper bound and number 10 has been chosen rather arbitrary. From our simulation it seems, however, that this is sufficiently high.

Similarly, we consider two local orders on each face s_i, $i \in \{1,\dots,6\}$, denoted by $p^{s_i,1}$, $p^{s_i,2}$. Those orders are in directions of a local two-dimensional system of coordinates on each face, that matches corresponding pair of three-dimensional coordinates ξ_1, ξ_2, ξ_3, always in lexicographic order. The same applies for edges. The order of the edge e_i is denoted by p^{e_i}.

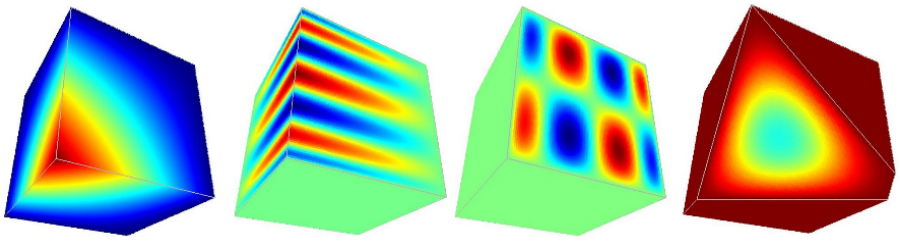


Figure 4.1: Vertex, edge, face and bubble local basis functions on the reference cube. Note, that the last cube has one of the vertices cut off, since bubble functions are local to the element interior and vanish on its boundary.

Polynomial space

Now we are ready to define the polynomial space on the reference element B:

$$W_B^1 = \{w \in Q_{p^{b,1},p^{b,2},p^{b,3}};\ w|_{s_i} \in Q_{p^{s_i,1},p^{s_i,2}},\ i = 1,\dots,6,\\ w|_{e_j} \in P_{p^{e_j}},\ j = 1,\dots,12\}. \quad (4.13)$$

Here

$$Q_{p_1,p_2,p_3} = \mathrm{span}\{\xi_1^{i_1}\xi_2^{i_2}\xi_3^{i_3};\ -1 \le \xi_1,\xi_2,\xi_3 \le 1,\\ i_1 = 0,\dots,p_1,\ i_2 = 0,\dots,p_2,\ i_3 = 0,\dots,p_3\} \quad (4.14)$$

and

$$Q_{p_1,p_2} = \text{span}\{\zeta_1^{i_1}\zeta_2^{i_2};\ i_1 = 0,\dots,p_1,\ i_2 = 0,\dots,p_2\}, \tag{4.15}$$

where ζ_1, ζ_2 are local coordinates on the appropriate face of the reference domain B. Finally

$$P_p = \text{span}\{\eta^i;\ i = 0,\dots,p\}, \tag{4.16}$$

where η is a local coordinate on the appropriate edge.

4.2.2 Construction of local basis functions

In this section we describe the exact way how to construct the local basis functions. Thank to the product geometry of the reference domain, it is natural to construct them as products of 1D polynomial functions. Lobatto shape functions have been chosen for their good conditioning properties.

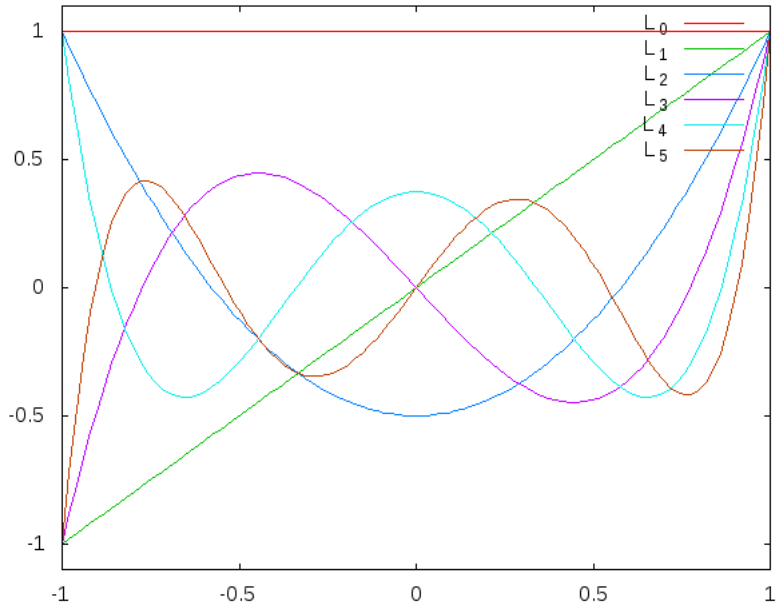


Figure 4.2: Legendre polynomials.

One-dimensional Lobatto shape functions

First let us remind the definition of Legendre polynomials. There are many ways to define them, we chose a recurrent definition.

Definition 4.2. Legendre polynomials are defined as follows:

$$\begin{aligned} L_0(x) &= 1, \\ L_1(x) &= x, \\ L_k(x) &= \frac{2k-1}{k} x L_{k-1}(x) - \frac{k-1}{k} L_{k-2}(x), \qquad k \geq 2. \end{aligned}$$

Those polynomials are well known and have many interesting properties. Several first Legendre polynomials are depicted in Figure 4.2.

Next let us define the Lobatto shape functions as integrals of previously defined legendre polynomials.

Definition 4.3. Lobatto polynomials are defined as follows:

$$\begin{aligned} l_0(x) &= \frac{1-x}{2}, \\ l_1(x) &= \frac{1+x}{2}, \\ l_k(x) &= \frac{1}{\sqrt{2/(2k-1)}} \int_{-1}^{x} L_{k-1}(\xi)\, d\xi, \qquad k \geq 2, \end{aligned}$$

where L_k denote Legendre polynomials.

It follows from their orthogonality that all functions l_k, $k \geq 2$, vanish at both endpoints of the interval $[-1,1]$, while l_0 and l_1 vanish at one endpoint and equal to 1 at the other. This fact is used when dividing basis functions into categories. Orthogonality of Legendre polynomials ensures good conditioning properties of the Lobatto basis. In Hermes we use Lobatto functions $l_0, \ldots, l_{10}$.

Local basis functions

To define the set Σ_B^1 of degrees of freedom we have to describe construction of concrete basis of the space W_B^1. It has to include functions associated to vertices, edges, faces and element interiors. It is in accordance with the nodal approach and conformity requirements of the space H^1.

- *Vertex functions* φ^{v_i}, $i = 1, \ldots, 8$ are defined as trilinear functions in the form

$$\varphi^{v_i} = l_{d_1}(\xi_1) l_{d_2}(\xi_2) l_{d_3}(\xi_3), \tag{4.17}$$

  where $d_j = 0, 1$ and $j = 1, 2, 3$. Thus, there are eight possible combinations corresponding to the eight vertices. For the construction of vertex functions

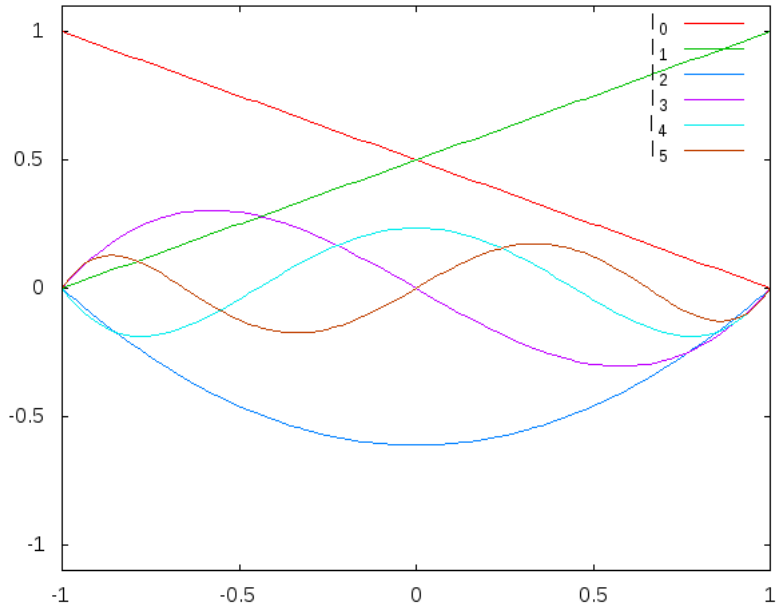


Figure 4.3: Lobatto shape functions l_k for $k = 0,\ldots,5$. Note, that for $k \geq 2$ l_k vanishes at endpoints of the interval $[0,1]$.

we use only first two lobatto functions l_0, l_1, which are linear and equal to 1 at one endpoint of the interval $[-1,1]$ and to 0 at the other. Their function values in vertices are 0 with the exception of one vertex, where all three composing functions have value 1. We say that vertex function is associated or related to the vertex, where it's function value is 1. Traces of vertex functions are linear on edges and bilinear on faces. Vertex functions form basis of the lowest-order element.

- *edge functions* $\varphi_k^{e_i}$, $i = 1,\ldots,12$, $k = 2,\ldots,p_{e_i}$ are defined in the form

$$\varphi_k^{e_i} = l_{d_1}(\xi_1) l_{d_2}(\xi_2) l_{d_3}(\xi_3). \tag{4.18}$$

Edge functions corresponding to the edge e_i are constructed in the following way. Let ξ_l be axis parallel to the edge e_i. Then component $d_l = k$, may be chosen from $2,\ldots,p_{e_i}$. Other two indices are either 0 or 1 (there are 4 combinations corresponding to 4 different edges parallel to ξ_l. The trace of edge function $\varphi_k^{e_i}$ equals to the 1D Lobatto function on the edge e_i and is zero on all other edges and in all vertices. Edge functions are present only for such edges e_j, for which $p^{e_j} \geq 2$ and their number is $p^{e_j} - 1$.

- *face functions* $\varphi^{s_i}_{k_1,k_2}$, $i = 1,\dots,6$, $k_1 = 2,\dots,p^{s_i,1}$, $k_2 = 2,\dots,p^{s_i,2}$ corresponding to the face s_i are constructed in such way, that the trace on face s_i is the product of lobatto functions l_{k_1} and l_{k_2} in directions parallel to the face. The remaining index d_l, where ξ_l is perpendicular to the face s_i, is 0 or 1 (there are two faces perpendicular to ξ_l). The formula is:

$$\varphi^{s_i}_{k_1,k_2} = l_{d_1}(\xi_1) l_{d_2}(\xi_2) l_{d_3}(\xi_3). \tag{4.19}$$

  A face function is zero on all other faces, on all edges and in all vertices. Face functions associated with the face s_i are present only if $p^{s_i,1} \geq 2$ and $p^{s_i,2} \geq 2$ and their number is $(p^{s_i,1}-1)(p^{s_i,2}-1)$.

- *bubble functions* $\varphi^{b}_{k_1,k_2,k_3}$, $k_l = 2,\dots,p^{b,l}$, $l = 1,\dots,3$ are equal to zero in all vertices, edges and faces and complete the basis of W^1_B:

$$\varphi^{b}_{k_1,k_2,k_3} = l_{d_1}(\xi_1) l_{d_2}(\xi_2) l_{d_3}(\xi_3). \tag{4.20}$$

  Bubble functions are present if all $p^{b,1}$, $p^{b,2}$ and $p^{b,3}$ are at least 2 and their number is $(p^{b,1}-1)(p^{b,2}-1)(p^{b,3}-1)$.

All types of local basis functions are illustrated in Figure 4.1.

Remark 4.1. One essential feature of this distinction of local basis function is that all vertex functions vanish in all vertices but one, all edge functions vanish in all vertices and all edges but one and all face functions vanish in all vertices, all edges and all faces but one. Bubble functions are nonzero only in the element interior. This plays a vital role in the projection based interpolation described in Section 2.5 and it is also used in the proof of the following theorem.

Theorem 4.4. (Local basis) *Local basis functions given by formulas (4.17), (4.18), (4.19), (4.20) constitute basis of the space W^1_B defined in (4.13).*

Proof. All functions (4.17), (4.18), (4.19) and (4.20) are obviously linearly independent. Now consider arbitrary function $u \in W^1_B$. We want to show, that this function can be expressed as a linear combination of functions (4.17), (4.18), (4.19), (4.20). Values of u in vertices determine values of coefficients of vertex functions α^{v_i} (since vertex function corresponding to each vertex is the only nonzero from all functions of the basis). This expression is called a *vertex interpolant*:

$$u^v = \sum_{i=1}^{4} \alpha^{v_i} \varphi^{v_i}.$$

From the way of its construction it is obvious, that $u - u^v$ vanishes in all vertices and $(u-u^v)|_{e_i} \in P_{p^{e_i}}$ for $i = 1,\dots,12$ (it holds from the definition of u and from

the fact, that u^v is a trilinear function). From this reason we can define edge interpolant u^{e_i} on each edge e_i, since traces $\varphi_2^{e_i},\dots,\varphi_{p_{e_i}}^{e_i}$ on e_i are polynomials of degrees $2,\dots,p_{e_i}$ and have zero values in endpoints of e_i. Therefore there are unique coefficients $\alpha_k^{e_i}$ such that

$$u^{e_i} = \sum_{k=2}^{p^{e_i}} \alpha_k^{e_i} \varphi_k^{e_i},$$

where $u^{e_i} = u - u^v$ on e_i. Summing edge interpolants u^{e_i} for all edges e_i we obtain a complete edge interpolant u^e:

$$u^e = \sum_{i=1}^{12} u^{e_i}.$$

We can see, that $u - u^v - u^e$ vanishes in all vertices and on all edges. In the same way as for the edges we can construct face interpolant u^{s_i} for each face s^i. From the definition of the space W_B^1 it follows that $u|_{s_i} \in Q_{p^{s_i,1},p^{s_i,2}}$, therefore also $(u-u^v-u^e)|_{s_i} \in Q_{p^{s_i,1},p^{s_i,2}}$ and because it is zero in vertices and edges, it can be expressed as linear combination of functions $\varphi_{k_1,k_2}^{s_i}$:

$$u^{s_i} = \sum_{k_1=2}^{p^{s_i,1}} \sum_{k_2=2}^{p^{s_i,2}} \alpha_{k_1,k_2}^{s_i} \varphi_{k_1,k_2}^{s_i}$$

and again, by summing all face interpolants u^{s_i} we obtain complete face interpolant u^s:

$$u^s = \sum_{i=1}^{6} u^{s_i}.$$

Function $u - u^v - u^e - u^s = u^b$ is zero on the whole boundary of B and its polynomial degree is at most the same as the polynomial degree of u (thanks to the minimum rule (2.2.1)), therefore it can be expressed as a linear combination of functions φ_{k_1,k_2,k_3}^b:

$$u^b = \sum_{k_1=2}^{p^{b,1}} \sum_{k_2=2}^{p^{b,2}} \sum_{k_3=2}^{p^{b,3}} \alpha_{k_1,k_2,k_3}^b \varphi_{k_1,k_2,k_3}^b.$$

This concludes the proof, since we have shown that u can be expressed as a linear combination of local basis functions. □

Theorem 4.5. *A triple (B, W_B^1, Σ_B^1), where degrees of freedom in Σ_B^1 are associated with particular basis functions, constitutes the finite element.*

Proof. It follows from the definition of finite element in Chapter 2. □

4.3 Construction of global basis functions

In the finite element method, solution of the problem is sought as a combination of basis functions. In the concept of hierarchic basis, each basis function is related to an entity in the mesh, which in the case of three dimensional mesh can be vertex, edge, face or element interior.

We start with the space H^1, which is used for discretization of elliptic problems. The regularity requirement of this space is the continuity. Therefore, all basis functions has to be created in such way, that they are continuous in all vertices, edges and faces. In the following text, a rather technical description of such a construction is described. At first for the regular mesh, where the situation is simpler. Even though using functions with correct orientations is a big issue, one has to consider only elements adjacent to a given vertex, edge or face. On all other elements the basis function equals to zero. Bubble (or interior) functions are simple, they are local to one element and zero elsewhere and therefore their continuity is clear.

For the case of meshes with hanging nodes, new problems arise. Here much more elements may be involved and great effort has to be made to keep everything conforming. A rather sophisticated algorithm has been developed in [49] for two dimensional case. We used the idea, but in the 3D setting everything is much more complicated.

4.3.1 Vertex basis functions

First let us describe the construction of global vertex functions. Vertex functions are the simplest, because they are trilinear on each element, there is always only one vertex function corresponding to each vertex[1] and finally there is no orientation of vertices, that has to be taken into account for edges and faces.

Definition of global vertex basis function is very straightforward and it is described in the next section. When hanging nodes are present, situation becomes more complicated. A support of the basis function can be constituted by many elements as constrains "spread" through the mesh. This is described in detail in the following

[1] There is always one degree of freedom related to the vertex, but only if the vertex is not constrained. If it is constrained, there is no degree of freedom and therefore no vertex function correspond to this vertex.

Regular mesh

On the regular mesh, construction of global vertex basis functions is simple. Vertex function associated with vertex v is defined on a patch formed by elements sharing v (it is zero elsewhere) by transforming local vertex function on the elements in such a way, that they can be "glued" together with value 1 in vertex v. More rigorous definition will be given for the irregular mesh with hanging nodes, since the regular mesh is its special case.

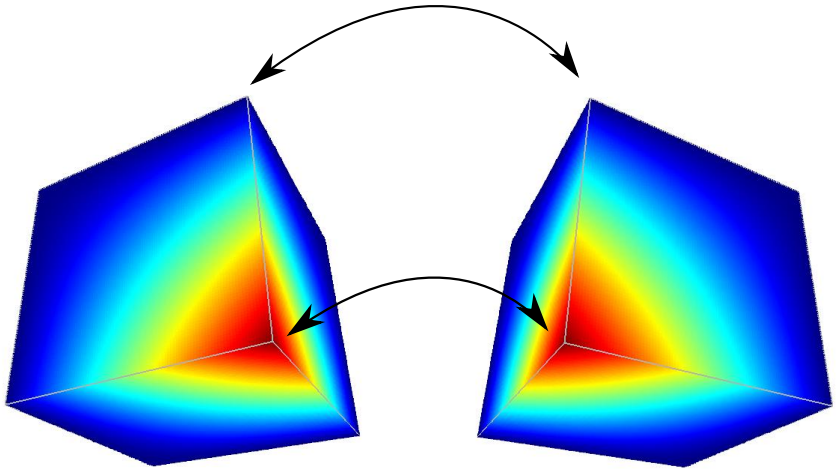


Figure 4.4: Two elements with images of the local basis vertex function being "glued" together to form a part of a global vertex function.

Mesh with hanging nodes

On the mesh with hanging nodes, more elements may be involved in construction of global basis function. The general definition will use the machinery of constrains, that has been developed in Chapter 3.

Definition 4.4. (Global vertex basis function) Let v_c be an unconstrained vertex and let the set $C^{Vv}(v_c)$ be calculated according to the previous definitions. Global vertex function w^{v_c} associated to v_c is defined on element K as follows:

$$w|_K = \sum_{v \in K,\ (v_c,\gamma) \in C_{Vv}(v)} \gamma \mathbf{\Phi}_K^1(\varphi^{\hat{v}}), \tag{4.21}$$

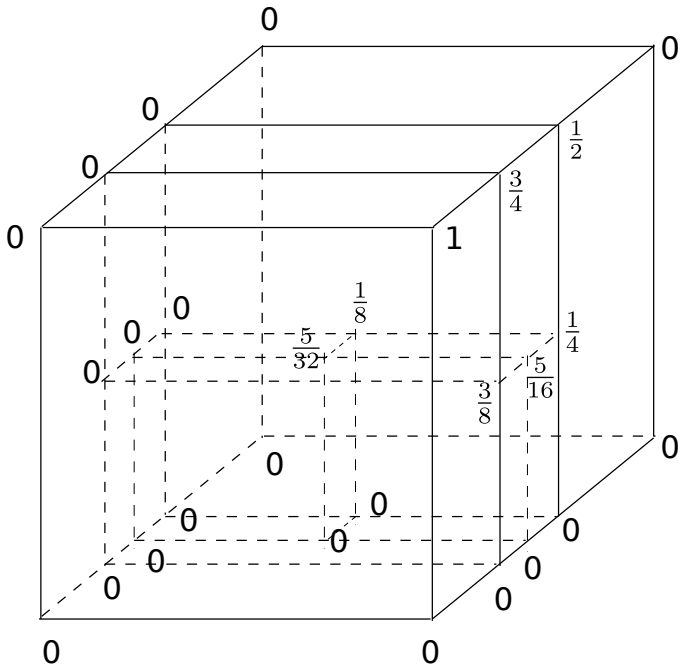


Figure 4.5: Example of one element of the coarse mesh with many refinements. Numbers assigned to vertices represent function γ defined when constructing vertex basis function (associated to a vertex with number 1).

where $\hat{v} = \boldsymbol{x}_K^{-1}(v)$ is the corresponding vertex of the reference domain and $\varphi^{\hat{v}}$ is a local vertex basis function associated with the vertex $\hat{v}$. The transformation $\boldsymbol{\Phi}_K^1$ from the reference element to K has been defined in Section 2.3.4.

The meaning of the definition is clear. When defining global basis vertex function in the mesh with hanging nodes, we have to add images of local basis vertex functions from many elements. They are added multiplied by coefficient γ, which corresponds to the "distance" of the vertex from the constraining vertex in the mesh and determines, how much is the contribution reduced. An example can be seen in Figure 4.5, where coefficients γ are shown for a part of a mesh with hanging nodes.

Remark 4.2. Let us note, that although many elements may be involved in the construction of the basis function, a support of the function is limited and cannot spread through the mesh without control. More precisely, a support of the vertex

basis function is always subset of the area formed by those elements from the coarse mesh[2], that share vertex v_c.

In the following theorem we state the key feature of the whole construction of the vertex global basis function from the previous text.

Theorem 4.6. (Conformity of global vertex basis function) *Global vertex basis function associated with vertex v defined in Definition 4.4 is continuous in* Ω *and therefore conforms to the space* H^1.

Proof. Coefficients γ are constructed in such a way, that the values of the global vertex basis function in all vertices are the same, no matter from which element we go. The reason is the following. Consider a constraining edge E with endpoints a and b and midpoint c. When we look at the value of the basis function at c from the side of element adjacent to edge E, it has to be equal to the average of values of coefficients γ at points a and b. This is because the value at c is obtained as a sum of values of trilinear shape functions associated with vertices a and b, multiplied by the coefficients γ.

When we look at the value of the basis function at c from the side of an element having c as is its vertex, we can see, that it is equal to the value of γ at c, since the only shape function involved in forming the global vertex basis function is that associated with vertex c, which has value 1 at c. The rest can be seen from definitions in Section 3.3.4 (and from an illustrative example presented in Figure 4.5). The value of γ at c equals to the average of values of γ at a and b. Therefore the value of the global vertex basis function at c is the same from all sides.

That implies, that the values are the same also on all edges and faces, since values there are images of linear and bilinear functions (local basis vertex function is linear on edges and bilinear on faces) and restrictions of mappings $\boldsymbol{x}_K$ and $\boldsymbol{x}_{K'}$ on the common edge or face shared by K and K' are the same. Therefore also its images are the same on all common edges and faces. Therefore the global vertex basis functions are continuous. □

4.3.2 Edge basis functions

In this section we will describe construction of global edge basis functions. Their construction is more complicated than construction of vertex functions. First of all, there may be more edge functions related to an edge. Another inconvenience

[2]By the coarse mesh we mean the initial mesh, that partitions the domain before any refinements are made, see Section 3.1.

is that we have to take into account the orientation of the edge and adjust projected local basis function accordingly.

For the regular mesh, the situation is simplified by the fact that support of an edge function is always composed of elements sharing the edge (usually but not necessarily 4 elements), while for the case of meshes with hanging nodes, it may be composed of many elements and coefficients for constrained edges has to be calculated similarly as for vertex nodes[3].

Regular mesh

As for the vertex functions, in this section we only want to suggest how the global edge functions are created. Rigorous definition will follow in the next, more general section.

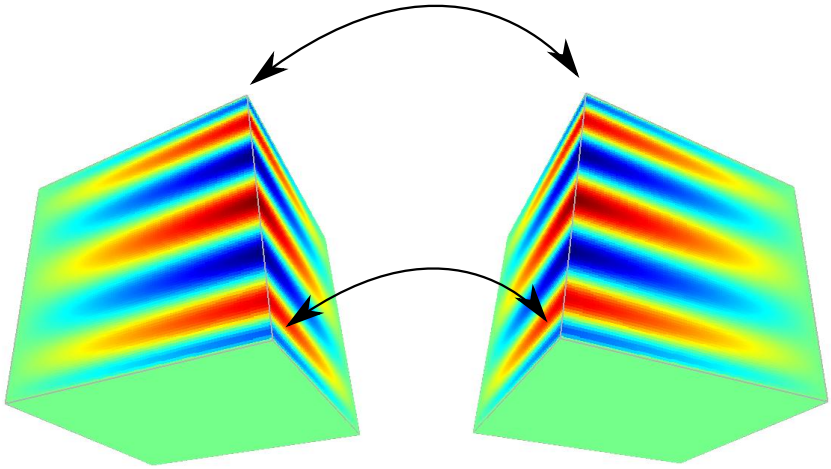


Figure 4.6: Creation of global edge basis function on two elements of a regular mesh.

The idea is to map local edge function from the reference domain onto elements adjacent to the edge which the constructed edge function is related to. This time we have to deal with the orientation. Edge functions have to be adjusted in such a way, that after "gluing" them together they will be continuous. An illustrative situation can be seen in Figure 4.6.

[3]This is one of those things that make the 3D implementation of the construction of basis functions much harder than implementation in 2D, where this is not necessary.

Constrained edge coefficients

Before we proceed to the definition of the global edge basis function on a mesh with hanging nodes, we need to prepare one more thing. In Chapter 3 we introduced a lot of geometrical data, that describe relations and constrains in the mesh. The main use of this data is to determine, which basis functions on constrained nodes will "contribute" to form globally continuous basis functions.

The core problem we have to solve is the following: Consider interval $[-1,1]$ with subinterval $[q^1,q^2]$, as shown in Figure 4.7. Now consider Lobatto function l_k defined on $[-1,1]$. The question is, how this function can be expressed by means of a linear function and Lobatto functions? This expression has the following form:

$$l_k \circ x_o^{o_c}\Big(\frac{q^2-q^1}{2}x+\frac{q^2-q^1}{2}\Big) = \\ l_k \circ x_o^{o_c}(q^1)\; l_0(x) + l_k \circ x_o^{o_c}(q^2) l_1(x) + \sum_{m=2}^{k} \delta_{m,o_e}^{k,(q^1,q^2),o_c} l_m \circ x_o^{o_e}(x) \quad (4.22)$$

for all $x \in [-1,1]$. On the left-hand side of the equation we have values of constraining function restricted to the interval $[q^1,q^2]$. On the right-hand side we have two linear functions with coefficients taken as function values of the constraining function in the endpoints of the interval $[q^1,q^2]$ and higher-order contributions with unknown coefficients. Orientation adjustments are discussed in the following remark. By plugging $k-1$ different values, we obtain $k-1$ linearly independent equations with $k-1$ unknown coefficients $\delta_{m,o_e}^{k,(q^1,q^2),o_c}$, $m=2,\ldots,k$. Those equations are solved and resulting coefficients used in construction of global basis functions in the mesh with hanging nodes.

Remark 4.3. Special attention, again, has to be given to proper orientation handling. At this place, we will not develop notation with such precision, as we did in Section 3.4. The symbol o_c represents the orientation flag of the constraining edge, while o_e represents the orientation flag of the constrained edge. We can see, that coefficients δ depend on both of them. The purpose of mappings $x_o^{o_c}$ and $x_o^{o_e}$ is to alter the coordinate direction, in accordance with definitions from Section 3.4.1.

From the efficiency reasons, we do not solve the system every time. Rather than that, for each position of edge/face on big face and for each degree of the constraining function we calculate unique number – the hash. When we need the coefficients, we first look into the hash table to find out, whether we have already solved this particular situation. If so, we do not calculate the coefficients again,

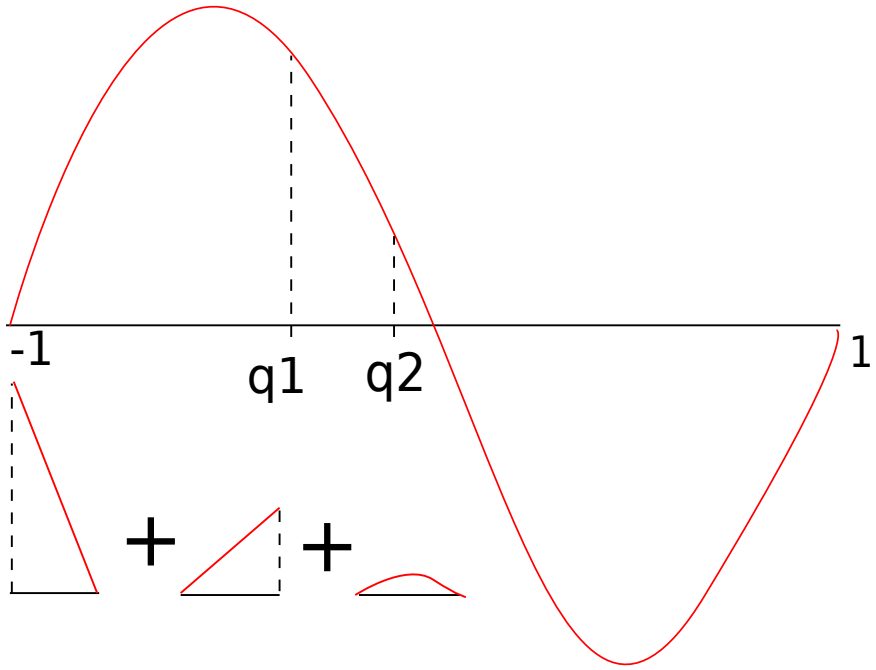


Figure 4.7: Calculation of constrained coefficients. A part of the function defined on interval $[-1,1]$ is reconstructed on sub-element $[q_1,q_2]$ by adding linear and higher-order part.

we just use them. If not, we find the coefficients by solving the linear system arising from the equation (4.22), as it was described above.

Mesh with hanging nodes

On the mesh with hanging nodes, more local basis functions has to be put together to form the global edge basis function, this time both vertex and edge local basis functions will be involved.

Definition 4.5. (Global edge basis function) Let e_c be unconstrained edge and let sets $C^{Ev}(e_c)$ and $C^{Ee}(e_c)$ be calculated according to the previous definitions. Global edge function $w_k^{e_c}$ associated to e_c, of order $k \leq p^{e_c}$ is defined on element K

as follows:

$$w_k^{e_c}|_K = \sum_{v\in K,\,(e_c,\gamma,p)\in C^{Ev}(v)} \gamma\, l_k(p)\, \boldsymbol{\Phi}_E^1(\varphi^{\hat{v}}) + \\ + \sum_{e\in K,\,(e_c,\gamma,(q^1,q^2))\in C^{Ee}(e)} \sum_{m=2}^{p_{e_c}} \gamma\, \delta_{m,o_e}^{k,(q^1,q^2),o_c}\, \boldsymbol{\Phi}_K^1(\varphi_m^{\hat{e}} \circ \boldsymbol{x}_o^{K,e}),$$

where

$$\hat{v} = \boldsymbol{x}_K^{-1}(v), \qquad \hat{e} = \boldsymbol{x}_K^{-1}(e)$$

are the corresponding vertex and edge of the reference domain. Function $\varphi^{\hat{v}}$ is the local vertex basis function associated with the vertex $\hat{v}$ and $\varphi_k^{\hat{e}}$ is the local edge basis function associated with the edge $\hat{e}$ of the order k. Transformation $\boldsymbol{\Phi}_K^1$ from the reference element to K has been defined in (2.3.4) and the orientation adjustment $\boldsymbol{x}_o^{K,e}$ has been defined in Section 3.4.1.

The definition is similar to the case of vertex global basis functions, but it is more complicated due to the fact, that both vertex and edge functions have to be included. First, contributions of vertices are added. Similarly as in the previous section, coefficient γ adjusts contribution of a vertex v according to its "distance" from constraining edge. Additional coefficient $l_k(p)$ alter contribution according to the position p on the constraining edge, from where the vertex v is constrained (and the value of the constraining function in that point is $l_k(p)$). On the second line we have contribution of edges, again multiplied by coefficient γ with similar meaning and with coefficient defined in the previous section, ensuring that proper linear combination of edge functions is used to ensure continuity on edges.

Remark 4.4. Again we want to point out, that even though support of the edge function may comprise many elements as the mesh is refined, it will never "extend" beyond limit given by union of elements of the coarse mesh, that share edge e_c.

The whole described procedure has the purpose of constructing continuous edge global basis function. Let us conclude it in a theorem similar to one we stated for vertex functions.

Theorem 4.7. (Conformity of global edge basis function) *Global edge basis function associated with edge e defined in Definition 4.5 is continuous in* Ω *and therefore conforms to the space* H^1.

Proof. First let us to show continuity on constraining edges, where the basis function has the form of higher-order polynomial. The construction of constrained coefficients has been described in the previous text. They are calculated according to

formula (4.22) which guarantees, that values on each such edge are the same from all sides (both from elements adjacent to the constraining edge and elements adjacent to smaller constrained edge, which is a part of bigger one). The continuity than can be shown from the linearity of other components of the edge functions, similarly as for vertex functions. □

4.3.3 Face basis functions

Face functions are even more complicated than those associated with edges. There are several reasons. For meshes with hanging nodes, all vertex, edge and face functions has to be combined on potentially many elements to form global basis functions. Moreover, there are many possible combinations of face and edge orientations, that has to be solved properly.

Regular mesh

Again, let us start with the case of regular meshes. The situation is quite simple. Two elements only share given face and the global face basis function is created by putting two images of the local face function together, as shown in Figure 4.8. The only thing that has to be taken care of is proper orientation handling, since there are eight possible orientations of the face (from the viewpoint of each element), as it has been described in Section 3.4.2.

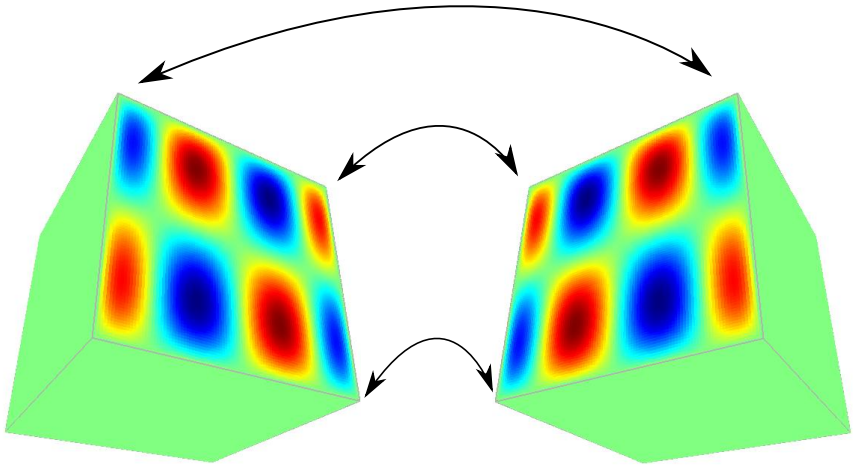


Figure 4.8: Construction of global face basis function on a regular mesh.

Constrained face coefficients

The idea behind is similar as for edges, just extended to two dimensions. For an edge, we first had to subtract linear part corresponding to two vertices, the rest was created by Lobatto functions, corresponding to the edge. For the face function, we similarly have to subtract parts corresponding to vertices and edges.

First let us define

$$l_{k_1,k_2}(x,y) = l_{k_1}(x)\, l_{k_2}(y),$$

which will be useful to simplify orientation handling. Constrained coefficients on edges were calculated in (4.22), according to Figure 4.7. Now we want to formulate a similar relation, that could be used to find constrained coefficients on a face, so that values on smaller faces "match" values on larger face to which they are adjacent. An example of such situation is presented in Figure 4.9, where also the decomposition of the constraining function into vertex, edge and face contributions is suggested. The coefficient of the vertex contribution is just the value of the constraining face function in the vertex. Coefficients of edge contributions can be found using the procedure described in the previous section. The following formula shows, how the coefficients of face contribution may be calculated:

$$\begin{aligned}
l_{k_1,k_2} \circ x_o^{o_c}\Big(\frac{q_1^2-q_1^1}{2}x + \frac{q_1^2-q_1^1}{2},\ \frac{q_2^2-q_2^1}{2}y + \frac{q_2^2-q_2^1}{2}\Big) = \\
l_{k_1,k_2} \circ x_o^{o_c}(q_1^1,q_2^1)\ l_0(x)l_1(y) + l_{k_1,k_2} \circ x_o^{o_c}(q_1^2,q_2^1)\ l_1(x)l_1(y) + \\
l_{k_1,k_2} \circ x_o^{o_c}(q_1^2,q_2^2)\ l_1(x)l_0(y) + l_{k_1,k_2} \circ x_o^{o_c}(q_1^1,q_2^2)\ l_0(x)l_0(y) + \\
l_{k_2} \circ x_o^{o_{e_1}}(q_2^2) \sum_{m=2}^{k_1} \delta_{m,o_{e_1}}^{k_1,(q_1^1,q_1^2),o_c} l_m(x)l_1(y) + l_{k_1} \circ x_o^{o_{e_2}}(q_1^2) \sum_{m=2}^{k_2} \delta_{m,o_{e_2}}^{k_2,(q_2^1,q_2^2),o_c} l_1(x)l_m(y) + \\
l_{k_2} \circ x_o^{o_{e_3}}(q_2^1) \sum_{m=2}^{k_1} \delta_{m,o_{e_3}}^{k_1,(q_1^1,q_1^2),o_c} l_m(x)l_0(y) + l_{k_1} \circ x_o^{o_{e_4}}(q_1^1) \sum_{m=2}^{k_2} \delta_{m,o_{e_4}}^{k_2,(q_2^1,q_2^2),o_c} l_0(x)l_m(y) + \\
\sum_{m_1=2}^{k_1}\sum_{m_2=2}^{k_2} \zeta_{m_1,m_2,o_f}^{(q_1^1,q_1^2),(q_1^1,q_1^2),o_c} l_{m_1,m_2} \circ x_o^{o_f}(x,y). \quad (4.23)
\end{aligned}$$

The whole scheme may seem rather complicated, but it is similar to the case of edges. On the first line, we have constraining function. On the two following lines, we have bilinear contributions of four vertices, with known coefficients. On the other two lines, there are contributions of edges, again with known coefficients. Finally, on the last line, there is contribution of face functions with unknown coefficients.

Remark 4.5. All local basis functions in the relation has to be equipped with proper orientation mapping. Notation introduced in Section 3.4 does not suit

exactly for this situation, but the idea is the same. The symbol o_c is the orientation flag of constraining face, o_f is the orientation flag of the constrained face and $o_{e_1},\ldots,o_{e_4}$ are orientation flags of edges involved. Those flags are used to construct mappings $x_o^{o_c}$, $x_o^{o_f}$ and $x_o^{o_{e_1}},\ldots,x_o^{o_{e_4}}$, whose purpose is to adjust coordinates in such a way, that calculated coefficients are in accordance with orientation handling defined in Section 3.4.

The equation has to be satisfied for all $x,y\in[-1,1]$. If we choose $(k_1-1)\times(k_2-1)$ grid points in $[-1,1]$, we obtain system of $(k_1-1)(k_2-1)$ linearly independent linear equations for $(k_1-1)(k_2-1)$ unknown coefficients $\alpha_f^{i,j}$. It is easy to show, that the system of equations solved have a unique solution. Again, we use caching to avoid repetitive calculations of the same situation.

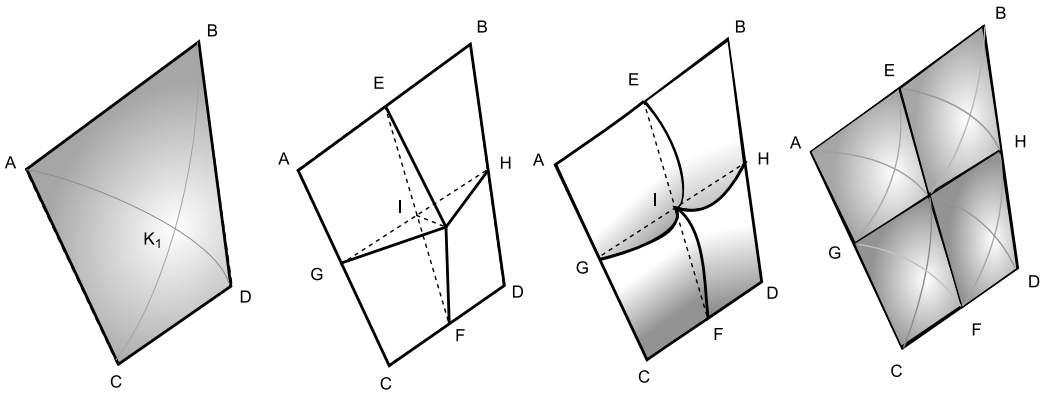


Figure 4.9: Illustrative figure showing, how face function related to the face *ABCD* may be expressed in terms of basis functions related to constrained nodes. First vertex interpolant is calculated in *I* and subtracted. Then edge interpolants are calculated for *EI*, *FI*, *GI* and *HI* and subtracted. Finally, face interpolants for small faces are calculated. This idea is used in our text, although we allow to calculate interpolation for arbitrary sub-face of the constraining face.

Mesh with hanging nodes

With this definitions in hand, we may proceed to the most complicated situation, the construction of global face functions on meshes with hanging nodes.

Definition 4.6. (Global face basis function) Let f_c be unconstrained face and let the sets $C^{Fv}(f_c)$, $C^{Fe}(f_c)$ and $C^{Ff}(f_c)$ be calculated according to the previous definitions. Global face function $w_{k_1,k_2}^{f_c}$ associated to f_c, of orders $k_1\leq p_1^{f_c}$, $k_2\leq$

$p_2^{f_c}$ is defined on element K as follows:

$$w_{k_1,k_2}^{f_c}|_K = \sum_{v\in K,\,(f_c,\gamma,p_1,p_2)\in C^{Fv}(v)} \gamma\, l_{k_1}(p_1)\, l_{k_2}(p_2)\, \mathbf{\Phi}_K^1(\varphi^{\hat{v}}) + \\ + \sum_{e\in K,\,(f_c,\gamma,p,(q^1,q^2),d)\in C^{Ee}(e)} \sum_{m=2}^{p_{e_c}} \gamma\, \varepsilon_{m,o_e}^{k_1,k_2,p,(q^1,q^2),d,o_c}\, \mathbf{\Phi}_K^1(\varphi_m^{\hat{e}} \circ \boldsymbol{x}_o^{K,e}) + \\ + \sum_{f\in K,\,(f_c,(q_1^1,q_1^2),(q_1^1,q_1^2))\in C^{Ff}(f)} \sum_{m_1=2}^{p_{fc}^1} \sum_{m_2=2}^{p_{fc}^2} \zeta_{m_1,m_2,o_f}^{(q_1^1,q_1^2),(q_1^1,q_1^2),o_c}\, \mathbf{\Phi}_K^1(\varphi_{m_1,m_2}^{\hat{f}} \circ \boldsymbol{x}_o^{K,f}), \tag{4.24}$$

where

$$\hat{v} = \boldsymbol{x}_K^{-1}(v), \qquad \hat{e} = \boldsymbol{x}_K^{-1}(e), \qquad \hat{f} = \boldsymbol{x}_K^{-1}(f)$$

are the vertex, the edge and the face on the reference cube, corresponding to the vertex v, the edge e and the face f in the physical mesh, respectively. Transformation $\mathbf{\Phi}_K^1$ from the reference element to K has been defined in (2.3.4) and the orientation adjustments $\boldsymbol{x}_o^{K,e}$ and $\boldsymbol{x}_o^{K,f}$ have been defined in Section 3.4.

The definition may seem complicated, but the structure is clear. First we sum up contributions of vertex functions, with proper coefficient γ reflecting "distance" of the vertex from constraining face and also multiplied by values of appropriate Lobatto functions in points p_1 and p_2, reflecting from which point on the constraining face is vertex constrained. On the second line, edge contributions are added with coefficient γ with the similar meaning and coefficient ε, that has been calculated in the section dedicated to edge functions, ensuring that edge functions of different orders form result corresponding to the constraining function. Finally, on the last line, we have face functions, again with appropriate coefficients defined in the previous section.

An illustrative example can be seen in Figures 4.10 and 4.11. We have shown a process of creation of a global face basis function on the mesh with hanging nodes. On the second figure we can see all vertices, edges and faces, that are involved in this still relatively simple situation. Constructed face functions have the following properties.

Remark 4.6. Global face basis function associated with face f_c defined in Definition 4.6 vanishes in all elements of the coarse mesh that do not share face f_c.

Theorem 4.8. (Conformity of global face basis function) *Global face basis function associated with face f defined in Definition 4.6 is continuous in Ω and therefore conforms to the space H^1.*

Proof. The idea is similar as for edge functions. Function values on faces and edges with higher polynomial orders "match" thanks to the construction. The rest follows from the way, how coefficients in vertices are calculated and from linearity of face function in one direction. □

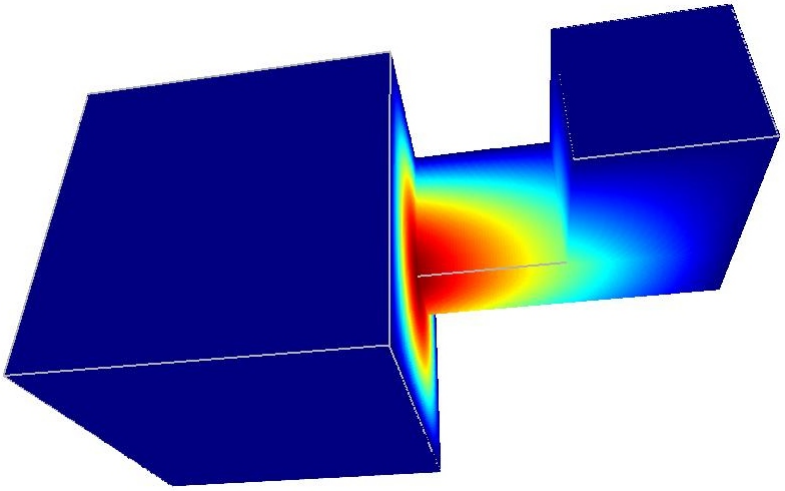


Figure 4.10: Global face function constructed on mesh with hanging nodes. Three of the elements involved are shown.

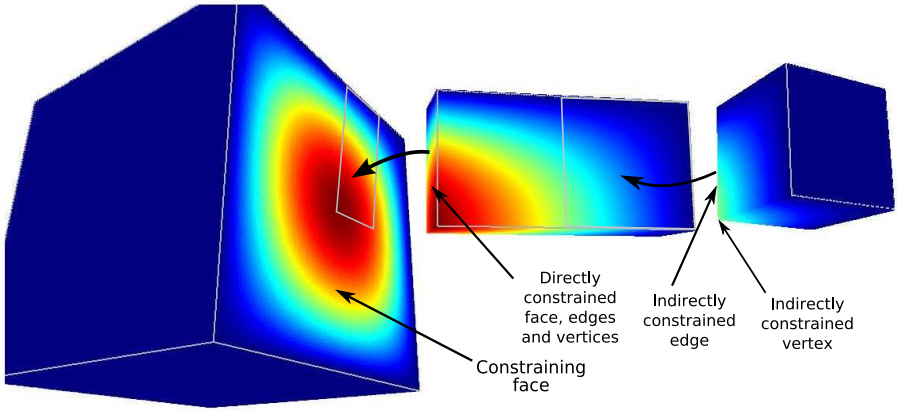


Figure 4.11: Decomposition of situation from Figure 4.10. All nodes constrained by the large face both directly and indirectly are marked by arrows. Basis functions from all those nodes has to be used to form continuous basis function.

4.3.4 Bubble functions

Construction of global bubble basis functions is simple, since they are local to one element only. Hence it is not necessary to ensure conformity requirements by "gluing" images of local basis functions on more elements of the physical mesh. A global bubble basis function is defined as follows.

Definition 4.7. (Global bubble basis function) Global bubble basis function $w^K_{k_1,k_2,k_3} \in V_{h,p}$, associated to the element K, of orders $k_1 \leq p^K_1$, $k_2 \leq p^K_2$ and $k_3 \leq p^K_3$, is defined as $w|_{K'} = 0$ on all elements $K' \neq K$ and

$$w^K_{k_1,k_2,k_3}|_K = \mathbf{\Phi}^1_K(\varphi^b_{k_1,k_2,k_3}), \tag{4.25}$$

where $\varphi^b_{k_1,k_2,k_3}$ is a local bubble function . Transformation $\mathbf{\Phi}^1_K$ from the reference element to K has been defined in (2.3.4).

Remark 4.7. Just for completeness let us state, that global bubble functions are local to one element only and vanish in all other elements of the mesh.

Theorem 4.9. (Conformity of global bubble basis function) *Global bubble basis function associated with element K defined in Definition 4.7 is continuous in* Ω *and therefore conforms to the space* H^1.

Proof. Obvious from the definition, since we construct the bubble function on one element only, by composing two continuous functions. □

CHAPTER

FIVE

HP-FEM IN 3D FOR ELECTROMAGNETIC PROBLEMS

In this chapter we want to address the usage of the *hp*-FEM for electromagnetic problems described by Maxwell's equations. It has similar structure like the previous chapter, that has been devoted to elliptic problems. First we want to state the problem in both classical and weak formulation and describe some of its key properties. In the next section we define vector-valued higher-order shape functions, that are suitable for discretization of the $\boldsymbol{H}^{\text{curl}}$ space. Shape functions are used in Section 5.3 to define global basis functions. As in the previous chapter, we will use mesh-related definitions and structures from Chapter 3 to ensure conformity of constructed global basis functions.

The finite element method has been used for Maxwell's equations for a long time. Soon it was shown, that standard H^1-conforming elements are not appropriate, see e.g. [9], [10], [31], [33] and [34]. It led to design of special elements for $\boldsymbol{H}^{\text{curl}}$ space. At first, the lowest-order Whitney elements appeared (see [51]). More recently it became obvious, that it would be beneficial to develop higher-order elements (see [1], [18], [32] and others).

The definition of higher-order $\boldsymbol{H}^{\text{curl}}$ shape functions in this work is strongly inspired by definitions used in the 2D code. For details see e.g. [44], [50]. The main difference, similarly as for the H^1 space, is in the way, how basis functions on individual elements are "glued" together in order to form global basis functions.

5.1 Time-harmonic Maxwell's equations

Due to the limited length of this text, we skip the derivation from the fundamental laws of electromagnetics and we directly introduce the model problem of time-harmonic Maxwell's equations. For details see e.g. [23] or [46].

5.1.1 Classical formulation

Let us start with classical formulation of time-harmonic Maxwell's equations. The problem is to find $\boldsymbol{E} \in [C^2(\Omega)]^3 \cap [C^1(\overline{\Omega})]^3$ such that

$$\boldsymbol{\nabla} \times \left(\mu_r^{-1} \boldsymbol{\nabla} \times \boldsymbol{E} \right) - k^2 \varepsilon_r \boldsymbol{E} = \boldsymbol{\Phi} \quad \text{in } \Omega, \tag{5.1}$$

$$\boldsymbol{E} \times \boldsymbol{\nu} = 0 \quad \text{on } \Gamma_{\mathrm{P}}, \tag{5.2}$$

$$\mu_r^{-1} (\boldsymbol{\nabla} \times \boldsymbol{E}) \times \boldsymbol{\nu} - jk\lambda \boldsymbol{E}_T = \boldsymbol{g} \quad \text{on } \Gamma_{\mathrm{I}}, \tag{5.3}$$

where $\Omega \subset \mathbb{R}^3$, is a bounded domain with Lipschitz boundary, $\boldsymbol{\nu}$ is the unit outer normal to the boundary $\partial\Omega$, the sets Γ_{P} and Γ_{I} are relatively open in $\partial\Omega$, $\overline{\Gamma}_{\mathrm{P}} \cap \overline{\Gamma}_{\mathrm{I}} = \emptyset$, and $\overline{\Gamma}_{\mathrm{P}} \cup \overline{\Gamma}_{\mathrm{I}} = \partial\Omega$, the symbol $\boldsymbol{E}_T$ stands for the tangential projection of $\boldsymbol{E}$ to the boundary

$$\boldsymbol{E}_T = (\boldsymbol{\nu} \times \boldsymbol{E}) \times \boldsymbol{\nu}$$

and λ is the impedance

$$\lambda = Z \sqrt{\frac{\mu_0}{\varepsilon_0}},$$

where Z is a material parameter. The boundary conditions (5.2) and (5.3) are called perfect conductor and impedance boundary condition, respectively.

5.1.2 Weak formulation

In order to derive the weak formulation of problem (5.1)–(5.3) we first multiply the equation (5.1) with sufficiently smooth complex vector-valued function $\boldsymbol{F}$ and integrate over the domain Ω:

$$\int_\Omega \left(\boldsymbol{\nabla} \times (\mu_r^{-1} \boldsymbol{\nabla} \times \boldsymbol{E}) \cdot \boldsymbol{F} - k^2 \varepsilon_r \boldsymbol{E} \cdot \boldsymbol{F} \right) \mathrm{d}\boldsymbol{x} = \int_\Omega \boldsymbol{\Phi} \cdot \boldsymbol{F} \, \mathrm{d}\boldsymbol{x}, \tag{5.4}$$

where the inner product for complex vectors $\boldsymbol{a}, \boldsymbol{b}$ is defined as

$$\boldsymbol{a} \cdot \boldsymbol{b} = \sum_{i=1}^{3} a_i \overline{b_i}.$$

After applying the Green's theorem, we obtain the form

$$\int_\Omega \Big((\mu_r^{-1}\nabla \times \boldsymbol{E}) \cdot (\nabla \times \boldsymbol{F}) \; - k^2 \varepsilon_r \boldsymbol{E} \cdot \boldsymbol{F} \Big) \, \mathrm{d}\boldsymbol{x} + \\ + \int_{\partial\Omega} \boldsymbol{\nu} \times (\mu_r^{-1}\nabla \times \boldsymbol{E}) \cdot \boldsymbol{F}_T \, \mathrm{d}\boldsymbol{s} = \int_\Omega \boldsymbol{\Phi} \cdot \boldsymbol{F} \, \mathrm{d}\boldsymbol{x}, \quad (5.5)$$

where $\boldsymbol{F}_T$ stands for the tangential projection of $\boldsymbol{F}$ to the boundary of the domain Ω:

$$\boldsymbol{F}_T = (\boldsymbol{\nu} \times \boldsymbol{F}) \times \boldsymbol{\nu}.$$

As a next step we use the boundary conditions. The space, where the test functions $\boldsymbol{F}$ are taken from will be specified later, but from now on we will use only such test functions, that satisfy the condition

$$\boldsymbol{F} \times \boldsymbol{\nu} = 0 \quad \text{on } \Gamma_\mathrm{P}. \quad (5.6)$$

The field $\boldsymbol{E}$ itself satisfies this condition thanks to the perfect conductor boundary condition (5.2). This implies that on the Γ_P part of the boundary, the surface integral in (5.5) vanishes. For the remaining part of the surface Γ_I we use the impedance boundary condition (5.3):

$$\int_\Omega \Big((\mu_r^{-1}\nabla \times \boldsymbol{E}) \cdot (\nabla \times \boldsymbol{F}) \; - k^2 \varepsilon_r \boldsymbol{E} \cdot \boldsymbol{F} \Big) \, \mathrm{d}\boldsymbol{x} - \\ - \int_{\Gamma_\mathrm{I}} jk\lambda \, \boldsymbol{E}_T \cdot \boldsymbol{F}_T \, \mathrm{d}\boldsymbol{s} = \int_\Omega \boldsymbol{\Phi} \cdot \boldsymbol{F} \, \mathrm{d}\boldsymbol{x} + \int_{\Gamma_\mathrm{I}} \boldsymbol{g} \cdot \boldsymbol{F}_T \, \mathrm{d}\boldsymbol{s}. \quad (5.7)$$

We can see that the appropriate space for the functions in the weak formulation is

$$V = \{ \boldsymbol{E} \in \boldsymbol{H}^{\mathrm{curl}}(\Omega); \quad \boldsymbol{\nu} \times \boldsymbol{E} = 0 \text{ on } \Gamma_\mathrm{P} \}, \quad (5.8)$$

where the space $\boldsymbol{H}^{\mathrm{curl}}(\Omega)$ has been defined in Section 2.1.

Assumptions

In order to introduce the weak solution, we need to assume some properties of the domain and the data of the problem:

- The domain $\Omega \subset \mathbb{R}^3$ is bounded and simply connected, with Lipschitz-continuous boundary $\partial\Omega$.
- The boundary of the domain consists of two relatively open parts Γ_P and Γ_I, $\partial\Omega = \overline{\Gamma_\mathrm{P}} \cup \overline{\Gamma_\mathrm{I}}$, $\Gamma_\mathrm{P} \cap \Gamma_\mathrm{I} = \emptyset$.

- The domain Ω can be split into several open simply connected sub-domains $\Omega_1, \Omega_2, \ldots, \Omega_n$ with Lipschitz-continuous boundary, such that $\overline{\Omega} = \bigcup_{i=1}^{n} \overline{\Omega}_i$ and $\Omega_i \cap \Omega_j = \emptyset$ for $i, j \in 1 \ldots n$, $i \neq j$. The permitivity and permeability parameters ε_r and μ_r can be discontinuous, but has to be smooth in each sub-domain Ω_i. This allows us to deal with problems with several different materials.
- $\varepsilon_r|_{\Omega_i} \in H^3(\Omega)$, $i \in 1, \ldots, n$
- There exists a positive constant $C_\varepsilon > 0$ such that for each sub-domain Ω_1, $\Omega_2, \ldots, \Omega_n$ either $\mathrm{Im}(\varepsilon_r) \geq C_\varepsilon$ or $\mathrm{Im}(\varepsilon_r) = 0$.
- The impedance function $\lambda \in L^\infty(\Gamma_\mathrm{I})$, $\lambda > 0$.
- Further, we assume $\boldsymbol{\Phi} \in (L^2(\Omega))^3$ and $\boldsymbol{g} \in (L^2(\Gamma_\mathrm{I}))^3$.

Weak formulation

Denote the sesquilinear form

$$a(\boldsymbol{E}, \boldsymbol{F}) = \int_\Omega \Big((\mu_r^{-1} \nabla \times \boldsymbol{E}) \cdot (\nabla \times \boldsymbol{F}) \; - k^2 \varepsilon_r \boldsymbol{E} \cdot \boldsymbol{F} \Big) \, \mathrm{d}\boldsymbol{x} - \int_{\Gamma_\mathrm{I}} jk\lambda \, \boldsymbol{E}_T \cdot \boldsymbol{F}_T \, \mathrm{d}\boldsymbol{s}$$

and the linear form

$$l(\boldsymbol{F}) = \int_\Omega \boldsymbol{\Phi} \cdot \boldsymbol{F} \, \mathrm{d}\boldsymbol{x} + \int_{\Gamma_\mathrm{I}} \boldsymbol{g} \cdot \boldsymbol{F}_T \, \mathrm{d}\boldsymbol{s}.$$

Definition 5.1. (Weak solution) Let us consider all the above assumptions. Let the function $\boldsymbol{E} \in V$ satisfy

$$a(\boldsymbol{E}, \boldsymbol{F}) = l(\boldsymbol{F}) \quad \forall \boldsymbol{F} \in V. \tag{5.9}$$

Then the function $\boldsymbol{E}$ is called the *weak solution* of problem (5.1)–(5.3).

Existence and uniqueness of the solution

Theorem 5.1. (Unique solution) *Assume that the domain Ω, boundary parts Γ_P and Γ_I, coefficients ε_r, μ_r, λ and data $\boldsymbol{\Phi}$ and $\boldsymbol{g}$ satisfy the assumptions from the definition 5.1. Let us further assume that either the impedance boundary Γ_I is not empty, or that the imaginary part of ε_r is positive in some open sub-domain of Ω.*

Then for any wave number $k > 0$, problem (5.9) has a unique solution $\boldsymbol{E} \in V$.

Proof. The proof can be found in [46]. □

At this point the description of mathematical properties of the equations is sufficient for our purposes. More details can be found in literature, which was mentioned at the beginning of this chapter.

5.2 Higher-order shape functions

In this section we present shape (or local basis) functions used in the space $\boldsymbol{H}^{\text{curl}}$. The procedure is similar to that used for H^1 space in Section 4.2. Again we consider only hexahedral elements with reference element described in Section 2.3.1 and depicted in Figure 2.2.

5.2.1 Construction of local basis functions

As we can see from the de Rham diagram (see Section 2.3), spaces W^1 and W^{curl} used for discretization of spaces H^1 and $\boldsymbol{H}^{\text{curl}}$ are connected through the gradient operator ∇. The local basis functions should also be constructed in this fashion. Consider product monomial $\xi_1^i \xi_2^j \xi_3^k \in W^1$. Its gradient should be included in the space W^{curl}:

$$\nabla \xi_1^i \xi_2^j \xi_3^k \in W^1 = \left(i\xi_1^{i-1} \xi_2^j \xi_3^k,\ j\xi_1^i \xi_2^{j-1} \xi_3^k,\ k\xi_1^i \xi_2^j \xi_3^{k-1} \right).$$

This is the general guideline, how to define local basis functions of the space $\boldsymbol{H}^{\text{curl}}$. But rather than using gradients of functions from H^1 as basis functions, we use simpler functions, which have always only one component nonzero.

Remark 5.1. The fact that $\boldsymbol{H}^{\text{curl}}$ is defined as a vector-valued space brings several complications not only for the practical implementation of the algorithms, but also for their description. Making pictures is also more complicated, since 3D plots containing vectors are usually not very lucid, but, on the other hand, 3 different figures showing one component each do not give the general picture very well.

For the basis functions, however, things are simplified by defining all of them with only one nonzero component. That means, that vector values are always parallel to one of the axis. So if we say, that tangential component of a basis function vanish on the edge, it means either that values are zero on the edge or that direction of the vectors is perpendicular to the edge (parallel to one of two remaining axis perpendicular to the edge). The same holds for the faces. If tangential component of a basis function is said to vanish on a face, it means, that it is either zero or

that it is perpendicular to this face (for faces there are two linearly independent tangential directions).

Orders of elements

We again want to allow different orders in each direction and the meaning of the order of the element $p^{b,1}$, $p^{b,2}$ and $p^{b,3}$, faces $p^{s_i,1}$ and $p^{s_i,2}$ and edges p^{e_i} are the same as in the elliptic case, see Section 4.2. The only difference is, that in the $\boldsymbol{H}^{\text{curl}}$ space the lowest order is 0, not 1. This may sound confusing, but may be seem as a logical consequence of the way how basis functions are derived, as it was described in the previous paragraph.

The minimum rule has the same form as for the H^1 reference domain, which means, that orders of faces $p^{s_i,1}$ and $p^{s_i,2}$ are smaller or equal to corresponding pair of orders of the element (two of $p^{b,1}$, $p^{b,2}$ and $p^{b,3}$ parallel to the local face coordinates) and the order of edge p^{e_i} is at most equal to the minimum of orders (in the direction of the edge e_i) of adjacent faces.

Local basis functions

The finite element $(B, W^{\text{curl}}, \Sigma^{\text{curl}})$ should be equipped with a polynomial space defined as follows:

$$W^{\text{curl}} = \{\boldsymbol{E} \in Q_{p^{b,1},p^{b,2}+1,p_{b,3}+1} \times Q_{p^{b,1}+1,p^{b,2},p_{b,3}+1} \times Q_{p^{b,1}+1,p^{b,2}+1,p_{b,3}},$$
$$\boldsymbol{E}_t|_{s_i} \in Q_{p^{s_i,1},p^{s_i,2}+1} \times Q_{p^{s_i,1}+1,p^{s_i,2}},\ i = 1,\ldots,6,$$
$$\boldsymbol{E}\cdot\boldsymbol{t}|_{e_j} \in P_{p_{e_j}}, j = 1,\ldots,12\}, \quad (5.10)$$

where $\boldsymbol{E}_t|_{s_i} = \boldsymbol{E} - \boldsymbol{n}_i(\boldsymbol{E}\cdot\boldsymbol{n}_i)$ is the projection of the vector $\boldsymbol{E}$ on the face s_i, with $\boldsymbol{n}_i$ being the outer normal vector to the face s_i. The scalar product $\boldsymbol{E}\cdot\boldsymbol{t}|_{e_j}$ represents the projection of the vector $\boldsymbol{E}$ on the edge e_j.

Conformity requirements of the space $\boldsymbol{H}^{\text{curl}}$, see Section 2.6 suggest, that shape functions will be divided into categories associated with edges, faces and element interior. Unlike in the case of H^1 space, there are no vertex functions. Shape functions are again constructed as products of 1D Lobatto functions l_i and also Legendre polynomials L_i, defined in Section 4.2.2. Since $\boldsymbol{H}^{\text{curl}}$ is a vector space, all shape functions have to be vector-valued. To keep the definition simple, all vector values are parallel to one of the axis ξ_1, ξ_2 or ξ_3, in other words, only one component is nonzero.

- *Edge functions* $\psi_k^{e_i}$, $i = 1,\dots,12$, $k = 0,\dots,p^{e_i}$ are defined as follows (positions of edges on the reference domain were defined in Section 2.3.1).

$$
\begin{aligned}
\psi_k^{e_1} &= L_k(\xi_1)l_0(\xi_2)l_0(\xi_3)\boldsymbol{\xi_1} && k = 0,\dots,p_{e_1}, \\
\psi_k^{e_2} &= l_1(\xi_1)L_k(\xi_2)l_0(\xi_3)\boldsymbol{\xi_2} && k = 0,\dots,p_{e_2}, \\
\psi_k^{e_3} &= L_k(\xi_1)l_1(\xi_2)l_0(\xi_3)\boldsymbol{\xi_1} && k = 0,\dots,p_{e_3}, \\
\psi_k^{e_4} &= l_0(\xi_1)L_k(\xi_2)l_0(\xi_3)\boldsymbol{\xi_2} && k = 0,\dots,p_{e_4}, \\
\psi_k^{e_5} &= l_0(\xi_1)l_0(\xi_2)L_k(\xi_3)\boldsymbol{\xi_3} && k = 0,\dots,p_{e_5}, \\
\psi_k^{e_6} &= l_1(\xi_1)l_0(\xi_2)L_k(\xi_3)\boldsymbol{\xi_3} && k = 0,\dots,p_{e_6}, \\
\psi_k^{e_7} &= l_1(\xi_1)l_1(\xi_2)L_k(\xi_3)\boldsymbol{\xi_3} && k = 0,\dots,p_{e_7}, \\
\psi_k^{e_8} &= l_0(\xi_1)l_1(\xi_2)L_k(\xi_3)\boldsymbol{\xi_3} && k = 0,\dots,p_{e_8}, \\
\psi_k^{e_9} &= L_k(\xi_1)l_0(\xi_2)l_1(\xi_3)\boldsymbol{\xi_1} && k = 0,\dots,p_{e_9}, \\
\psi_k^{e_{10}} &= l_1(\xi_1)L_k(\xi_2)l_1(\xi_3)\boldsymbol{\xi_2} && k = 0,\dots,p_{e_{10}}, \\
\psi_k^{e_{11}} &= L_k(\xi_1)l_1(\xi_2)l_1(\xi_3)\boldsymbol{\xi_1} && k = 0,\dots,p_{e_{11}}, \\
\psi_k^{e_{12}} &= l_0(\xi_1)L_k(\xi_2)l_1(\xi_3)\boldsymbol{\xi_2} && k = 0,\dots,p_{e_{12}}.
\end{aligned}
\tag{5.11}
$$

  The edge functions in H^1 space vanish at all vertices and on all edges but one. Here the situation is similar, but this time not the function value, but the tangential component of $\psi_k^{e_i}$ only vanish on all edges except for e_i. Such edge shape function is said to be associated with this edge. The trace of the function equals to the Legendre polynomial L_k. There are always p_{e_i} edge functions associated with edge e_i .

  Edge functions of the lowest order $k = 0$ are always present. They are called Whitney functions and they form the standard lowest-order basis.

- *Face functions* are defined for each face s_i, $i = 1,\dots,6$ in two groups: $\psi_{k_1,k_2}^{s_i,1}$ with vectors in direction of the first local face coordinate and $\psi_{k_1,k_2}^{s_i,2}$ with vectors in direction of the second local face coordinate. The easiest way how to define them is again to list them all.
  Face s_1:

$$
\begin{aligned}
\psi_{k_1,k_2}^{s_1,1} &= l_0(\xi_1)L_{k_1}(\xi_2)l_{k_2}(\xi_3)\boldsymbol{\xi_2}, \\
&\quad k_1 = 0,\dots,p^{s_1,1},\ k_2 = 2,\dots,p^{s_1,2}+1, \\
\psi_{k_1,k_2}^{s_1,2} &= l_0(\xi_1)l_{k_1}(\xi_2)L_{k_2}(\xi_3)\boldsymbol{\xi_3}, \\
&\quad k_1 = 2,\dots,p^{s_1,1}+1,\ k_2 = 0,\dots,p^{s_1,2}.
\end{aligned}
\tag{5.12}
$$

Face s_2:

$$\begin{aligned}
\psi^{s_2,1}_{k_1,k_2} &= l_1(\xi_1)L_{k_1}(\xi_2)l_{k_2}(\xi_3)\boldsymbol{\xi_2}, \qquad (5.13)\\
&\quad k_1 = 0,\ldots,p^{s_2,1},\ k_2 = 2,\ldots,p^{s_2,2}+1,\\
\psi^{s_2,2}_{k_1,k_2} &= l_1(\xi_1)l_{k_1}(\xi_2)L_{k_2}(\xi_3)\boldsymbol{\xi_3},\\
&\quad k_1 = 2,\ldots,p^{s_2,1}+1,\ k_2 = 0,\ldots,p^{s_2,2}.
\end{aligned}$$

Face s_3:

$$\begin{aligned}
\psi^{s_3,1}_{k_1,k_2} &= L_{k_1}(\xi_1)l_0(\xi_2)l_{k_2}(\xi_3)\boldsymbol{\xi_1}, \qquad (5.14)\\
&\quad k_1 = 0,\ldots,p^{s_3,1},\ k_2 = 2,\ldots,p^{s_3,2}+1,\\
\psi^{s_3,2}_{k_1,k_2} &= l_{k_1}(\xi_1)l_0(\xi_2)L_{k_2}(\xi_3)\boldsymbol{\xi_3},\\
&\quad k_1 = 2,\ldots,p^{s_3,1}+1,\ k_2 = 0,\ldots,p^{s_3,2}.
\end{aligned}$$

Face s_4:

$$\begin{aligned}
\psi^{s_4,1}_{k_1,k_2} &= L_{k_1}(\xi_1)l_1(\xi_2)l_{k_2}(\xi_3)\boldsymbol{\xi_1}, \qquad (5.15)\\
&\quad k_1 = 0,\ldots,p^{s_4,1},\ k_2 = 2,\ldots,p^{s_4,2}+1,\\
\psi^{s_4,2}_{k_1,k_2} &= l_{k_1}(\xi_1)l_1(\xi_2)L_{k_2}(\xi_3)\boldsymbol{\xi_3},\\
&\quad k_1 = 2,\ldots,p^{s_4,1}+1,\ k_2 = 0,\ldots,p^{s_4,2}.
\end{aligned}$$

Face s_5:

$$\begin{aligned}
\psi^{s_5,1}_{k_1,k_2} &= L_{k_1}(\xi_1)l_{k_2}(\xi_2)l_0(\xi_3)\boldsymbol{\xi_1}, \qquad (5.16)\\
&\quad k_1 = 0,\ldots,p^{s_5,1},\ k_2 = 2,\ldots,p^{s_5,2}+1,\\
\psi^{s_5,2}_{k_1,k_2} &= l_{k_1}(\xi_1)L_{k_2}(\xi_2)l_0(\xi_3)\boldsymbol{\xi_2},\\
&\quad k_1 = 2,\ldots,p^{s_5,1}+1,\ k_2 = 0,\ldots,p^{s_5,2}.
\end{aligned}$$

Face s_6:

$$\begin{aligned}
\psi^{s_6,1}_{k_1,k_2} &= L_{k_1}(\xi_1)l_{k_2}(\xi_2)l_1(\xi_3)\boldsymbol{\xi_1}, \qquad (5.17)\\
&\quad k_1 = 0,\ldots,p^{s_6,1},\ k_2 = 2,\ldots,p^{s_6,2}+1,\\
\psi^{s_6,2}_{k_1,k_2} &= l_{k_1}(\xi_1)L_{k_2}(\xi_2)l_1(\xi_3)\boldsymbol{\xi_2},\\
&\quad k_1 = 2,\ldots,p^{s_6,1}+1,\ k_2 = 0,\ldots,p^{s_6,2}.
\end{aligned}$$

Tangential components of all face functions $\psi^{s_i,1}_{k_1,k_2}$, $\psi^{s_i,2}_{k_1,k_2}$ vanish on all edges and all faces with the exception of s_i. There are $(p^{s_i,1}+1)p^{s_i,2}+p^{s_i,1}(p^{s_i,2}+1)$ face functions associated with the face s_i and they are present if $p^{s_i,1} \geq 1$ or $p^{s_i,2} \geq 1$.

- *Bubble functions* $\psi^{b,1}_{k_1,k_2,k_3}$, $\psi^{b,2}_{k_1,k_2,k_3}$, and $\psi^{b,3}_{k_1,k_2,k_3}$ are the last to be added to the basis:

$$\begin{aligned}
\psi^{b,1}_{k_1,k_2,k_3} &= L_{k_1}(\xi_1)l_{k_2}(\xi_2)l_{k_3}(\xi_3)\boldsymbol{\xi_1}, \qquad (5.18)\\
&\quad k_1 = 0,\ldots,p^{b,1},\ k_2 = 2,\ldots,p^{b,2}+1,\ k_3 = 2,\ldots,p^{b,3}+1,\\
\psi^{b,2}_{k_1,k_2,k_3} &= l_{k_1}(\xi_1)L_{k_2}(\xi_2)l_{k_3}(\xi_3)\boldsymbol{\xi_2},\\
&\quad k_1 = 2,\ldots,p^{b,1}+1,\ k_2 = 0,\ldots,p^{b,2},\ k_3 = 2,\ldots,p^{b,3}+1,\\
\psi^{b,3}_{k_1,k_2,k_3} &= l_{k_1}(\xi_1)l_{k_2}(\xi_2)L_{k_3}(\xi_3)\boldsymbol{\xi_3},\\
&\quad k_1 = 2,\ldots,p^{b,1}+1,\ k_2 = 2,\ldots,p^{b,2}+1,\ k_3 = 0,\ldots,p^{b,3}.
\end{aligned}$$

  There are $(p^{b,1}+1)p^{b,2}p^{b,3}+p^{b,1}(p^{b,2}+1)p^{b,3}+p^{b,1}p^{b,2}(p^{b,3}+1)$ bubble functions, which are present only if at least two of three orders $p^{b,1}$, $p^{b,2}$, $p^{b,3}$ are greater or equal to 1.

We do not present figures of local basis functions, but their shape can be seen very clearly from Figures 5.1–5.5, where global basis functions constructed on regular mesh are shown.

Remark 5.2. Section 4.2 presents properties of local basis functions of the space H^1, namely the property of function values being zero on certain types of nodes (vertices, edges and faces) for each class of local basis functions (associated with vertices, edges, faces and element interior). In the case of the $\boldsymbol{H}^{\mathrm{curl}}$ space the idea is similar, with the difference, that we are not speaking about vanishing function values, but just tangential components of the vectors. Tangential components of edge functions vanish on all edges with the exception of the edge to which the function is related. Tangential components of face functions vanish on all edges and all faces but that to which the function is related. And finally, tangential components of bubble functions vanish on all edges and faces. These properties play the essential role in the projection-based interpolation described in Section 2.5 and in the proof of the following theorem.

Theorem 5.2. (Local basis) *Functions (5.11), (5.12)–(5.17) and (5.18) constitute a hierarchic basis of the space* W^{curl} *defined in (5.10).*

Proof. The proof will be performed in a similar way as for Theorem 4.4. All functions (5.11), (5.12)–(5.17) and (5.18) are linearly independent. Consider arbitrary function $\boldsymbol{E} \in W^{\mathrm{curl}}$. In the following we will show, that this function can be expressed as a linear combination of functions (5.11), (5.12)–(5.17) and (5.18). Function $\boldsymbol{E}$ is vector valued, $\boldsymbol{E} = (E_1, E_2, E_3)$. As we have seen above, all local basis functions are defined in such a way, that their vectors are parallel to one

of the axis ξ_1, ξ_2, ξ_3. Therefore the process of finding linear combination of local basis functions to be equal to $\boldsymbol{E}$ can be divided into three parts. In each part we find the linear combination of local basis functions parallel to ξ_1 to be equal to $\boldsymbol{E}^1 = (E_1, 0, 0)$, the linear combination of those parallel to ξ_2 to be equal to $\boldsymbol{E}^2 = (0, E_2, 0)$ and, finally, the linear combination of local basis functions parallel to ξ_3 to be equal to $\boldsymbol{E}^3 = (0, 0, E_3)$.

In the following we will construct the linear combination that constitutes $\boldsymbol{E}^1$. Linear combinations that equal to $\boldsymbol{E}^2$ and $\boldsymbol{E}^3$ can be constructed analogically. First let us find an edge interpolant $\boldsymbol{E}^1_{e_i}$ of $\boldsymbol{E}^1$. From the definition of edge local basis functions (5.11) we can see, that the only edges parallel to ξ_1 and thus relevant for constructing $\boldsymbol{E}^1$ are e_1, e_3, e_9 and e_{11}. From the Definition 5.10 we can see, that the degree of the tangential component of $\boldsymbol{E}$ on e_i is at most p^{e_i}. Tangential components of functions $\psi^{e_i}_0, \ldots, \psi^{e_i}_{p^{e_i}}$ on the edge e_i are Legendre polynomials of degrees from 0 to p^{e_i}. Therefore there are unique coefficients $\alpha^{e_i}_0, \ldots, \alpha^{e_i}_{p^{e_i}}$ such that

$$\boldsymbol{E}_{e_i} = \sum_{k=0}^{p^{e_i}} \alpha^{e_i}_k \psi^{e_i}_k$$

and $\boldsymbol{E}_{e_i}$ is the edge interpolant on edge e_i, meaning that *tangential* components of $\boldsymbol{E}$ and $\boldsymbol{E}_{e_i}$ are equal on edge e_i. By summing appropriate edge interpolants we get edge interpolant

$$\boldsymbol{E}^1_e = \sum_{i \in \{1,3,9,11\}} \boldsymbol{E}_{e_i}$$

in the direction of ξ_1.

Let us proceed to the face functions. Since we are projecting first component of $\boldsymbol{E}$, the only faces relevant are s_3, s_4, s_5 and s_6, always with only one part of face functions[1] $\psi^{s_i,1}_{k_1,k_2}$, $i = 3, \ldots, 6$, $k_1 = 0, \ldots, p^{s_i,1}$, $k_2 = 2, \ldots, p^{s_i,2} + 1$. We can see, that first component of $\boldsymbol{E} - \boldsymbol{E}^1_e$ vanishes on edges e_1, e_3, e_9 and e_{11}. Now $E_1|_{s_i} \in Q_{p^{s_i,1}, p^{s_i,2}+1}$, therefore there are coefficients $\alpha^{s_i,1}_{k_1,k_2}$ such that

$$\boldsymbol{E}^1_{s_i} = \sum_{k_1=0}^{p^{s_i,1}} \sum_{k_2=2}^{p^{s_i,2}} \alpha^{s_i,1}_{k_1,k_2} \psi^{s_i,1}_{k_1,k_2}$$

is the projection of $\boldsymbol{E} - \boldsymbol{E}^1_e$ on the face s_i. This is possible thank to the fact, that $\boldsymbol{E} - \boldsymbol{E}^1_e$ is in the first local face direction polynomial of maximal degree $p^{s_i,1}$ and corresponding components of local basis functions $\psi^{s_i,1}_{k_1,k_2}$ are Legendre polynomials of degrees $0, \ldots, p^{s_i,1}$ and in the second local face direction it is a polynomial

[1] For the second component of $\boldsymbol{E}$ we would use $\psi^{s_1,1}_{k_1,k_2}$, $\psi^{s_2,1}_{k_1,k_2}$, $\psi^{s_5,2}_{k_1,k_2}$ and $\psi^{s_6,2}_{k_1,k_2}$ and for the third component we would use $\psi^{s_i,2}_{k_1,k_2}$, $i = 1, \ldots, 4$.

of maximal degree $p^{s_i,2}+1$ and corresponding components of $\psi^{s_i,1}_{k_1,k_2}$ are Lobatto polynomials of degree $2,\dots,p^{s_i,2}+1$ (and that is sufficient to express $\boldsymbol{E}-\boldsymbol{E}^1_e$ as linear combination, since it vanishes on edges e_1, e_3, e_9 and e_{11}, as it was mentioned before.

By summing particular face interpolants we get the face interpolant

$$\boldsymbol{E}^1_s = \sum_{i=3}^{6} \boldsymbol{E}^1_{s_i}$$

in the direction of ξ_1.

From the previous construction we can see, that $\boldsymbol{E}-\boldsymbol{E}^1_e-\boldsymbol{E}^1_s$ vanish on all edges and on faces s_3, s_4, s_5 and s_6. The last step is to find the coefficients $\alpha^{b,1}_{k_1,k_2,k_3}$ of the linear combination of bubble functions $\psi^{b,1}_{k_1,k_2,k_3}$, $k_1=0,\dots,p^{b,1}$, $k_2=2,\dots,p^{b,2}+1$, $k_3=2,\dots,p^{b,3}+1$ in the direction of ξ_1

$$\boldsymbol{E}^1_b = \sum_{k_1=0}^{p^{b,1}} \sum_{k_2=2}^{p^{b,2}} \sum_{k_3=2}^{p^{b,2}} \alpha^{b,1}_{k_1,k_2,k_3} \psi^{b,1}_{k_1,k_2,k_3}$$

such that $\boldsymbol{E}-\boldsymbol{E}^1_e-\boldsymbol{E}^1_s=\boldsymbol{E}^1_b$. This can be done, since local basis functions $\psi^{b,1}_{k_1,k_2,k_3}$, $k_1=0,\dots,p^{b,1}$, $k_2=2,\dots,p^{b,2}+1$, $k_3=2,\dots,p^{b,3}+1$ vanish on all edges and faces with the exception of s_1 and s_2 and are formed as products of Legendre polynomials of degrees $0,\dots,p^{b,1}$ in the direction of ξ_1 and Lobatto polynomials of degrees $2,\dots,p^{b,2}+1$ and $2,\dots,p^{b,3}+1$ in the directions ξ_2 and ξ_3, respectively.

We have shown, that $\boldsymbol{E}$ can be expressed as a linear combination of local basis functions in the form

$$\boldsymbol{E} = \sum_{d=1}^{3} \left(\boldsymbol{E}^d_e + \boldsymbol{E}^d_s + \boldsymbol{E}^d_b\right),$$

which concludes the proof. □

5.3 Construction of global basis functions

In this section we will describe construction of global basis functions in $\boldsymbol{H}^{\mathrm{curl}}$. The idea is the same as for H^1 basis functions, that were described in Section 4.3. From that reason, we will not go into such details, as we did for the H^1 space. There are two main differences. First, the construction is complicated by the fact, that basis functions are vector-valued. Luckily this does not complicate the situation significantly, since we constructed local basis functions to be nonzero

in one direction only. However, for the face functions, we have to construct two types of basis functions (with two possible directions of the vectors) and also only edges parallel to the vectors are constrained. Second difference is the absence of vertex functions, given by different conformity requirements. Not only we do not have to construct vector basis functions, but we can also ignore constrains of vertices by edges and faces, which simplifies things a little.

5.3.1 Edge functions

Edge functions in $\boldsymbol{H}^{\text{curl}}$ are constructed similarly as in the H^1 space, which has been described in Section 4.3.2. We do not want to repeat unnecessarily, so we will not go into details in features, that are analogical. As in the elliptic case, we will use sets defined in Chapter 3, describing geometric relations in the mesh. We will not, however, use all of them, since there are no basis functions associated to vertices, due to different conformity requirements of the space $\boldsymbol{H}^{\text{curl}}$.

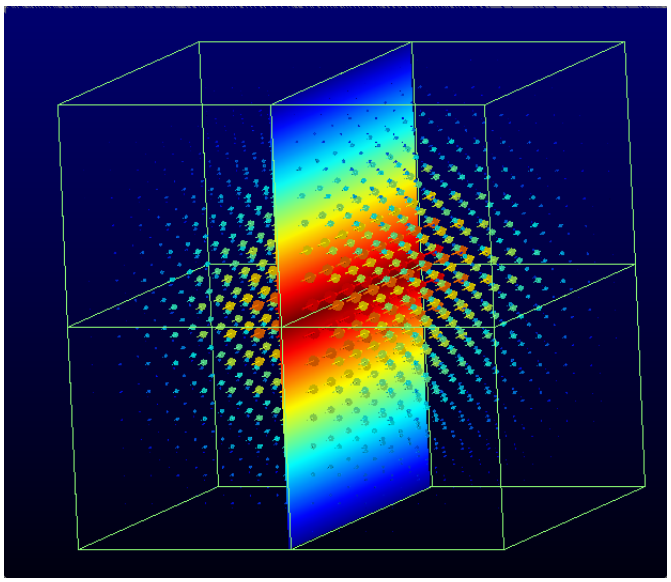


Figure 5.1: The lowest-order $\boldsymbol{H}^{\text{curl}}$ edge function.

Regular mesh

For the regular mesh, the situation is simple. In Figures 5.1 and 5.2 we can see a part of a regular mesh, consisting of four elements only. In the first figure, we can

see the lowest-order edge (Whitney) basis function. Its values are constant in the direction of the edge. In the later figure we can see the higher-order edge basis function. In both figures the vector values are depicted by arrows and in addition there is a cutting plane in the middle with color showing magnitude of the vectors in that place.

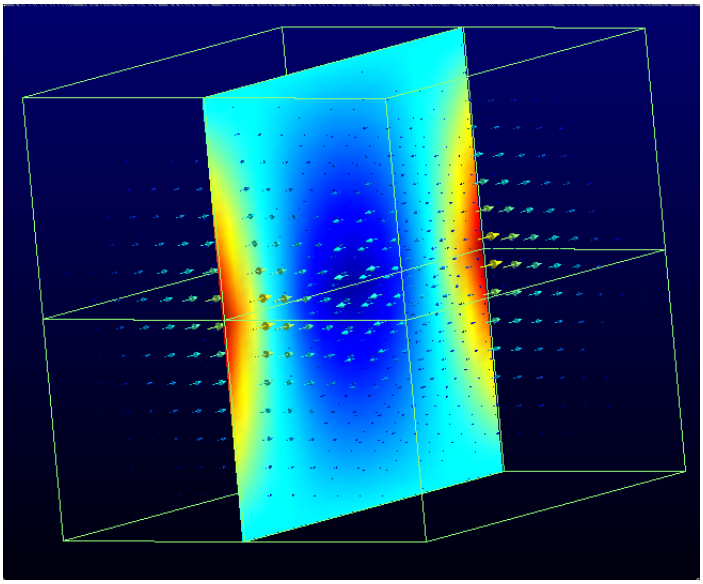


Figure 5.2: Higher-order $\boldsymbol{H}^{\text{curl}}$ edge function.

In both figures, the four shown elements constitute the support of the basis functions. We can see, that on some of the faces on the boundary of the domain, the function values are not zero. However, *tangential* components are zero (vectors are perpendicular to the boundary faces), which is in accordance with the conformity requirements of the space $\boldsymbol{H}^{\text{curl}}$.

Mesh with hanging nodes

In the following we will give a general definition of global edge basis functions. As in the case of the H^1 space, more elements may be involved, but this time we have to consider edge functions only. When we were constructing global edge basis functions in H^1, we first derived coefficients $\delta_{m,o_e}^{k,(q^1,q^2),o_c}$, that describe, how would basis function of order m contribute in the linear combination, that produce function identical to basis function of order k on interval (q^1, q^2) of the edge. For details see Section 4.3.2. We will not repeat the process, since the idea is the

same. The difference, though, is, that the component of the edge function in the direction of the edge is the Lagrange, not the Lobatto polynomial. Values of the coefficients will be therefore different and we denote them as $\bar{\delta}_{m,o_e}^{k,(q^1,q^2),o_c}$.

Definition 5.2. (Global edge basis function) Let e_c be unconstrained edge and let the $C^{Ee}(e_c)$ be calculated according to definitions in Chapter 3. Global edge function $\boldsymbol{E}_k^{e_c}$ associated to e_c, of order $k \leq p^{e_c}$ is defined on element K as follows:

$$\boldsymbol{E}_k^{e_c}|_K = \sum_{e\in K,\ (e_c,\gamma,(q^1,q^2))\in C^{Ee}(e)} \sum_{m=2}^{p_{e_c}} \gamma\, \bar{\delta}_{m,o_e}^{k,(q^1,q^2),o_c}\, \boldsymbol{\Phi}_K^{\mathrm{curl}}(\boldsymbol{\psi}_m^{\hat{e}} \circ \boldsymbol{x}_o^{K,e}), \qquad (5.19)$$

where

$$\hat{e} = \boldsymbol{x}_K^{-1}(e)$$

is the corresponding edge of the reference domain. Function $\boldsymbol{\psi}_k^{\hat{e}}$ is an edge local basis function associated with the edge $\hat{e}$ of the order k. Transformation $\boldsymbol{\Phi}_K^{\mathrm{curl}}$ from the reference element to K has been defined in Section 2.3.4 and the orientation adjustment $\boldsymbol{x}_o^{K,e}$ has been defined in Section 3.4.1.

The definition has the same structure as its counterpart from Chapter 4 and it does not need any further comments.

Remark 5.3. From the same reasons as for the H^1 space, a global edge basis function associated with the edge e_c defined in Definition 5.2 vanishes in all elements of the coarse mesh that do not share edge e_c.

Theorem 5.3. (Conformity of global edge basis function) *Global edge basis function associated with edge e defined in Definition 5.2 has continuous tangential components on all edges and faces of the mesh and therefore it conforms to the space* $\boldsymbol{H}^{\mathrm{curl}}(\Omega)$.

Proof. Analogous to the H^1 case, see Theorem 4.7. □

5.3.2 Face functions

Similarly as for edge functions, we will not go into as many details here as we did for the H^1 face functions in Section 4.3.3. As in the previous section, we do not have to consider vertex functions, which leaves us to edge and face functions. Otherwise we again use the sets, that has been defined in Chapter 3.

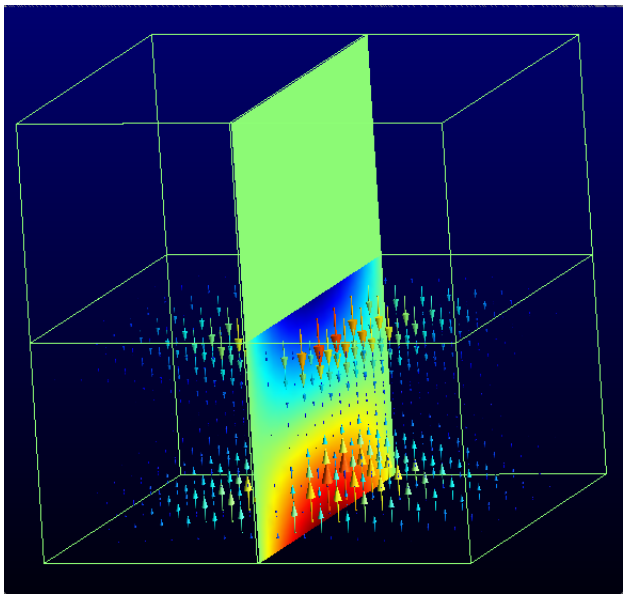


Figure 5.3: $\boldsymbol{H}^{\mathrm{curl}}$ face function.

Regular mesh

As usually, let us first describe the face functions in the regular mesh. In such case, the face function is nonzero on two elements sharing the face only. In Figures 5.3 and 5.4 they are the two bottom elements (we are using the same part of the mesh as for the edge functions).

Unlike in the H^1 space, there are two types of face functions for each face, since there are two directions parallel to the edge and vectors of the function may lay in either of them. In Figures 5.3 and 5.4 we show both possibilities (and also different orders of the basis functions). Note that again, on all boundary faces of the domain the vectors are either zero or they are perpendicular to the face.

Mesh with hanging nodes

Now let us proceed to the most complicated case, construction of global face basis functions on mesh with hanging nodes. In Section 4.3.3 we described this process in detail for the case of the H^1 space. Even though there are some differences (absence of vertex functions, functions are vector-valued), we can skip substantial part of the description, that may be done analogically. It applies especially to construction of coefficients $\bar{\varepsilon}_{m,o_e}^{k_1,k_2,p,(q^1,q^2),d,o_c}$ and $\bar{\zeta}_{m_1,m_2,o_f}^{(q_1^1,q_1^2),(q_1^1,q_1^2),o_c}$, that are again

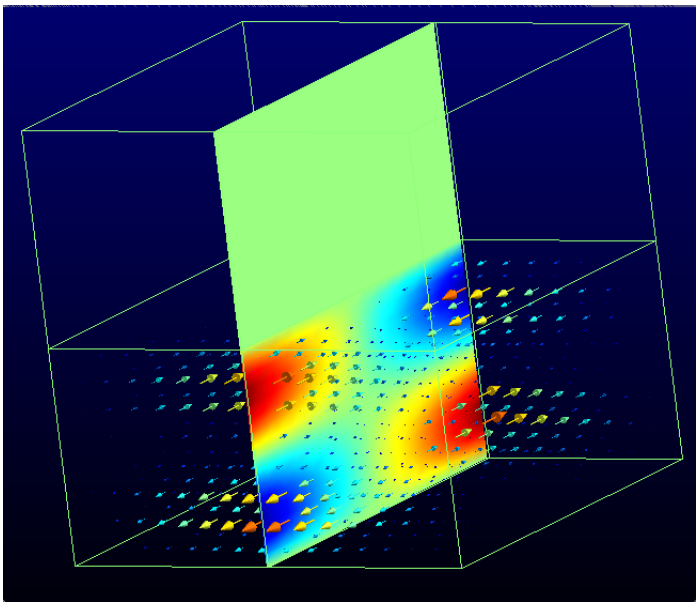


Figure 5.4: Another $\boldsymbol{H}^{\text{curl}}$ face function having different direction of vectors and different order then the on Figure 5.3.

constructed in order to ensure continuity.

There is, however, one more significant difference. In Chapter 3, $C^{Fe}(f_c)$, has been defined to contain all edges lying on the face f_c and further all edges constrained by those edges. All such edges are constrained by f_c. In the $\boldsymbol{H}^{\text{curl}}$ space, there are two variants of face functions for each face with two possible directions of vectors. Of course, face function constrains only edges in the direction of its vectors. Therefore we have to split the set $C^{Fe}(f_c)$ up into two disjoint sets $C^{Fe,1}(f_c)$ and $C^{Fe,2}(f_c)$, each containing edges constrained by the corresponding type of face functions.

Definition 5.3. (Global face basis function) Let f_c be unconstrained face and sets $C^{Fe}(f_c)$ and $C^{Ff}(f_c)$ be calculated according to definitions in Chapter 3. Global face function $\boldsymbol{E}^{f_c,t}_{k_1,k_2}$ associated to face f_c, of orders $k_1 \leq p^{f_c}_1$, $k_2 \leq p^{f_c}_2$ is defined on element K as follows:

$$\boldsymbol{E}^{f_c,t}_{k_1,k_2}|_K = \sum_{e\in K,\ (f_c,\gamma,p,(q^1,q^2),d)\in C^{Ee}(e)} \sum_{m=2}^{p_{e_c}} \gamma \bar{\varepsilon}^{k_1,k_2,p,(q^1,q^2),d,o_c}_{m,o_e} \boldsymbol{\Phi}^{\text{curl}}_K(\boldsymbol{\psi}^{\hat{e}}_m \circ \boldsymbol{x}^{K,e}_o) +$$

$$+ \sum_{f\in K,\ (f_c,(q^1_1,q^2_1),(q^1_1,q^2_1))\in C^{Ff,t}(f)} \sum_{m_1=2}^{p^1_{f_c}} \sum_{m_2=2}^{p^2_{f_c}} \bar{\zeta}^{(q^1_1,q^2_1),(q^1_1,q^2_1),o_c}_{m_1,m_2,o_f} \boldsymbol{\Phi}^{\text{curl}}_K(\boldsymbol{\psi}^{\hat{f}}_{m_1,m_2} \circ \boldsymbol{x}^{K,f}_o),$$

where

$$\hat{e} = \boldsymbol{x}_K^{-1}(e) \quad \text{and} \quad \hat{f} = \boldsymbol{x}_K^{-1}(f)$$

are the edge and the face on the reference cube, corresponding to the edge e and the face f in the physical mesh, adjacent to the element K. Symbol $\boldsymbol{\psi}^{\hat{e}}$ represents a local edge basis function associated to edge $\hat{e}$. Similarly, $\boldsymbol{\psi}^{\hat{f}}$ stands for a local face basis function associated to face $\hat{f}$. Finally, $t \in \{1,2\}$ represents one of two possible directions of function vectors, as defined in Section 5.2.1. The transformation $\boldsymbol{\Phi}_K^{\text{curl}}$ from the reference element to K has been defined in Section 2.3.4 and the orientation adjustments $\boldsymbol{x}_o^{K,e}$ and $\boldsymbol{x}_o^{K,f}$ have been defined in Section 3.4.

Remark 5.4. Global face basis function associated with face f_c defined in Definition 5.3 vanishes in all elements of the coarse mesh that do not share face f_c.

Theorem 5.4. (Conformity of global face basis function) *Global face basis function associated with face f defined in Definition 5.3 has continuous tangential components on all edges and faces of the mesh and therefore it conforms to the space $\boldsymbol{H}^{\text{curl}}(\Omega)$.*

Proof. Analogous to the H^1 case, see Theorem 4.8. □

5.3.3 Interior functions

Global interior basis functions (also called bubble functions) are again defined on one element only, so no effort has to be done to ensure the conformity. An example of an interior function can be seen in Figure 5.5. Again, vectors are either zero on each face of the element, or perpendicular to it.

A global interior basis function is defined as follows.

Definition 5.4. (Global interior basis function) Global interior basis function $\boldsymbol{E}^{K,t}_{k_1,k_2,k_3}$ defined on Ω and associated to an element K, of orders $k_1 \leq p_1^K$, $k_2 \leq p_2^K$ and $k_3 \leq p_3^K$, is defined as $\boldsymbol{K}|_{K'} = 0$ on all elements $K' \neq K$ and

$$\boldsymbol{E}^{K,t}_{k_1,k_2,k_3}|_K = \boldsymbol{\Phi}_K^{\text{curl}}(\boldsymbol{\psi}^{b,t}_{k_1,k_2,k_3}), \tag{5.20}$$

where $\boldsymbol{\psi}^{b,t}$ is a local bubble function and $t \in \{1,2,3\}$ represents one of three possible directions of function vectors, as defined in Section 5.2.1. Transformation $\boldsymbol{\Phi}_K^{\text{curl}}$ from the reference element to K has been defined in (2.3.4).

Remark 5.5. Again, global interior functions vanish in all elements but one.

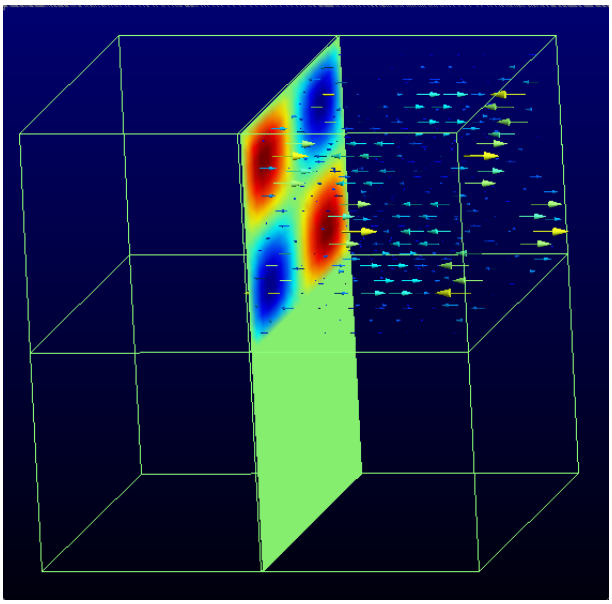


Figure 5.5: $\boldsymbol{H}^{\mathrm{curl}}$ interior function.

Theorem 5.5. (Conformity of global interior basis function) *Global bubble basis function associated with element K defined in Definition 5.4 has continuous tangential components on all edges and faces of the mesh and therefore it conforms to the space* $\boldsymbol{H}^{\mathrm{curl}}(\Omega)$.

Proof. It is obvious from the definition, since we construct the bubble function on one element only, the tangential components vanish. □

CHAPTER

SIX

NUMERICAL QUADRATURE

Integration of higher-order basis functions is another issue, that is not as straightforward as it may seem. In traditional low-order FEM codes, the bulk of the computational time is the solution of the resulting system of linear equations. The assembling of the stiffness matrix, which comprises evaluations of integrals in the weak formulation, is just negligible in comparison. Therefore there is no need to think about it.

In the case of higher-order elements, the situation is different. The question of numerical integration becomes an important issue for higher-order shape functions, especially in three dimensions. If we want to calculate stiffness matrix exactly[1], the order of the quadrature has to correspond to the degree of polynomial basis functions on the given element. As the polynomial degree of the basis grows, number of integration points in the appropriate integration rule grows as well, which makes the calculation more expensive.

Note that all integration in our code is being performed on the reference domain, which is possible thanks to the reference mapping technique, which is described in Section 2.3.2. It means, that we never transform integration points into the physical domain, rather than that we transform the integrand in such a way, that it can be integrated on the reference domain. Therefore all the quadrature techniques mentioned in the following text are designed for the reference cube only.

[1] In reality (when elements in the mesh are not cuboids), we are unable to calculate integrals exactly anyway, since the inverse reference mapping is not polynomial and therefore it can not be evaluated exactly by the numerical quadrature. On the other hand, if the elements are not degenerated, this error does not spoil the convergence of the method.

In this chapter, ideas from [30] and [21] are used, extended to 3D and simplified to a form suitable for our implementation. This chapter is based on paper [26].

6.1 Gauss quadrature rules

The choice of the quadrature type is very important. Even though two quadrature rules integrate exactly polynomials up to certain degree, their performance can differ significantly when integrating non-polynomial functions (which is in reality often the case, since the inverse Jacobi matrix is non-polynomial for general mesh elements). The usual choice for higher-order integration are Gauss quadrature rules. The 1D integral over the segment $(-1,1)$ is then approximated by the formula

$$\int_{-1}^{1} f(\xi)\mathrm{d}\xi \approx \sum_{i=1}^{n} w_{n,i} f(\xi_{n,i}). \tag{6.1}$$

The integration points $\xi_{n,i}$ and the weights $w_{n,i}$ can be found by inserting linearly independent functions with known integrals (such as polynomials of different degrees) into the equation. We have $2n$ unknowns, so if we insert polynomials of degrees $0,1,\ldots,2n-1$, we obtain a system of $2n$ linearly independent nonlinear equations with $2n$ unknowns. If solved, the resulting rule will be exact for all polynomials of degree up to $2n-1$, thanks to the linearity of polynomials.

6.2 Product quadrature rules

Since the integration is performed on reference element, which is a cube in our case, the most natural choice of the integration rules is to use tensor products of 1D Gauss rules.

6.2.1 Computational cost of the integration

In this chapter, for simplicity, let us stick to the case of elliptic problems. Now let us estimate the computational cost of the calculation of the local stiffness matrix. Assume that we have a cubic element of polynomial degree p in all directions. Such element has to be equipped with basis functions with polynomial degree up to p in each direction. In total we have $(p+1)^3$ basis functions on such element, since it has this number of degrees of freedom (see Chapter 4 for details). In order to evaluate the local stiffness matrix, we have to calculate the integral in (4.12) for each pair of basis functions on this element. Therefore we have $(p+1)^3 \times (p+$

$1)^3$ evaluations of the integral. We integrate products of basis functions, whose polynomial degree is up to p^2 in each direction. Quadrature rule that will calculate these integrals exactly (obtained as a product of 1D Gauss quadrature rules) has approximately p^3 points (each 1D rule has approximately p points). The value of the integral is calculated as a sum of products of function values in each point of the integration rule, so if we are interested in the asymptotic complexity of the evaluation, we can estimate the time to calculate an integral as $O(p^3)$. Since we have to do $(p+1)^3 \times (p+1)^3$ such calculations, total asymptotic complexity of the evaluation of the local stiffness matrix is $O(p^9)$.

It is obvious, that this is extremely unfavorable and makes the assembling procedure very time-consuming for higher values of p. For the numerical solution of partial differential equations in more than 3 dimensions, this estimate is even more severe and makes it virtually impossible to use such straightforward approach towards the evaluation of integrals. For truly high-dimensional calculations, which are becoming more and more desirable for example for financial problems, completely different ways towards estimation of the values of the integrals, such as Smoljak's schemes are used. For more details see Section 6.5.

6.2.2 Hierarchic elements

We have seen in the previous section, that the simplest implementation of the numerical quadrature leads to extremely high computational costs. In the following we describe several ideas how to make the calculation more economical. The first one is very simple. If we use hierarchic rather then nodal basis, which is our case, we can take advantage of the structure of the basis functions. In the hierarchic case, the basis of the element of order p is obtained by adding several polynomial functions of degree p to the basis of the element of order $p-1$. Therefore, the basis consist of polynomials of various degrees from 1 up to p and obviously it would be wasteful to integrate the product of two low-degree polynomials with quadrature rule which is exact for product of polynomials of degree p. It would be better to pick different quadrature rule for each pair of shape functions according to their degree.

As we consider basis functions on cube (cube is the reference domain), they have different degrees in each direction. Assume we have to calculate product of two functions of degrees (p_x, p_y, p_z) and (q_x, q_y, q_z). Obviously, the rule capable of exact calculation is of order $(p_x+q_x, p_y+q_y, p_z+q_z)$. This approach, however, has a slight drawback. From the efficiency reasons, the system HERMES precalculates the values of the shape functions in the integration points of the particular rule. If we precalculated values of all shape functions for all rules of order (p_x, p_y, p_z),

$p_x, p_y, p_z \in \{1, \ldots, P\}$, where P is the maximal degree of polynomials used in the basis, the size of the tables would occupy a big portion of the computer memory.[2] Possible solution of this problem is to use only quadrature rules with the same order in all directions, i.e. instead of the rule of order (p_x, p_y, p_z) we use the rule of order (p_m, p_m, p_m), where $p_m = \max(p_x, p_y, p_z)$. The advantage is that we do not precalculate so many quadrature rules. The drawback is that we use schemes with more integration points than necessary, which slow down the process of assembling. The comparisons can be found in Section 6.6.

6.3 Alternative approaches to quadrature

In the previous section we described, how the simple numerical quadrature works. We have seen, however, that this approach may lead to quadrature rules with very high number of integration points and therefore to long time of calculation. In this section we want to describe two different approaches towards the numerical quadrature. The first one is based upon the works [30] and [21]. Ideas used there for 2D basis functions are adapted to our slightly different approach, which allows it's substantial simplification. Thanks to it, the extension to 3D is possible without much trouble.

The second alternative is presented mainly for reference. Smoljak's schemes are used for integration in partial differential equations in more dimensions, where all conventional approaches fail due to the "curse of dimensionality". If the standard product integration rule (even with just two points in each dimension) is used, the total number of integration points would rise exponentially with number of dimensions. It seems that the only solution is to construct sparse integration meshes, which do not integrate exactly, but, if constructed properly, convergence is not spoiled. It seems, however, that this idea starts to be useful for truly high-dimensional problems and is not that suitable for 3D.

6.4 Reordering of quadrature

So far it seemed, that the right way is to optimize the amount of work needed for calculation of every single integral, which can be achieved by selecting different integration rules with appropriate order. Here the approach is different. We as-

[2] Not speaking about the time needed to perform the precalculation. It would be impossible to precalculate all tables at the beginning. Even though we implemented system of precalculation on demand, where only those tables, which are really needed, are precalculated on the fly, the amount of memory consumed is still huge.

semble the whole local stiffness matrix at once, using the integration rule capable of integration of functions of maximal degree on the element. It means, that we in fact over-integrate functions of lower order. This waste of computational time is justified by bigger saving in different way.

We can use the fact, that both basis functions and integration rules are constructed as Cartesian products of 1D functions and integration rules. Thanks to this structure, we can reorder the whole calculation, save some results into auxiliary fields and use them multiple times. The core idea of this approach is that even though each basis function is different, they are all generated as combination of relatively small set of 1D functions. Therefore, for higher order there can be thousands of basis functions, but they can be divided into ten groups only, in which all functions are created by the same function in the x variable. In some sense this part of integral can be calculated only once for the whole stiffness matrix comprising many integrals.

One has to realize, that the idea is not just a simple change of the order of integration, like in Fubini's theorem. Here we not only split the integration into three successive 1D integrations, but also split the integrand to a product of three functions of one variable (x, y and z). Those two operations together allow us to precalculate auxiliary fields, which are used during the calculation multiple times. This saves a significant amount of computational time, as will be shown in the following.

6.4.1 The algorithm

In the articles [30], [21], the authors distinguish between vertex, edge and bubble basis functions and use slightly different algorithm for each group. Our algorithm does not do that and treats all types of basis functions in the same way. We consider basis functions on the reference domain $K = [-1,1]^3$ in the form

$$F_{k_1,k_2,k_3}(\xi_1,\xi_2,\xi_3) = f^1_{k_1}(\xi_1) f^2_{k_2}(\xi_2) f^3_{k_3}(\xi_3), \tag{6.2}$$

where $(k_1,k_2,k_3) \in M = \{1,\dots,n_1\} \times \{1,\dots,n_2\} \times \{1,\dots,n_3\}$. Our goal is to calculate all the integrals

$$\int_K F_{\boldsymbol{k}}(\boldsymbol{\xi}) F_{\boldsymbol{k}'}(\boldsymbol{\xi}) Z(\boldsymbol{\xi}) \,\mathrm{d}\boldsymbol{\xi}, \tag{6.3}$$

where $\boldsymbol{k},\boldsymbol{k}' \in M$. The integrals will be approximated by one quadrature rule obtained as a product of three 1D rules with sufficiently high order in each direction.

Individual 1D rules may have different order:

$$\begin{aligned} R_1 &= \{(w_i^1, \xi_i^1), \quad i = 1, \ldots, m_1\}, \\ R_2 &= \{(w_i^2, \xi_i^2), \quad i = 1, \ldots, m_2\}, \\ R_3 &= \{(w_i^3, \xi_i^3), \quad i = 1, \ldots, m_3\}, \end{aligned}$$

where w_i^j stands for the weight and ξ_i^j for the integration point. The compound rule then has the form:

$$R = \{(w_{i_1}^1 w_{i_2}^2 w_{i_3}^3,\ (\xi_{i_1}^1, \xi_{i_2}^2, \xi_{i_3}^3)),\ i_1 = 1, \ldots, m_1,\ i_2 = 1, \ldots, m_2,\ i_3 = 1, \ldots, m_3\},$$

the number of integration points being $m = m_1 m_2 m_3$. Using the product structure of basis functions and integration rules, the integral from (6.3) can be approximated as

$$\sum_{i_1=1}^{m_1} \sum_{i_2=1}^{m_2} \sum_{i_3=1}^{m_3} w_{i_1}^1 w_{i_2}^2 w_{i_3}^3 \, f_{k_1}^1(\xi_{i_1}^1) f_{k_2}^2(\xi_{i_2}^2) f_{k_3}^3(\xi_{i_3}^3) \, f_{k_1'}^1(\xi_{i_1}^1) f_{k_2'}^2(\xi_{i_2}^2) f_{k_3'}^3(\xi_{i_3}^3) Z(\xi_{i_1}^1, \xi_{i_2}^2, \xi_{i_3}^3).$$

Now the summation can be reordered:

$$\sum_{i_1=1}^{m_1} w_{i_1}^1 f_{k_1}^1(\xi_{i_1}^1) f_{k_1'}^1(\xi_{i_1}^1) \sum_{i_2=1}^{m_2} w_{i_2}^2 f_{k_2}^2(\xi_{i_2}^2) f_{k_2'}^2(\xi_{i_2}^2) \sum_{i_3=1}^{m_3} w_{i_3}^3 f_{k_3}^3(\xi_{i_3}^3) f_{k_3'}^3(\xi_{i_3}^3) Z(\xi_{i_1}^1, \xi_{i_2}^2, \xi_{i_3}^3).$$

Let us introduce an auxiliary field $G(k_3, k_3', i_1, i_2)$, where

$$G(k_3, k_3', i_1, i_2) = \sum_{i_3=1}^{m_3} w_{i_3}^3 f_{k_3}^3(\xi_{i_3}^3) f_{k_3'}^3(\xi_{i_3}^3) Z(\xi_{i_1}^1, \xi_{i_2}^2, \xi_{i_3}^3). \tag{6.4}$$

It is important to realize, that the just defined term really depends only on k_3, k_3', i_1 and i_2. Indeed, all terms depending on k_1, k_1', k_2 and k_2 were put in front of the last sum and i_3 is being summed over.

Similarly, let us introduce another auxiliary field $H(k_2, k_2', k_3, k_3', i_1)$:

$$H(k_2, k_2', k_3, k_3', i_1) = \sum_{i_2=1}^{m_2} w_{i_2}^2 f_{k_2}^2(\xi_{i_2}^2) f_{k_2'}^2(\xi_{i_2}^2) G(k_3, k_3', i_1, i_2). \tag{6.5}$$

This field depends on k_2 and k_2' in addition, but, thanks to the summation, it does not depend on i_2. Now the integral (6.3) can be approximated as

$$\int_K F_{\boldsymbol{k}}(\boldsymbol{\xi}) F_{\boldsymbol{k}'}(\boldsymbol{\xi}) Z(\boldsymbol{\xi}) \, \mathrm{d}\boldsymbol{\xi} \approx \sum_{i_1=1}^{m_1} w_{i_1}^1 f_{k_1}^1(\xi_{i_1}^1) f_{k_1'}^1(\xi_{i_1}^1) H(k_2, k_2', k_3, k_3', i_1). \tag{6.6}$$

When generating the (stiffness) matrix of the integrals, we first precalculate the field G, than the field H and finally use it to calculate all the integrals (6.6), where $\boldsymbol{k}, \boldsymbol{k}' \in M$.

6.4.2 Asymptotic analysis

Now let us estimate the amount of work needed to generate the stiffness matrix according to the above described procedure. The numerical comparisons are presented in Section 6.6, here we want to do just a rough estimate. As in Section 6.2.1, we assume, that the polynomial degree of our basis functions is up to p in each direction. Therefore we have p^3 functions and the 1D integration rules, that comprise the final integration rule, have approximately p integration points.

In Section 6.2.1, we approximated the work needed to generate the stiffness matrix to $O(p^9)$. In the algorithm described above, we first precalculate field G, which requires asymptotically $O(p^4)$ operations. Than the field H is precalculated, which requires $O(p^5)$ operations. That should be negligible in comparison with the main part, which are calculations using the formula (6.6). There is p^3 functions, therefore we have to calculate p^6 integrals. But in the formula (6.6) there is only one summation, with respect to i_1. Other summations are hidden in the auxiliary fields. Therefore the complexity of this part is $O(p^7)$.

This seems to be promising, but, on the other hand, we have to realize, that degrees p are usually relatively small (up to 10 in our code), so the asymptotic analysis is not sufficient to prove the value of the proposed algorithm. Comparisons of real numbers of operations needed to calculate the stiffness matrix are presented in Section 6.6.

6.5 Sparse schemes

The idea of sparse schemes was first introduced by Smolyak in [41]. The goal of this approach is to construct an integration grid, similar to the simple product grid, but with fewer points. The reason, why this is possible, is that slight under-integration does not always spoil the convergence. However, this is a rather complicated matter from both theoretical and practical point of view and therefore we will restrict ourselves to this brief description.

Moreover, from the experiments and comparisons we made it seems that this approach is not the most successful for problems in three dimensions. It's role starts to be vital for problems in much more dimensions, which arise in various fields including financial math. There, sparse grids seems to be the only method capable to cope with the "curse of dimensionality".

6.6 Comparisons

In this section we want to compare different approaches to quadrature with respect to the number of operations needed and to the CPU time. First of all we want to show, that the CPU time needed to assembly the stiffness matrix (and, of course, most of this time is consumed by numerical integration) is significant for calculations in 3D. This justifies the effort done to speed up the integration process.

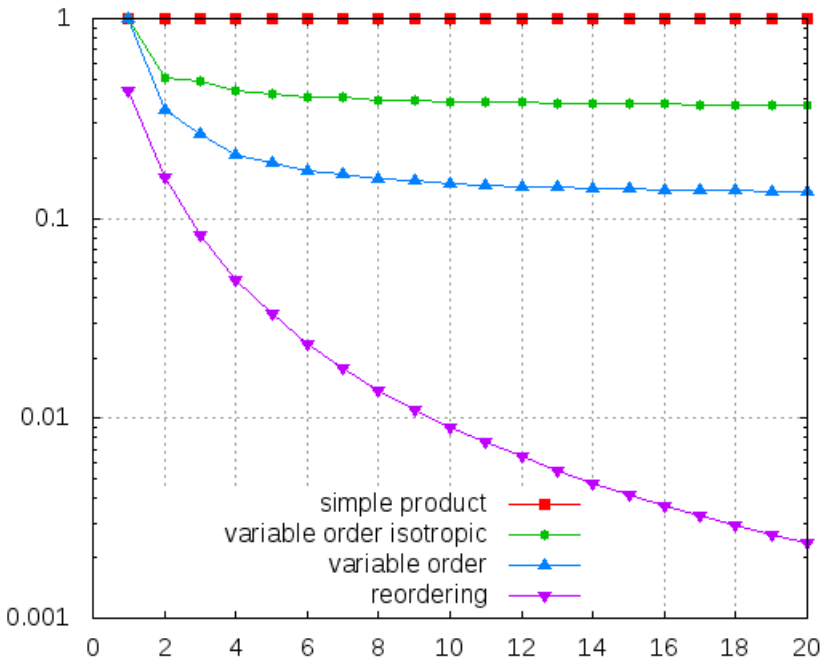


Figure 6.1: The comparison of the performance of described methods with respect to the simple product method. The horizontal axis corresponds to the order of elements and the vertical one shows the ratio N/N_{simple}, where N_{simple} and N stand for the numbers of operations of the simple product method and of the particular one, respectively.

6.6.1 CPU time of assembling

The solving process in our code has two main parts, assembling and solving the stiffness matrix. It is very complicated to compare the CPU time needed to per-

form each part, because it very strongly depends on the problem setting. It depends not only on the number of elements of the mesh and the orders used in the finite element space, but also on the structure of the mesh and on the presence of hanging nodes.

So far we did not work out a systematic and representative comparison of CPU times for different problems, but sometimes the assembling time exceeds the time needed to solve the resulting linear system, so faster quadrature can be very welcomed in some cases.

6.6.2 Performance of different quadrature techniques

In Figure 6.1 we can see a comparison of number of operations needed to assembly the mass matrix for elements of various orders. On the graph we can see the ratios of the number of operations of individual methods with respect to the simple product method (therefore it is 1 for all polynomial degrees in the simple product case.) *Simple product* method is the simplest, where each individual integral is calculated by the same rule, whose order is given by the order of the element. This is, of course, extremely expensive. *Variable order* methods use different order for each integral depending on order of integrated functions. *Isotropic* variant uses the same order for all three directions, as it was described in Section 6.2.2. Finally, the curve described as *reordering* corresponds to the method, that has been described in Section 6.4. We chose to calculate the number of operations for mass matrix, but for the stiffness matrix, the ratios would be similar.

6.6.3 Conclusions

We have shown, that the concept of reordering of summation works well and decreases the number of operations needed to construct the local stiffness matrix. It's effect grows with growing order of elements. Even though incorporating into the code might bring certain complications (we have to keep in mind, that the method is not just a reordering of summation for one integral, but that generates whole element's stiffness matrix at once), it is definitively worth considering.

CHAPTER

SEVEN

COMPUTER IMPLEMENTATION

In the previous chapters we presented theoretical background and described algorithms used for practical implementation of *hp*-adaptive finite element method for 3D problems. Algorithms are presented in a way, which is suitable to theoretical analysis. In this chapter we would like to make few remarks regarding practical implementation, as it is done in HERMES 3D. Of course, the computer code is rather extensive and it would make no sense to describe it as a whole. Instead of that we will focus on few areas, where the algorithmization is highly nontrivial and may have a huge impact to the resulting code, not only from the viewpoint of time-efficiency, but also regarding further maintainability.

7.1 Meshes with hanging nodes

The idea of dealing with hanging nodes in the mesh has been described in detail in Chapter 3. The division of an element has been described there as removing of one element from the mesh and adding two its sons instead. This is not exactly what we do in the code. Elements are never removed from the mesh, we only set an active flag to false. This approach has several advantages. It allows us to keep the structure of the mesh, when the divided element (father) has access to indices of its sons. In addition to this, we keep similar structure of faces. Again, we do not remove the divided faces, we only mark them as non-active and keep indices of its sub-faces. These structures allows us to use recursive algorithms to traverse the mesh. Moreover, when element is to be unrefined (coarsened), we can simple remove its sons and mark bigger element as active again.

7.1.1 Recursive data structures

Let us describe mesh data structures very briefly. Since the code is written in C++, most of the entities of the mesh (elements, faces, edges, vertices, boundaries, etc.) are declared as classes. For the implementation of the algorithms it is extremely important to be able to follow relations in the mesh (such as finding sons of refined elements, faces of elements, neighboring elements, vertices of elements or faces, elements and faces that are defined by given set of vertices, etc.), there are dozens of different types of queries, that the system has to be able to handle.

An obvious solution of these needs would be to connect corresponding objects by pointers. When pointers are used, we do not have to allocate memory in advance – everything can be done dynamically as the number of mesh entities grows during the adaptivity process. Pointers, however, suffer from various drawbacks. Therefore we use indexing of mesh entities and Judy[1] arrays to manage the data. Judy arrays have many advantages. They combine sparse dynamic arrays with hash tables to optimize speed, memory efficiency and scalability.

Some of the demanded queries, such as to find vertices of the given element, are simple and straightforward. The class modeling a mesh element contains an array of indices of its vertices and with this index in hand, one can retrieve vertex objects from the array. Other queries, such as to retrieve a face given by four vertex indices or to retrieve the vertex that lies between two given vertices, are more complex. However, thanks to the Judy arrays they can by simply implemented. All we have to do is to use an extra hash table for each such relation.

7.2 Assembling

In most of the FEM codes it is usual to separate the local and the global assembly step. First local stiffness matrices are created for individual elements and then global stiffness matrix is created, using some kind of connectivity information. Our approach is slightly different. During the process of resolving the constrains, we obtain information attached to each node (vertex, edge, face and element interior), a set of quadruples consisting of index of degree of freedom, index of local basis function, its orientation variant and coefficient, with which it should be multiplied. Now we can, for each element, perform double loop over this set and for each pair evaluate the weak form. The result of the integral is added to the appropriate entry of the global stiffness matrix. The correct row and column correspond to the indices of degrees of freedom associated with it. We can see, that no local

[1] `http://judy.sourceforge.net`

stiffness matrices are constructed and also global basis functions are in fact not constructed. Assembling is carried out element by element and global stiffness matrix is updated according to assembly information, constructed for each node.

7.3 Automatic adaptivity algorithm

The automatic adaptivity algorithm lies in the very core of the *hp*-FEM code. It's purpose is to identify elements, where the solution seems not to be satisfactorily resolved and to propose an optimal way to deal with. In the case of *h*-adaptivity, the situation is quite straightforward. The only thing we need is to identify elements, where the solution has a higher error. Then we can simply refine these elements, obtain a new mesh and continue with the process. There are various sophisticated ways how to estimate the error on individual element which work fine for the *h*-adaptivity. They are, however, not very useful for the case of *hp*-adaptivity. The problem is, that they produce only one number as an output (the error estimate). That is enough to identify problematic elements and also to refine them, if we have only one option, which is the case of the *h*-adaptivity. For the *hp*-adaptivity, more information is needed. We have to know something about the *shape* of the error to be able to choose from wide range of refinement candidates available: we can increase the order of the element (or even imply different order in each direction) or refine the element and choose different orders for each of it's sub-elements.

There is no way, how an estimator producing one number only could guide the *hp*-adaptivity. There are attempts to design similar estimators, that would also decide what is the best way to refine, or at least whether it is better to refine the element and keep the order small or to increase the order (see, e.g. [16]). Such methods may be successful for specialized problems (mainly elliptic), but they are not robust enough for the use in more general setting, like multi-physics problems, where more physical fields with completely different behavior are coupled together. Extreme complexity of such problems together with the lack of theoretical results lead to skepticism about further development of general, robust enough estimator.

At the present moment, we use the concept of a reference solution. This is a solution calculated on a finer grid and therefore more accurate. The difference between the solution and the reference solution then can be used as an error estimate. The main advantage is the robustness, this approach should give reasonable results for most types of physical fields. The main disadvantage is the computational cost, which is enormous – the cost to calculate the reference solution is bigger than the cost of the solution itself. This is of course far from optimal, but, on the other

hand, it is not as bad as it may seem. The reference solution is very expensive, but it produces hp-meshes, that use only a small fraction of degrees of freedom that h-mesh would need. Effort should be made to find another way of estimating the shape of the error, but even with the reference solution, the hp-adaptivity has respectable results. The automatic adaptivity algorithm may be described as follows:

1. Start with a coarse initial mesh $\tau_{h,p}$.
2. Calculate a solution $u_{h,p} \in V_{h,p}$ on the mesh $\tau_{h,p}$.
3. Construct an enriched finite element space $V_{ref} \supset V_{h,p}$ and find a solution $u_{ref} \in V_{ref}$.
4. Construct the error function $e_{h,p} = u_{ref} - u_{h,p}$ and calculate its energy norm E_i on each element K_i.
5. Calculate the approximation of the global error

$$E = \sum_{K_i \in \tau_{h,p}} E_i.$$

   If E is smaller than prescribed tolerance, $u_{h,p}$ is the final solution of the algorithm.
6. From the elements, that has not yet been considered and marked for refinement take the element $K_i \in \tau_{h,p}$ with the highest error.
7. Construct a list of refinement candidates for this element.
8. Project the reference solution on each of the candidates. Choose the candidate with smallest projection error and mark the element for refinement.
9. According to prescribed criterion, if not enough elements were marked for refinement, continue with step 6.
10. Perform chosen refinements for all marked elements and create a new mesh.
11. Continue with step 2.

Refinement candidates

Special attention has to be given to step 7 – the creation of the list of refinement candidates. The question is, whether the list should be large (then we have to spend a lot of time performing interpolations of the reference solution), or small (then we risk that the convergence may be slower, since we will not choose the optimal refinement).

There are many ways, how candidates may be constructed. The first way is increasing the order of the element – it may be increased by 1, by 2 (or more), or we can increase the order in some directions only. The second possibility is to split the element to more sub-elements. It can be halved (3 possible ways), divided to 4 (another 3 ways) or 8 sub-elements. Now there is question of redistributing orders. Even if we consider the case of 8 sub-elements only, if we wanted to employ all possible combinations of 3 different smaller orders to be issued to individual sub-elements, we would have 3^8 possibilities how to do it. And if we would like to consider anisotropic orders, we would end up with tens or even hundreds of thousands of candidates (if all possibilities of choice were combined).

It is hard to set objective criterion to decide the optimal way. But according to our simulations, limiting the number of candidates does not damage the convergence significantly. In our experiments we used anisotropic orders only for candidates without elements splitting (increasing the order in different ways) and we limited the number of different order combinations for element splitting. This area should be, however, studied more rigorously to find an optimal strategy, although it may be problem-dependent.

7.4 Solution of the linear system

The final step of the calculation is solving a very large set of linear equations with relatively sparse matrix. Of course, using meshes with hanging nodes results in more dense matrices, since the support of basis functions involve more elements and therefore there are more "interactions" between individual global basis functions. Despite this fact, matrices still may be described as sparse.

As usually, there are two main possibilities to solve the system, to use iterative (see e.g. [38]) or direct (see e.g. [13]) solvers. Both of them have well known advantages and drawbacks. We usually prefer to use the direct solver UMFPACK[2] because of the possibility to use it for various types of matrices resulting from different physical fields, without the necessity of employing preconditioners to

[2] `http://www.cise.ufl.edu/research/sparse/umfpack`

ensure convergence of the method. But system HERMES allows to use any type of method, the choice is on the user.

CHAPTER

EIGHT

NUMERICAL EXAMPLES

In this chapter we want to present some numerical examples, that prove abilities of the implementation of the method. Their aim is not to calculate real engineering problems, but to show advantages of the *hp*-adaptivity on meshes with arbitrary-level hanging nodes over simpler methods.

8.1 Distribution of electrostatic potential in the Fichera corner domain

The first example to be presented is finding the distribution of the electrostatic potential in Fichera corner domain. The problem is to solve the Poisson's equation in a domain, where it exhibits singularity. It will show the idea, that it is most efficient to use small low-order elements close to the singularity and large higher-order elements in areas, where the solution is smooth. Of course, the mesh is a product of completely automatic adaptivity process, that has been described in the previous text. The problem is given by the following equation:

$$\begin{aligned} -\Delta u &= f \quad \text{in } \Omega, \\ u &= u_D \quad \text{on } \partial\Omega, \end{aligned}$$

where $\Omega = (-1,1)^3 \setminus [0,1]^3$ and f and u_D are chosen to comply with the exact solution

$$u(x_1,x_2,x_3) = (x_1^2+x_2^2+x_3^2)^{1/4}.$$

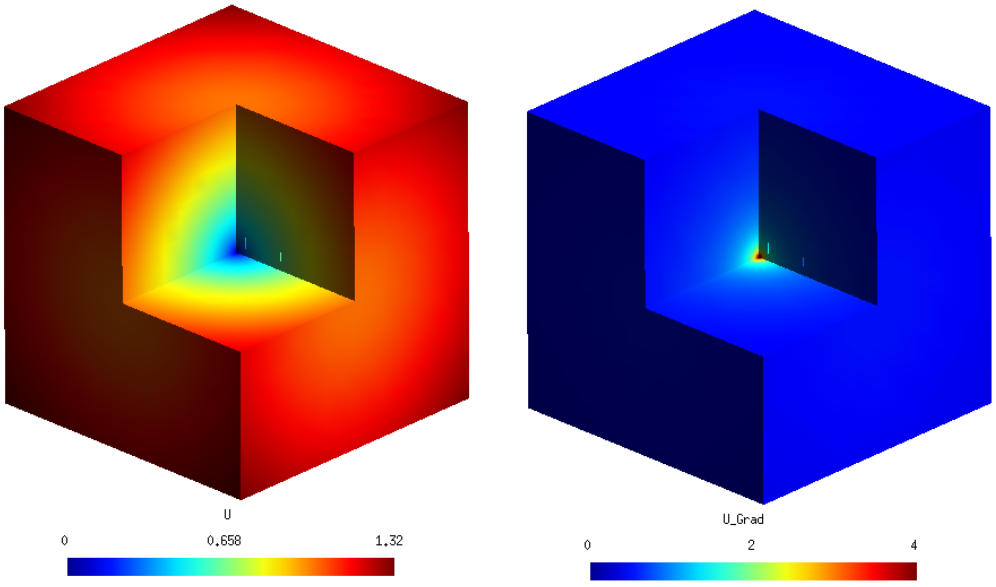


Figure 8.1: Exact solution and its gradient on the Fichera corner domain.

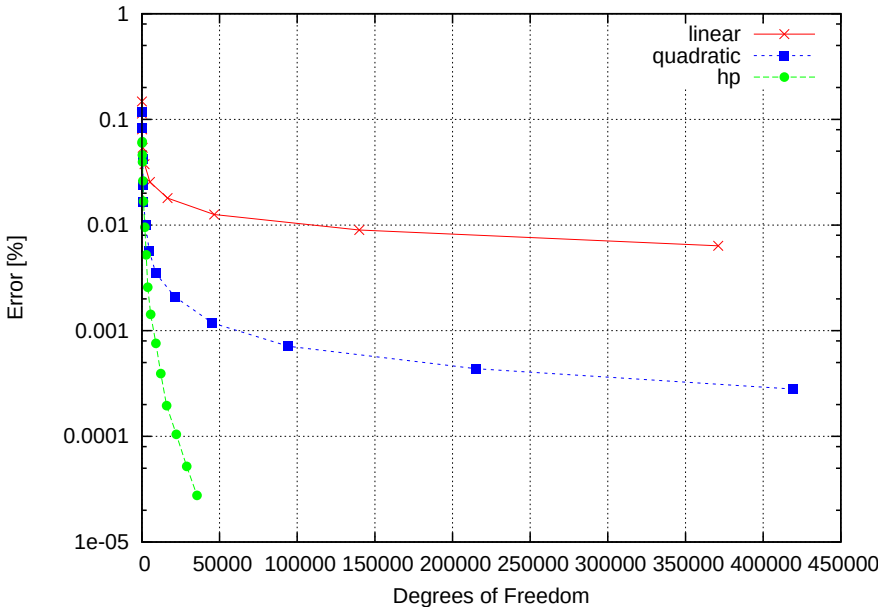


Figure 8.2: The convergence curves for adaptive FEM on meshes with arbitrary level hanging nodes using linear, quadratic and various orders (hp-FEM), respectively. Relative error in H^1 norm is shown on the vertical axis as a function of the number of DOFs.

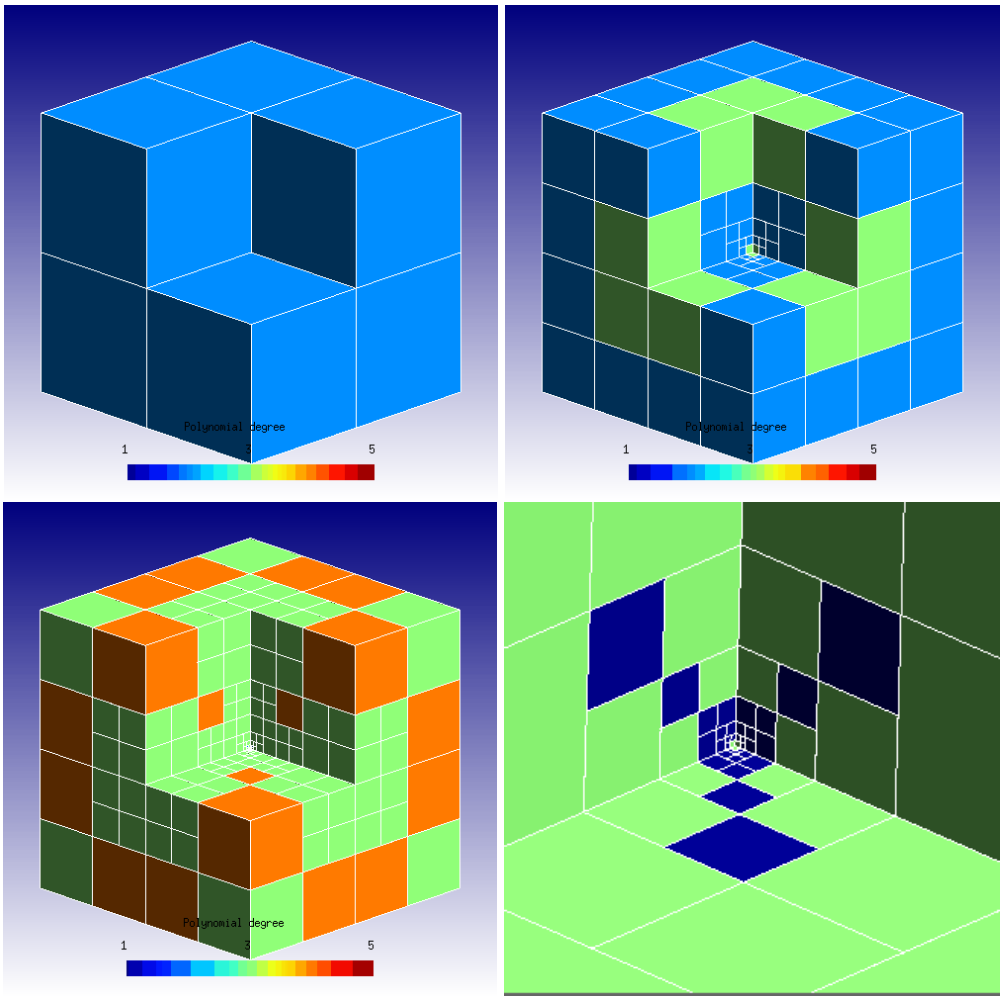


Figure 8.3: The *hp* meshes for the Fichera corner domain problem. We can see initial mesh (top-left), mesh after 9 refinement steps (top-right) and final mesh with detail of the singularity (bottom).

The missing part of the cube represents a metallic object. The solution, representing the electric potential in the surrounding air is smooth, but its gradient exhibits a strong singularity near the re-entrant corner. The exact solution and the gradient are depicted in Figure 8.1.

In Figure 8.3 we can see several meshes, that are obtained successively in the adaptivity process. It starts with simple mesh consisting of seven quadratic elements only. During the adaptivity process, elements are refined towards the singularity and also large higher-order elements appear in areas, where the solution

is smooth. This is in a good accordance with the theory.

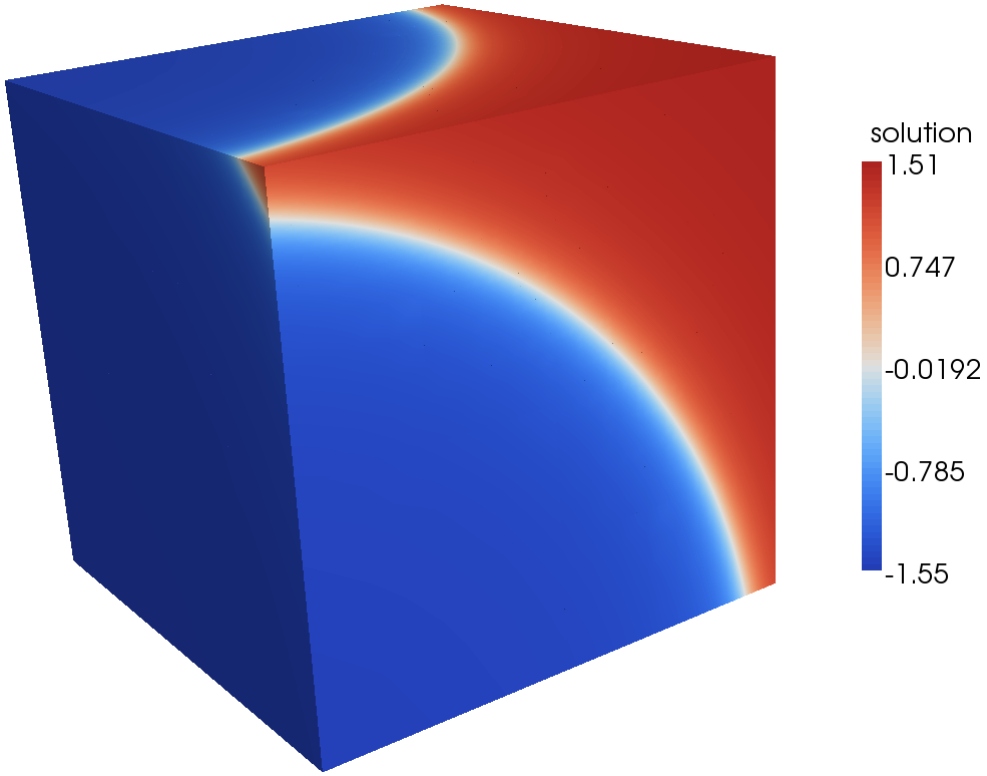


Figure 8.4: Exact solution of the shock problem.

It can be seen from the convergence curve in Figure 8.2 that the *hp*-FEM outperforms both piecewise-linear and piecewise-quadratic FEM significantly. Obviously if one does not require very small relative error, quadratic, or even linear elements can be used. On the other hand, if one needs really good approximation with relative error below 0.1 percent, both linear and quadratic approximations become too expensive and *hp*-FEM is the most suitable from compared methods.

8.2 Shock problem

In the second example we solve the same elliptic equation, but this time we construct the exact solution in such a way that instead of point singularity there is internal layer with large gradient. The problem is taken from [14]. We solve

Poisson's equation with mixed boundary conditions:

$$\begin{aligned} -\Delta u &= f \quad \text{in } \Omega, \\ u &= u_D \quad \text{on } \Gamma_{\mathrm{D}}, \\ \frac{\partial u}{\partial n} &= g \quad \text{on } \Gamma_{\mathrm{N}}, \end{aligned}$$

where $\Omega = [0,1]^3$, Γ_{D} consists of faces of Ω lying in planes $x = 0$, $y = 0$ and $z = 0$ and Γ_{N} consist of the remaining faces.

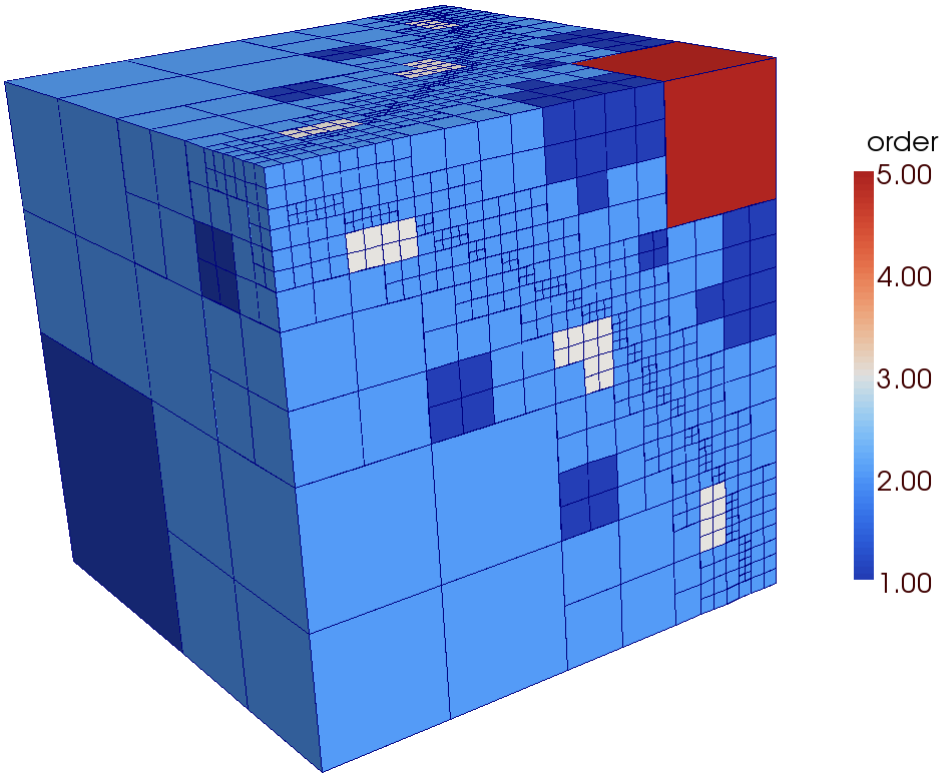


Figure 8.5: Mesh after 8 iterations of the shock problem.

The data f, u_D and g are chosen to comply with the exact solution

$$u(r) = \tan^{-1}(\alpha(r - r_0)),$$

where r stands for the distance of a point $\boldsymbol{x}$ from the point $(1/4, 1/4, 1/4)$ representing the center of the sphere, $r = |\boldsymbol{x} - (1/4, 1/4, 1/4)|$ and $r_0 = \sqrt{3}$ represents radius of the sphere. The internal layer is created by function $\tan^{-1}(\alpha x)$, which, for large α, grows very steeply close to $x = 0$, while being smooth (almost

constant) otherwise. We choose parameter $\alpha = 40$, where this sharp transition of $\tan^{-1}(\alpha x)$ around $x = 0$ (and therefore sharp transition of the exact solution around the described sphere) is very significant. Exact solution with sharp internal layer can be seen in Figure 8.4.

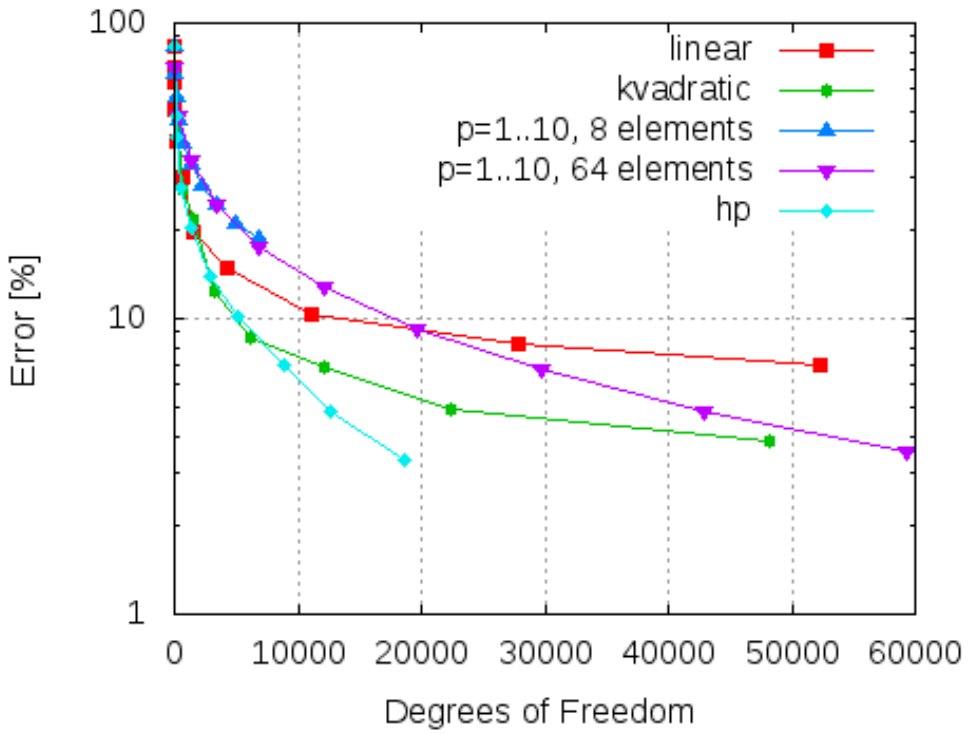


Figure 8.6: The convergence curves for different methods. First two correspond to adaptivity with linear quadratic elements. The other two correspond to calculations on regular fixed meshes consisting of 8 and 64 elements, respectively, using order from 1 to 10. Last curve shows *hp*-adaptivity. Relative error in energy norm is shown on the vertical axis as a function of the number of DOF.

We performed several calculations for different variants of the finite element method. The comparison of convergence curves can be seen in Figure 8.6. We used *h*-adaptivity for linear and quadratic elements, as in the previous example. In addition, we compare also the *p*-method. We performed calculations on regular fixed meshes obtained by splitting the computational domain into 8 and 64 cubic elements, respectively. On these meshes we always prescribe constant order p for $p = 1, \ldots, 10$. Finally we used *hp*-FEM method.

An example of the *hp* mesh can be seen in Figure 8.5. We can see very small

elements around the internal layer. More of the higher-order elements are inside of the domain, so they cannot be visible in the figure. We can also see examples of second-level hanging nodes. In convergence comparison in Figure 8.6 we can see, that, again, *hp*-adaptivity outperforms the other methods that have been considered.

8.3 H^{curl} example

Finally, let us present a simple numerical example using the $\boldsymbol{H}^{\text{curl}}$ space. The $\boldsymbol{H}^{\text{curl}}$ space is used to model electromagnetic problems. In this work we want to show a simple situation only. It will prove the benefits of higher-order $\boldsymbol{H}^{\text{curl}}$ elements, that have been described in Chapter 5.

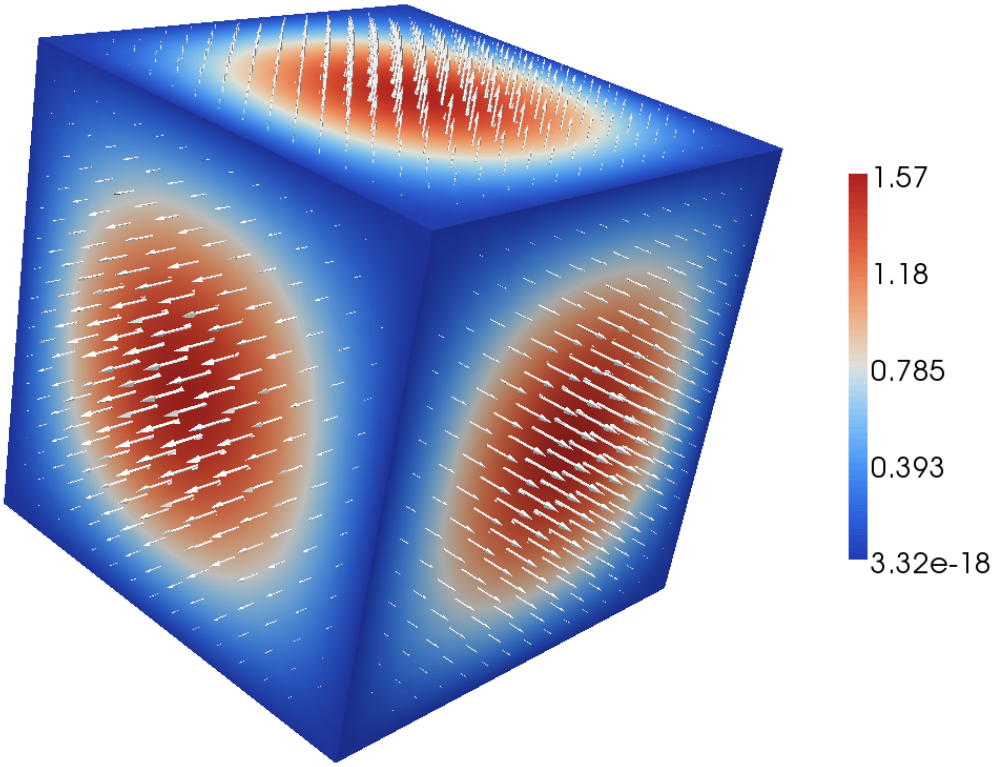


Figure 8.7: Exact solution of the $\boldsymbol{H}^{\text{curl}}$ problem.

In this example we compare the rate of convergence of uniform refinements in space (with all elements having the same order) with the rate of convergence of calculation on a fixed mesh with increasing order of elements. So no adaptivity is

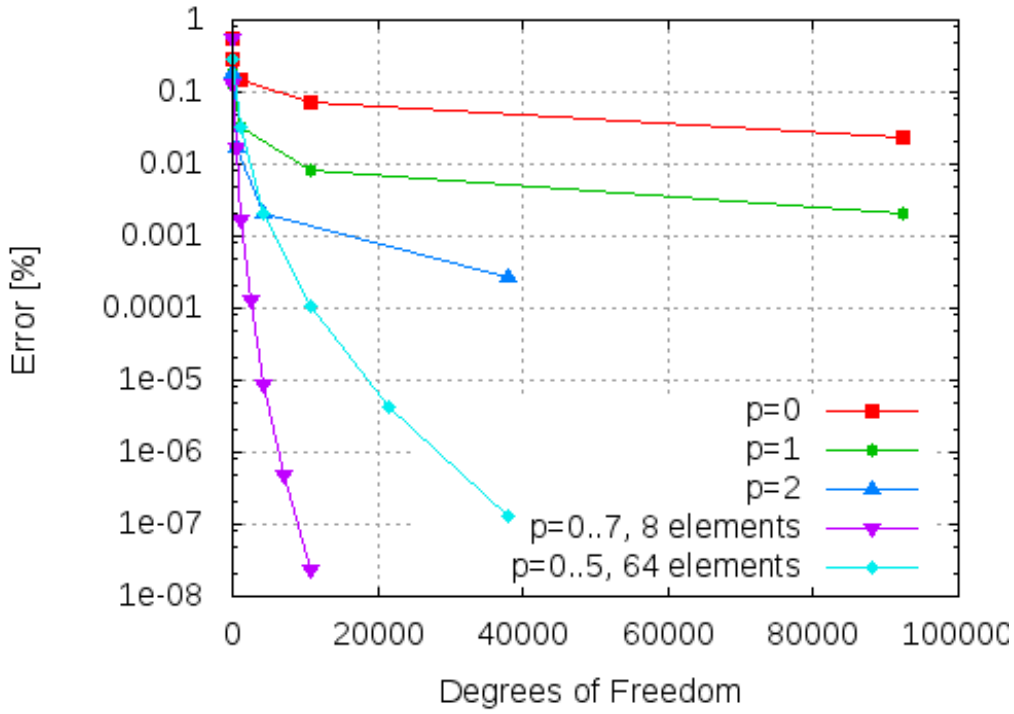


Figure 8.8: Comparison of convergence. First three curves correspond to uniform refinements of the whole mesh using the same order 0, 1 or 2, respectively, for all elements. Two other curves show convergence of p method on fixed mesh consisting of 8 and 64 elements with uniformly increasing orders. Unlike in the previous examples, no adaptivity is used.

involved in this case. We solve the equation

$$\nabla \times \nabla \times \boldsymbol{E} - \boldsymbol{E} = \boldsymbol{\Phi} \quad \text{in } \Omega,$$

which is a special case of (5.1). We consider boundary condition

$$\boldsymbol{E} \times \boldsymbol{\nu} = 0$$

on the whole boundary. The right hand side of the equation $\boldsymbol{\Phi}$ is chosen in such way, that the exact solution is

$$\boldsymbol{E}(x,y,z) = (x\cos(y)\cos(z), y\cos(z)\cos(x), z\cos(x)\cos(y))^T.$$

The exact solution is depicted in Figure 8.7. We can see, that the boundary condition is satisfied for this function, since all vectors on the boundary are perpendicular to it.

In Figure 8.8 we can see the comparison of errors, obtained by solving the problem with constant low order on successively uniformly refined meshes on one side and using one mesh with increasing order of elements on the other. The exact solution is very smooth, therefore it is not surprising, that better results are obtained using few elements with higher order. In fact, the best convergence is obtained using mesh with eight elements only with increasing order.

CHAPTER

NINE

CONCLUDING REMARKS

In this short final chapter we want to conclude our work and also point out some directions, in which the further development might go. As was mentioned in the preface, the main goal of this work was to create, implement and test algorithms for *hp*-adaptive finite element method, that uses conforming basis functions on meshes with arbitrary distribution of hanging nodes in three spatial dimensions. The main goals have been achieved, as we can see from the above description of algorithms and from the presented numerical examples, that have been solved using our code.

What first might have seemed as quilt straightforward extension of the 2D code, turned out to be algorithmically much more demanding in 3D. Namely handling of hanging nodes in 3D meshes and proper construction of global basis functions become very difficult to implement, since there are many different situations that have to be taken into account. In addition to that, visualization and therefore also testing is much more demanding in 3D. Finally, a necessity of more systematic approach become clear. The results of this effort were presented in the main part of this thesis. Despite all described problems, we succeeded to deliver functional code capable of solving benchmark problems.

The main part of this thesis is formed by a formal description and analysis of the most difficult parts of the implementation, which comprise mesh handling, following relations and constrains in the mesh and construction of conforming global basis functions of higher order. This is preceded by short, but hopefully sufficiently comprehensive introduction to the theory. It also suggests sources for more detailed study of the field. The last part of the thesis is dedicated to

computer realization of the described algorithms. First we analyze possibilities of faster numerical integration and make few remarks about the implementation. Then, in the last chapter, several numerical results are presented. The included convergence comparisons show advantages of the selected method.

The development of the HERMES 3D project is an ongoing process. The next step is the extension of internal data structures and algorithms necessary for solving coupled and time-dependent problems. The recent development goes in the direction of unifying common parts of HERMES 3D with HERMES 2D. This should solve the problems, since the missing features are similar for both 2D and 3D codes. This will conclude the first phase of the development of HERMES 3D.

There are, however, many efficiency-related issues, that have to be addressed. They are basically the same for 2D and 3D, but are more necessary for 3D, since calculations there are much more time demanding and without some improvements, the software would not be suitable for complex engineering problems solved on complicated geometries. The first group of improvements are strictly computer-related. Some of them are simple optimizations, that may save a lot of computer time in the phase of numerical integration and construction of stiffness matrix. For example, when elements in the mesh are cuboids and the weak form does not depend explicitly on space coordinates, one can precalculate local stiffness matrix and use it for all elements, which will speed up the assembly process dramatically.

When such conditions are not satisfied, a more general approach of evaluating each individual local stiffness matrix has to be used (as it is implemented now). However, there is still a possibility of huge savings of the CPU time by employing parallelism. Introduction of shared memory multiprocessing would be quite straightforward – by using OpenMP we can parallelize evaluation of integrals or even the whole local stiffness matrices, since those calculations are independent. Using MPI in order to turn HERMES into fully parallel application, capable of running on clusters, would require more effort. On the other hand, it could be simplified by using specialized libraries such as PETSc, that would take care of parallel matrix handling and solution of the linear problem.

Probably the most demanding possible improvements require more substantial changes to the whole adaptive algorithm. The most time-consuming part of the calculation is finding of the reference solution. If we found a way, how to avoid it, we would save a lot. The standard way, how to guide adaptivity procedure by using analytical estimator of the error (see, e.g. [2]) does not, however, meet our needs. For the hp-adaptivity we need to estimate not only the size of the error, but also its shape if we want to be able to select appropriate refinement candidates. Moreover, we want the method to be robust and problem-independent, so that it

can be used for various types of coupled problems without further adjustments. Deriving analytical estimators with such conditions would be extremely hard, if not impossible.

One of the possible solutions to this problem would be to solve more reference problems on smaller parts of the domain, which may be faster, but hopefully will still provide enough information to guide the *hp*-adaptivity. This idea is among those, that should be investigated in the future.

BIBLIOGRAPHY

[1] M. Ainsworth and J. Coyle. Hierarchic *hp*-edge element families for Maxwell's equations on hybrid quadrilateral/triangular meshes. *Comput. Methods Appl. Mech. Engrg.*, 190:6709–6733, 2001.

[2] M. Ainsworth and J. T. Oden. *A Posteriori Error Estimation in Finite Element Analysis*. John Wiley & Sons, Inc, New York, 2000.

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

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

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

[6] D. Bachman. *A Geometric Approach to Differential Forms*. Birkhäuser, Boston, 2006.

[7] W. Bangerth, R. Hartmann, and G. Kanschat. deal.II - A general-purpose object-oriented finite element library. *ACM Trans. Math. Softw.*, 33(4):24, 2007.

[8] P. Bochev and J. Hyman. Principles of mimetic discretizations of differential operators. *Compatible Spatial Discretizations*, pages 89–120, 2006.

[9] A. Bossavit. *Computational Electromagnetism: Variational Formulation, Edge Elements, Complementarity*. Academic Press, Boston, 1998.

[10] A. Bossavit and J. Verite. A mixed fem-bem method to solve 3D eddy current problems. *IEEE Trans. Magn.*, 18:431–435, 1982.

[11] J. Brandts, S. Korotov, M. Křížek, and J. Šolc. On nonobtuse simplicial partitions. *SIAM Rev.*, 51(2):317–335, 2009.

[12] R. D. Cook, D. S. Malkus, M. E. Plesha, and R. J. Witt. *Concepts and Applications of Finite Element Analysis*. John Wiley & Sons, Inc, Hoboken, 1974.

[13] T. A. Davis. *Direct Methods for Sparse Linear Systems*. SIAM, Philadelphia, 2006.

[14] L. Demkowicz, J. Kurtz, D. Pardo, M. Paszynski, W. Rachowicz, and A. Zdunek. *Computing with hp-Adaptive Finite Elements, Volume 2*. Chapman & Hall/CRC Press, Boca Raton, 2008.

[15] L. Demkowicz, D. Pardo, and W. Rachowicz. 3D *hp*-adaptive finite element package (3dhp90). *TICAM Report 02-24, University of Texas at Austin*, 2002.

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

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

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

[19] L. Dubcová. *hp*-FEM for coupled problems in fluid dynamics. *Doctoral thesis, Charles University, Prague*, 2010.

[20] L. Dubcová, P. Šolín, J. Červený, and P. Kůs. Space and time adaptive two-mesh hp-finite element method for transient microwave heating problems. *Electromagnetics*, 30(1):23–40, 2010.

[21] T. Eibner and J. M. Melenk. Fast algorithms for setting up the stiffness matrix in *hp*-fem: a comparison. *Numerical Analysis Report 3/05*.

[22] P. Frauenfelder and C. Lage. Concepts - An object-oriented software package for partial differential equations. *M2AN*, 36(5):937–951, 2002.

[23] J. Jin. *The Finite Element Method in Electromagnetics*. John Wiley & Sons, Inc, New York, 2002.

[24] G. E. Karniadakis and S. Sherwin. *Spectral/hp Element Methods for Computational Fluid Dynamics*. Oxford University Press, New York, 2005.

[25] S. Korotov and M. Křížek. Local nonobtuse tetrahedral refinements of a cube. *Appl. Math. Letters*, 16:1101–1104, 2003.

[26] P. Kůs. Integration in higher-order finite element method in 3D. *Proceedings PANM 2010, Dolní Maxov*, pages 131–136, 2010.

[27] P. Kůs, P. Šolín, and I. Doležel. On adaptive hp-FEM with arbitrary-level hanging nodes in 3D. *Proceedings ENUMATH 2007, Graz, Austria*, 2007.

[28] P. Kůs, P. Šolín, and I. Doležel. Solution of 3D singular electrostatics problems using adaptive *hp*-FEM. *COMPEL*, 27(4):939–945, 2008.

[29] J. M. Melenk. *hp-Finite Element Methods for Singular Perturbations*. Springer-Verlag, Berlin Heidelberg, 2002.

[30] J. M. Melenk, K. Gerdes, and C. Schwab. Fully discrete *hp*-finite elements: Fast quadrature. *Research Report No. 99-15*, 1999.

[31] P. Monk. An analysis of Nédélec's method for spatial discretization of Maxwell's equations. *J. Comput. Appl. Math*, pages 103–121, 1993.

[32] P. Monk. On the *p*- and *hp*-extension of Nédélec's h(curl)-conforming elements. *J. Comput. Appl. Math.*, pages 117–137, 1994.

[33] P. Monk. *Finite Element Methods for Maxwell's Equations*. Clarendon Press, Oxford, 2002.

[34] G. Mur. Edge elements, their advantages and their disadvantages. *IEEE Trans. Magn.*, 30:3552–3557, 1994.

[35] J. C. Nédélec. A new family of mixed finite elements in R^3. *Numer. Math.*, 50:57–81, 1986.

[36] A. Quarteroni and A. Valli. *Numerical Approximation of Partial Differential Equations*. Springer-Verlag, Berlin, 1997.

[37] P. A. Raviart and J. M. Thomas. Primal hybrid finite element methods for second-order elliptic equations. *Math. Comput.*, 31:391–413, 1997.

[38] Y. Saad. *Iterative Methods for Sparse Linear Systems*. SIAM, Philadelphia, 2003.

[39] A. Schröder. Constraints coefficients in *hp*-FEM. *Proceedings of ENUMATH Conference, Graz, Austria*, pages 183–190, 2007.

[40] C. Schwab. *p- and hp-Finite Element Methods*. Calderon Press, Oxford, 1998.

[41] S. A. Smolyak. Quadrature and interpolation formulas for tensor product of certain classes of functions. *Dokl. Akad. Nauk*, pages 240–243, 1963.

[42] B. Szabo and I. Babuška. *Finite element analysis*. John Wiley & Sons, New York, 1991.

[43] J. Červený. Adaptive finite element methods for nonlinear coupled problems. *Master Thesis, University of Texas at El Paso*, 2007.

[44] T. Vejchodský, P. Šolín, and M. Zítka. On some aspects of the *hp*-FEM for time-harmonic Maxwell's equations. *Numerical Mathematics and Advanced Applications (Proceedings of ENUMATH 2005)*, pages 691–699, 2006.

[45] T. Vejchodský, P. Šolín, and M. Zítka. Modular *hp*-FEM system HERMES and its application to the Maxwell's equations. *Math. Comp. Simul.*, 76:223–228, 2007.

[46] P. Šolín. *Partial Differential Equations and the Finite Element Method*. John Wiley & Sons, Inc, Hoboken, New Jersey, 2004.

[47] P. Šolín, L. Dubcová, and I. Doležel. Adaptive hp-FEM with arbitrary-level hanging nodes for Maxwell's equations. *Adv. Apl. Math. Mech*, 2(4):518–532, 2010.

[48] P. Šolín, K. Segeth, and I. Doležel. *Higher-Order Finite Element Methods*. Chapman & Hall/CRC Press, Boca Raton, 2004.

[49] P. Šolín, J. Červený, and I. Doležel. Arbitrary-level hanging nodes and automatic adaptivity in the *hp*-FEM. *Math. Comput. Simulation*, 2007.

[50] P. Šolín, M. Zítka, and I. Doležel. On a *hp*-finite element method for singular electro- and magnetostatic problems. *Proceedings of ISEF, Baiona, Spain*, 2005.

[51] H. Whitney. *Geometric Integration Theory*. Princeton University Press, Princeton, 1957.

[52] M. Zítka. On some aspects of adaptive higher-order finite element method for three-dimensional elliptic problems. *Doctoral thesis, Charles University, Prague*, 2008.